Mathematical Modelling and Numerical Simulation with Applications, 2024, 4(3), 351–369

https://dergipark.org.tr/en/pub/mmnsa
ISSN Online: 2791-8564 / Open Access
https://doi.org/10.53391/mmnsa.1406374

RESEARCH PAPER

A multi-step mathematical model-based predictive strategy for software release timing during testing stage

Poonam Panwar [1,‡], **Satish Kumar** [2,‡], **Shakuntala Singla** [3,‡] **and Yeliz Karaca** [4,*,‡]

[1]University School of Computing, Rayat Bahra University, Mohali-140301, India, [2]Faculty of Agriculture, Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala-133207, India, [3]Department of Mathematics and Humanities, MMEC, Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala-133207, India, [4]University of Massachusetts Chan Medical School, Worcester, MA 01655, USA

[*] Corresponding Author
[‡] rana.poonam1@gmail.com (Poonam Panwar); satishrana.biotech@gmail.com (Satish Kumar); shakusingla@gmail.com (Shakuntala Singla); yeliz.karaca@ieee.org (Yeliz Karaca)

Abstract

Mathematically precise modeling is important to be established to accurately examine the quantitative relationship between software testing and software reliability. Software testing process is complex since it is concerned with various factors such as test case execution, defect debugging, tester expertise, test case selection, and so forth. For this reason, it is required to be meticulous in formulating the software testing process in a manner which is mathematically concise. The software release life cycle or sequential release timeline, referring to the process related to the development, testing and distribution of a software product comprises several critical stages, and the length of this particular life cycle reveals variations depending on different factors like the type of product, the intended use of it, industry security, general standards and compliance. One consideration software engineers have is related to the release date of the software so that future commitments about the software's release time can be formulated beforehand. In view of these aspects, a multi-step strategy for predicting software release dates is proposed in the current study along with the following stages: firstly, the proposed technique selects the utmost reliability growth model that very well fits the observed test data halfway through the testing period, and then employs it to forecast the probable date of release. This technique entails approximating the unknown parameters of suitable Software Reliability Growth Models (SRGMs). Finally, the chosen SRGM is used to forecast the release date of the software under test by fitting it to available fault data. The proposed method is straightforward and applied to test on a total of ten actual datasets collected from the literature. The results of the proposed technique reveal that in the majority of the situations, nearly exact approximation of date of release can be made halfway through the testing period. Moreover, the proposed method's performance is also compared to that of a number of previous strategies present in the literature. The outcomes obtained by our study demonstrate that the proposed strategy may be used to forecast the release date of software in practical situations.

Keywords: Nonhomogeneous Poisson process; mathematical prediction modeling; software reliability model; stochastic processes; error content function; software source code; goodness of fit

AMS 2020 Classification: 60G35; 93E10; 60G55; 60J45; 94B70

1 Introduction

Mathematical modeling, describing a system by a set of equations and variables, is employed for establishing relationships among them, and in control of the system, it has a critical value for the accurate examination of the quantitative relationship between software testing and software reliability. Software testing process, as a complex one, is concerned with various factors such as test case execution, defect debugging, tester expertise, test case selection, and so on. For this reason, it is required to be meticulous in formulating the software testing process in a reliable manner. Computer software is used in practically every facet of human endeavor, and it is of utmost significance to devise, build and test the software appropriately before being released. Software development takes a long time and comes with a substantial amount of financial burden. When software is developed, it is thoroughly tested before being released to ensure that it is bug-free and hence trustworthy. In reality, reliability is the most crucial characteristic for a well-designed software application. Accordingly, a software reliability model indicates the form of a random process defining the behavior of software failures to time, and these models have emerged as more understanding has become a requisite to examine the features of the way and reason software fails, with an attempt to quantify software reliability. Musa and Okumoto [1] defined reliability of any software application as the likelihood of operation with no failures in any given environment for a specific amount of time. In practice, project managers find it challenging to assess software reliability. A variety of Software Reliability Growth Models (SRGMs) has been proposed since the early 1970s [1–4] for the evaluation of reliability growth of systems throughout software developments specially during the completing and testing periods of the software concerned. The number of expected failures within a certain time period is a widely accepted indicator for assessing a product's reliability. Failures are the result of software code faults, and even a single flaw can result in several failures. Furthermore, software engineers are often interested in projecting the software's expected release date while it is still in development so that future delivery commitments can be made timely. With this in mind, software engineers used specialized development approaches to reduce the overall risk and support rapid change. As a result, there is a significant issue in predicting the likely release date of software in development with sufficient accuracy. Existing techniques, such as a cumulative flow methodology, release backlogs are used in software development to anticipate and set release dates; however, because this does not take software reliability into account when projecting release dates, there is a risk that software at the predicted release date may be unreliable. Software system availability depends on reliability, and SRGMs can be used to determine whether sufficient defects have been eliminated in order to release the software.

A software economic policy was developed by Huang et al. [5] offering a thorough examination of software based on test efficiency and cost. Project managers may also benefit from the strategy by using it to assist them decide when to finish testing in preparation for market release. A SRGM that takes into account the impact of imprecise fault debugging and error creation was proposed by Kapur et al. [6]. The suggested model is used to define the release time problem, which minimizes the estimated cost while meeting the minimal dependability level that must be met by the release time. By creating a software cost model with a risk component, Singh

➤ **Received:** 29.12.2023 ➤ **Revised:** 14.09.2024 ➤ **Accepted:** 25.09.2024 ➤ **Published:** 30.09.2024

and Kumar [7] provided a technique for determining when to conclude the testing phase and deliver the program to the end user. They addressed the question of how to determine when to finish testing and release the product. A method for building a software reliability growth model based on the Non-Homogeneous Poisson Process was presented by Quadri et al. [8]. Despite the fact that several testing-effort functions based on the non-homogeneous Poisson process (NHPP) have been developed for the software reliability growth model. They examined the scenario in which the Generalized Exponential Distribution (GED) describes the time-dependent behaviors of testing-effort expenditures. The NHPP is used to create SRGMs, which include the (GED) testing-effort spent during the software-testing phase. A mathematical modeling approach for numerous software product releases is proposed by Kapur et al. [9]. Their suggested model uses a Cobb Douglas production function to simulate the failure process using a software reliability growth model, accounting for the combined effects of schedule pressure and resource constraints. A technique for choosing SRGMs to forecast the overall amount of errors in software was suggested by Panwar and Lal [10]. To assess how effectively the technique predicts the predicted total number of software failures, it is used to a case study consisting of three datasets of defect reports from system testing of three versions of a big medical record system. In order to offer more accurate predictions, Choudhary and Baghel [11] provide an efficient software dependability modeling based on Cuckoo Search optimization, Ensemble Empirical Mode Decomposition, and Autoregressive Integrated Moving Average (ARIMA) modeling of time series. Panwar and Kaur [12] suggest a method for estimating the number of software defects that remain by utilizing both perfect and imperfect software reliability growth models. A software metrics-based technique for software reliability prediction is presented by Shi et al. [13]. Metric measurement outcomes are linked to quantitative reliability forecasts by taking into account defect data and operational conditions.

Although numerous models have been presented researched, and implemented, the majority of them are failure count models that do not account for the many development scenarios like developers team structure or a substantial reduction in development time. As a result, standard models are unable to reliably estimate the release dates. Hence, in the present study, a method for obtaining reliability estimations is proposed which can determine the product's likely release date during the product testing stage. Previously, only basic SRGMs were employed in the studies, however the proposed method, as a novelty, suggests that NHPP SRGMs can model the circumstances more practically. The objective of this study is to respond to the following questions:

- Is it possible to forecast software release dates using NHPP SRGMs?
- Is our proposed method more accurate than the previously proposed methods in terms of predicting the release dates?

The following can be put forth among the contributions to the proposed work.

- A multi-step strategy for predicting software release time by dividing development time into various degrees of testing.
- A method for estimating the release date forecast precision by specifying a desired level of confidence.
- An evaluation result demonstrating that our prediction method outperforms previous models.

The following is a breakdown of the paper's structure. Section 2 provides a basic introduction to SRGMs. Section 3 describes the suggested strategy, which is then tested on 10 real datasets in Section 4 to see how effective it is. Finally, in Section 5, conclusions based on the current study along with the future directions are drawn and discussed.

2 Non-homogeneous Poisson process software reliability growth models

NHPP class of SRGMs has been broadly used in the literature [14–18]. These models use test data from failure history, predicting the software application's projected total number of faults and forthcoming reliability. This class of models can also be used to approximate the sum of remaining software faults and the amount of time period it will take to identify these. Mean value function is used is each model of this category, which is dependent on diverse conventions about the equations of error content and the defect finding rate. The NHPP SRGMs models are commonly categorized as follows based on the patterns of their mean value functions, which are the concave models and S-shaped models.

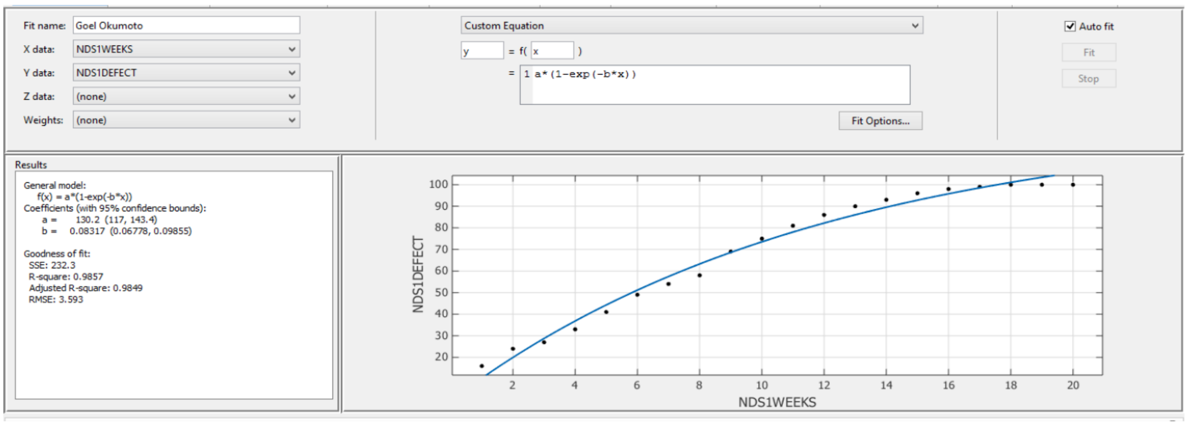
The defect debugging process is depicted naturally in concave models, in which the faults discovered accumulate as the testing activity proceeds, and the aggregated faults propagates at a gradual pace before approaching asymptotic behaviour as the software behavior stabilizes. On the other side, S-shaped curve models show a steady fault discovery at the beginning of the debugging stage [19]. As testing progresses, the rate of defect identification increases, and the cumulative defects curve finally approaches asymptotic behavior [20]. This class also distinguishes between finite and infinite failure models. Finite failure models presume that a fault-free product may be developed in the end, as well as an asymptotic methodology to a predictable value. The failure models of infinite class, on the other hand, presume that the count of observed faults is inestimable, implying that the function of mean value is unrestrained. Numerous models similarly imply that whilst correcting existing issues, new bugs may be introduced inadvertently. These are denoted as imperfect debugging models [21]. Five concave, nine S-shaped, and two more models that can perform as a concave otherwise S-shaped are employed in our proposed study.

