An Investigation of Entropy Based Models for Measuring Reliability Growth of Open Source Software Projects and Related Release Time Planning

Kamlesh Kumar Raghuvanshi[1], Anil Rajput[2], Meera Sharma[3]
[1]*Ramanujan College, University of Delhi, Delhi, India*
[2]*Chandra Shekhar Azad Government P.G. Nodal College, Sehore, MP, India*
[3]*Swami Shraddhanand College, University of Delhi, Delhi, India*

Abstract **-** Open Source Software (OSS) projects have wide applications in different domains. Software Reliability Growth Models (SRGMs) are proven to successful in the reliability growth measurement of close source projects. In this paper, we have investigated the applicability of SRGMs in measuring the reliability growth of open source projects. During bug fixing process, software source code need to be changed. As a result of these code changes, uncertainty increases in the software and a measure of this is called entropy. In this paper, entropy based models have been used to predict the potential number of bugs lying dormant in the software. The performance of the models has been evaluated on the basis of Rsquared (R^2), Mean Squared Error (MSE) and Sum of Squared Error (SSE). The goodness of fit has been tested for Avro product of Apache open source project.

Keywords Entropy, Software reliability, Bugs, Open Source Software

Notations

$m(t)$	Expected number of software failures detected at time t, also called the mean value function
N	Expected number of initial faults that exist in the software before testing
$b(t)$	Time dependent logistic function per quantity per unit time (i.e., fault detection rate per fault per unit of time).
β	Constant
b	Constant
$H(t)$	Entropy at time t

I. INTRODUCTION

Due to automation of different activities and penetration of information technology, the quality of software products can not compromised. It is important to understand the software quality and also to measure it. The quantitative quality evaluation of the software products is an important aspect of software development life cycle. However, the software development process can be time consuming and expensive this is because of the complexity of software systems. Enhancing the reliability of software systems and reducing cost to acceptable levels have became the main focus of the software industry [1]. Meanwhile, many SRGMs have been proposed and research has long been performed in software reliability.

The quantitative quality evaluation based software reliability growth models [1-15] have been developed to estimate reliability metrics such as the number of residual faults, failure rate and reliability of software. Over the past three decades many SRGMs have been formulated under Non Homogeneous Poisson Process (NHPP) modeling framework. Many researchers have used NHPP based SRGMs to capture the reliability growth of a software from the processes of testing and debugging. These models include Goel and Okumoto model [2] which describes the fault detection rate, as an NHPP assuming the hazard rate is proportional to remaining fault number. Ohba and Yamada [5] proposed the delayed s-shaped SRGM and the inflection s-shaped SRGMs describe an error detection process in which the detectability of an error increases with the progress of software testing. Pham et. al [6] proposed PNZ-SRGM that accounts the impact of both the imperfect debugging and the learning effects. Pham et. al [7] discussed an imperfect debugging fault detection dependent-parameter model with a common parameter metric used in both the fault content rate function and fault detection rate function. In the area of software reliability modeling, many models [1-3, 5-7, 15-17] have been developed in the past three decades based on logistic growth function to estimate the reliability metrics such as the number of residual faults, failure rate and reliability of software. In this paper, we have discussed the models proposed in [23] and then proposed entropy based models to predict the potential bugs over a long run in the software. The models given in Table 1 have been discussed in [23].

Table 1. Existing software growth logistic models

Software Growth Logistic Models	
Goel-Okumoto(GO) [2]	$m(t) = a(1 - e^{-bt})$
Delayed S-shaped [5]	$m(t) = a(1 - (1 + bt)e^{-bt})$
Pham Dependent-parameter[7]	$m(t) = \alpha(1 + \gamma t)(\gamma t + e^{-\gamma t} - 1)$
Inflexion S-shaped[4]	$m(t) = \frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$

II. DATA COLLECTION AND MODEL BUILDING

The proposed mathematical models have been validated on dataset of Avro product of Apache open source project [18]. The time period of data collection is from April 2009 to April 2014. Figure 1 shows sample of different feature reports (bugs, new features and feature improvements) for Avro product.

