International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

SDE based Unified Scheme for Developing Entropy Prediction Models for OSS

Deepika
Department of Operational Research,
University of Delhi, Delhi, India.
E-mail: deepika.sre@gmail.com

Ompal Singh
Department of Operational Research,
University of Delhi, Delhi, India.
E-mail: drompalsingh1@gmail.com

Adarsh Anand
Department of Operational Research,
University of Delhi, Delhi, India.
Corresponding author: adarsh.anand86@gmail.com

Jagvinder Singh
University School of Management and Entrepreneurship,
Delhi Technological University, Delhi, India.
E-mail: jagvinder.singh@dtu.ac.in

(Received January 13, 2020; Accepted April 30, 2020)

Abstract
Today, so as to meet the user's requirement, modification of software is necessarily required. But at the same time, to incorporate these modifications and requirements there are enormous changes which are made to the coding of the software and over a period of time these changes make the software complex. Largely there are three types of code changes occur in the source code namely, bug repair, feature enhancement & addition of new features, but these changes bring the uncertainty in the bug removal rate. In this paper, these uncertainties have been explicitly modeled and using three-dimensional wiener processes that define the three types of fluctuation; we have come up with an entropy prediction modeling framework with a unified approach. The analytical solution of the equation is interpreted using Itô's process. The models are fitted on three real life projects namely Avro, Hive and Pig of Apache open source software (OSS) The experimental findings show that present models exhibit accurate estimation results and have strong prediction skills.

Keywords- Complexity of code changes (CCC), Differential equation (DE), Distribution, Entropy, Fluctuations, Open source software (OSS), Unification.

1. Introduction

Software's role in smooth functioning and productivity enhancement of almost every sector cannot be denied. "The advancements in computing capabilities of devices have proved to be one of the significant reasons for the introduction of software to time critical systems. OSS has brought major breakthroughs in development trends for large scale software systems. For such software, the prototype is developed and is evolved with voluntary contribution across the globe. The development pattern followed by OSS doesn't rigidly comply with standards described under the traditional software engineering lifecycle (Bhatt et al., 2017). With the enormous rise in

software demand, IT industry has become highly competitive. Quality is the most critical characteristic that ensures the sustainability of software in market and reliability is an important metric to quantify the system's quality". In past decades numerous researches have been carried out in the direction of reliability modeling and assessment (Kapur et al., 2011a).

Open Source Software (OSS) varies characteristically from closed source software. Features such as irregular volunteer participation, lack of hierarchical management, no delivery schedules, and lack of rigid SDLC principles etc. make it different from closed source systems. OSS development is initiated with an idea or concept and a working prototype is worked out by few developers or a small team. "The core system with limited functionality is then released over the internet along with source code for further enhancement and refinement with global volunteer participation. The most distinguishing feature over closed source software is that this is released with very little functionality testing and therefore it is the user population who adopts such systems become the deciding factor in reliability growth phenomenon for such software. The OSS keeps on evolving due to the contributions from volunteers across the world. The software is developed by the core team with few members and released over the internet".

"The OSS community generally follows a layered architecture which describes the users into various categories like core members, active developers, peripheral developers, bug fixer, bug reporter, passive users etc. from the core to the periphery. The users are assigned participation levels which may change with time and contributions. People start working on software and when they find some glitch in software workflow, they report it as a failure and the fault is identified". After fault identification developers try to remove the fault.

Software source codes are enhanced to satisfy the user's needs. "The huge changes in the code make the software complex over a period of time. These changes increase the complexity of the code which also leads to the introduction of the bugs. The changes are being made continuously in the software to remove the bugs, enhance the features and implement the new functionalities (Singh and Chaturvedi, 2013; Deepika et al., 2017). The open source software is being built through the contributors from diverse communities and keeps on moving rapidly. The code complexity is one of the major attributes which determines the quality and reliability of the software. Many contributors are making changes in the source code of the software to produce the quality of the software in a limited time. It is getting difficult to remember all those changes committed in the code which makes the source code complex. The contributors are interacting with each other through the discussion forum. Three types of code changes occur in the source code namely, bug repair, features enhancement and addition of new features. Bugs are generated in the software mainly due to miscommunication or no communication among active users, frequently changing requirements, early release pressures, the occurrence of programming errors, increase software complexity".