3 Multi-step mathematical model-based predictive strategy

In the literature, various methods and practices for selecting appropriate SRGM are suggested [22–32]. However, the majority of those are dependent on the particular situation and may not be applied with certainty in all situations. Present study proposes a method for selecting the best SRGM along with using that one to forecast the likely date of release with the intention to make required arrangements and revisions ahead of time to fulfil any deadlines. The procedure is straightforward and has shown to be beneficial in the recent study. It entails choosing an SRGM that almost fits the existing error content and then use it next to forecast the date of software release. The proposed technique demands calculating the unknown variables/parameters of applicable SRGMs before ordering these according to their behavior on observed failure data. Finally, best chosen SRGM is utilized for forecasting the date of release of any software under test. The strategy proposed is not scenario-specific and can be used to any situation. It works in the way whose specific details are provided in the remaining parts of our study.

Estimation of model parameters

NHPP SRGM has some unknown parameters that must be determined from observed test failure data. Maximum Likelihood Estimation (MLE) or Least Squares Estimate (LSE) are the two methods which can be used on the currently available test data, to determine the value of these unknown parameters [33, 34]. The MLE method estimate these parameters by solving a set of simultaneous equations whereas LSE reduces the TSS (total sum of square of variation) between observed and probable faults depending on the hypothetical chosen model. There are also a number of tools available to estimate the value of these unknown parameters like Curve Fitting MATLAB [35] which is based on the LSE technique. Moreover, our individual proficiency has also proved that the LSE provides more appropriate parameter values as contrasted to MLE, allowing the model



Fit name: Goel Okumoto
X data: NDS1WEEKS
Y data: NDS1DEFECT
Z data: (none)
Weights: (none)

Custom Equation
y = f(x)
= a*(1-exp(-b*x))

[✓] Auto fit

```
Results
General model:
    f(x) = a*(1-exp(-b*x))
Coefficients (with 95% confidence bounds):
      a =      130.2  (117, 143.4)
      b =    0.08317  (0.06778, 0.09855)

Goodness of fit:
  SSE: 232.3
  R-square: 0.9857
  Adjusted R-square: 0.9849
  RMSE: 3.593
```

Figure 1. Estimation of unknown parameters by MATLAB curve fitting tool

to better fit the actual data. As a result, we chose to employ the LSE technique for parameter estimation in using MATLAB's curve fitting tool. After analyzing the existing test data, we first determine which models appear to be more suitable for fitting this data, and then calculate the values of RSq and RMSE for each of these SRGMs to determine which one best matches the data. Figure 1 shows how the Curve Fitting tool fits the SRGM to the given test data and computes the values of parameters that are unknown in nature.

Ranking of models

In the second step of proposed technique, a comparison criterion (1) is proposed to compare models realistically in order to examine the efficiency of software reliability growth models employed in the proposed study. Based on our experience, using a vast set of comparison criteria is not necessary, and in most situations, it does not even assure trustworthy forecasts. Hence, we discovered that the subsequent modest criterion may be implemented to rank rival models of software reliability in order to choose an optimal SRGM for more accurate release date predictions.

$$\text{Rank Index} = \frac{1}{2}\left[\frac{RSq_j}{max_j^n(RSq_j)} + \frac{minjn(RMSEj}{RMSEj)}\right]. \tag{1}$$

The relative amount of variation in the actual test data and the test data estimated by the matching SRGM is shown by RSq. The higher the RSq score, the greater variation there is in the actual and estimated test data values. The RSq is computed as the proportion of the residuals sum of squares (SSR) and the total sum of squares (SST). Here, j denotes the number of the SRGM as provided in Table 1. Also, we have

$$RS_q = \frac{SSR}{SSQ}, \tag{2}$$

where SSR is defined as

$$SSR = \left(\sum_{i=1}^{n}\widehat{m(t_i)} - \sum_{i=1}^{n}\frac{(m(t_i))}{n}\right)^2. \tag{3}$$

The sum of squares about the mean, or SST, is defined as: In (3) and (4) i signifies the test period and $m(t_i)$ the real number of faults discovered up to time t_i. Next $m(t_i)$ represents the calculated value of cumulative failures until time ti as determined by SRGM under study and $m(t)$ represent the mean value of reported total failures. The regression's fit standard error is denoted by $RMSE$ in (1). It is a calculation of the random component's standard deviation, and it is defined as:

$$RMSE = \sqrt{MSE}. \tag{4}$$

$MSE = SSE/v$ is the "mean square error" or "the residual mean square" whereas SSE is the aggregate divergence of the genuine measured faults from the approximated of faults using SRGM. SSE can be calculated by using equation $SSE = \sum_{i=1}^{n}(m(t_i) - (mt_i)^2$ and v is the degree of freedom. The number of fitted coefficients m subtracted from the total count of response values n is the degree of freedom. All competing models' rank index values are obtained in (1), and next they are ordered in increasing sequence of these values. The model with the highest rank index value receives rank 1. If it results in a draw (any two or more models have identical values of rank index), they are together regarded to be of same rank. The most appropriate model that best captures the behavior of the test data is model ranked one.

Forecasting the release date

In the next step, using the chosen model and the error content function $(a(t))$ of the selected rank 1 SRGM, we estimate the total number of predicted errors in the software. The total number of errors that may occur in software over its lifetime is the value of the error content function. The error content function $(a(t))$ of each model is given in Table 1. The following equation is used to measure the software's reliability over time using the mean value function and error content function stated in Table 1.

$$R(t) = m(t)/a(t). \tag{5}$$

The conditional reliability $(R(s|t))$ in interval $(t, t+s)$ is estimated using

$$R(s|t) = e^{[m(t+s)-m(t)]}. \tag{6}$$

The likelihood that the obtained reliability at any point of time t may not alter in this gap is given by conditional reliability $(t, t+s)$. By increasing the value of time t stepwise in Eq. (5) and Eq. (6) the future prediction about reliability and conditional reliability is done. We increase the value of t by 1 in each step and finally time of release t is considered the time when $R(t) \geq 0.960$ and $R(s|t) \geq 0.500$ for $s = 1$ and $R(s|t) \geq 0.350$ for $s = 2$.

The proposed method with its relevant stages

- Estimate the length of the testing period when the program is ready for testing and continue testing until at least 50% of the testing time has passed.
- Choose the acceptable models from Table 1 that should fit the data into the best of your ability.
- Calculate the unknown parameters of the selected models using Section 3, Then use Section 3 to choose the model with the highest rank.
- This model is then used to calculate $R(t)$ and $R(s|t)$.
- Take this as the time of release if $R(t)$ which meets the necessary level of reliability and $R(s|t)$ for the next two-time units is acceptable. If the anticipated release date is to be met, adapt the testing infrastructure accordingly. When around 75% of the expected release time has passed, it is often recommended to update the estimations again.

