Comput Syst Sci & Eng (2019) 1: 33–46

International Journal of
Computer Systems
Science & Engineering

Modeling and analysis of leftover issues and release time planning in multi-release open source software using entropy based measure

Meera Sharma[1], **H. Pham**[2] **and V.B. Singh**[3]

[1] *Department of Computer Science, University of Delhi, India. E-mail: meerakaushik@gmail.com*
[2] *Department of Industrial and Systems Engineering, Rutgers, the State University of New Jersey hoang84pham@gmail.com*
[3] *Delhi College of Arts and Commerce, University of Delhi, India. E-mail: vbsingh@dcac.du.ac.in*

In Open Source Software (OSS), users report different issues on issues tracking systems. Due to time constraint, it is not possible for developers to resolve all the issues in the current release. The leftover issues which are not addressed in the current release are added in the next release issue content. Fixing of issues result in code changes that can be quantified with a measure known as complexity of code changes or entropy. We have developed a 2-dimensional entropy based mathematical model to determine the leftover issues of different releases of five Apache open source products. A model for release time prediction using entropy is also proposed. This model maximizes the satisfaction level of user's in terms of number of issues addressed.

Keywords: Open Source Software; Software Repositories; Entropy; Cobb-Douglas; New feature; Feature improvement; Release time problem

Acronyms

OSS	Open Source Software
NHPP	Non-Homogeneous Poisson Process
NFs	New Features
IMPs	Feature Improvements
SPSS	Statistical Package for Social Sciences
MSE	Mean Squared Error
RMSPE	Root Mean Squared Prediction Error
VAR	Variation
GA	Genetic Algorithm

1. INTRODUCTION

Release engineering focuses on qualitative and quantitative approach for developing a roadmap, which can shape the product that's ready for release. A systematic review for understanding the different approaches for release planning has been conducted by reviewing the research papers published in various academic journals and conferences [50]. In the OSS development model, users are categorized into various categories depending upon their skills and involvement in software development. In many cases, users also act as developers [48]. The contributors located at different geographical locations request for addressing NFs, IMPs and fixing the bugs through a centralized software platform [1 and 51]. In order to address the different issues reported by users and to fix the bugs the source code of the software need to be changed and these code changes are updated in source code

repositories. These source code changes have been quantified using Shannon entropy [2, 3 and 12].

In every organization, the software development team always desires to produce a software release of high quality with a low fault content. To meet the enormous requirements, they also want to produce a software with frequent releases [14]. In literature, the release time problem for proprietary software has been discussed widely by considering one factor, the bugs which have been fixed in different releases. A release planning study has been conducted by interviewing the persons involved in release planning of the OSS and found that release planning based on the implementation of features requested by users faces various challenges and prefer for release planning based on a given interval of time [25].

In feature implementation based release planning, the practitioners observed that all the features are not taking uniform time in their implementation but some features take enormous time. In OSS development and even in closed source development the developers/ release managers mainly prefer time based release as the feature based release planning has not been addressed adequately.

The OSS development has been carried by the volunteer contributors (active users) and has not taken into account the cost criteria [42, 43].

To release the next version of the software one need to consider the number of addressed issues requested by the users. In OSS, active user's satisfaction can be measured in terms of number of the issues fixed. Different issues are characterized by different severity levels where severity of an issue shows the extent of its impact [44 and 52]. By considering the severity levels of different issues, we have calculated active user's satisfaction as number of issues fixed multiplied by severity levels. In order to increase the active user's satisfaction level, we have maximized this quantity.

In this paper, we propose models for a multi-release software product. We determine the release time by considering the fixing of NFs, IMPs, bugs and code changes. The models have been validated by using datasets of various products of Apache software project. We have taken into consideration the two existing software reliability growth models [37 and 38] for comparison. We observed that our models give highest cases of maximum performance.

We have used "ga-Genetic Algorithm" in MATLAB software [28]. Genetic Algorithm has been used in various reliability applications [29-31and 45-47].

Remaining part of the paper has seven sections. In section 2, we have summarized the related work. Data collection and model construction have been discussed in section 3. The experimental setup has been given in section 4. Section 5 documents the results and discussion. The optimal release planning has been discussed in section 6. Threats to validity have been given in section 7 and the conclusion of the paper has been given in section 8.

2. RELATED WORK

The monolithical software development offers a way for multiple releases as a result of functionality enhancements [21]. A better understanding of release process assists developers to design frequent and fault free software releases [14]. "A timely introduction of a clunker and a delayed entry of a masterpiece can destroy a product's chance of success" [27].

The uncertainty analysis has been carried out in different disciplines in order to measure the uncertainty arises due to internal or external factors. In software development too, the uncertainty analysis has been used by applying Shannon information theory in order to quantify frequent source code changes [2]. A study based on the quantified code changes has been conducted [3]. In this study, the authors also estimated the rate at which source code changes are taking place.

The quantified approach in understanding the different activities relating to software development and maintenance activities has been proven to be very useful. In literature, the search based software engineering approach has been used in discussing the next release problem [8 and 9]. The release time of a software can be affected by many factors like complexity, bugs, software architecture, software domains and tools [22, 23 and 24].

During the past three decades, various mathematical models have been proposed to quantify the quality of the software. Various release time planning approaches have been discussed in literature for commercial/proprietary software. These approaches have considered the fixing of bugs of current and just previous release [7]. In case of OSS, the release time strategy is different from the proprietary/closed source software in the sense that it considers not only the number of bugs fixed but the number of NFs and IMPs implemented. "When using a feature-based strategy, an open source project might make a release even if not all planned features have been implemented" [25].

For a single release problem in closed source software the authors proposed to determine the optimal release time by minimizing the development cost that include testing cost, debugging cost and fixing cost or maximizing the reliability level subject to predefined budget [4]. When the addons are included and the faults are removed a proportion of faults are generated, a release time problem by considering this phenomenon has been developed in [36]. An attempt has been made to develop multiple release time problem [7]. The approach determined the optimal time and the optimal amount of resources. But, again the next release has been decided on the basis of number of bugs removed in testing phase and operational phase. The Dempster-Shafer theory and differential evolution based multi software reliability allocation for a multimedia system has been proposed in [32]. An optimal software release time planning by considering delay incurred cost has been proposed in [33]. Multi-criteria based model has been proposed for the software reliability prediction [34]. In a study [35], the failure processes of testing have been investigated by considering the delay effect in fault fixing. Recently, a mathematical model for optimal time determination in multi-release software has been proposed in [53]. The authors considered different types of users, namely innovators and imitators in predicting the release time.

In this paper, next release planning has been proposed based on the predefined addressed features and bugs.

An economy outlook is presented by Cobb-Douglas function [6, 7 and 13].

In our work, we have used the Cobb-Douglas function to model the growth of fixing of different issues.

3. DATA COLLECTION AND MODEL BUILDING

3.1 Data collection

We have taken data from five products, namely Avro, jUDDI, Hive, Pig and Whirr of Apache open source project [10 and 42] where bugs, NFs and IMPs have been presented with different signs as shown in Figure 1.

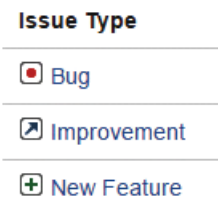

Figure 1 Different issue types for Apache products.

The fault tracking data of Mozilla project for three successive versions Firefox 3.0, 3.5, and 3.6 (https://bugzilla.mozilla.org/) [39] and for the product of gnome-control-center (https://bugzilla.gnome.org/) [40], for 4 successive versions from 2.0 to 2.3 have been selected [35]. The historical code change data has been extracted using GitHub tool [11]. Figure 2 shows the sample of different issue reports of Avro product.

Figure 3 shows the screen shot to download code change history for Avro product from the GitHub repository.

The process of data collection for Apache project and Entropy calculation has been carried out using the methods and formulas as discussed and proposed in [2, 12 and 53].

3.2 Modeling for Multi-Release Software Product

In OSS, once the issues are reported, the triaging takes place and different issues are assigned to developers. The source code of different files gets changed during fixing of these issues and the product goes for next release. But, there are some issues which are still left in the current release, which get fixed in the next release. A mathematical model is necessary here, which will predict the leftover issues of a release to be fixed in the next releases.

The class of time-domain software reliability models assumes that software failures display the behavior of a Non-Homogeneous Poisson Process (NHPP) [16-20, 26].

Let Poisson probability mass function of a random variable, $N(t)$, with parameter $X(t)$ is defined as

$$p\{N(t)=n\}=\frac{[X(t)]^n * e^{-X(t)}}{n!}, n=0,1,2,.....,\infty \quad (1)$$

Various time domain models have appeared in the literature which describe the stochastic failure process by an NHPP [26].

'a' : potential number of issues. Issues can have 1) bugs/NFs/IMPs, 2) NFs+IMPs and 3) bugs+NFs+IMPs. These issues in infiite time with finite failure NHPP can be written as with the following differential equation (2) as follows in [7 and 53].

$$\frac{d}{dt}X(t)=\frac{f(t)}{1-F(t)}[a-X(t)] \quad (2)$$

Here, $F(t)$ is a distribution function and $f(t)=\frac{d}{dt}F(t)$ is a density function. The quantity $[a-X(t)]$ denotes the expected number of issues remaining in the software at time t.

Solving above equation at $t=0$, $X(0)=0$ we get the following

$$X(t)=aF(t) \quad (3)$$

Here, $X(t)$ is the cumulative value of fixed bugs/new features/feature improvements / (new features + feature improvements) at any given time t.

During data collection, we observe that in the beginning, the cumulative number of issues fixed is slow and after that it increases and then stabilizes in subsequent releases. To model such behaviour of issue fixing, we proposed a model based on logistic function, i.e.

$$F(t)=\left[\frac{1}{1+\gamma\exp(-\varphi t))}\right] \quad (4)$$

By using equation (4) in (3), we get

$$X(t)=a\left[\frac{1}{1+\gamma\exp(-\varphi t))}\right] \quad (5)$$

Here, γ is a constant and depending upon its value, models can capture different types of issue fixing growth curves. The φ denotes an issue fixing rate per remaining issue.