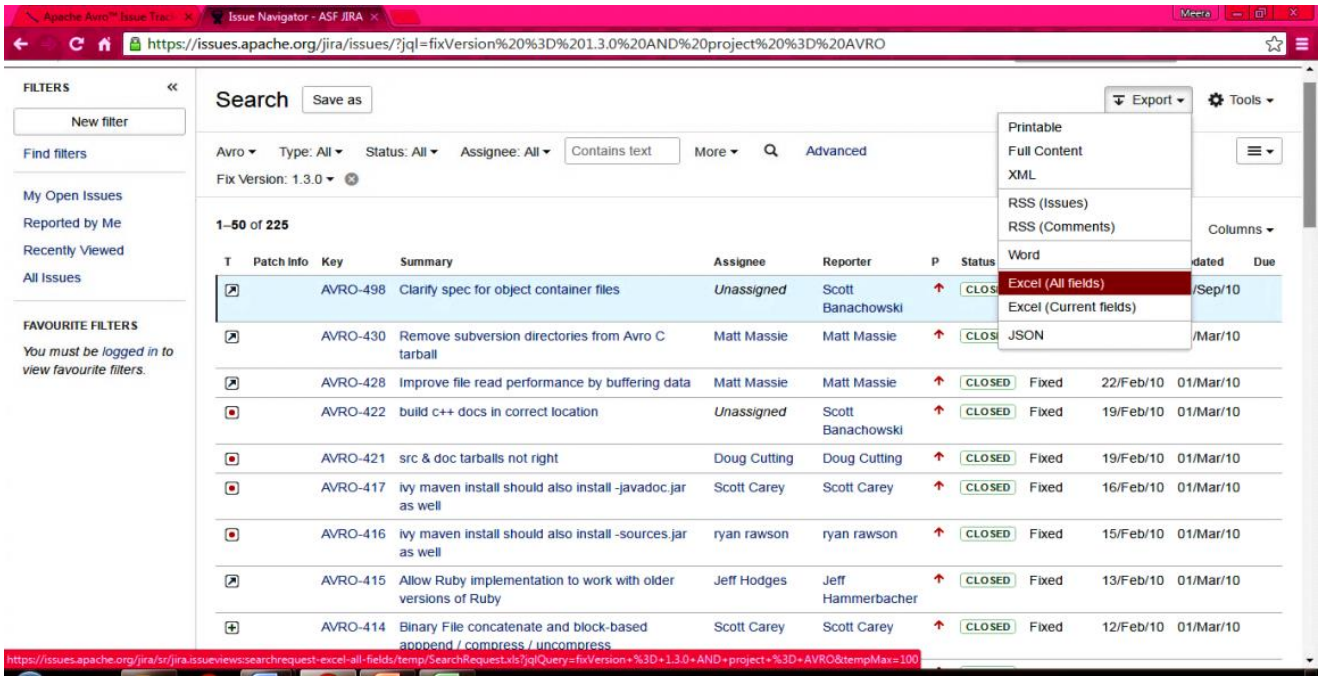

Figure 1. Sample features (bugs, new features and feature improvements) reports for Avro product

GitHub tool [19] has been used to collect the code change history of Avro product. We have downloaded fixed bugs on monthly basis. Entropy calculation has been discussed below.
In [20], the authors have defined Shannon entropy as

$$H_n(P) = -\sum_{k=1}^{n}(p_k * \log_2 p_k) \quad \text{where} \quad p_k \geq 0$$

$$\sum_{k=1}^{n} p_k = 1$$

where, p_k is the probability of change occurrence and defined as the ratio of the number of times kth file changed during a period and the total number of changes for all files in that period. Normalized Static Entropy, H is defined in [21] as

$$H(P) = \frac{1}{Max\ Entropy\ for\ Distribution} * H_n(P)$$

$$= \frac{1}{\log_2 n} * H_n(P) = -\frac{1}{\log_2 n} * \sum_{k=1}^{n}(p_k * \log_2 p_k)$$

$$= -\sum_{k=1}^{n}(p_k * \log_n p_k)$$

$$\text{where } p_k \geq 0, \forall\ k \in 1, 2, \ldots\ldots, n \text{ and } \sum_{k=1}^{n} p_k = 1.$$