Table 1. SRGMs investigated

No	Model	Category	Mean Value (m(t)) Equation	Remarks
1	Goel-Okumoto (GO) [2]	Concave	$m(t) = a(1 - e^{-bt})$, $a(t) = a,\ \ b(t) = b$,	Known as the exponential growth model
2	Generalized Goel [2]	Concave	$m(t) = a(1 - e^{-bt^c})$, $a(t) = a,\ \ b(t) = b$,	Goodness-of-fit is better than the GO-Model. For $c = 1$, the same as the GO-Model
3	Modified Duane [19]	Concave	$m(t) = a[1 - (b/(b+t))^c]$, $a(t) = a,\ \ b(t) = b$,	Assume independence of failure occurrences
4	Musa-Okumoto [2]	Concave	$m(t) = a\ln(1 + bt)$, $a(t) = a,\ \ b(t) = b$,	Assumes that the severity of failure decreases exponentially as the predicted number of errors increases
5	Yamada Exponential [36]	Concave	$m(t) = a(1 - e^{r\alpha}(1 - e^{\beta t}))$, $a(t) = a, b(t) = r\alpha\beta e^{-\beta t}$,	Make an attempt to account for the time spent testing
6	Gompert [37]	S-Shaped	$m(t) = ake^{-bt}$, $a(t) = a$, $b(t) = b$,	Estimates the severity of software errors. In addition, it forecasts demand, economic growth, and future population
7	Inflection S-Shaped [38]	S-Shaped	$m(t) = (a(1 - e^{bt}))/(1 + \beta e^{-bt})$, $a(t) = a$, $b(t) = b/1 + \beta e^{-bt}$,	With the GO model, a technical problem is solved. If $k = 0$, the result is the same as GO-Model
8	Logistic Growth [24]	S-Shaped	$m(t) = a/(1 + ke^{-bt})$,	Calculates the amount of error in software systems
9	Delayed S-Shaped [36]	Concave	$m(t) = a(1 - (1 + bt)e^{-bt}$, $a(t) = a$, $b(t) = (b^2 t)/(1 + bt)$,	The GO model has been modified to become S-shaped
10	Yamada-Imperfect-Debugging Model I [36]	S-Shaped	$m(t) = ab/(\alpha + b)(e^{\alpha t} - e^{bt})$, $a(t) = ae^{\alpha} t$, $b(t) = b$,	Assumes a constant fault detection rate and an exponential fault content function
11	Yamada-Imperfect-Debugging Model II [36]	S-Shaped	$m(t) = a[1 - e^{-bt}][1 - \alpha/b]$ $+ \alpha at$, $a(t) = a(1 + \alpha t)$, $b(t) = b$,	Assumes constant rate of introduction α and a constant rate of fault detection
12	Yamada-Rayleigh [36]	S-Shaped	$m(t) = a(1 - e^{-r\alpha(1 - e^{(\beta t^2/2)})})$, $a(t) = a$, $b(t) = r\alpha\beta te^{-\beta t^2/2}$,	Make an effort to report the time spent testing
13	Pham-Zhang-IFD [39]	S-Shaped	$m(t) = a - ae^{-bt}(1 + (b + d)t$ $+ bdt^2)$, $a(t) = a$, $b(t) = b$,	Maintains an initial constant function fault count and an imperfect detection rate of fault considering fault introduction phenomenon
14	Zhang-Teng-Pham [40] model (ZT Pham)	S-Shaped	$m(t) = a/(p - \beta)[(1 - (1 +$ $\alpha)e^{-bt}/1 + \alpha e^{-bt})^{(c/b(p-\beta))}]$, $a(t) = \beta(t)m(t)$, $b(t) = c/(1 + \alpha e^{-bt}), \beta(t) = \beta$,	Considers a constant rate of fault introduction and a non-decreasing function of fault detection rate
15	Pham-Nordman-Zhang [41] (PNZ Model)	S-Shaped & Concave	$m(t) = (a(1 - e^{-bt})(1 - \alpha/b)$ $+ \alpha at)/(1 + \beta e^{-bt})$, $a(t) = a(1 + \alpha t)$, $b(t) = b/(1 + \beta e^{(} - bt))$,	Considers that the fault detection rate is non-decreasing and the introduction rate is a linear
16	Pham-Zhang model (PZ Model) [40]	S-Shaped & Concave	$m(t) = 1/(1 + \beta e^{-bt})$ $((c + a)(1 - e^{-bt})$ $- ab/(b - \alpha)(e^{-\alpha t} - e^{-bt}))$, $a(t) = c + a(1 - e^{-\alpha t})$, $b(t) = b/(1 + \beta e^{-bt})$,	The exponential rate of introduction is and non-decreasing rate of fault detection

Table 2. Tandom computer software failure (dataset 1)

Time (in weeks)	Processor hours	Faults found	Time (in weeks)	Processor hours	Faults found
1	519	16	11	6539	81
2	968	24	12	7083	86
3	1430	27	13	7487	90
4	1893	33	14	7846	93
5	2490	41	15	8205	96
6	3058	49	16	8564	98
7	3625	54	17	8923	99
8	4422	58	18	9282	100
9	5218	69	19	9641	100
10	5823	75	20	10000	100

4 Implementation on testing data

In this section, the suggested approach is applied on 10 real datasets from the literature [4, 22, 23, 25, 26, 42]. In this study, 16 NHPP SRGMs, which are given in Table 1 are applied on all the datasets. In the same table, the characteristics of these SRGMs are also summarized. The proposed strategy is used at three levels of testing, when 50% of the testing is finished, next after 75% of the testing is ready, and lastly, once the testing of software finished. The presented technique's operation is described in more depth using various examples below.

Example 1 *A dataset (DS1) with 100 observed faults was gathered from the public domain of literature for examination; the dataset is listed in Table 2 and was acquired from a subsection of artifacts for four different Tandem Computers Company software versions. The count of errors was normalized as 0 to 100 to eliminate confidentiality concerns, and the amount of energy consumed was translated correspondingly into the scale (0 to 10,000) [23, 25, 26].*
To evaluate the unknown parameters of SRGMs, LSE approach was utilized for NHPP SRGMs considered under study, with confidence bounds of 95%. The parameters were estimated using MATLAB at time $t = 10$ weeks, as indicated in Section 3; this is when 50% of the testing is completed. Table 3 shows the estimated values of the parameters for each of the 16 models. Following that, using Eqs. (2)-(5), the values of the comparison criteria (RSq and RMSE) presented in this study paper were obtained. Table 4 shows the estimated RSq and RMSE values for dataset 1 with $t = 10$ weeks (i.e. this is the time when almost half of the testing is complete).
The rank index is then determined using (1). The models are next ranked accordingly in descending rank index values based on the value of the derived rank index (i.e. model with the highest value of rank index is allotted rank 1). Table 5 shows the predicted rank index values and model ranking of dataset 1 considering the fault data of ten weeks.
Table 5 further demonstrates that at this stage of testing, when only half of the test data is available, Gompertz has ranked one model. As a result, Gompertz is now utilized to predict software delivery dates utilizing Eqs. (5) and (6). With the given level of reliability and conditional reliability, the projected release time at this stage of testing is 35 weeks. Table 6 shows the estimated value of the release date, reliability, and conditional reliability with this data.
The same process is repeated again when testing has been done up to 15 weeks (i.e., 75% test plan is complete). At this point, the top-ranking model has been identified as logistic growth. Table 6 shows the estimated time of release with conditional reliability and reliability which can be reached upon this day. The method was again repeated using 20 weeks of full test data to get the conditional reliability and reliability values at the factual date of release. Table 6 summarizes all the results. We may deduce from the outcomes of this dataset that predicting the release is doable even when only 50% of the failure data is available. We also tried to see if we could make accurate forecasts sooner than ten weeks. For this, we carried out the

Table 3. Unknown parameter approximation of SRGMs for dataset 1

Model Name	Values
Delayed S-Shaped	$a = 126.8, b = 0.2426$
Generalized Goel	$a = 172.5, b = 0.04078, c = 1.235$
Goel Okumoto	$a = 528, b = 0.01773$
Gompertz	$a = 152.7, b = 0.08441, c = 0.8835$
Inflection S-Shaped	$a = 127.3, b = 0.2412, \beta = 3.524$
Logistic Growth	$a = 104, b = 0.2767, k = 6.371$
Modified Duane	$a = 420.7, b = 27.19, c = 1.206$
Musa Okumoto	$a = 497.8, b = 0.0188$
Pham Zhang IFD	$a = 127.4, b = 0.2414, d = 0.22e - 14$
Pham Nordman Zhang (PNZ Model)	$a = 9.163, \alpha = 0.709, b = 21.89, \beta = 0.001809$
Pham Zhang model (PZ Model)	$a = 0.001194, \alpha = 3570, b = 0.2412, \beta = 3.524, c = 127.2$
Yamada Exponential	$a = 300, \alpha = 2.307, \beta = 0.006354, r = 2.361$
Yamada Imperfect Debugging Model 1	$a = 528.1, \alpha = 1.595e - 08, b = 0.01773$
Yamada Imperfect Debugging Model II	$a = 8.997, \alpha = 0.7221, b = 49.81$
Yamada Rayleigh	$a = 142.2, \alpha = 1.198, \beta = 0.02026, r = 1.302$
Zeng Teng Pham	$a = 29.91, \alpha = 5.214, b = 0.2286, \beta = 0.6015, c = 0.841, p = 0.8238$

Table 4. Estimation of RSquare and RMSE using ten weeks failure data for dataset 1

Model Name	RSq	RMSE
Delayed S-Shaped	0.903	6.524
Generalized Goel	0.984	2.852
Goel Okumoto	0.972	3.529
Gompertz	0.994	1.707
Inflection S-Shaped	0.972	3.773
Logistic Growth	0.993	1.849
Modified Duane	0.984	2.849
Musa Okumoto	0.974	3.382
Pham Zhang IFD	0.903	6.524
Pham Nordman Zhang (PNZ Model)	0.993	2.001
Pham Zhang model (PZ Model)	0.991	2.515
Yamada Exponential	0.927	6.532
Yamada Imperfect Debugging Model 1	0.975	3.544
Yamada Imperfect Debugging Model II	0.993	1.853
Yamada Rayleigh	0.866	8.860
Zeng Teng Pham	0.995	2.204

computations at seven weeks (about 35% data) also but the results obtained at this stage were not compatible with the later date of prediction.

Example 2 *We used a separate dataset to assess the applicability of the suggested technique to diverse datasets [23]. This failure dataset was compiled out of three versions of a big medical record software with 188 components. Numerous files are included in each component. The package originally comprised of 173 software components. All the three updates have improved the product's functionality. A total of 15 components were added to the three releases. In each release, three to seven new components were included. As a result of the increased capability, some other components were adjusted in all three editions. Table 7 shows the results of applying the proposed approach to release 1 of this dataset. The same step-by-step process was used for this dataset as it was for the SRGM rating and release date prediction in Example 1. Table 8 provides the results, which show that logistic growth is ranked first using the results acquired (1). The same model of logistic growth is ranked 1 in all stages of testing for this dataset. At all three stages, the*

Table 5. SRGMs ranking using rank index for dataset 1 at $t = 10$ weeks

Model Name	Rank Index	Rank
Delayed S-Shaped	0.5848	14
Generalized Goel	0.7939	8
Goel Okumoto	0.7303	11
Gompertz	0.9999	1
Inflection S-Shaped	0.7147	12
Logistic Growth	0.9609	2
Modified Duane	0.7943	7
Musa Okumoto	0.7420	9
Pham Zhang IFD	0.5848	15
PNZ Model	0.9259	4
PZ Model	0.8376	6
Yamada Exponential	0.5968	13
Yamada Imperfect Debugging Model 1	0.7310	10
Yamada Imperfect Debugging Model II	0.9601	3
Yamada Rayleigh	0.5316	16
Zeng Teng Pham	0.8873	5