Here, we consider that in equation (4), if $t=0$ then $F(0)=\frac{1}{1+\gamma}$, in case of open source development environment where contributors are located at different geographical locations, during the initial time period ($t=0$) issue fixing is very slow and γ takes a very large value. In this scenario, $F(0)=0$. If we take $t=\infty$, $F(\infty)=1$. It means, the value of $F(t)$ given in (4) satisfies the condition to be a distribution function strictly in the case of open source development environment.

The model given in equation (5) is used to predict the potential number of bugs, new features, feature improvements, and (new features + feature improvements) at any given time for Release 1 data.

Now, we consider that source code is changed to fix bugs, new features and feature improvements. The changes in the source code of the software can be quantified using entropy based measure [2, 3 and 12] To incorporate and consider the source code changes for fixing the issues, we extend the Cobb-Douglas type function [13]. We considered time and entropy both simultaneously instead of only time. In (6), s and u represent the time and entropy (the complexity of code changes) respectively. β is the code change elasticity to issues fixing time.

$$t\equiv s^{\beta}u^{1-\beta}(0\le\beta\le 1) \quad (6)$$

We can develop 2-dimensional model by using calendar time 's' and entropy 'u' which results in equation (7).

$$X(s,u)=a\left[\frac{1}{1+\gamma\exp\left(-\varphi(s^{\beta}u^{1-\beta})\right)}\right] \quad (7)$$

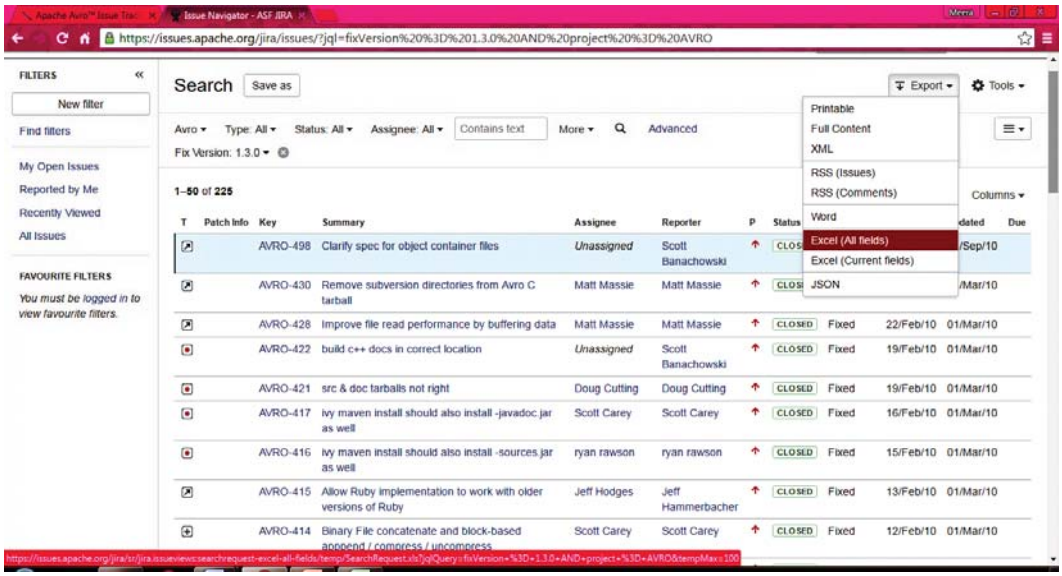

Figure 2 Sample issue reports for Avro product.

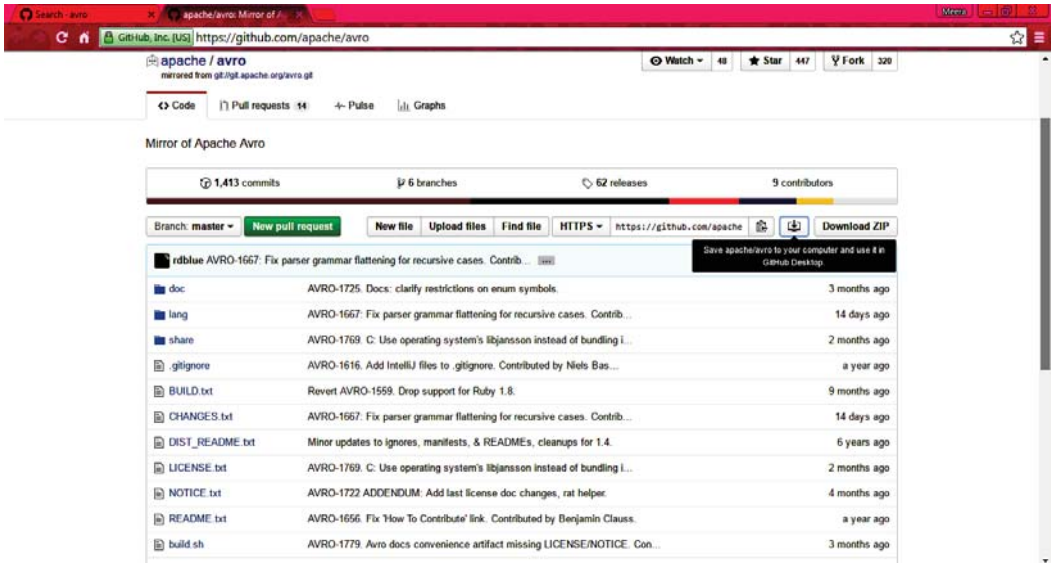

Figure 3 Avro product code change history on GitHub.

The above model can also be written in the following form

$$X(s,u) = aF(s,u)$$

and

$$F(s,u) = \left[\frac{1}{1+\gamma \exp\left(-\varphi(s^{\beta}u^{1-\beta})\right)}\right] \tag{8}$$

We used model given in equation (7) to predict the potential number of total issues at any given time for Release 1 data. We consider the same fixing rate of issues across different releases for the sake of simplicity.

Multi-Release Modeling based on Leftover Issues of just previous Release [53 and 54]

In software, different issues, namely bugs, new features and feature improvements are reported and get fixed in the current release. The remaining unresolved issues which are leftover move to the next release. The mean value function of issues based on calendar time and entropy has been evaluated by using (5) and (7) respectively

During our empirical investigation, we found that the issues, namely, new features and feature improvements are fixed in the current release and the unresolved are fixed in next release, means next release considers the leftover new features and feature improvements of the just released version of the software. But, in case of issues, namely bugs, leftover bugs of Release1 are fixed in Release 2, Release 3 and Release 4. It means, in an open source development environment leftover bugs of different releases are passed on to higher releases (up to next three-four releases). Here, we consider that leftover bugs of Release 1 can pass up to Release 4. Based on this empirical evidence, the different mean value functions for different releases have been modeled as follows:

We consider that in the first release different bugs are reported and get fixed are modeled by the following equation

$$X_1(t) = a_1 F(t), \quad 0 < t < t_1 \tag{9}$$

where a_1 is the potential bugs to be fixed in the first release at time t_1. The leftover bugs of first release, i.e. $a_1(1 - F_1(t_1))$ are added to the potential bugs of second release with fixing rate $F_2(t - t_1)$. Therefore, the mathematical equation representing the cumulative number of bugs fixed in the second release is given by

$$X_2(t) = a_2 F_2(t - t_1) + a_1(1 - F_1(t_1)) F_2(t - t_1), t_1 < t < t_2 \tag{10}$$

In above equation a_2 is the potential bugs to be fixed in the second release. In the line of modeling for the second release and along with taking into consideration the fact that the next release will contain the remaining bugs of all the previous releases, we can write the expressions for Release 3 and Release 4.

$$\begin{aligned} X_3(t) = {} & a_3 F_3(t - t_2) + a_2(1 - F_2(t_2)) F_3(t - t_2) \\ & + a_1(1 - F_1(t_1))(1 - F_2(t_2)) F_3(t - t_2), t_2 < t < t_3 \end{aligned} \tag{11}$$

$$\begin{aligned} X_4(t) = {} & a_4 F_4(t - t_3) + a_3(1 - F_3(t_3)) F_4(t - t_3) \\ & + a_2(1 - F_2(t_2))(1 - F_3(t_3)) F_4(t - t_3) \\ & + a_1(1 - F_1(t_1))(1 - F_2(t_2))(1 - F_3(t_3)) \\ & F_4(t - t_3), \quad t_3 < t < t_4 \end{aligned} \tag{12}$$

Multi-Release Modeling based on unresolved Bugs passed on to different Releases [36, 53 and 54]

If we consider that the next release consists of remaining issues of just previous release, then we can write the following expressions for different releases.

The mathematical modeling for cumulative addressed issues for Release 1 is given as equation (13).

$$X_1(t) = a_1 F(t), \quad 0 < t < t_1 \tag{13}$$

a_1 is the addressed potential issues of Release 1. The leftover issues, i.e. $a_1(1 - F_1(t_1))$ with fixing rate $F_2(t - t_1)$ are added to the issue content of Release 2. Mathematical equations for cumulative addressed issues estimation in Release 2 and Release 3 are given in equation (14) and equation (15).

$$X_2(t) = \left(a_2 + a_1\left(1 - F_1\left(t_1\right)\right)\right) F_2(t - t_1), t_1 < t < t_2 \tag{14}$$

$$X_3(t) = (a_3 + a_2(1 - F_2(t_2))) F_3(t - t_2), \ t_2 \le t < t_3 \tag{15}$$

Similarly, we can write equation (16) for the n^{th} release.

$$\begin{aligned} X_n(t) = {} & (a_n + a_{n-1}(1 - F_{n-1}(t_{n-1}))) F_n(t - t_{n-1}), \\ & t_{n-1} \le t < t_n \end{aligned} \tag{16}$$

4. EXPERIMENTAL SETUP

In this section, we have discussed multi-release based on unresolved Bugs passed on to different releases and multi-release based on the Leftover Issues of just previous release.

