INVESTIGATION ON LOGISTIC GROWTH MODELS IN SOFTWARE RELIABILITY

Maria Vasileva, Georgi Penchev

Abstract. The talk is devoted to Software Reliability Growth Models based on the Verhulst model also known as the logistic growth model. It is well known that it is a mathematical model that describes population growth, where growth is limited by resources or carrying capacity. This provides a natural way for modeling the fault detection and removal process during software testing. We examine the applicability of logistic-type SRGMs using empirical data from real open-source software projects. The analysis focuses on the models' ability to capture the nonlinear dynamics of software reliability growth and to provide accurate approximations of the fault discovery process. The results demonstrate the potential of logistic SRGMs as an effective quantitative tool for assessing the reliability of open-source software systems.

Key Words: *software reliability growth model, logistic growth model, software reliability engineering, open-source projects*

Introduction

Software reliability is an essential part of software quality according to ISO/IEC 25010:2023. It shows how well a system can operate without failure during real use. Software systems increase in size and complexity. This growth makes the need for clear and reliable methods to measure their behavior more important. The central focus of this work is on software reliability growth models (SRGM) and their use as a primary tool for the quantitative analysis of software reliability. This study examines several logistic-type models and explores how they can support a more realistic and practical understanding of reliability in modern software development.

The logistic growth model, commonly referred to the Verhulst logistic growth model, is a fundamental mathematical model for describing growth processes under resource constraints. It is originally introduced by Pierre François Verhulst and is defined by

$$\frac{dN(t)}{dt} = rN(t)\left(1 - \frac{N(t)}{K}\right), \tag{1}$$

where r is intrinsic rate of natural increase (growth rate) and $N(t)$ is a population size. This model considers environmental limitations, introducing carrying capacity K to prevent infinite exponential growth. It is designed to predict population growth, but finds wide applications in various fields, including epidemiology, ecology, demography, biology, and economics. The model exhibits characteristic sigmoidal (S-shaped) behavior, where an initial phase of slow growth is followed by a period of rapid increase and eventually a saturation stage as the system approaches equilibrium. Its widespread use is based on its ability to capture nonlinear behavior and saturation effects.

In the context of SRGM, the behavior of the logistic model is especially useful because it reflects realistic testing processes. In such processes, the effectiveness of defect detection does not simply decrease over time. Instead, it first accelerates and then stabilizes as testing progresses. The Verhulst model often fits empirical software failure data better than classical exponential NHPP models. This is particularly true when the observed failure datasets show clear shaped behavior. Thus, the logistic model provides a more accurate and practical way to describe how reliability grows during testing.

In 2018, Pham [1] proposed estimating the mean value function $m(t)$ using a logistic growth model defined by

$$\frac{dm(t)}{dt} = b(t)m(t)\left(1 - \frac{m(t)}{a(t)}\right), \quad m(0) \neq 0, \tag{2}$$

where $a(t)$ is fault content function and $b(t)$ is software fault detection rate per unit of time. Using this model with functions $a(t) = a$ and time-dependent fault detection rate $b(t) = \frac{b}{1+\beta e^{-bt}}$ Pham introduced the following SRGM model

$$m_{Ph}(t) = \frac{a}{1 + h\left(\frac{1+\beta}{\beta + e^{bt}}\right)}. \tag{3}$$

Haque and Ahmad [2] proposed the software reliability model that takes into account the improvement of the testing team's skills over time. Its mean value function is expressed by

$$m_{HA}(t) = \frac{a}{1 + \beta e^{\left(-\frac{bt^{\alpha}}{\alpha}\right)}}. \tag{3}$$

They used (2) with $a(t) = a$ and $b(t) = bt^{\alpha-1}$. Function $b(t)$ is the same as in the Generalized Goel Model and represents an initial increase followed by a decrease in the failure rate.

Song, Kim and Chang [3] consider a software reliability model that considers the number of finite faults and dependent faults with the mean value function defined by

$$m_{SKC}(t) = \frac{a}{1 + \left(\frac{a}{h} - 1\right)(1 + bt)e^{-bt}}. \quad (4)$$

It is obtained from (2) with $a(t) = a$ and $b(t) = \frac{b^2 t}{1+bt}$.

Haque and Ahmad [4] assume that the function $\$b(t) = b\log(\alpha)\, t^{\beta-1}\alpha^{t^{\beta}}$ follows Vtub-shaped based on log–log distribution. Using this and (2) with $a(t) = a$ they proposed the following SRGM model

$$m_{HA2}(t) = \frac{a}{1 + e^{\left(\gamma - \alpha^{t^{\beta}}\right)}}.$$

The main aim of this study is to examine how well Software Reliability Growth Models describe the behavior of a contemporary open-source software project. Our focus is on logistic-type SRGMs. Earlier studies have tested these models mostly on older and smaller datasets. This leaves open the question of how they perform on current OSS systems. We present a detailed analysis to one OOS project to support this aim. We also use heatmap visualizations to show how reliability changes over time. These visual tools help us understand defect behavior and offer practical guidance for applying SRGM-based methods in the reliability assessment of open-source software. Further discussions, as well as open problems in debugging and test theory, can be found in related papers and monographs [5–10].