Table 6. SRGMs ranking using rank index for dataset 1 at $t = 10$ weeks

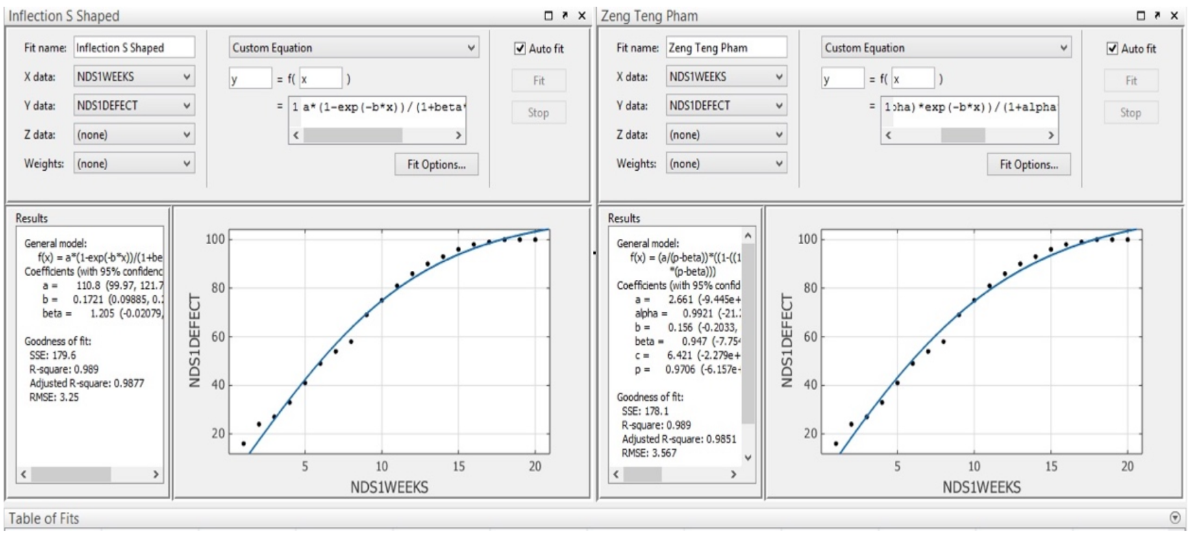
The dataset as well as the actual release date	Testing data used (in weeks)	Model chosen	Date of expected release (in weeks)	Expected level of reliability $(R(t))$	Conditional reliability $(R(s\|t))$ to be accomplished	
					For $s = 1$	For $s = 2$
	10	Gompertz	35	0.970	0.570	0.350
(20 weeks)	15	Logistic Growth	21	0.980	0.590	0.400
	20	ZT Pham	18	0.980	0.580	0.400

Table 7. Data of a significant medical record system's failures: release-1 (dataset 2)

Weeks	Aggregated failures	Weeks	Aggregated failures
1	28	10	125
2	29	11	139
3	29	12	152
4	29	13	164
5	29	14	164
6	37	15	165
7	63	16	168
8	92	17	170
9	116	18	176

estimated release date is the same. As a result, it can be stated that if the selected model is the same at each step of testing, more accurate predictions about the software release date can be made.

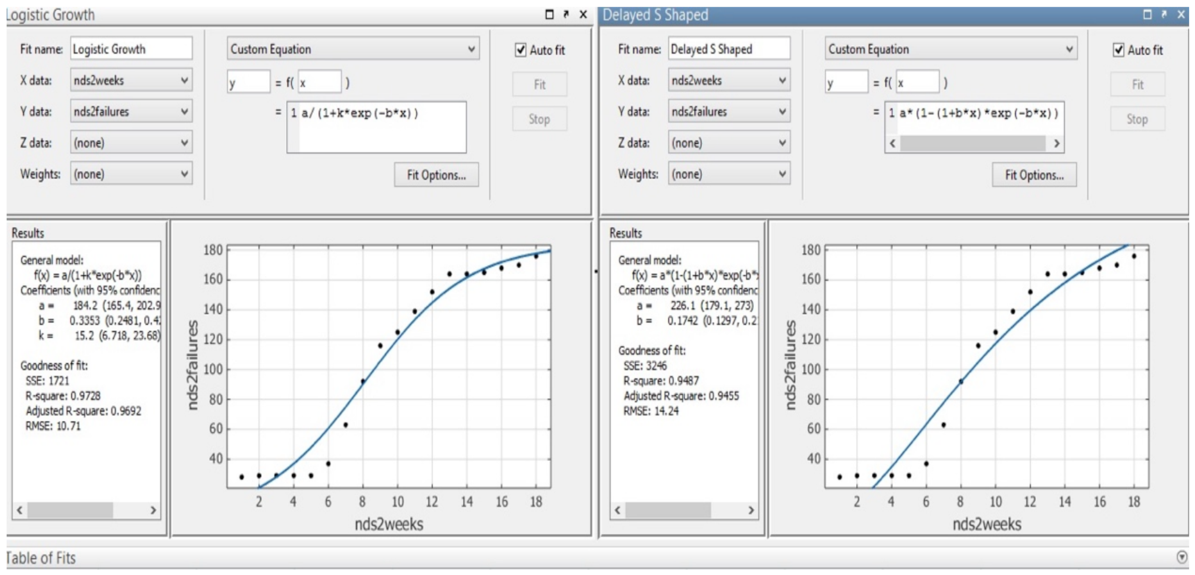
Similarly, eight more datasets from the available literature were used to assess the applicability of the suggested approach. Table 9 shows the anticipated value of release time for all the datasets, as well as to be expected value of conditional reliability and reliability. We have also evaluated the anticipated date of release by our presented technique with the factual date of release and



```
Inflection S Shaped
Fit name: Inflection S Shaped      Custom Equation          ☑ Auto fit
X data:   NDS1WEEKS                y = f( x )
Y data:   NDS1DEFECT               = 1 a*(1-exp(-b*x))/(1+beta
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = a*(1-exp(-b*x))/(1+be
Coefficients (with 95% confidenc
    a =      110.8  (99.97, 121.7
    b =     0.1721  (0.09885, 0.
    beta =   1.205  (-0.02079,

Goodness of fit:
  SSE: 179.6
  R-square: 0.989
  Adjusted R-square: 0.9877
  RMSE: 3.25
```

```
Zeng Teng Pham
Fit name: Zeng Teng Pham           Custom Equation          ☑ Auto fit
X data:   NDS1WEEKS                y = f( x )
Y data:   NDS1DEFECT               = 1 ɔha)*exp(-b*x))/(1+alpha
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = (a/(p-beta))*((1-((1
        *(p-beta)))
Coefficients (with 95% confid
    a =      2.661  (-9.445e+
    alpha =  0.9921  (-21.
    b =      0.156  (-0.2033,
    beta =   0.947  (-7.75
    c =      6.421  (-2.279e+
    p =     0.9706  (-6.157e-

Goodness of fit:
  SSE: 178.1
  R-square: 0.989
  Adjusted R-square: 0.9851
  RMSE: 3.567
```

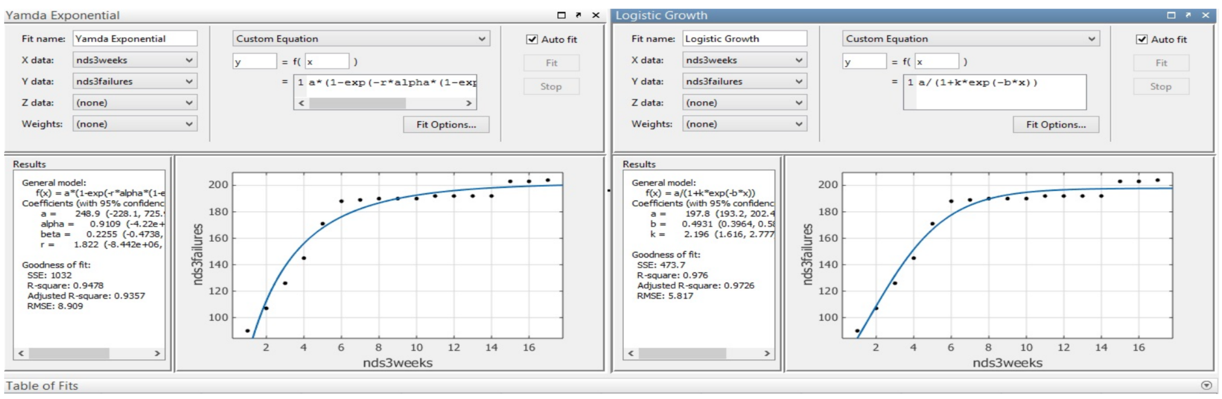
Figure 2. Comparison for dataset 1 using the models identified by presented methodology with the best model considered in existing studies



```
Logistic Growth
Fit name: Logistic Growth          Custom Equation          ☑ Auto fit
X data:   nds2weeks                y = f( x )
Y data:   nds2failures             = 1 a/(1+k*exp(-b*x))
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = a/(1+k*exp(-b*x))
Coefficients (with 95% confidenc
    a =      184.2  (165.4, 202.9
    b =     0.3353  (0.2481, 0.4
    k =       15.2  (6.718, 23.68)

Goodness of fit:
  SSE: 1721
  R-square: 0.9728
  Adjusted R-square: 0.9692
  RMSE: 10.71
```

```
Delayed S Shaped
Fit name: Delayed S Shaped         Custom Equation          ☑ Auto fit
X data:   nds2weeks                y = f( x )
Y data:   nds2failures             = 1 a*(1-(1+b*x)*exp(-b*x))
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = a*(1-(1+b*x)*exp(-b*
Coefficients (with 95% confidenc
    a =      226.1  (179.1, 273)
    b =     0.1742  (0.1297, 0.2

Goodness of fit:
  SSE: 3246
  R-square: 0.9487
  Adjusted R-square: 0.9455
  RMSE: 14.24
```