Table 1 Next Release Planning.

Release time Planning using different issues fixed based on calendar time (month)	
Case 1	Bugs prediction
Case 2	IMPs prediction
Case 3	NFs prediction
Case 4	NFs+IMPs prediction
Case 5	Bugs+NFs+IMPs prediction
Case 6	Bugs+NFs+IMPs prediction

Multi-Release based on unresolved Bugs passed on to different Releases

In an open source development environment, leftover bugs of different releases are passed on to the higher releases. We validated the proposed models given in equations (9), (10) and (11) for case 1 (in Table 1), on weekly bugs fixed data of three releases of Firefox and Gnome-control-center projects [35]. Results have been presented in Table 3 given in the next section.

Multi-Release based on the Leftover Issues of just previous Release

If we consider that the next release consists of the remaining issues of just previous release, then potential values of different issues (cases 1-5 of Table 1) in Release 1 can be estimated using equation (13) and the potential value of all the issues (case 6 of Table 1) can be estimated using model given in (7) as this model takes care of entropy. For Release 2 we have used model given in (14). For Release 3 we have used model given in (15). The generalized model for n^{th} release given in equation (16) has been used to estimate parameters for rest of the further releases by considering just previous release leftover issues. For the first release, the leftover issues have been predicted by using the parameters estimation results of first release. Release 2 parameters are estimated by considering Release 1 leftover issues along with Release 2 dataset. For this we have used equation (14). With the resultant leftover issues, along with Release 3 dataset, we have estimated Release 3 parameters. For this we have used equation (15). For Release 4 and Release 5 parameters estimation we have followed the same process and used equation (16).

We have validated the proposed models discussed here for Apache open source products, namely Hive, Pig, Avro, jUDDI and Whirr [10]. We have used Nonlinear Regression (NLR) in Statistical Package for Social Sciences (SPSS) software to estimate the parameters (a, ϕ, β and α). Table 2 shows the values of different parameters we have used in NLR.

Table 2 Parameters used in Nonlinear Regression.

Parameter	Estimation method		Step limit	Infinite step size
Value	Sequential Programming	Quadratic	2	1.00E+20

From the results, it has been observed that the proposed models representing all cases of Table 1 give a high goodness of fit.

Table 3 Estimated parameter values and numerical results of bug fixing model (case 1 in Table I) for three releases of Firefox and Gnome-control-center

Product and No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs fixed (A_i)	$a_i - A_i$ (Leftover bugs of i^{th} release added to different releases)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
Firefox 3.0 (Release 1)	Prediction of bugs based on calendar time(month)	$a_1 = 50$, $\varphi_1 = .070$, $\gamma_1 = 4.827$	48	2	6.436	-0.103	2.559	2.561	.995
Firefox 3.5 (Release 2)	Prediction of bugs based on calendar time(month)	$a_1 = 45$, $\varphi_1 = .190$, $\gamma_1 = 12.857$	45	0	2.813	-0.170	1.699	1.707	.986
Firefox 3.6 (Release 3)	Prediction of bugs based on calendar time(month)	$a_1 = 32$, $\varphi_1 = .135$, $\gamma_1 = 7.394$	35	-3	2.699	-0.104	1.656	1.659	.972
Gnome 2.0 (Release 1)	Prediction of bugs based on calendar time(month)	$a_1 = 43$, $\varphi_1 = .366$, $\gamma_1 = 38.720$	42	1	1.396	-0.035	1.217	1.217	.993
Gnome 2.1 (Release 2)	Prediction of bugs based on calendar time(month)	$a_2 = 95$, $\varphi_2 = .179$, $\gamma_2 = 19.751$	35	60	4.613	-0.082	2.233	2.235	.943
Gnome 2.2 (Release 3)	Prediction of bugs based on calendar time(month)	$a_2 = 243$, $\varphi_2 = .125$, $\gamma_2 = 103.139$	23	220	6.311	0.23	2.583	2.593	.787

We have used various performance measures (Bias, Variation (VAR), Mean Squared Error (MSE), Root Mean Squared Prediction Error (RMSPE) and Rsquare (R^2)) to measure goodness of fit of different models.

5. NUMERICAL EXAMPLES

Numerical application examples are given here for the illustration purpose for Firefox, Gnome-control-center and Apache software products. We have measured the performance of different estimation models for all the releases.

Multi-Release based on unresolved Bugs passed on to different Releases

Table 3 shows the parameter estimates of case 1 defined in Table 1, for Firefox and Gnome-control-center projects. Here, we consider that leftover bugs of Release 1 can pass upto Release 3. The number of fixed real bugs of i^{th} release is shown by 'A_i'. The potential number of bugs of i^{th} release is shown by 'a_i' and the leftover bugs of i^{th} release is shown by '$a_i - A_i$'. The issues fixing efficiency for i^{th} release is represented by the parameter ϕ_i. γ_i shows the variation in bug fixing pattern of i^{th} release.

In Table 3, we have documented the leftover bugs of each release, which are the addons to the fault content of the next releases. In case of Firefox, the estimation results show that 50 initial bugs are there in Release 1. But, fixed bugs are 48. Hence $50 - 48 = 2$ bugs remained unresolved, which added to the next release fault content. We can draw similar inferences for other releases. For Release 2, 45 bugs have been fixed and the estimation also shows $(a_2 + a_1(1 - F_1(t_1))) = 45$ bugs, which means all the bugs have been detected and fixed in Release 2.

For Release 3, model shows under fit performance.

In case of Gnome-control-center, the estimation results show that 43 initial bugs are there in Release 1. But, fixed bugs are 42. Hence $43 - 42 = 1$ bug remained unresolved, which added to the next release fault content. We can draw similar inferences for other releases. For Release 2, 35 bugs have been fixed and the estimation shows $(a_2 + a_1(1 - F_1(t_1))) = 95$ bugs, which means 60 bugs remained unresolved and added to the third release fault content. Similarly, we observe that 220 leftover bugs added to next release from Release 3.

We observe that estimation models, exhibits good fit in terms of MSE, Bias, VAR, RMSPE and R^2 for all the releases.

Multi-Release based on the Leftover Issues of just previous Release

Tables 4 to 8 present parameter estimates of Case 1 to Case 4 (Table 1) for different Apache datasets. Here, we considered that leftover issues of just previous release are added to the issue content of next release. The number of fixed real issues (bugs/IMPs/NFs/NFs+IMPs) of i^{th} release is shown by 'A_i'. The potential number of different issues of i^{th} release is shown by 'a_i' and the leftover issues of i^{th} release is shown by '$a_i - A_i$'. The issues fixing efficiency for i^{th} release is represented by the parameter ϕ_i. γ_i shows the variation in bug fixing pattern of i^{th} release. γ_i is the code change elasticity to issues fixing time for i^{th} release.

In Table 4, we have documented the leftover bugs of each Avro release, which are the addons to the fault content of the next releases. The estimation results show that 149 initial bugs are there in Release 1. But, fixed bugs are 134. Hence $149 - 134 = 15$ bugs remained unresolved, which added to the next Release 2 fault content. We can draw similar inferences for other releases. For Release 2, 81 bugs have been fixed and the estimation also shows $(a_2 + a_1(a - F_1(t_1))) = 91$ bugs, which means 10 bugs remained unresolved and added to the third release fault content. Similarly, we observe that 16 leftover bugs added to Release 4 from Release 3. From Release 4, 2 unresolved bugs are added to Release 5.

The estimation results of IMPs show that 161 initial IMPs are there in Release 1. But, addressed IMPs are 153. Hence $161 - 153 = 8$ IMPs issues remained unresolved, which added to the Release 2 issue content. We can draw similar inferences for other releases. For fourth release, 41 IMPs have been addressed and the estimation shows 46 IMPs, which means 5 IMPs remained

Table 4 Estimated parameter values and numerical results of different models (cases 1 to 4 mentioned in Table 1) for different releases of Avro.