The entropy, H, depends on the number of files, i.e. 'n' in a software system. To avoid the rarely modified files from

reducing the entropy measure, we have divided by the number of recently modified files rather than the actual current number of files [21, 22]. Table 2 Show the monthly data of Avro product of Apache open source project, which has been used to validate the proposed models.

Table 2. Avro Dataset

Time (month)	Bugs	Entropy
1	7	.9767
2	10	.9768
3	11	.9744
4	10	.9718
5	17	.9662
6	8	.9430
7	26	.9413
8	18	.9724
9	9	.9533
10	14	.9558
11	4	.9806
12	5	.9656
13	4	.9617
14	17	.9507
15	7	.9924
16	5	.9902
17	4	.9695
18	9	.9874
19	16	.9382
20	7	.9641
21	4	.9939
22	3	1
23	6	.9902
24	4	.9808
25	6	.9859
26	1	.9832
27	3	.9856
28	27	.9692
29	3	1
30	6	.9867
31	8	.9823
32	14	.9837
33	6	1
34	3	.9971
35	25	.9616
36	6	.9647
37	5	1
38	3	1
39	20	.9694
40	1	1
41	11	.9794
42	3	1
43	4	.9609
44	13	.9906
45	1	.9859
46	6	.9862
47	6	.9884
48	6	.9672
49	2	.9844
50	4	1
51	1	1
52	0	.9848
53	9	.9935
54	11	.9901
55	1	1
56	5	.9165
57	0	1
58	3	1
59	6	1
60	5	1

Figure 2 shows the monthly bug distribution for Avro product.

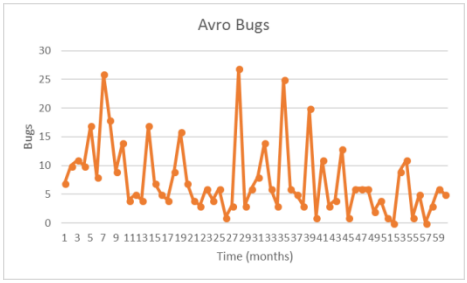

Figure 2. Monthly bug distribution of Avro

Figure 3 shows the monthly entropy distribution for Avro product.

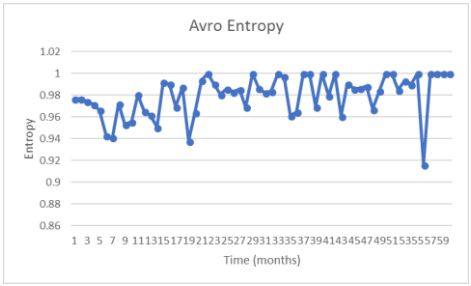

Figure 3. Monthly bug distribution of Avro

Many existing reliability models [1-15] assume that the rate of change of the number of software faults is proportional to the residual fault contents. Let $m(t)$ denote the number of faults at time t also called the mean value function. Assume that the rate of change of quantity function $m(t)$ is directly proportional to its remaining quantity for growth by the time-dependent three-parameter logistic function per quantity per unit time $b(t)$. We can obtain the differential equation as follows:

$$\frac{dm(t)}{dt} = b(t)[N - m(t)]$$

At time $t = 0$, $m(0) = 0$ then the solution for the function $m(t)$ can be obtained by solving the above differential equation :

$$m(t) = \left[1 - e^{-\int_0^t b(x)dx}\right]$$

By considering the entropy in modeling for reliability growth of the software products, we can derive the following models given in Table 3.