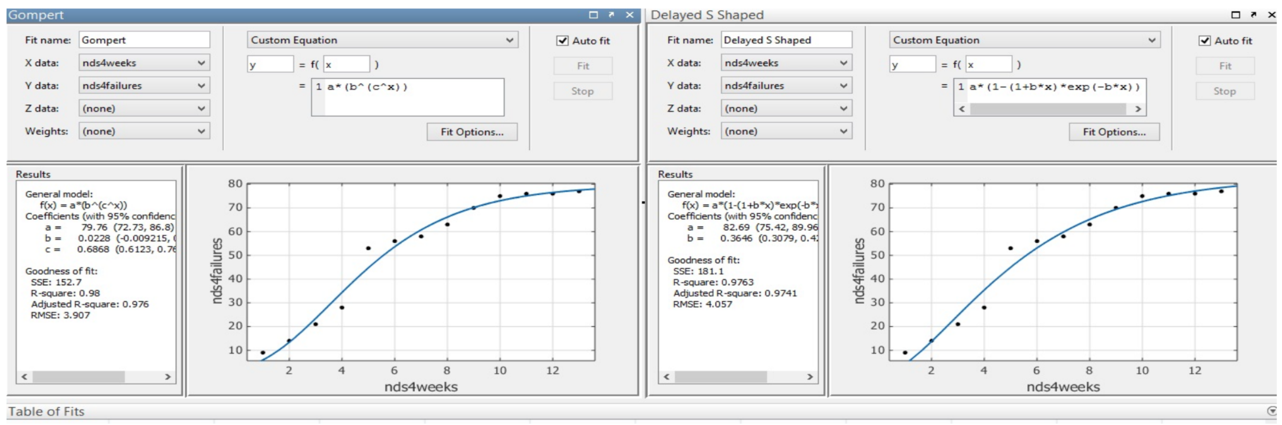
Figure 3. Comparison for dataset 2 using the models identified by presented methodology with the best model considered in existing studies



```
Yamda Exponential
Fit name: Yamda Exponential        Custom Equation          ☑ Auto fit
X data:   nds3weeks                y = f( x )
Y data:   nds3failures             = 1 a*(1-exp(-r*alpha*(1-exp
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = a*(1-exp(-r*alpha*(1-e
Coefficients (with 95% confidenc
    a =      248.9  (-228.1, 725.
    alpha =  0.9109  (-4.22e+
    beta =   0.2255  (-0.4738,
    r =      1.822  (-8.442e+06,

Goodness of fit:
  SSE: 1032
  R-square: 0.9478
  Adjusted R-square: 0.9357
  RMSE: 8.909
```

```
Logistic Growth
Fit name: Logistic Growth          Custom Equation          ☑ Auto fit
X data:   nds3weeks                y = f( x )
Y data:   nds3failures             = 1 a/(1+k*exp(-b*x))
Z data:   (none)
Weights:  (none)

Results
General model:
  f(x) = a/(1+k*exp(-b*x))
Coefficients (with 95% confidenc
    a =      197.8  (193.2, 202.4
    b =     0.4931  (0.3964, 0.5
    k =      2.196  (1.616, 2.777

Goodness of fit:
  SSE: 473.7
  R-square: 0.976
  Adjusted R-square: 0.9726
  RMSE: 5.817
```

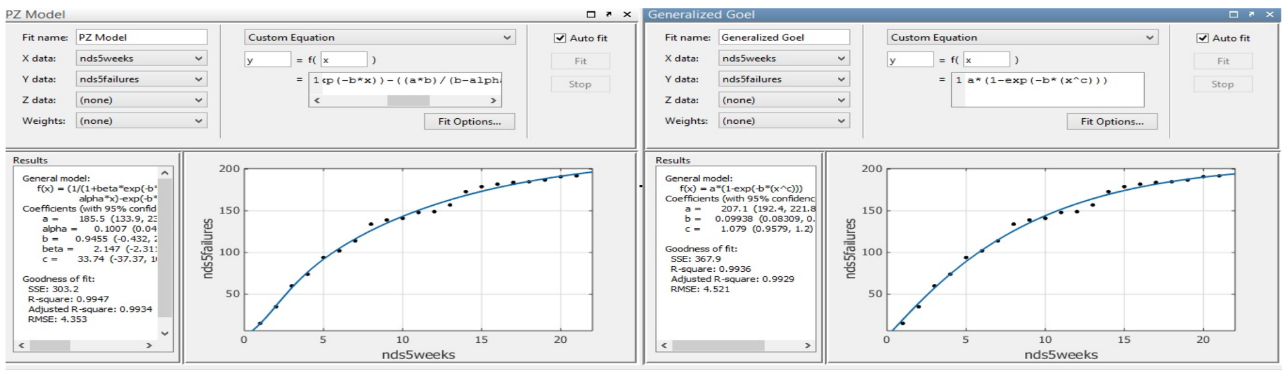
Figure 4. Comparison for dataset 3 using the models identified by presented methodology with the best model considered in existing studies



```
Gompert
Fit name: Gompert
X data: nds4weeks
Y data: nds4failures
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 a*(b^(c^x))
[✓] Auto fit

Results
General model:
  f(x) = a*(b^(c^x))
Coefficients (with 95% confidenc
    a =      79.76  (72.73, 86.8)
    b =     0.0228  (-0.009215, 
    c =     0.6868  (0.6123, 0.76

Goodness of fit:
  SSE: 152.7
  R-square: 0.98
  Adjusted R-square: 0.976
  RMSE: 3.907
```

```
Delayed S Shaped
Fit name: Delayed S Shaped
X data: nds4weeks
Y data: nds4failures
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 a*(1-(1+b*x)*exp(-b*x))
[✓] Auto fit

Results
General model:
  f(x) = a*(1-(1+b*x)*exp(-b*
Coefficients (with 95% confidenc
    a =      82.69  (75.42, 89.96
    b =     0.3646  (0.3079, 0.4

Goodness of fit:
  SSE: 181.1
  R-square: 0.9763
  Adjusted R-square: 0.9741
  RMSE: 4.057
```

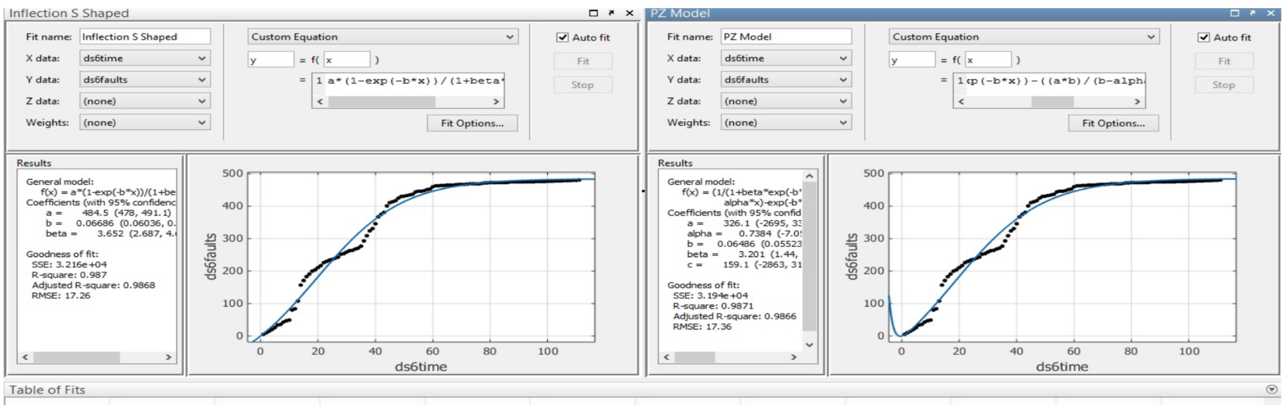
Figure 5. Comparison for dataset 4 using the models identified by presented methodology with the best model considered in existing studies



```
PZ Model
Fit name: PZ Model
X data: nds5weeks
Y data: nds5failures
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 xp(-b*x))-((a*b)/(b-alph

Results
General model:
  f(x) = (1/(1+beta*exp(-b*
             alpha*x)-exp(-b*
Coefficients (with 95% confid
    a =       185.5  (133.9, 23
    alpha =   0.1007  (0.04
    b =      0.9455  (-0.432, 2
    beta =    2.147  (-2.31
    c =       33.74  (-37.37, 1

Goodness of fit:
  SSE: 303.2
  R-square: 0.9947
  Adjusted R-square: 0.9934
  RMSE: 4.353
```

```
Generalized Goel
Fit name: Generalized Goel
X data: nds5weeks
Y data: nds5failures
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 a*(1-exp(-b*(x^c)))

Results
General model:
  f(x) = a*(1-exp(-b*(x^c)))
Coefficients (with 95% confidenc
    a =      207.1  (192.4, 221.8
    b =    0.09938  (0.08309, 0.
    c =      1.079  (0.9579, 1.2)

Goodness of fit:
  SSE: 367.9
  R-square: 0.9936
  Adjusted R-square: 0.9929
  RMSE: 4.521
```

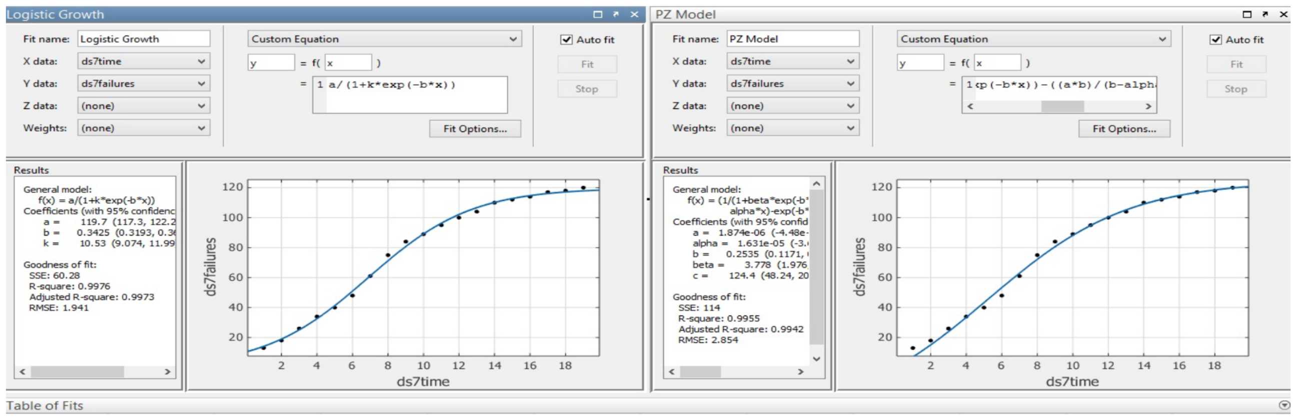
Figure 6. Comparison for dataset 5 using the models identified by presented methodology with the best model considered in existing studies



```
Inflection S Shaped
Fit name: Inflection S Shaped
X data: ds6time
Y data: ds6faults
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 a*(1-exp(-b*x))/(1+beta*

Results
General model:
  f(x) = a*(1-exp(-b*x))/(1+be
Coefficients (with 95% confidenc
    a =       484.5  (478, 491.1)
    b =     0.06686  (0.06036, 0.
    beta =    3.652  (2.687, 4.

Goodness of fit:
  SSE: 3.216e+04
  R-square: 0.987
  Adjusted R-square: 0.9868
  RMSE: 17.26
```

```
PZ Model
Fit name: PZ Model
X data: ds6time
Y data: ds6faults
Z data: (none)
Weights: (none)
Custom Equation
y = f( x )
= 1 xp(-b*x))-((a*b)/(b-alph

Results
General model:
  f(x) = (1/(1+beta*exp(-b*
             alpha*x)-exp(-b*
Coefficients (with 95% confid
    a =       326.1  (-2695, 33
    alpha =   0.7384  (-7.0
    b =     0.06486  (0.05523
    beta =    3.201  (1.44,
    c =       159.1  (-2863, 31

Goodness of fit:
  SSE: 3.194e+04
  R-square: 0.9871
  Adjusted R-square: 0.9866
  RMSE: 17.36
```