No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs / IMPs / NFs/ (NFs + IMPs) fixed (A_i)	$a_i - A_i$ (Leftover bugs of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	Case 1	$a_1 = 149$, $\phi_1 = .487$, $\gamma_1 = 21.035$	134	15	8.42	−0.08	3.04	3.04	0.996
1	Case 2	$a_1 = 161$, $\phi_1 = .696$, $\gamma_1 = 73.014$	153	8	20.43	0.1	4.74	4.74	0.994
1	Case 3	$a_1 = 49$, $\phi_1 = .538$, $\gamma_1 = 25.072$	45	4	2.05	0.07	1.5	1.5	0.991
1	Case 4	$a_1 = 209$, $\phi_1 = .657$, $\gamma_1 = 55.783$	198	11	27.74	0.26	5.52	5.52	0.995
2	Case 1	$a_2 = 91$, $\phi_2 = .429$, $\gamma_2 = 12.829$	81	10	15.2	−0.2	4.08	4.09	0.977
2	Case 2	$a_2 = 107$, $\phi_2 = .265$, $\gamma_2 = 10.964$	68	39	16.34	−0.17	4.24	4.24	0.955
2	Case 3	$a_2 = 44$, $\phi_2 = .29$, $\gamma_2 = 7.415$	34	10	4.11	−0.08	2.12	2.13	0.952
2	Case 4	$a_2 = 148$, $\gamma_2 = .272$, $\gamma_2 = 9.463$	102	46	35.86	−0.24	6.28	6.28	0.955
3	Case 1	$a_3 = 128$, $\gamma_3 = .341$, $\gamma_3 = 19.897$	112	16	30.98	−0.28	5.79	5.79	0.971
3	Case 2	$a_3 = 73$, $\gamma_3 = .464$, $\gamma_3 = 18.285$	73	0	15.51	−1.83	3.63	4.07	0.973
3	Case 3	$a_3 = 23$, $\phi_3 = .576$, $\gamma_3 = 26.588$	24	−1	2.15	−0.04	1.53	1.53	0.964
3	Case 4	$a_3 = 98$, $\gamma_3 = .439$, $\gamma_3 = 16.722$	97	1	26.55	−1.23	5.21	5.35	0.973
4	Case 1	$a_4 = 87$, $\phi_4 = .384$, $\gamma_4 = 11.085$	85	2	15.69	−0.27	4.11	4.12	0.975
4	Case 2	$a_4 = 46$, $\phi_4 = .408$, $\gamma_4 = 22.837$	41	5	1.37	−0.1	1.22	1.22	0.992
4	Case 3	$a_4 = 19$, $\phi_4 = .335$, $\gamma_4 = 19.859$	16	3	1.3	0.42	1.11	1.18	0.973
4	Case 4	$a_4 = 69$, $\phi_4 = .336$, $\gamma_4 = 17.422$	57	12	2.08	−0.08	1.5	1.5	0.993
5	Case 1	$a_5 = 44$, $\phi_5 = .463$, $\gamma_5 = 18.344$	42	2	7.9	0.54	2.89	2.94	0.957
5	Case 2	$a_5 = 20$, $\phi_5 = .95$, $\gamma_5 = 99.752$	20	0	0.71	0.12	0.88	0.88	0.986
5	Case 3	$a_5 = 13$, $\phi_5 = .982$, $\gamma_5 = 146.158$	13	0	0.19	0.07	0.46	0.46	0.991
5	Case 4	$a_5 = 34$, $\phi_5 = .558$, $\gamma_5 = 1$	33	1	6.83	−1.39	2.32	2.7	0.951

unresolved and added to the fifth release issue content.

In Release 1, 45 NFs have been addressed and the potential estimated value is 49. Hence, $49 - 45 = 4$ NFs issues remained unresolved, which added to the Release 2 issue content. For fourth release, 16 NFs have been addressed and the estimation shows 19 NFs, which means 3 NFs remained unresolved and added to the fifth release issue content.

In Release 1, 198 IMPs+NFs have been addressed and the potential estimated value is 209. Hence, $209 - 198 = 11$ IMPs+NFs issues remained unresolved, which added to the Release 2 issue content. For fourth release, 57 IMPs+NFs have been addressed and the estimation shows 69 IMPs+NFs, which means 12 IMPs+NFs remained unresolved and added to the fifth release issue content.

Similar, interpretations can be drawn for other products.

By analyzing the above empirical results, we can evaluate the quality of maintainability and the adaptability of software in fixing of different issues. The number of issues left in different releases determines the release readiness and the stability of the software. In case of Apache datasets, we observed that for Avro product 61.5% of the potential issues get fixed before the next release. For Pig product except new features fixing in Release 4, all releases are following the same fixing level. For Hive product, all the issues fixing process achieved 61.5% performance. We observe that for jUDDI product issues fixing process for different issues, namely bugs, NFs and IMPs follow the same pattern except bug fixing for Release 4 and total issues fixing for Release 2.

The releases of Whirr product follow the same pattern except improvements and new features fixing in fourth release. Results show that 61.5% of potential issues of current release get fixed before the next release except the four cases in all five products. All the estimation models (Table 1) exhibits a good fit in terms of R^2, MSE, RMSPE, Bias and VAR. The models give R^2 greater than 0.90 in 123 cases out of 138 total cases across all the releases of Apache.

Table 5 Estimated parameter values and numerical results of different models (cases 1 to 4 mentioned in Table 1) for different releases of Pig.

No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs / IMPs / NFs / (NFs + IMPs) fixed (A_i)	$a_i - A_i$ (Leftover bugs of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	Case 1	$a_1 = 299, \phi_1 = .419, \gamma_1 = 30.657$	231	68	19.74	−0.28	4.65	4.66	0.996
1	Case 2	$a_1 = 75, \phi_1 = .575, \gamma_1 = 53.178$	69	6	4.61	0.25	2.24	2.25	0.992
1	Case 3	$a_1 = 43, \phi_1 = .439, \gamma_1 = 24.829$	34	9	3.57	0.1	1.98	1.98	0.973
1	Case 4	$a_1 = 118, \phi_1 = .522, \gamma_1 = 39.343$	103	15	11.7	0.37	3.57	3.59	0.99
2	Case 1	$a_2 = 408, \phi_2 = .34, \gamma_2 = 11.678$	376	32	100.75	−0.17	10.45	10.45	0.991
2	Case 2	$a_2 = 121, \phi_2 = .431, \gamma_2 = 8.763$	123	−2	36.83	−0.81	6.26	6.31	0.968
2	Case 3	$a_2 = 42, \phi_2 = .432, \gamma_2 = 8.272$	43	−1	8.11	−0.6	2.9	2.96	0.947
2	Case 4	$a_2 = 169, \phi_2 = .333, \gamma_2 = 5.213$	166	3	119.36	−5.86	9.6	11.24	0.944
3	Case 1	$a_3 = 407, \phi_3 = .273, \gamma_3 = 4.774$	354	53	99.54	−0.27	10.42	10.42	0.987
3	Case 2	$a_3 = 114, \phi_3 = .298, \gamma_3 = 8.104$	94	20	5.13	−0.08	2.36	2.37	0.992
3	Case 3	$a_3 = 28, \phi_3 = .325, \gamma_3 = 10.49$	24	4	0.44	−0.03	0.7	0.7	0.991
3	Case 4	$a_3 = 143, \phi_3 = .303, \gamma_3 = 8.518$	118	25	7.08	−0.11	2.78	2.78	0.993
4	Case 1	$a_4 = 187, \phi_4 = .468, \gamma_4 = 15.136$	157	30	17.5	−0.42	4.42	4.44	0.992
4	Case 2	$a_4 = 72, \phi_4 = .436, \gamma_4 = 7.407$	63	9	8	−0.12	3	3	0.975
4	Case 3	$a_4 = 31, \phi_4 = .318, \gamma_4 = 13.255$	18	13	0.85	0	0.98	0.98	0.967
4	Case 4	$a_4 = 98, \phi_4 = .402, \gamma_4 = 7.902$	81	17	11.18	−0.12	3.54	3.55	0.978
5	Case 1	$a_5 = 162, \phi_5 = .479, \gamma_5 = 6.764$	154	8	25.44	−0.27	5.34	5.35	0.985
5	Case 2	$a_5 = 47, \phi_5 = .579, \gamma_5 = 9.968$	47	0	3.04	−0.12	1.84	1.85	0.983
5	Case 3	$a_5 = 18, \phi_5 = .366, \gamma_5 = 8.808$	16	2	1.63	−0.05	1.35	1.35	0.906
5	Case 4	$a_5 = 63, \phi_5 = .532, \gamma_5 = 8.808$	63	0	8.17	−0.18	3.03	3.03	0.974

Table 6 Estimated parameter values and numerical results of different models (cases 1 to 4 mentioned in Table 1) for different releases of Hive.

No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs/ IMPs/NFs/ (NFs + IMPs)fixed (A_i)	$a_i - A_i$ (Leftover issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	Case 1	$a_1 = 308, \varphi_1 = .387, \gamma_1 = 8.533$	271	37	51.17	−0.42	7.53	7.54	0.991
1	Case 2	$a_1 = 96, \varphi_1 = .479, \gamma_1 = 10.938$	96	0	16.84	−0.24	4.32	4.33	0.997
1	Case 3	$a_1 = 75, \varphi_1 = .597, \gamma_1 = 18.381$	75	0	8.2	−0.21	3.01	3.02	0.986
1	Case 4	$a_1 = 171, \varphi_1 = .531, \gamma_1 = 13.607$	171	0	42.29	−0.45	6.84	6.85	0.984
2	Case 1	$a_2 = 290, \varphi_2 = .321, \gamma_2 = 13.746$	244	46	15.97	−0.34	4.14	4.16	0.997
2	Case 2	$a_2 = 203, \varphi_2 = .256, \gamma_2 = 13.929$	136	67	8.55	−0.04	3.04	3.04	0.994
2	Case 3	$a_2 = 88, \varphi_2 = .287, \gamma_2 = 6.997$	77	11	6.38	−0.07	2.63	2.63	0.985
2	Case 4	$a_2 = 289, \varphi_2 = .256, \gamma_2 = 10.413$	213	76	9.88	−0.14	3.27	3.27	0.997
3	Case 1	$a_3 = 290, \varphi_3 = .362, \gamma_3 = 15.835$	253	37	11.4	0.11	3.51	3.51	0.998
3	Case 2	$a_3 = 160, \varphi_3 = .392, \gamma_3 = 15.921$	153	7	9.92	−0.25	3.27	3.28	0.995
3	Case 3	$a_3 = 51, \varphi_3 = .372, \gamma_3 = 17.936$	47	4	1.97	−0.08	1.46	1.46	0.99
3	Case 4	$a_3 = 211, \varphi_3 = .387, \gamma_3 = 16.289$	200	11	15.51	−0.33	4.08	4.1	0.996
4	Case 1	$a_4 = 798, \varphi_4 = .271, \gamma_4 = 20.188$	523	275	243.48	−1.38	16.18	16.24	0.989
4	Case 2	$a_4 = 217, \varphi_4 = .245, \gamma_4 = 15.833$	135	82	6.67	−0.18	2.68	2.69	0.995
4	Case 3	$a_4 = 76, \varphi_4 = .293, \gamma_4 = 16.133$	57	19	1.63	−0.03	1.33	1.33	0.994
4	Case 4	$a_4 = 287, \varphi_4 = .26, \gamma_4 = 15.602$	192	95	9.45	−0.23	3.19	3.2	0.997
5	Case 1	$a_5 = 895, \varphi_5 = .412, \gamma_5 = 18.615$	831	64	824.3	−3.18	29.7	29.87	0.989
5	Case 2	$a_5 = 47, \varphi_5 = .579, \gamma_5 = 9.968$	191	4	16.63	−0.75	4.17	4.24	0.996
5	Case 3	$a_5 = 18, \varphi_5 = .366, \gamma_5 = 8.808$	44	3	2.59	−0.18	1.67	1.68	0.989
5	Case 4	$a_5 = 63, \varphi_5 = .532, \gamma_5 = 8.808$	235	7	29.42	−0.94	5.56	5.64	0.996

Comparison of Performance for the proposed models

The proposed issue estimation models (Case 5 and Case 6 in Table 1) have been compared with Goel-Okumoto model [37] (equation (17)) and Yamada delayed S-shaped model [38] (equation (18)).

$$X_i(t) = a_i(1 - \exp(-b_i t)) \tag{17}$$

$$X_i(t) = a_i(1 - (1 + b_i t)\exp(-b_i t))\ a_i, b_i > 0 \tag{18}$$

$X_i(t)$ denotes the i^{th} release cumulative number of fixed errors at time t. 'a_i' is the potential number of errors. 'b_i' is the fixing rate of errors for i^{th} release.