Numerical experiments

We present a numerical example that involves fitting software reliability growth models to an open-source dataset containing detected software faults. We perform a comparative analysis of several classical SRGMs, namely the Goel–Okumoto model m_{GO}, Delayed S-shaped m_{DS} and Inflection S-shaped m_{IS}. In addition, we include four logistic-type SRGMs models: Pham (2018) m_{Ph}, Haque–Ahmad (2021) m_{HA1}, Song-Kim-Chang (2023) m_{SKC} and Haque–Ahmad (2025) m_{HA2} models.

Let us recall the mean value functions of classical SRGMs:

- Goel-Okumoto [12]: $m_{GO}(t) = a(1 - e^{-bt})$;
- Delayed S-shaped [13]: $m_{DS}(t) = a(1 - (1 + bt)e^{-bt})$;
- Inflection S-shaped [14]: $m_{IS}(t) = \frac{a\left(1-(1+bt)e^{-bt}\right)}{1+\beta e^{-bt}}$.

SwiftUICharts is a composable, SwiftUI-native chart library for iOS 13+. It is an open source project designed to help create and present charts with minimal code. It supports line, bar, and pie charts and is designed to be interactive, animated, and easy to customize. We look at data from GitHub for version 2.0.0

(2.03.2026). It is a major release focused on immutable configuration, environment-driven composition, and modifier-based APIs. The data is normalized into 80-time intervals based on open issues over time (see https://github.com/AppPear/ChartView).

Table 1. Parameter estimation for SwiftUICharts dataset

Model	a	b	α	β	γ	h
Goel-Okumoto	260.239	0.00513	–	–	–	–
Delayed S-shaped	101.518	0.04051	–	–	–	–
Inflection S-shaped	115.918	0.03398	–	3.19071	–	–
Pham (2018)	83.861	−0.06925	–	−0.1227	–	13.4258
Haque-Ahmad (2021)	123.360	0.39546	0.40844	126.653	–	–
Song-Kim-Chang (2023)	81.043	0.10091	–	–	–	9.1496
Haque-Ahmad (2025)	116.254	–	2.60156	0.17145	6.42264	–

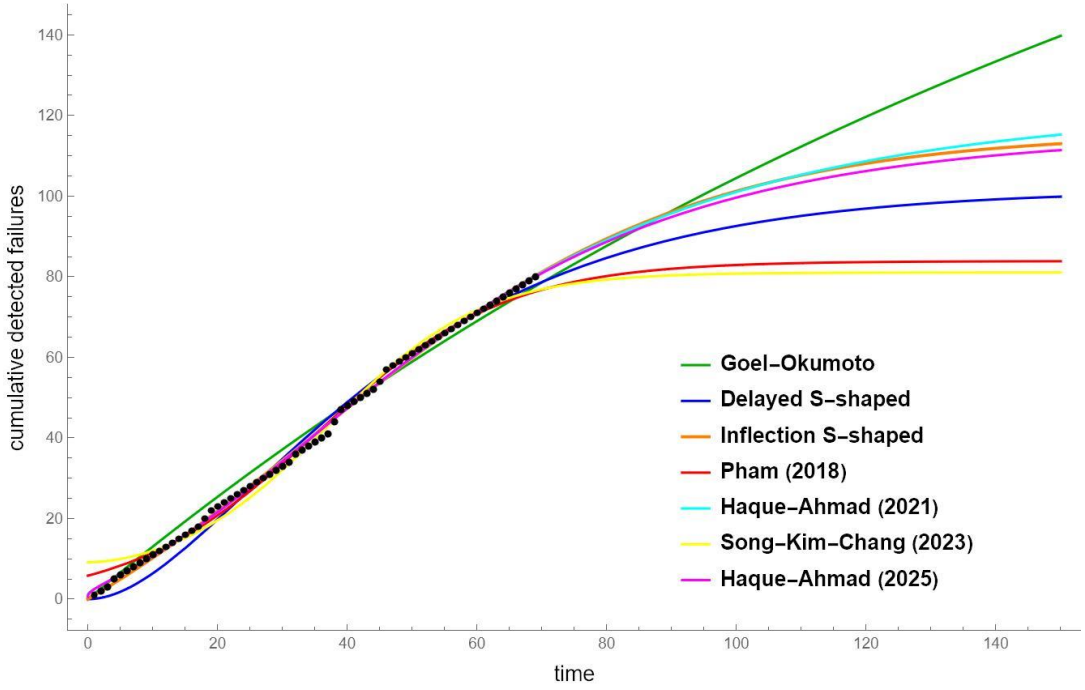

Figure 1. Fitting results for SwiftUICharts 2.0.0.