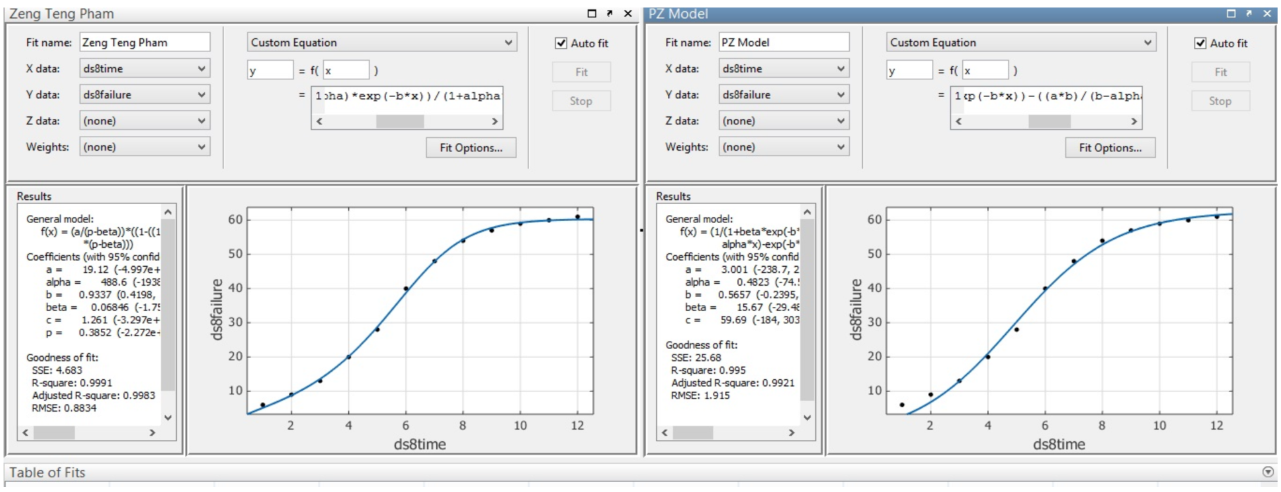
Figure 7. Comparison for dataset 6 using the models identified by presented methodology with the best model considered in existing studies



```
Logistic Growth
Fit name: Logistic Growth        Custom Equation        [✓] Auto fit
X data: ds7time                  y = f( x )
Y data: ds7failures              = 1 a/(1+k*exp(-b*x))
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = a/(1+k*exp(-b*x))
Coefficients (with 95% confidenc
    a =     119.7  (117.3, 122.2
    b =    0.3425  (0.3193, 0.3
    k =     10.53  (9.074, 11.99

Goodness of fit:
  SSE: 60.28
  R-square: 0.9976
  Adjusted R-square: 0.9973
  RMSE: 1.941
```

```
PZ Model
Fit name: PZ Model               Custom Equation        [✓] Auto fit
X data: ds7time                  y = f( x )
Y data: ds7failures              = 1 xp(-b*x))-((a*b)/(b-alph
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = (1/(1+beta*exp(-b*
          alpha*x)-exp(-b*
Coefficients (with 95% confid
    a =  1.874e-06  (-4.48e-
    alpha =  1.631e-05  (-3.
    b =    0.2535  (0.1171,
    beta =     3.778  (1.976
    c =     124.4  (48.24, 20

Goodness of fit:
  SSE: 114
  R-square: 0.9955
  Adjusted R-square: 0.9942
  RMSE: 2.854
```

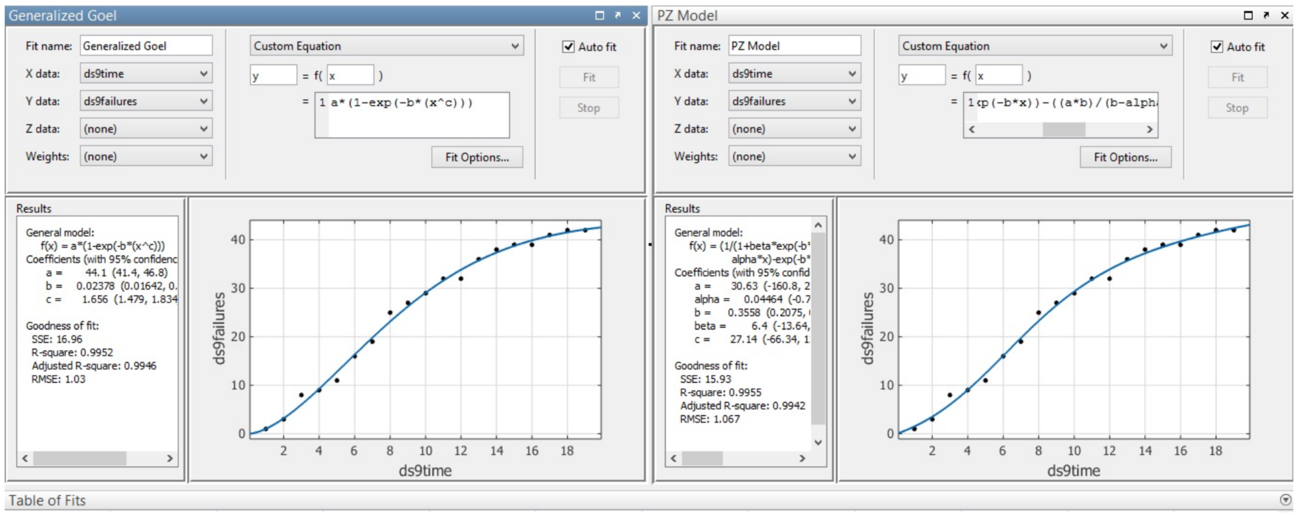
Figure 8. Comparison for dataset 7 using the models identified by presented methodology with the best model considered in existing studies



```
Zeng Teng Pham
Fit name: Zeng Teng Pham         Custom Equation        [✓] Auto fit
X data: ds8time                  y = f( x )
Y data: ds8failure               = 1 pha)*exp(-b*x))/(1+alpha
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = (a/(p-beta))*((1-((1
          *(p-beta)))
Coefficients (with 95% confid
    a =     19.12  (-4.997e+
    alpha =     488.6  (-193
    b =    0.9337  (0.4198,
    beta =   0.06846  (-1.7
    c =     1.261  (-3.297e+
    p =    0.3852  (-2.272e-

Goodness of fit:
  SSE: 4.683
  R-square: 0.9991
  Adjusted R-square: 0.9983
  RMSE: 0.8834
```

```
PZ Model
Fit name: PZ Model               Custom Equation        [✓] Auto fit
X data: ds8time                  y = f( x )
Y data: ds8failure               = 1 xp(-b*x))-((a*b)/(b-alph
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = (1/(1+beta*exp(-b*
          alpha*x)-exp(-b*
Coefficients (with 95% confid
    a =     3.001  (-238.7, 2
    alpha =    0.4823  (-74.
    b =    0.5657  (-0.2395,
    beta =     15.67  (-29.4
    c =     59.69  (-184, 303

Goodness of fit:
  SSE: 25.68
  R-square: 0.995
  Adjusted R-square: 0.9921
  RMSE: 1.915
```

Figure 9. Comparison for dataset 8 using the models identified by presented methodology with the best model considered in existing studies



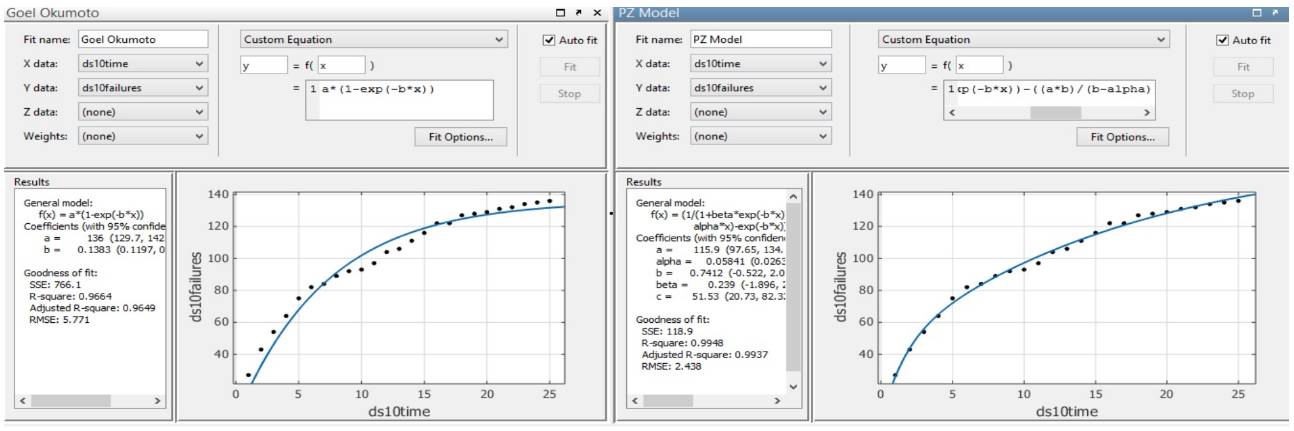
```
Generalized Goel
Fit name: Generalized Goel       Custom Equation        [✓] Auto fit
X data: ds9time                  y = f( x )
Y data: ds9failures              = 1 a*(1-exp(-b*(x^c)))
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = a*(1-exp(-b*(x^c)))
Coefficients (with 95% confidenc
    a =      44.1  (41.4, 46.8)
    b =   0.02378  (0.01642, 0.
    c =     1.656  (1.479, 1.834

Goodness of fit:
  SSE: 16.96
  R-square: 0.9952
  Adjusted R-square: 0.9946
  RMSE: 1.03
```

```
PZ Model
Fit name: PZ Model               Custom Equation        [✓] Auto fit
X data: ds9time                  y = f( x )
Y data: ds9failures              = 1 xp(-b*x))-((a*b)/(b-alph
Z data: (none)
Weights: (none)

Results
General model:
  f(x) = (1/(1+beta*exp(-b*
          alpha*x)-exp(-b*
Coefficients (with 95% confid
    a =     30.63  (-160.8, 2
    alpha =   0.04464  (-0.7
    b =    0.3558  (0.2075,
    beta =       6.4  (-13.64,
    c =     27.14  (-66.34, 1

Goodness of fit:
  SSE: 15.93
  R-square: 0.9955
  Adjusted R-square: 0.9942
  RMSE: 1.067
```