Tables 9 to 13 present the results of these two models in comparison of proposed models. For Avro product (Table 9), the

Table 7 Estimated parameter values and numerical results of different models (cases 1 to 4 mentioned in Table I) for different releases of jUDDI.

No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs/ IMPs/NFs/ (NFs + IMPs)fixed (A_i)	$a_i - A_i$ (Leftover issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	$R^2{}_i$
1	Case 1	$a_1 = 160, \varphi_1 = .253, \gamma_1 = 18.653$	109	51	12.95	−0.072	3.724	3.724	0.989
1	Case 2	$a_1 = 9, \varphi_1 = .434, \gamma_1 = 10.628$	9	0	0.3	0.023	0.566	0.567	0.962
1	Case 3	$a_1 = 2, \varphi_1 = .688, \gamma_1 = 3.292$	2	0	0.025	0.003	0.166	0.166	0.836
1	Case 4	$a_1 = 11, \varphi_1 = .406, \gamma_1 = 6.626$	11	0	0.319	0.016	0.584	0.585	0.967
2	Case 1	$a_2 = 104, \varphi_2 = .204, \gamma_2 = 14.994$	64	40	9.453	0.013	3.175	3.175	0.974
2	Case 2	$a_2 = 9, \varphi_2 = .459, \gamma_2 = 105.781$	8	1	0.351	0.067	0.608	0.612	0.962
2	Case 3	$a_2 = 3, \varphi_2 = .326, \gamma_2 = 7.477$	3	0	0.076	−0.005	0.286	0.286	0.912
2	Case 4	$a_2 = 13, \varphi_2 = .344, \gamma_2 = 27.784$	11	2	0.695	0.027	0.86	0.861	0.955
3	Case 1	$a_3 = 21, \varphi_3 = .187, \gamma_3 = 1.117$	19	2	0.358	0.002	0.621	0.621	0.953
3	Case 2	$a_3 = 2, \varphi_3 = 1, \gamma_3 = 0$	2	0	0	0	0	0	*
3	Case 3	$a_3 = 3, \varphi_3 = .185, \gamma_3 = 2.951$	2	1	0.058	0.002	0.25	0.25	0.769
3	Case 4	$a_3 = 6, \varphi_3 = .073, \gamma_3 = 1.293$	4	2	0.06	0	0.255	0.255	0.757
4	Case 1	$a_4 = 382, \varphi_4 = .226, \gamma_4 = 86.485$	103	279	29.157	−0.544	5.56	5.56	0.966
4	Case 2	$a_4 = 51, \varphi_4 = .311, \gamma_4 = 46.207$	39	12	6.326	−0.37	2.575	2.601	0.954
4	Case 3	$a_4 = 29, \varphi_4 = .408, \gamma_4 = 62.549$	27	2	0.757	−0.07	0.898	0.9	0.991
4	Case 4	$a_4 = 75, \varphi_4 = .355, \gamma_4 = 49.988$	66	9	9.917	−0.472	3.222	3.257	0.977

Table 8 Estimated parameter values and numerical results of different models (cases 1 to 4 mentioned in Table 1) for different releases of Whirr.

No. of release	Different models	Estimated parameters (a_i, φ_i and γ_i)	Real no. of Bugs/ IMPs/NFs/ (NFs + IMPs)fixed (A_i)	$a_i - A_i$ (Leftover issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	Case 1	$a_1 = 60, \varphi_1 = .665, \gamma_1 = 17.067$	58	2	0.938	0.022	1.026	1.027	0.997
1	Case 2	$a_1 = 88, \varphi_1 = .455, \gamma_1 = 13.508$	74	14	8.426	−0.117	3.076	3.078	0.982
1	Case 3	$a_1 = 19, \varphi_1 = .549, \gamma_1 = 41.637$	14	5	0.162	0.055	0.422	0.422	0.992
1	Case 4	$a_1 = 110, \varphi_1 = .449, \gamma_1 = 15.338$	88	22	7.715	−0.106	2.943	2.945	0.989
2	Case 1	$a_2 = 35, \varphi_2 = .47, \gamma_2 = 18.442$	24	11	0.813	0.028	0.963	0.964	0.985
2	Case 2	$a_2 = 37, \varphi_2 = .707, \gamma_2 = 12.288$	37	0	5.922	0.002	2.601	2.601	0.95
2	Case 3	$a_2 = 10, \varphi_2 = .663, \gamma_2 = 6.294$	10	0	0.088	−0.013	0.371	0.371	0.988
2	Case 4	$a_2 = 47, \varphi_2 = .681, \gamma_2 = 10.163$	47	0	5.689	−0.015	2.549	2.549	0.968
3	Case 1	$a_3 = 61, \varphi_3 = .387, \gamma_3 = 15.835$	45	16	31.643	0.535	5.986	6.01	0.822
3	Case 2	$a_3 = 34, \varphi_3 = .325, \gamma_3 = 2.674$	32	2	6.308	−0.372	2.655	2.681	0.846
3	Case 3	$a_3 = 6, \varphi_3 = .372, \gamma_3 = 16.205$	4	2	0.542	−0.036	0.786	0.786	0.68
3	Case 4	$a_3 = 39, \varphi_3 = .4, \gamma_3 = 4.776$	36	3	15.126	0.988	4.021	4.141	0.795
4	Case 1	$a_4 = 7, \varphi_4 = .374, \gamma_4 = 7.315$	6	1	1.096	0.075	1.144	1.466	0.575
4	Case 2	$a_4 = 16, \varphi_4 = .42, \gamma_4 = 12.016$	11	5	2.331	0.371	1.622	1.664	0.714
4	Case 3	$a_4 = 6, \varphi_4 = .51, \gamma_4 = 11.679$	4	2	0.259	0.015	0.558	0.558	0.824
4	Case 4	$a_4 = 26, \varphi_4 = .568, \gamma_4 = 28.416$	15	11	2.442	0.601	1.711	1.814	0.837

proposed models (Case 5 and Case 6 in Table 1) estimation results show that 332 issues (bugs+NFs+IMPs) have been fixed in Release 1. The estimated potential value of issues is 357. This means 25 issues which would have been fixed in Release 1 now will be fixed in Release 2. Similarly, 14 issues which would have been fixed in Release 4 will be fixed in Release 5.

Similar, interpretations can be drawn for other products.

Out of total 23 releases, in 20 releases proposed models give better R^2 than the two existing models.

We designed the experiment to test the statistical significance of the proposed model as discussed in case 6 in Table 1. The statistical significance has been validated using non-parametric Kolmogorov-Smirnov (K-S) test.

Null hypothesis (H0): The distribution of observed and estimated values are same.

Alternate hypothesis (H1): The distribution of observed and estimated values are not same.

The P-values of the experiment have been given in Table 14. We have taken level of significance $\alpha = 0.025$. We observed that the proposed model is statistically significant.

6. OPTIMAL RELEASE PLANNING

In closed source software, due to time and resource constraints it is not possible to address/fix all the issues in the current release

Table 9 Estimated parameter values and numerical results for Avro.