Table 3. Entropy based software reliability growth models

Entropy based Software Reliability Growth Models	
Model-1	$m\left(H\left(t\right)\right) = a(1-\exp\left(-b\left(H\left(t\right)\right)\right)$
Model-2	$m\left(H\left(t\right)\right) = a(1-\left(1+b\left(H\left(t\right)\right)\right)\exp\left(-b\left(H\left(t\right)\right)\right)$
Model-3	$m\left(H\left(t\right)\right) = \alpha(1+\gamma\left(H\left(t\right)\right)\left(\gamma\left(H\left(t\right)\right)+\exp\left(-\gamma H\left(t\right)\right)-1\right)$
Model-4	$m\left(H\left(t\right)\right) = \dfrac{a(1-\exp\left(-b\left(H\left(t\right)\right)\right)}{1+\beta\left(\exp\left(-b\left(H\left(t\right)\right)\right)\right)}$

In this paper, we have validated the new entropy based models using Avro failure data based on some common goodness of fit criteria such as R^2, Mean Squared Error (MSE), Sum of Squared Error (SSE).

III. MODEL COMPARISON

To compare the goodness of fit for all models, we have used performance measures, namely Sum of Squared Error (SSE) and Mean Squared Error (MSE) as described below.

Sum of Squared Error (SSE)

The criterion we use to judge the performance of the models is the sum of squared error (SSE), which sum up the squares of the residuals of the actual data and the mean value function $m(t_k)$ of each model in terms of the number of actual faults at any time points.

The SSE function can be expressed as follows:

$$\text{SSE} = \sum_{k=1}^{n} [y_k - \hat{\boldsymbol{m}}(t_k)]^2$$

where y_k is the total number of faults observed at time t_k according to the testing data and $m(t_k)$ is the estimated cumulative number of faults at time t_k obtained from the fitted mean value function, $k = 1, 2 \ldots n$.

Mean Squared Error (MSE)

The Mean Squared Error (MSE) refers to the mean value of the deviation between the prediction value and the observation value as follows:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n} (\hat{m}(t_i) - y_i)^2}{n - N}$$

where y_i is the total number of faults observed at time t_i according to the testing data and $m(t_k)$ is the estimated cumulative number of faults at time t_i obtained from the. fitted mean value function, n and N are the number of observations and the number of parameters, respectively.

IV. MODEL RESULTS

The entropy based models mentioned in Table 3 have been evaluated using given Avro dataset (Table 2). Various comparison criteria, namely MSE, SSE and R^2 have been used to validate the models. Table 4 presents the values of the estimated parameters of the models mentioned in the Table 3.

Table 4. Estimates of different parameters for entropy based models

Models	Parameter estimates
Model-1	a=944.787, b=0.012
Model-2	a=501.462, b=0.065
Model-3	α=431, γ=0.023
Model-4	a=639.492, b=0.03, β=1.000

In Table 5 we have given the values for different performance measures of the models mentioned in the Table 3.

Table 5. Estimation for the goodness of fit of entropy based models

Models	MSE	SSE	R^2
Model-1	115.349	7304.349	0.994
Model-2	346.697	21841.92	0.983
Model-3	11839.253	745872.985	0.475
Model-4	103.923	6547.181	0.995

From the above table we observed that R^2 for entropy based models are giving significant values except for Pham Dependent-parameter model where R^2 is 0.475.
Figures 4 to 7 show goodness of fit curves for different entropy based models given in Table 3.