We use Nonlinear regression model to evaluate parameters for considered SRGM models. We should note that the nonlinear regression model obtained from the SRGM model is correct if the residuals or differences between the observed data and the model results are a normal distribution. This fact has been verified for the considered SRGM models by standard statistical tests at 5 percent level

such as Shapiro-Wilk. Table 1 presents the estimated parameters for the models under considerations. Figure 1 provides a graphical representation of the cumulative number of failures (indicated by black dots) alongside the estimated mean value functions of the considered software reliability models.

We use several common statistical measures to evaluate how well the models fit the data. We look at the coefficient of determination R^2, the adjusted coefficient of determination ($Adj\ R^2$), and three information criteria: the AIC, $AICc$, and BIC. These measures help us judge both how accurate each model is and how complex it is. When we compare different models, the best one is the model that gives the highest values of R^2 and $Adj\ R^2$, and at the same time produces the lowest values of AIC, $AICc$, and BIC. Table 2 provides the numerical results for all models based on these standard goodness-of-fit criteria.

Table 2. Comparison of criteria values for SwiftUICharts dataset

Model	AIC	$AICc$	BIC	R^2	$Adj\ R^2$
Goel-Okumoto	328.404	328.773	335.107	0.997215	0.997132
Delayed S-shaped	320.918	321.287	327.62	0.997502	0.997427
Inflection S-shaped	194.649	195.274	203.585	0.999611	0.999593
Pham (2018)	276.479	277.431	287.649	0.998763	0.998687
Haque-Ahmad (2021)	190.174	191.126	201.344	0.999646	0.999624
Song-Kim-Chang (2023)	324.356	324.981	333.293	0.99745	0.997334
Haque-Ahmad (2025)	**187.664**	**188.616**	**198.834**	**0.999659**	**0.999638**

Based on the results of standard goodness-of-fit tests, we can conclude that the Haque-Ahmad 2025 software reliability model $m_{HA2}$is the most suitable for forecasting future fault occurrences. For this model we obtain $a = 116.254$, which represents the expected maximum number of defects. According to the model, the value of 116 will be reached around the 236th time period.

Reliability Assessment

In NHPP-based SRGMs, the reliability function $R(t)$ [15] is defined as the probability that no failures occur in the time interval $(0, t)$. The reliability over a future operation period $[t, t + x]$ is expressed through the conditional probability $R(x \mid t)$, defined as

$$R(x \mid t) = e^{-[m(t+x)-m(t)]}, \quad (7)$$

where $t \geq 0$ and $x > 0$. This function represents the probability that no software failures will occur during the additional operating time x, given that the software as already executed without failure up to time t.

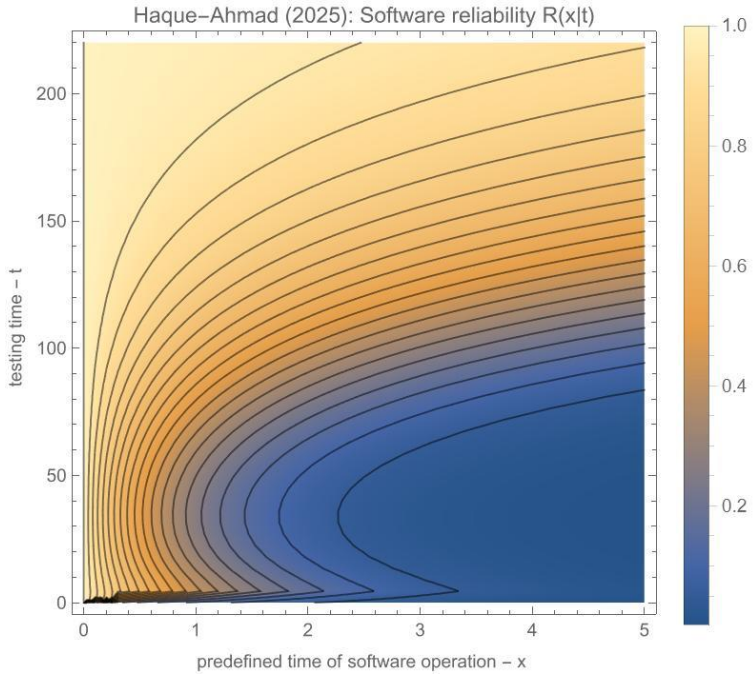

Figure 2. Reliability function for SwiftUICharts dataset

We use a heatmap visualization to track how software reliability changes during the testing process. The color scale shows the values of $R(x|t)$. Darker colors indicate lower reliability, while lighter colors show higher reliability, closer to one. The plot presents the conditional reliability as a function of two variables: the accumulated testing time t and the future operating period x. The heatmap makes the main trends easy to see. Reliability increases when t becomes larger, and it decreases when x grows. The colors become lighter in the upper part of the plot, where the testing time is long. This means that the expected number of remaining defects is small. The colors become darker in the right part of the plot, where x is large. This shows that the chance of failure increases when the required failure-free period becomes longer. The contour lines mark equal levels of reliability and highlight the nonlinear relation between the two variables. For small values of x, reliability rises quickly as t increases. For large values of x, even long testing cannot fully offset the growing risk of failure. The heatmap gives a clear visual summary of the main idea behind conditional reliability in SRGM models: longer testing improves reliability but demanding long failure-free operation always reduces it.