No. of release	Different models	Estimated parameters (a_i, b_i, φ_i and γ_i)	Real no. of bugs/issues fixed (A_i)	$a_i - A_i$ (Leftover bugs/issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	GO model	$a_1 = 360, b_1 = .121$	$A_1 = 332$	28	2549.20	0.97	52.94	52.95	0.81
1	S-shaped model	$a_1 = 621, b_1 = .173$		289	202.30	−0.76	14.90	14.92	0.985
1	Case 5	$a_1 = 357, \varphi_1 = .581, \gamma_1 = 35.664$		25	59.84	0.18	8.11	8.11	0.996
1	Case 6	$a_1 = 357, \varphi_1 = .594,$ $\gamma_1 = 36.419, \beta_1 = .805$		25	60.04	0.21	8.12	8.13	0.996
2	GO model	$a_2 = 197, b_2 = .13$	$A_2 = 183$	14	283.14	1.41	17.59	17.64	0.901
2	S-shaped model	$a_2 = 221, b_2 = .268$		38	108.00	1.94	10.71	10.88	0.962
2	Case 5	$a_2 = 223, \varphi_2 = .343, \gamma_2 = 9.989$		40	86.95	−0.48	9.77	9.78	0.97
2	Case 6	$a_2 = 223, \varphi_2 = .348,$ $\gamma_2 = 9.954, \beta_2 = .816$		40	87.91	−0.48	9.82	9.83	0.969
3	GO model	$a_3 = 264, b_3 = .078$	$A_3 = 209$	55	442.06	−2.88	21.68	21.87	0.892
3	S-shaped model	$a_3 = 300, b_3 = .174$		91	71.38	0.15	8.79	8.79	0.983
3	Case 5	$a_3 = 207, \varphi_3 = .423, \gamma_3 = 20.116$		−2	101.84	−0.77	10.47	10.50	0.975
3	Case 6	$a_3 = 214, \varphi_3 = .398,$ $\gamma_3 = 18.831, \beta_3 = .808$		5	107.69	0.57	10.78	10.8	0.974
4	GO model	$a_4 = 219, b_4 = .063$	$A_4 = 142$	77	102.51	1.39	10.44	10.53	0.944
4	S-shaped model	$a_4 = 181, b_4 = .211$		38	26.45	1.06	5.24	5.34	0.986
4	Case 5	$a_4 = 156, \varphi_4 = .352,$ $\gamma_4 = 12.586$		14	28.21	−0.37	5.52	5.53	0.985
4	Case 6	$a_4 = 156, \varphi_4 = .355,$ $\gamma_4 = 12.586, \beta_4 = .814$		14	28.19	−0.36	5.51	5.52	0.985
5	GO model	$a_5 = 85, b_5 = .116$	$A_5 = 75$	10	104.55	0.40	10.72	10.72	0.837
5	S-shaped model	$a_5 = 104, b_5 = .233$		29	24.42	−0.32	5.17	5.18	0.962
5	Case 5	$a_5 = 71, \varphi_5 = .768, \gamma_5 = 54.475$		−4	11.01	0.23	3.47	3.48	0.983
5	Case 6	$a_5 = 77, \varphi_5 = .521,$ $\gamma_5 = 16.845, \beta_5 = .803$		2	20.96	−1.3	4.6	4.78	0.967

and in OSS due to active user's demands. By putting a constraint on minimum number of fixed issues with different severity levels before the next release of the software, we can maximize the active user's satisfaction. The following objective function considers two different severity groups, S_{n1} (average severity for n^{th} release issues) and S_{n2} (average severity for $(n-1)^{th}$ and n^{th} release issues which will be addressed in $(n+1)^{th}$ release).

$$\begin{aligned} \textit{Maximize } S_n(t) &= S_{n1} X_n(t) + S_{n2} \\ &\times \left[a_n + a_{n-1}\left(1 - F_{n-1}(t_{n-1})\right) - \left(X_n(t)\right)\right] \end{aligned} \tag{19}$$

In IEEE 982.2 defect indices definition [41] 10 weight has been used for high severity issues, 3 weight for medium severity issues and 1 for low severity issues. We have also used the same weights for different severity issues as given in [41]. By using (7) and (8), we can write the following function to consider active user's satisfaction.

Maximize

$$\begin{aligned} S_n(s^{\beta}, u^{1-\beta}) &= S_{n1}\left[\frac{a_n}{1+\gamma_n \exp\left(-\varphi_n s^{\beta_n} u^{1-\beta_n}\right)}\right] \\ &+ S_{n2}\left[\begin{array}{l} a_n + a_{n-1}\left(1 - \frac{1}{1+\gamma_{n-1}\exp\left(-\varphi_{n-1} s^{\beta_{n-1}} u^{1-\beta_{n-1}}\right)}\right) \\ -\left(\frac{a_n}{1+\gamma_n \exp\left(-\varphi_n s^{\beta_n} u^{1-\beta_n}\right)}\right) \end{array}\right] \end{aligned}$$

Subject to

$$\begin{aligned} &X_n\left(s^{\beta_n}, u^{1-\beta_n}\right) \\ &\geq \rho\left(a_n + a_{n-1}\left(1 - \frac{1}{1+\gamma_{n-1}\exp\left(-\varphi_{n-1} s^{\beta_{n-1}} u^{1-\beta_{n-1}}\right)}\right)\right) \end{aligned} \tag{20}$$

We consider $t \equiv s^{\beta}\, u^{1-\beta} (0 \leq \beta \leq 1)$.

We have solved the above nonlinear problem by using Genetic Algorithm (GA) [28].

To calculate optimal release time of n^{th} release, estimated parameters of n^{th} and $(n-1)^{th}$ releases have been used. We have solved equation (20) in MATLAB with an optimization tool by using solver "ga-Genetic Algorithm". Table 15 shows different parameter values to obtain optimal solution. We have taken Heuristic crossover function and Tournament selection function with Eliete count 2.

Similarly, we have observed that the estimated optimal release time for jUDDI third release is 19 months with 92% active user's satisfaction. The real release time we have observed is 14 months. The estimated optimal release time is close to the real release time. For Hive and Pig, we have estimated optimal release time of 14 months and 10 months for fourth releases at 98% and 98.5% user's satisfaction levels respectively. In case of Hive, the real release time is 13 months and in case of Pig it is 9 months. This shows estimated optimal release time is close to the real release time.

In case of Whirr product, we have estimated optimal release

Table 10 Estimated parameter values and numerical results for Pig.

No. of release	Different models	Estimated parameters (a_i, b_i, φ_i and γ_i)	Real no. of bugs/issues fixed (A_i)	$a_i - A_i$ (Leftover bugs/issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	GO model	$a_1 = 450$, $b_1 = .077$	$A_1 = 334$	116	2003.73	−7.07	46.36	46.89	0.829
1	S-shaped model	$a_1 = 1208$, $b_1 = .095$		874	94.22	1.58	10.05	10.17	0.992
1	Case 5	$a_1 = 409$, $\varphi_1 = .451$, $\gamma_1 = 32.201$		75	42.47	0.00	6.83	6.83	0.996
1	Case 6	$a_1 = 410$, $\varphi_1 = .456$, $\gamma_1 = 32.466$, $\beta_1 = .813$		76	43.19	0.01	6.89	6.89	0.996
2	GO model	$a_2 = 739$, $b_2 = .080$	$A_2 = 542$	197	815.55	0.27	29.72	29.72	0.991
2	S-shaped model	$a_2 = 603$, $b_2 = .253$		61	452.20	5.64	21.34	22.07	0.981
2	Case 5	$a_2 = 556$, $\varphi_2 = .360$, $\gamma_2 = 10.087$		14	257.65	−0.70	16.69	16.71	0.989
2	Case 6	$a_2 = 556$, $\varphi_2 = .365$, $\gamma_2 = 10.148$, $\beta_2 = .801$		14	257.43	−0.69	16.68	16.7	0.989
3	GO model	$a_3 = 718$, $b_3 = .084$	$A_3 = 472$	246	229.77	3.06	15.51	15.81	0.984
3	S-shaped model	$a_3 = 464$, $b_3 = .364$		−8	1086.62	9.07	33.10	34.32	0.923
3	Case 5	$a_3 = 553$, $\varphi_3 = .277$, $\gamma_3 = 5.472$		81	131.88	−0.36	11.99	11.99	0.991
3	Case 6	$a_3 = 553$, $\varphi_3 = .279$, $\gamma_3 = 5.438$, $\beta_3 = .802$		81	133.61	−0.37	12.07	12.07	0.99
4	GO model	$a_4 = 288$, $b_4 = .142$	$A_4 = 238$	50	423.07	−8.91	19.66	21.59	0.913
4	S-shaped model	$a_4 = 317$, $b_4 = .283$		79	67.18	2.32	8.34	8.65	0.986
4	Case 5	$a_4 = 287$, $\varphi_4 = .440$, $\gamma_4 = 11.878$		49	38.60	−0.49	6.57	6.59	0.992
4	Case 6	$a_4 = 286$, $\varphi_4 = .444$, $\gamma_4 = 11.849$, $\beta_4 = .814$		48	38.71	−0.49	6.58	6.6	0.992
5	GO model	$a_5 = 391$, $b_5 = .087$	$A_5 = 217$	174	12.49	−0.22	3.74	3.75	0.996
5	S-shaped model	$a_5 = 222$, $b_5 = .450$		5	77.37	1.96	9.09	9.30	0.978
5	Case 5	$a_5 = 225$, $\varphi_5 = .493$, $\gamma_5 = 7.278$		8	55.80	−0.44	7.91	7.92	0.984
5	Case 6	$a_5 = 225$, $\varphi_5 = .497$, $\gamma_5 = 7.240$, $\beta_5 = .811$		8	56.85	−0.44	7.98	8	0.984

time of 10 months for third release at 97% user's satisfaction level which is 2 months more than the real release time (8 months).

We observed that the release time problem based on the complexity of code change metric is more practical and gives a close prediction.

7. THREATS TO VALIDITY

Factors affecting the validity of proposed work are as follows:

Internal Validity: Some of the assumptions made by us may not always reflect the reality (e.g. leftover issues of n^{th} release may not have a larger severity when they are added in the initial issue content of $(n+1)^{th}$ release). Moreover, we have not empirically validated the real number of leftover issues.

External Validity: We have studied five different Apache open source products. Although, these open source products have good quality of code change history, our results may not generalize to all software products.

Construct Validity: We used the information available in GitHub repository, such as code change history for calculating entropy. This information we calculated manually, and may contain some manual errors. The total code changes in files resulting from fixing of different issues has been considered in our work. Statement level code changes need to be considered instead of file level changes. We have taken some versions collectively (versions have less data points) as a release. The selection of the releases is based on the equal data points for every release. We do not claim any causal findings for other choices of releases.

8. CONCLUSION

We estimated the potential value of different issues based on the time and the complexity of code changes (entropy) in different releases of Apache open source products. The issues left unresolved in different releases have also been calculated. Results show that in all the five products out of total 23 releases in 19 releases, at least 61.5% of potential issues have been addressed before the next release of the software. The leftover issue content contributes in the upcoming releases. The proposed model (case 6 of Table 1) performance has been compared with two models (Goel-Okumoto model and Yamada delayed S-shaped model). Out of 23 releases for 20 releases, we observed that the proposed entropy based model results in maximum cases of maximum R^2 in comparison of these two models.