Figure 10. Comparison for dataset 9 using the models identified by presented methodology with the best model considered in existing studies

Table 8. Predicted dataset 2 release time, with expected values of conditional reliability and reliability

The dataset as well as the actual release date	Testing data used (in weeks)	Model chosen	Date of expected release (in weeks)	Expected level of reliability (R(t))	Conditional reliability (R(s ∣ t)) to be accom-plished	
					For $s = 1$	For $s = 2$
Dataset 2 (18 weeks)	10	Logistic Growth	24	0.990	0.550	0.350
	15	Logistic Growth	24	0.990	0.690	0.520
	18	Logistic Growth	22	0.990	0.610	0.430



```
General model:
  f(x) = a*(1-exp(-b*x))
Coefficients (with 95% confide
     a =        136  (129.7, 142
     b =     0.1383  (0.1197, 0

Goodness of fit:
  SSE: 766.1
  R-square: 0.9664
  Adjusted R-square: 0.9649
  RMSE: 5.771
```

```
General model:
  f(x) = (1/(1+beta*exp(-b*x))
               alpha*x)-exp(-b*x)
Coefficients (with 95% confiden
     a =       115.9  (97.65, 134.
     alpha =  0.05841  (0.0263
     b =      0.7412  (-0.522, 2.0
     beta =    0.239  (-1.896, 2
     c =       51.53  (20.73, 82.3

Goodness of fit:
  SSE: 118.9
  R-square: 0.9948
  Adjusted R-square: 0.9937
  RMSE: 2.438
```

Figure 11. Comparison for dataset 10 using the models identified by presented methodology with the best model considered in existing studies

the estimated date of release by the best models identifies in existing studies for the datasets utilised in current study to evaluate the performance of our proposed method. From Figure 2 to Figure 11 depict the comparison. The findings shown in Figure 11 reveal that with the exception of datasets 3 and 6, our suggested approach can forecast dependability early and timely virtually in all circumstances.

5 Conclusions, outcomes and future directions

Formulating the software testing process in a mathematically rigorous manner is important in software testing which acts as a major apparatus for software quality assurance, and this process is known to be complex since it comprises many factors such as test case execution, test case selection, defect debugging, tester's knowledge and experience, and so forth. This study has investigated how to choose the best software reliability model for predicting the most likely release date. Section 3 outlines the proposed strategy, allowing the user to anticipate the expected release date even after nearly half of the estimated test period has passed. We used the proposed approach at various phases of testing (e.g., once 50% of the testing is done, 75% of the testing is accomplished, at the actual release date). Our findings reveal that when the current technique is employed to the test dataset 7 and 50% of the test plan period has passed, the proposed method's anticipated release date is nearly identical to the actual release date. In the case of datasets 1, 2, 4, 5, 7, 8, 9, and 10, the anticipated date of release based on 50% of the data is inside 1 to 2 weeks of the factual release date. The expected date of release for dataset 3 is, however, significantly sooner

Table 9. Anticipated time of release for datasets considered in present work with expected to be attained value of conditional reliability and reliability

Factual date of release (in weeks and dataset)	Prediction time (in weeks)	Selected model	Release date (in weeks)	Expected level of reliability	To be attained conditional reliability For $s = 1$	For $s = 2$
Dataset 1	10	Gompertz	35	0.970	0.570	0.350
(20 weeks)	15	Logistic Growth	21	0.980	0.590	0.400
	20	ZT Pham	18	0.980	0.580	0.400
Dataset 2	10	Logistic Growth	24	0.990	0.550	0.350
(18 weeks)	15	Logistic Growth	24	0.990	0.690	0.520
	18	Logistic Growth	22	0.990	0.610	0.430
Dataset 3	10	ZT Pham	10	1.000	0.900	0.860
(17 weeks)	15	Logistic Growth	11	0.990	0.540	0.370
	17	Logistic Growth	12	0.990	0.640	0.480
Dataset 4	7	Logistic Growth	9	0.980	0.530	0.400
(13 weeks)	14	Gompertz	14	0.980	0.620	0.440
Dataset 5	10	Generalized Goel	19	0.990	0.560	0.360
(21 weeks)	15	Generalized Goel	27	0.980	0.580	0.370
	21	Generalized Goel	30	0.980	0.580	0.360
Dataset 6	55	Generalized Goel	232	0.970	0.590	0.350
(111 weeks)	84	Generalized Goel	90	0.980	0.600	0.370
	111	Inflection S-Shaped	85	0.980	0.620	0.390
Dataset 7	10	Logistic Growth	19	0.990	0.610	0.430
(19 weeks)	15	Logistic Growth	18	0.980	0.560	0.370
	19	Logistic Growth	19	0.980	0.590	0.400
Dataset 8	7	Logistic Growth	14	0.980	0.600	0.440
(12 weeks)	12	ZT Pham	10	0.980	0.740	0.650
Dataset 9	10	Inflection S-Shaped	15	0.980	0.750	0.620
(19 weeks)	15	Delayed S-Shaped	26	0.970	0.770	0.620
	19	Generalized Goel	20	0.960	0.690	0.520
Dataset 10	13	Generalized Goel	23	0.960	0.690	0.520
(25 weeks)	19	Gompertz	25	0.970	0.590	0.380
	25	PZ Model	28	0.990	0.590	0.350

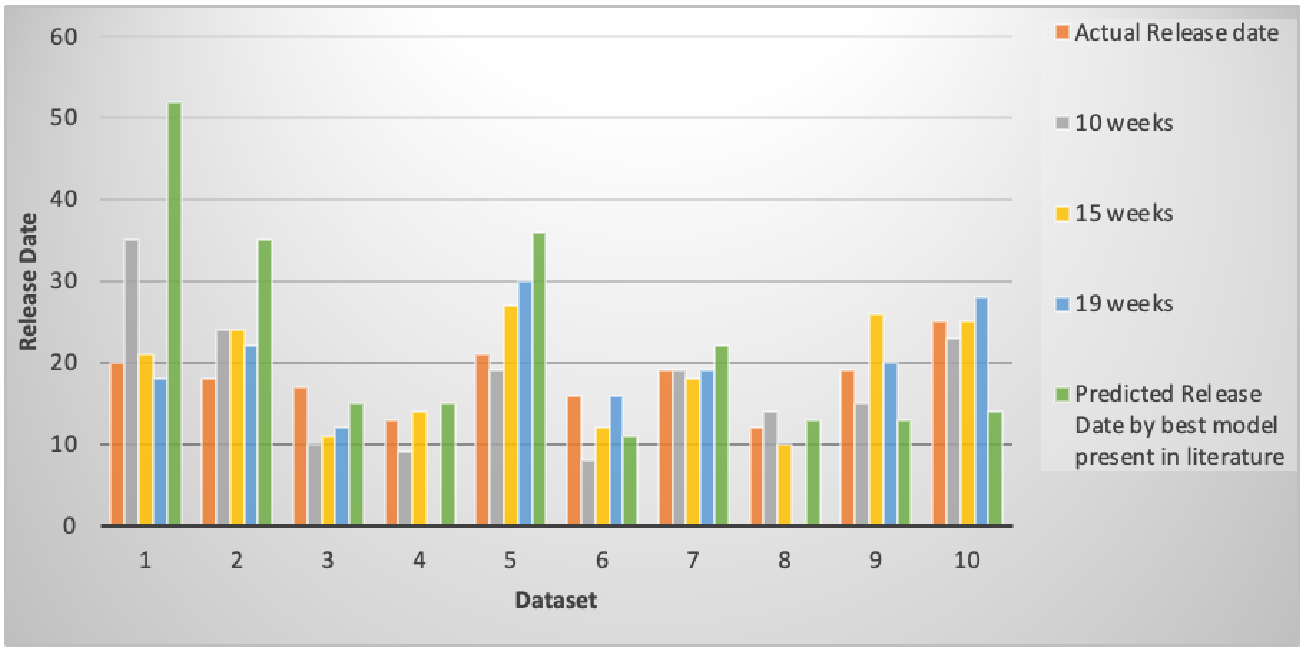


Figure 12. Comparison of the expected release time of the datasets in the proposed study with the actual release date, as well as the best models provided in the literature

than the actual release date. Interestingly, when 50% of the dataset is used to forecast the release date, the estimated date is 232 weeks, which is substantially far ahead than the actual date of 111 weeks. When a likelihood is generated using around 75% of the data, the estimated date of release is once again quite near to the factual release date, and it is dramatically lowered to 90 weeks. Even if all available data is used, the estimated release timeframe is 85 weeks. This indicates that testing may have been overdone, or that software adjustments were made in the interim. In all situations, we also tried with lower than 50% of test plan data and found that estimates were not reliable in common. Table 9 and Figure 11 show that when utilizing the proposed method, the anticipated release dates with models picked by us, even when using midway test data, are generally better than the similar outcomes achieved for these datasets when exploring the methods given in literature. Since NHPP SRGMs cannot handle time-dependent variables, the suggested approach is limited to software development circumstances that are time-independent. We intend to change the time-dependence of these models in the future, which will allow us to more exactly anticipate the number of faults found. We also intend to apply the proposed strategy to other software development methodologies, such as agile development. The comparison of the results with those available in literature shows that the proposed approach is able to select a model that fits the present data closely. Therefore, the selected model can be used for future predictions, and the selected models estimates by our proposed method are closer to the actual number of failures found by that time in each case.