Acknowledgments

The work was supported by the project No. FP25-FMI-010, “Innovative interdisciplinary research in Informatics, Mathematics, and Pedagogy of

Education", of the Scientific Fund of the Paisii Hilendarski University of Plovdiv, Bulgaria.

References

[1] H. Pham, A Logistic Fault-Dependent Detection Software Reliability Model, *J. Univers. Comput. Sci*., 2018, 24, 1717–1730, DOI: 10.3217/jucs-024-12-1717, ISSN: 0948-6968

[2] M. Haque, N. Ahmad, An effective software reliability growth model, *Saf. Reliab*., 2021, 40, 209–220, DOI: 10.1080/09617353.2021.1921547, ISSN: 2469-4126

[3] K. Song, Y. Kim, I. Chang, Software reliability model for open-source software that considers the number of finite faults and dependent faults, *Math. Biosci. Eng.*, 2023, 20, 11785–11804, DOI: 10.3934/mbe.2023524, ISSN: 1551-0018

[4] M. Haque, N. Ahmad, A logistic software reliability model with Loglog fault detection rate, *Iran J. Comput. Sci.*, 2025, 8, 1–7, 10.1007/s42044-024-00192-x, ISSN: 2520-8446

[5] N. Pavlov, A. Iliev, A. Rahnev, N. Kyurkchiev, *Some Software Reliability Models: Approximation and Modeling Aspects*, LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2018, ISBN: 978-613-9-82805-0

[6] N. Pavlov, A. Iliev, A. Rahnev, N. Kyurkchiev, *Nontrivial Models in Debugging Theory (Part 2)*, LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2018, ISBN: 978-613-9-87794-2

[7] V. Kyurkchiev, A. Iliev, A. Rahnev, N. Kyurkchiev, *Some new logistic differential models: properties and applications*, LAP LAMBERT Academic Publishing: Saarbrucken, Germany, 2019, ISBN: 978-620-0-43442-5

[8] O. Rahneva, A. Golev, G. Spasov, *Investigations on some new models in debugging and growth theory (Part 3)*, LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2020, ISBN: 978-620-66655-8

[9] N. Kyurkchiev, A. Iliev, A. Golev, A. Rahnev, *Some Non-Standard Models in "Debugging and Test Theory" (Part 4)*, Plovdiv University Press: Plovdiv, Bulgaria, 2020, ISBN: 978-619-2-02584-7

[10] M. Vasileva, G. Penchev, Modeling and Assessing Software Reliability in Open-Source Projects, *Computation*, 2025, 13, 214, DOI: 10.3390/ computation13090214, ISSN: 2079-3197

[11] M. Vasileva, G. Penchev, Reliability analysis of open-source projects using software reliability growth models with fault removal efficiency, *Int. J. Differ. Equ. Appl.*, 2025, 24 (1), 65–80, DOI: 10.12732/ijdea.v24i1.6, ISSN: 1314-6084

[12] A. Goel, K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliab.*, 1979, 28, 206–211, DOI: 10.1109/TR.1979.5220566, ISSN: 1558-1721

[13] S. Yamada, M. Ohba, S. Osaki, S-shaped reliability growth modeling for software error detection, *IEEE Trans. Reliab.* 1983, 32, 475–484, DOI: 10.1109/TR.1983.5221735, ISSN: 1558-1721

[14] M. Ohba, Inflexion S-shaped software reliability growth models, In *Stochastic Models in Reliability Theory*; S. Osaki, Y. Hatoyama, Eds.; Springer: Berlin, Germany, 1984, pp. 144–162, DOI: 10.1007/978-3-642-45587-210, ISBN: 978-3-642-45587-2

[15] H. Pham, *System Software Reliability*, Springer Series in Reliability Engineering, Springer, London, UK, 2006, ISBN: 978-185-2-33950-0

Maria Vasileva[1,*], Georgi Penchev[1]

[1] Paisii Hilendarski University of Plovdiv,

Faculty of Mathematics and Informatics,

24, Tzar Asen Str., 4000 Plovdiv, Bulgaria

Corresponding author: mariavasileva@uni-plovdiv.bg