A model for release time prediction using entropy is also proposed. We optimized the objective function of release time problem using Genetic algorithm in MATLAB. The estimated optimal release time is close to the real release time of jUDDI, Whirr, Hive, Pig and Avro products at 92, 97, 98, 98.5 and 99.7% satisfaction levels.

Table 11 Estimated parameter values and numerical results for Hive.

No. of release	Different models	Estimated parameters (a_i, b_i, φ_i and γ_i)	Real no. of bugs/issues fixed (A_i)	$a_i - A_i$ (Leftover bugs/issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	GO model	$a_1 = 807, b_1 = .070$	$A_1 = 442$	365	272.63	0.22	17.40	17.40	0.983
1	S-shaped model	$a_1 = 506, b_1 = .316$		64	246.24	4.19	15.94	16.48	0.984
1	Case 5	$a_1 = 474, \varphi_1 = .438, \gamma_1 = 9.900$		32	171.91	−0.85	13.79	13.82	0.989
1	Case 6	$a_1 = 474, \varphi_1 = .452,$ $\gamma_1 = 9.837, \beta_1 = .811$		32	175.29	−0.86	13.93	13.95	0.989
2	GO model	$a_2 = 526, b_2 = .097$	$A_2 = 457$	69	1527.73	−4.76	40.38	40.66	0.911
2	S-shaped model	$a_2 = 623, b_2 = .189$		166	360.76	6.73	18.49	19.67	0.979
2	Case 5	$a_2 = 574, \varphi_2 = .290, \gamma_2 = 11.866$		117	40.27	−0.46	6.59	6.60	0.998
2	Case 6	$a_2 = 580, \varphi_2 = .298,$ $\gamma_2 = 12.041, \beta_2 = .806$		123	38.98	−0.45	6.48	6.5	0.998
3	GO model	$a_3 = 588, b_3 = .078$	$A_3 = 453$	135	1790.54	2.67	43.95	44.04	0.907
3	S-shaped model	$a_3 = 664, b_3 = .179$		211	148.43	3.83	12.04	12.63	0.992
3	Case 5	$a_3 = 501, \varphi_3 = .372, \gamma_3 = 15.971$		48	25.76	−0.23	5.28	5.28	0.999
3	Case 6	$a_3 = 501, \varphi_3 = 0.387,$ $\gamma_3 = 15.877, \beta_3 = .808$		48	25.56	−0.23	5.26	5.26	0.999
4	GO model	$a_4 = 795, b_4 = .081$	$A_4 = 715$	80	5874.65	−4.00	79.67	79.77	0.858
4	S-shaped model	$a_4 = 1434, b_4 = .120$		719	723.74	8.66	26.51	27.89	0.983
4	Case 5	$a_4 = 1050, \varphi_4 = .271, \gamma_4 = 18.321$		335	339.05	−1.73	19.08	19.16	0.992
4	Case 6	$a_4 = 1073, \varphi_4 = .274,$ $\gamma_4 = 18.668, \beta_4 = .804$		358	333.76	−1.56	18.95	19.01	0.992
5	GO model	$a_5 = 1286, b_5 = .092$	$A_5 = 1066$	220	13904.41	0.31	122.73	122.73	0.996
5	S-shaped model	$a_5 = 1557, b_5 = .188$		491	385.22	−0.84	20.41	20.43	0.997
5	Case 5	$a_5 = 1127, \varphi_5 = .439, \gamma_5 = 20.303$		61	1092.66	−4.15	34.13	34.38	0.991
5	Case 6	$a_5 = 1127, \varphi_5 = .448,$ $\gamma_5 = 20.451, \beta_5 = .813$		61	1068.21	−4.11	33.75	34	0.991

Table 12 Estimated parameter values and numerical results for jUDDI.

No. of release	Different models	Estimated parameters (a_i, b_i, φ_i and γ_i)	Real no. of bugs/issues fixed (A_i)	$a_i - A_i$ (Leftover bugs/issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	GO model	$a_1 = 158, b_1 = .074$	$A_1 = 120$	38	157.15	−5.21	11.80	12.90	0.882
1	S-shaped model	$a_1 = 207, b_1 = .131$		87	34.67	1.80	5.80	6.08	0.974
1	Case 5	$a_1 = 167, \varphi_1 = .250, \gamma_1 = 15.127$		47	12.80	−0.10	3.70	3.70	0.990
1	Case 6	$a_1 = 168, \varphi_1 = .254,$ $\gamma_1 = 15.003, \beta_1 = .813$		48	13.034	−0.098	3.735	3.736	0.99
2	GO model	$a_2 = 81, b_2 = .081$	$A_2 = 75$	6	86.81	0.71	9.59	9.62	0.832
2	S-shaped model	$a_2 = 136, b_2 = .117$		61	22.03	1.44	4.61	4.83	0.957
2	Case 5	$a_2 = 113, \varphi_2 = .169, \gamma_2 = 6.052$		38	14.44	1.01	3.78	3.92	0.972
2	Case 6	$a_2 = 79, \varphi_2 = 0.411,$ $\gamma_2 = 31.944, \beta_2 = .805$		4	20.582	1.781	4.309	4.662	0.96
3	GO model	$a_3 = 22, b_3 = .530$	$A_3 = 23$	−1	5.52	0.20	2.43	2.44	0.469
3	S-shaped model	$a_3 = 21, b_3 = 1$		−2	10.08	0.49	3.25	3.29	.030
3	Case 5	$a_3 = 26, \varphi_3 = .166, \gamma_3 = 1.064$		3	0.61	0.00	0.81	0.81	0.941
3	Case 6	$a_3 = 31, \varphi_3 = .199,$ $\gamma_3 = 1.385, \beta_3 = .802$		8	20.582	1.781	4.309	4.662	0.96
4	GO model	$a_4 = 194, b_4 = .052$	$A_4 = 169$	25	603.44	−4.07	25.08	25.40	0.758
4	S-shaped model	$a_4 = 196, b_4 = .136$		27	248.69	0.36	16.32	16.32	0.900
4	Case 5	$a_4 = 290, \varphi_4 = .271, \gamma_4 = 48.602$		121	69.80	−1.05	8.58	8.64	0.972
4	Case 6	$a_4 = 273, \varphi_4 = .284,$ $\gamma_4 = 31.088, \beta_4 = .811$		104	0.181	0	0.425	0.425	0.983

Table 13 Estimated parameter values and numerical results for Whirr.

No. of release	Different models	Estimated parameters (a_i, b_i, φ_i and γ_i)	Real no. of bugs/issues fixed (A_i)	$a_i - A_i$ (Leftover bugs/issues of i^{th} release added to $(i+1)^{th}$ release)	MSE_i	$Bias_i$	VAR_i	$RMSPE_i$	R_i^2
1	GO model	$a_1 = 395, b_1 = .054$	$A_1 = 176$	219	172.28	−1.70	13.80	13.91	0.938
1	S-shaped model	$a_1 = 290, b_1 = .221$		114	50.05	1.85	7.24	7.48	0.982
1	Case 5	$a_1 = 219, \varphi_1 = .449, \gamma_1 = 15.338$		43	30.85	−0.22	5.89	5.89	0.989
1	Case 6	$a_1 = 218, \varphi_1 = .462,$ $\gamma_1 = 15.300, \beta_1 = .803$		42	9.556	1.7	2.738	3.223	0.989
2	GO model	$a_2 = 188, b_2 = .086$	$A_2 = 94$	94	28.57	−0.31	5.70	5.71	0.960
2	S-shaped model	$a_2 = 100, b_2 = .478$		6	27.65	0.64	5.58	5.62	0.962
2	Case 5	$a_2 = 94, \varphi_2 = .681, \gamma_2 = 10.163$		0	22.75	−0.03	5.10	5.10	0.968
2	Case 6	$a_2 = 103, \varphi_2 = .558,$ $\gamma_2 = 7.887, \beta_2 = .815$		9	44.216	0.485	7.089	7.106	0.968
3	GO model	$a_3 = 81, b_3 = .189$	$A_3 = 72$	9	61.76	1.07	8.32	8.39	0.730
3	S-shaped model	$a_3 = 58, b_3 = .734$		−14	103.06	1.54	10.73	10.84	0.549
3	Case 5	$a_3 = 78, \varphi_3 = .373, \gamma_3 = 4.158$		6	21.85	0.24	4.99	5.00	0.815
3	Case 6	$a_3 = 80, \varphi_3 = .360,$ $\gamma_3 = 3.539, \beta_3 = .802$		8	11.706	0.745	7.048	7.087	0.855
4	GO model	$a_4 = 40, b_4 = .123$	$A_4 = 30$	10	19.08	0.05	4.79	4.79	0.699
4	S-shaped model	$a_4 = 109, b_4 = .157$		79	15.26	1.29	4.04	4.24	0.746
4	Case 5	$a_4 = 61, \varphi_4 = .245, \gamma_4 = 8.109$		31	16.67	−0.36	4.46	4.47	0.772
4	Case 6	$a_4 = 42, \varphi_4 = .360,$ $\gamma_4 = 10.886, \beta_4 = .801$		12	8.939	0.031	3.275	3.275	0.851

Table 14 Goodness of Fit Test Using Kolmogorov-Smirnov (K-S) Test.