Declarations

Use of AI tools

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Data availability statement

There are no external data associated with the manuscript.

Ethical approval (optional)

The authors state that this research complies with ethical standards. This research does not involve either human participants or animals.

Consent for publication

Not applicable

Conflicts of interest

The authors declare that they have no conflict of interest.

Funding

No funding was received for this research.

Author's contributions

All authors have contributed equally to the manuscript. All authors have read and agreed to the published version of the manuscript.

Acknowledgements

Not applicable

Abbreviations

NHPP: Non-Homogeneous Poisson Process; SRGMs: Software Reliability Growth Models; LSE: Least Squared Estimation; GOF: Goodness of Fit; RMSE: Root-Mean-Square Error; RSq: R-Square; SSE: Sum-of-Squared Errors.

References

[1] Musa, J.D. and Okumoto, K. A logarithmic Poisson execution time model for software reliability measurement. In Proceedings, *7th International Conference on Software Engineering*, pp. 230–238, Orlando, Florida, (1984, March).

[2] Goel, A.L. and Okumoto, K. An analysis of recurrent software errors in a real-time control system. In Proceedings, *Proceedings of the 1978 Annual Conference (ACM)*, pp. 496–501, Washington, DC, USA, (1978, December). [CrossRef]

[3] Goel, A.L. Software reliability models: assumptions, limitations, and applicability. *IEEE Transactions on Software Engineering*, SE-11(12), 1411–1423, (1985). [CrossRef]

[4] Bauer, E., Zhang, X. and Kimber, D.A. *Practical System Reliability*. John Wiley & Sons: New Jersey, (2009).

[5] Huang, C.Y. and Lyu, M.R. Optimal release time for software systems considering cost, testing-effort, and test efficiency. *IEEE Transactions on Reliability*, 54(4), 583–591, (2005). [CrossRef]

[6] Kapur, P.K., Gupta, D., Gupta, A. and Jha, P.C. Effect of introduction of fault and imperfect debugging on release time. *Ratio Mathematica*, 18, 62–90, (2008).

[7] Singh, Y. and Kumar, P. Determination of software release instant of three-tier client server software system. *International Journal of Software Engineering*, 1(3), 51–62, (2010).

[8] Quadri, S.M.K., Ahmad, N. and Farooq, S.U. Software reliability growth modeling with generalized exponential testing-effort and optimal software release policy. *Global Journal of Computer Science and Technology*, 11(2), 26-41, (2011).

[9] Kapur, P.K., Pham, H., Aggarwal, A.G. and Kaur, G. Two dimensional multi-release software reliability modeling and optimal release planning. *IEEE Transactions on Reliability*, 61(3), 758–768, (2012). [CrossRef]

[10] Panwar, P. and Lal, A.K. Predicting total number of failures in a software using NHPP software reliability growth models. In Proceedings, *Third International Conference on Soft Computing for Problem Solving (SocProS)*, pp. 715–727, New Delhi, India, (2014, December). [CrossRef]

[11] Choudhary, A. and Baghel, A.S. Software reliability prediction using cuckoo search optimization, empirical mode decomposition, and ARIMA model: CS-EEMD-ARIMA based SRGM. *International Journal of Open Source Software and Processes*, 7(4), 39–54, (2016). [CrossRef]

[12] Panwar, P. and Kaur, R. Effect of imperfect debugging on prediction of remaining faults in software. In Proceedings, *Fifth International Conference on Soft Computing for Problem Solving (SocProS)*, pp. 175–185, Uttarakhand, India, (2016, December). [CrossRef]

[13] Shi, Y., Li, M., Arndt, S. and Smidts, C. Metric-based software reliability prediction approach and its application. *Empirical Software Engineering*, 22, 1579–1633, (2017). [CrossRef]

[14] Pandey, S. and Kumar, K. Software fault prediction for imbalanced data: A survey on recent developments. *Procedia Computer Science*, 218, 1815-1824, (2023). [CrossRef]

[15] Luo, H., Xu, L., He, L., Jiang, L. and Long, T. A novel software reliability growth model based on generalized imperfect debugging NHPP framework. *IEEE Access*, 11, 71573-71593, (2023). [CrossRef]

[16] Samal, U., Kushwaha, S. and Kumar, A. A testing-effort based Srgm incorporating imperfect debugging and change point. *Reliability: Theory & Applications*, 18(1), 86-93, (2023). [CrossRef]

[17] Pradhan, V., Kumar, A. and Dhar, J. Modeling multi-release open source software reliability growth process with generalized modified weibull distribution. *Evolving Software Processes: Trends and Future Directions*, 123–133, (2022). [CrossRef]

[18] Quadri, S.M.K., Ahmad, N. and Peer, M.A. Software optimal release policy and reliability growth modeling. In Proceedings, *2nd National Conference on on Computing for Nation Development (INDIACom)*, pp. 423–431, New Delhi, India, (2008).

[19] Kapur, P.K., Pham, H., Anand, S. and Yadav, K. A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. *IEEE Transactions on Reliability*, 60(1), 331–340, (2011). [CrossRef]

[20] Huang, C.Y., Lyu, M.R. and Kuo, S.Y. A unified scheme of some nonhomogenous poisson process models for software reliability estimation. *IEEE Transactions on Software Engineering*, 29(3), 261–269, (2003). [CrossRef]

[21] Shatnawi, O. Discrete time NHPP models for software reliability growth phenomenon. *The International Arab Journal of Information Technology*, 6(2), 124-131, (2009).

[22] Wood, A. Software reliability growth models. *Tandem Technical Report*, 96(130056), 900, (1996).

[23] Stringfellow, C. and Andrews, A.A. An empirical method for selecting software reliability growth models. *Empirical Software Engineering*, 7, 319–343, (2002). [CrossRef]

[24] Andersson, C. A replicated empirical study of a selection method for software reliability growth models. *Empirical Software Engineering*, 12, 161–182, (2007). [CrossRef]

[25] Garg, R., Sharma, K., Kumar, R. and Garg, R.K. Performance analysis of software reliability models using matrix method. *International Journal of Computer and Information Engineering*,

5(2), 113-120, (2010).

[26] Sharma, K., Garg, R., Nagpal, C.K. and Garg, R.K. Selection of optimal software reliability growth models using a distance based approach. *IEEE Transactions on Reliability*, 59(2), 266–276, (2010). [CrossRef]

[27] Ullah, N., Morisio, M. and Vetrò, A. Selecting the best reliability model to predict residual defects in open source software. *Computer,* 48(6), 50–58, (2014). [CrossRef]

[28] Kumar, V., Singh, V.B., Garg, A. and Kumar, G. Selection of optimal software reliability growth models: a fuzzy DEA ranking approach. In *Quality, IT and Business Operations,* (pp. 347–357). Singapore: Springer, (2018). [Crossref]

[29] Li, Q. and Pham, H. A generalized software reliability growth model with consideration of the uncertainty of operating environments. *IEEE Access*, 7, 84253–84267, (2019). [CrossRef]

[30] Kumar, V., Saxena, P. and Garg, H. Selection of optimal software reliability growth models using an integrated entropy–Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) approach. *Mathematical Methods in the Applied Sciences,* 1-21, (2021). [CrossRef]

[31] Asraful Haque, M. and Ahmad, N. Modified Goel-Okumoto software reliability model considering uncertainty parameter. In Proceedings, *Mathematical Modeling, Computational Intelligence Techniques and Renewable Energy (MMCITRE)*, pp. 369–379. Gandhinagar, India, (2022). [CrossRef]

[32] Bibyan, R. and Anand, S. Ranking of multi-release software reliability growth model using weighted distance-based approach. *Optimization Models in Software Reliability,* (pp. 355–373). Singapore, Springer, (2022). [CrossRef]

[33] Shanmugam, L. and Florence, L. A comparison of parameter best estimation method for software reliability models. *International Journal of Software Engineering & Applications,* 3(5), 91-102, (2012). [CrossRef]

[34] Song, K.Y., Chang, I.H. and Lee, S.W. Predictions of MLE and LSE in NHPP Software Reliability Model. *Journal of the Chosun Natural Science,* 6(2), 111–117, (2013). [CrossRef]

[35] The MathWorks, Inc., MATLAB 2022 version 9.12.0 (R2022a), Software, Natick, MA, (2022).

[36] Yamada, S., Tokuno, K. and Osaki, S. Imperfect debugging models with fault introduction rate for software reliability assessment. *International Journal of Systems Science,* 23(12), 2241–2252, (1992). [CrossRef]

[37] Pham, H. *Software Reliability*. Springer Science & Business Media, Singapore, (2000).

[38] Pham, H. *System Software Reliability*. Springer, London, (2007).

[39] Pham, H. and Zhang, X. An NHPP software reliability model and its comparison. *International Journal of Reliability, Quality and Safety Engineering,* 4(03), 269–282, (1997). [CrossRef]

[40] Zhang, X., Teng, X. and Pham, H. Considering fault removal efficiency in software reliability assessment. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* 33(1), 114–120, (2003). [CrossRef]

[41] Pham, H., Nordmann, L. and Zhang, Z. A general imperfect-software-debugging model with S-shaped fault-detection rate. *IEEE Transactions on Reliability,* 48(2), 169–175, (1999). [CrossRef]

[42] Wood, A. Predicting software reliability. *Computer,* 29(11), 69–77, (1996). [CrossRef]

How to cite this article: Panwar, P., Kumar, S., Singla, S. & Karaca, Y. (2024). A multi-step mathematical model-based predictive strategy for software release timing during testing stage. *Mathematical Modelling and Numerical Simulation with Applications*, 4(3), 351-369. https://doi.org/10.53391/mmnsa.1406374