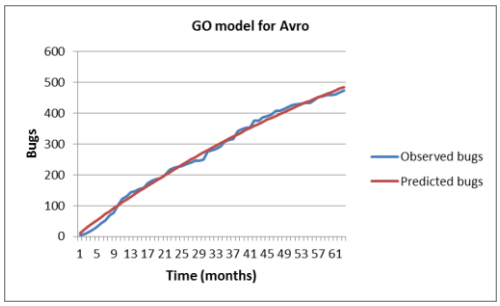

Figure 4. Goodness of fit curve for GO Model

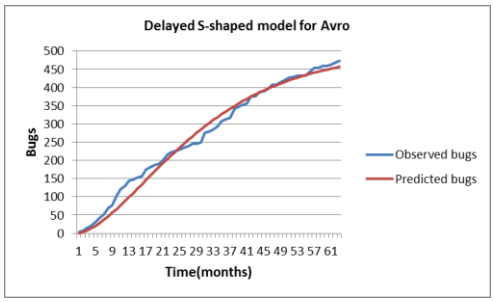

Figure 5. Goodness of fit curve for Delay S-shaped model

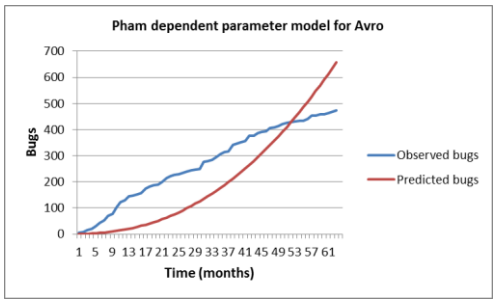

Figure 6. Goodness of fit curve for Pham dependent parameter model

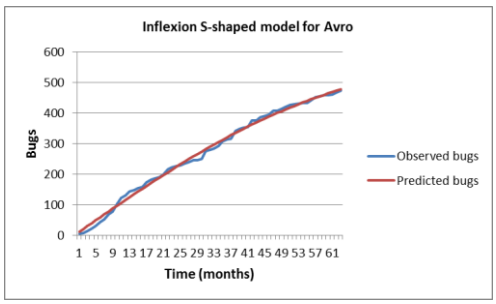

Figure 7. Goodness of fit curve for inflexion S-Shaped Model

We observed that the predicted bug values are close to the observed values for bug.

Modeling for Unresolved Bugs for Different Releases

Let '*a*' denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can also be written as (Musa et al. [24]):

$$X(t) = aF(t)$$

where $F(t)$ is a distribution function.

Each time a failure is observed, an immediate debugging effort takes place to find the cause of the failure in order to remove it; the instantaneous failure intensity $\lambda(t)$ in case of the finite failure NHPP models is given by:

$$\lambda(t) = aF'(t)$$

The above equation can be rewritten as:

$$\lambda(t) = [a - X(t)]\frac{F'(t)}{1-F(t)} = [a - X(t)]s(t)$$

Where $s(t)$ is the failure occurrence/observation/detection rate per remaining fault of the software, or the rate at which the individual faults manifest themselves as failures during testing or hazard rate. $[a - X(t)]$ denotes the expected number of faults remaining in the software at time *t*. Since $[a - X(t)]$ is monotonically non-increasing function of time, the nature of the overall failure intensity, $\lambda(t)$, is governed by the nature of failure occurrence rate per fault $s(t)$.

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. 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

$$X_1(t) = a_1 F(t) \qquad 0 < t < t_1$$

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

$$X_2(t) = a_2 F_2(t - t_1) + a_1(1 - F_1(t_1))F_2(t - t_1), \; t_1 < t < t_2$$

In above equation a_2 are 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.

$$X_3(t) = a_3 F_3(t - t_2) + a_2(1 - F_2(t_2))F_3(t - t_2) + a_1(1 - F_1(t_1))(1 - F_2(t_2))F_3(t - t_2), \quad t_2 < t < t_3$$

$$X_4(t) = a_4 F_4(t - t_3) + a_3(1 - F_3(t_3))F_4(t - t_3) + a_2(1 - F_2(t_2))(1 - F_3(t_3))F_4(t - t_3) + a_1(1 - F_1(t_1))(1 - F_2(t_2))(1 - F_3(t_3))F_4(t - t_3), \quad t_3 < t < t_4$$

V. CONCLUSION

Software reliability growth models are proven to be very useful in measuring the reliability growth of the software products. The open source software products are also getting an edge over the closed source software. In this paper an empirical validation of entropy based and without entropy based approach has been discussed. The estimated parameters have been also presented in the paper with their performance measure. The models presented in the paper are quite useful for real world applications. The presented models will be useful in measuring the reliability growth of the software products.