Product	Release no.	D-Value	Associated P-Value
Avro	1	0.0909	1.0000
	2	0.1818	0.985
	3	0.1538	0.995
	4	0.0769	1.0000
	5	0.1818	0.985
Pig	1	0.0909	1.0000
	2	0.0769	1.0000
	3	0.0833	1.0000
	4	0.1000	1.0000
	5	0.1000	1.0000
Hive	1	0.1000	1.0000
	2	0.0769	1.0000
	3	0.0769	1.0000
	4	0.1538	0.995
	5	0.1538	0.995
jUDDI	1	0.1333	0.998
	2	0.0625	0.999
	3	0.1429	0.997
	4	0.2667	0.589
Whirr	1	0.1000	1.000
	2	0.2000	0.975
	3	0.7000	0.700
	4	0.6000	0.031

Table 15 Parameters of GA.

Parameter size	Population generations	Number of probability	Cross over function	Mutation
Value	100	150	0.8	Adaptive feasible

REFERENCES

1. K.K. Chaturvedi, P. Singh and V.B. Singh, "Tools in Mining Software Repositories," in *CPS-IEEE Proceedings of the 13th International Conference on Computational Science and Its Applications* (ICCSA 2013), pp. 89-98.
2. A.E. Hassan, "Predicting faults based on complexity of code change," *in Proceedings of the International Conference on Software Engineering* (ICSE 2009), pp. 78-88.
3. K.K. Chaturvedi, P.K. Kapur, S. Anand and V.B. Singh, "Predicting the complexity of code changes using entropy based measures," *International Journal of System Assurance Engineering and Management*, Springer, vol. 5, pp.155-164.
4. P. K. Kapur and R. B. Garg, "Optimal release policies for software systems with testing effort," *International J. System Science*, vol. 22, no. 9, pp. 1563–1571, 1991.
5. C. Y. Huang and M. R. Lyu, "Optimal release time for software systems considering cost, testing effort, and testing efficiency," *IEEE Transactions on Reliability*, vol. 54, no. 4, Dec. 2005.
6. S. Inoue and S. Yamada, "Two-dimensional software reliability measurement technologies," in *Proceedings of IEEE IEEM*, pp. 223-227, 2009.
7. P. K. Kapur, H. Pham, A. G. Aggarwal and G. Kaur "Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning," *IEEE Transactions on reliability*, vol. 61, no. 3, pp. 758-768, September 2012
8. R. Günther and O.S. Moshood, 'The Art and Science of Software Release Planning," *IEEE Software*, 22(6), pp. 47 – 53, 2005.
9. X.O. Wei and Z. Huang (2012), "Evolutionary Approaches For Multi-Objective Next Release Problem," *Computing And Informatics*, vol. 31, no. 4, pp. 847–875, 2012.

10. https://issues.apache.org/
11. https://github.com/
12. C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, 27: pp. 379–423,623–656, Jul., Oct. 1948.
13. H.R. Varian, *Intermediate Microeconomics —A Modern Approach*. 2nd ed., W.W. Norton & Company, New York, 1991.
14. H. K. Wright, "Release engineering processes, models, and metrics," *The doctoral symposium for ESEC/FSE*, ACM, pp. 27-28, 2009.
15. http://www.wolfram.com/mathematica/
16. W. Farr, *Handbook of Software Reliability Engineering*, M. R. Lyu, Editor, chapter Software Reliability Modeling Survey, pp. 71–117, 1996. McGraw-Hill, New York, NY.
17. M. Obha, "Software reliability analysis models," *IBM Journal of Research and Development*, vol. 28, no. 4, pp. 428–443, 1984.
18. H. Pham*, System Software Reliability*.: Springer, 2006.
19. P. K. Kapur, H. Pham, A. Gupta and P. C. Jha*, Software Reliability Assessment with OR Applications*. London, U.K.: Springer, 2011.
20. M. Xie, *Software Reliability Modeling*.: World Scientific, 1999.
21. G. Ruhe, *Handbook of Software Engineering and Knowledge Engineering,* S. K. Chang, Editor, chapter Software release planning, pp. 365-393, 2005. Vol. 3, World Scientific.
22. H.K. Wright and D.E. Perry, "Subversion 1.5: A Case Study in Open Source Release Mismanagement," in *Proceedings of 2nd FLOSS Workshop*, pp. 13–18, 2009.
23. H.K. Wright, "Release Engineering Processes, Their Faults and Fail- ures," PhD dissertation, University of Texas at Austin, 2012.
24. H.K. Wright and D.E. Perry, "Release Engineering Practices and Pitfalls," in *Proceedings of International Conference Software Engineering* (ICSE 12), pp. 1281–1284, 2012.
25. M. Michlmayr, B. Fitzgerald and K. Stol, " Why and How Should Open Source Projects Adopt Time-Based Releases?," *IEEE Software*, pp. 55-63, 2015.
26. S.S. Gokhale and K.S. Trivedi, "Log-Logistic Software Reliability Growth Model," in Proceedings of Third IEEE International High-Assurance Systems Engineering Symposium, pp. 34-41, 1998.
27. D. Spinellis, "The Strategic Importance of Release Engineering', IEEE software, Volume 32 issue 2, pp 3-5, 2015.
28. D. E. Goldberg, "Genetic Algorithms in Search of Optimization and Machine Learning," 1989, New York: Addison-Wesley.
29. J. E. Yang, M. J. Hwang, T. Y. Sung, and Y. Jin, "Application of genetic algorithm for reliability allocation in nuclear power plants," Reliability Engineering Systems Safety, 1999, vol. 65, no. 3, pp. 229–238.
30. L. Painton and J. Campbell, "Genetic algorithms in optimization of system reliability," IEEE Trans. on Reliability, 1995, vol. 44, no. 2, pp. 172–178.
31. P. Roy, G. S. Mahapatra and K.N. Dey, "Neuro-genetic approach on logistic model based software reliability prediction," *Expert Systems With Applications*, 2015, Vol. 42 Issue: 10, pp. 4709-4718.
32. F. Yue, G. Zhang, Z. Su and T. Zhang, "Multi-software reliability allocation in multimedia systems with budget constraints using Dempster-Shafer theory and improved differential evolution," *Neurocomputing*, 2015, Vol. 169, Special Issue: SI, pp. 13-22.
33. R. Peng, Li. Yan-Fu, Jun-G. Zhan and Li. Xiang, "A risk-reduction approach for optimal software release time determination with the delay incurred cost," *International Journal Of Systems Science*, 2015, Vol. 46, Issue: 9 pp. 1628-1637.
34. J. Park and J. Baik, "Improving software reliability prediction through multi-criteria based dynamic model selection and combination," *Journal of Systems And Software*, 2015, Vol.101, pp. 236-244.
35. J. Yang, Y. Liu, M. Xie and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes," *Journal of Systems and Software*, 2016, Vol. 115, pp. 102-110.
36. P.K. Kapur, A. Tandon and G. Kaur, "Multi up-gradation software reliability model," *in the proceedings of 2nd International conference on reliability, safety & hazard (ICRESH-2010),* pp 468–474.
37. A. L. Goel, K. Okumoto, "Time-dependent error detection rate model for software reliability and other performance measures," IEEE Trans. Rel., R-28(3), pp. 206-211, August 1979.
38. S. Yamada, M. Ohba and S. Osaki, "S-Shaped Reliability Growth Modeling for Software Error Detection," IEEE Transactions on Reliability, R-32, 5, pp. 475-478, December 1983.
39. https://bugzilla.mozilla.org/
40. https://bugzilla.gnome.org/
41. [IEEE88] "IEEE Standard Dictionary of Measures to Produce Reliable Software", IEEE Std 982.1-1988, Institute of Electrical and Electronics Engineers, 1989.
42. Li, Xiang, Li, Y. Fu, Xie, M. and Ng, S. H., "Reliability analysis and optimal version-updating for open source software," *Information and Software Technology*, 2011, Vol. 53(9), pp. 929-936.
43. I. Samoladas, L. Angelis and I. Stamelos, "Survival analysis on the duration of open source projects," *Information and Software Technology*, 2010, Vol. 52, pp. 902–922.
44. T. Zhang, J. Chen, G. Yang, B. Lee and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," *Journal of Systems and Software*, 2016, Vol. 117, pp. 166-184.
45. G. Ruhe and D. Greer, "Quantitative Studies in Software Release Planning under Risk and Resource Constraints," In Proceedings of the International Symposium on Empirical Software Engineering (ISESE'03). 2003, IEEE, Rome, Italy, pp. 262–270.
46. G. Ruhe and A. Ngo-The, "Hybrid Intelligence in Software Release Planning," *International Journal of Hybrid Intelligent Systems*, 2004, 1-2, pp. 99–110.
47. D. Grees and G. Ruhe, "Software Release Planning: An Evolutionary and Iterative Approach," *Information & Software Technology*, 2004, Vol. 46, 4, pp. 243–253.
48. G. Cristina and A. Budi, "The Many Meanings of Open Source," IEEE Software, 2004, Vol. 21, no. 1, pp. 34-40, January 2004, doi 10.1109/MS.2004.1259206.
49. M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, M.U. Shafique, "A systematic review on strategic release planning models," *Information and Software Technology,* 2010, Vol. 52, pp. 237–248.
50. G. Ruhe, Product Release Planning: Methods, Tools, and Applications. Auerbach Publications, 2010.
51. S. Chawla, S. Srivastava and P. Bedi, "Goal and Scenario based Web Requirements Engineering," International Journal of Computer Systems Science and Engineering, 2016, Vol. 31,1.
52. K.H. Nehemiah, A. Gladston, and K. Arputharaj, "An optimal tabu prioritization algorithm for regression testing,"2016, Vol. 31, pp.385-392.
53. V. B., Singh, M., Sharma and H. Pham, "Entropy Based Software Reliability Analysis of Multi-Version Open Source Software," *IEEE Trans. Software Eng.*, 2018, Vol. 44, 12, pp. 1207–1223.
54. P. K. Kapur, R. B. Garg, A. G. Aggarwal, and A. Tandon, "Two dimensional flexible software reliability growth model and related release policy," in Proceedings of the 4th National Conference; INDIACom-2010, February 25âĽ"26, 2010.