VI. REFERENCES

[1] Garg, R., Sharma, K., Kumar, R., & Garg, R. K. (2010). Performance analysis of software reliability models using matrix method. *World Academy of Science, Engineering and Technology*, *71*, 31-38.

[2] Goel, A. L., & Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, *28*(3), 206-211.

[3] Yamada, S., Ohba, M., & Osaki, S. (1983). S-shaped reliability growth modeling for software error detection. *IEEE Transactions on reliability*, *32*(5), 475-484.

[4] Osaki, S., & Hatoyama, Y. (1984). Inflexion S-shaped software reliability growth models. In *Stochastic Models in Reliability Theory* (pp. 144-162). Springer-Verlag Merlin.

[5] Ohba, M., & Yamada, S. (1984). S-shaped software reliability growth models. In *International Colloquium on Reliability and Maintainability, 4th, Tregastel, France* (pp. 430-436).

[6] Pham, H., Nordmann, L., & Zhang, Z. (1999). A general imperfect-software-debugging model with S-shaped fault-

detection rate. *IEEE Transactions on reliability*, *48*(2), 169-175.

[7] Pham, H. (2007). An imperfect-debugging fault-detection dependent-parameter software. *International Journal of Automation and Computing*, *4*(4), 325.

[8] Kapur, P. K., Pham, H., Anand, S., & Yadav, K. (2011). 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.

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

[10] Keilman, N. (2001). Demography: uncertain population forecasts. *Nature*, *412*(6846), 490.

[11] Persona, A., Pham, H., & Sgarbossa, F. (2010). Age replacement policy in a random environment using systemability. *International Journal of Systems Science*, *41*(11), 1383-1397.

[12] Pham, H. (1993). Software reliability assessment: imperfect debugging and multiple failure types in software development. *EG&GRAMM-10737, Idaho National Engineering Laboratory*.

[13] Pham, H. (1996). A software cost model with imperfect debugging, random life cycle and penalty cost. *International Journal of Systems Science*, *27*(5), 455-463.

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

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

[16] Pham, H., & Deng, C. (2003, August). Predictive-ratio risk criterion for selecting software reliability models. In *Proceedings of the 9th International Conference on Reliability and Quality in Design* (pp. 17-21).

[17] Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *European Journal of Operational Research*, *145*(2), 443-454.

[18] http://www.apache.org/

[19] https://github.com/

[20] Shannon, C. E. (1948). A mathematical theory of communication, bell System technical Journal 27: pp. 379–423 and 623–656. Mathematical Reviews (MathSciNet): MR10, 133e.

[21] Hassan, A. E. (2009, May). Predicting faults using the complexity of code changes. In *Proceedings of the 31st International Conference on Software Engineering* (pp. 78-88). IEEE Computer Society.

[22] Chaturvedi, K. K., Kapur, P. K., Anand, S., & Singh, V. B. (2014). Predicting the complexity of code changes using entropy based measures. *International Journal of System Assurance Engineering and Management*, *5*(2), 155-164.

[23] Pham, H., Pham, D. H., & Pham Jr, H. (2014). A new mathematical logistic model and its applications. *International Journal of Information and Management Sciences*, *25*(2), 79-99.

[24] John D.. Musa, Iannino, A., & Okumoto, K. (1987). *Software reliability: measurement, prediction, application*. McGraw-Hill.