The remainder of the study is structured into the following sections: In Section 2, a detailed review of past literature is presented. Section 3 describes the modeling framework and also notations and assumptions are used for formulating the models. Following that, in Section 4, the parameter estimation and comparison criteria results of the proposed models are produced. Finally, in the next section, we summarize the key results and provide the concluding remarks which are then followed by the references.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

2. Literature Review

Numerous studies have been carried out for assessing the reliability of OSS systems. Tamura and Yamada (2008) investigated reliability assessment combining it with neural networks. Some of the latest SRGMs proposed to assess the reliability growth of software systems incorporating practical phenomena are Chatterjee and Shukla (2016), Li and Pham (2017), Zhu and Pham (2017). Stochastic differential equation-based models have been proposed for OSS and further discussed its application in the optimal version update problem by Tamura et al. (2008). Li et al. (2011) presented another model to predict optimal version update time for OSS. Yang et al. (2016) included the process of fault detection and correction and performed reliability modeling for multi-release OSS. Many other studies tried to model reliability for OSS. Rahmani et al. (2009, 2010) and Zhou and Davis (2005) studied the OSS reliability model by experiments and also performed the comparative analysis.

The frequent change in the source code of open source software makes the software complex and more error-prone. "The CCC has been quantified and design using an information theory-based procedure called entropy. In literature, researchers have worked on entropy and it is widely being exercised in software reliability. It was Shannon (1948, 1951) who provided the foundation for the area of information theory and discussed measures for the efficiency of the communication channel and entropy. The CCC matrix is also used for the maintenance of software (Kafura and Reddy, 1987). In the earlier stage, an article was written by Fenton and Neil in 1999 where the detection rate was used to predict defects with an objective (Fenton and Neil, 1999). After Fenton and Neil (1999), Graves et al. (2000) gave the work related to product measures (Graves et al., 2000). Some researchers proposed in the literature to resolve the fault potential. (Khoshgoftaar et al., 1999; Leazak et al., 2002; Arisholm and Briand, 2006). Then some researchers have discussed the bugs prediction of various projects with the absence of single metrics (Nagappan and Ball, 2005). In 2008 Moser has come up with a comparative study of change and code metrics (Moser et al., 2008). Some researchers gave different approaches to compare the history of complexity metric for bug prediction". Then other researchers have given the comparison of complexity metric and product measures (Hassan, 2009; Kamei et al., 2010).

Later on, D'Ambros have come up with a study of decay based models (D'Ambros et al., 2010, 2012). Then a model was suggested by Singh and Chaturvedi (2012) on the basis of entropy for bug prediction utilizing support vector regression. Further, they have proposed a mathematical model to study the diffusion of the CCC. The release time of software has been predicted by Chaturvedi et al. (2013) using the CCC. Further, the researchers have extended their work and discussed about the prediction of subsequently year expected bugs based on the current year CCC (Singh and Chaturvedi, 2013).

Along with several other measures of entropy, Arora et al. (2014) considered bug prediction models using different types of measures. Chaturvedi et al. (2014) used measures of entropy in the prediction of CCC. A defect prediction model was studied to predict how many defects would arise throughout the improvement of software lines on the basis of entropy (Jeon et al., 2014). Singh et al. (2015) proposed three models- software reliability growth models, models based on CCC to predict bugs in software. Recently Anand et al. have characterized the CCC into two aspects: features improvement and a bug fix for OSS (Anand et al., 2019a).

In the present paper, we have come up with an entropy prediction modeling framework using various types of uncertainties with the help of a three-dimensional wiener process. This process

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

defines the three types of fluctuations namely; bug fix, improvement and addition of new characteristics. The proposed framework is capable of handling the distribution function and is thus an important step towards the unification of SDE based models, which depends on specific distribution functions.

3. Model Development

In this section, the emphasis is laid on describing the proposed entropy-based models. Section 3.1 lists the set of notations and assumptions that have been used.

3.1 Set of Notations

a	Potential CCC to be diffused over a period of time
$b(t)$	Rate at which CCC is diffused in the code
$H(t)$	CCC at any given time.
$E(H(t)) = G(t)$	Expected number of CCC
s_1	Positive constant irregular fluctuations due to bug fixes
s_2	Positive constant irregular fluctuations due to new features
s_3	Positive constant irregular fluctuations due to features enhancement
$\lambda(t)$	Standardized Gaussian white noise
k	Shape parameter for Weibull Distribution
μ	Mean parameter for Normal Distribution
σ	Standard Deviation parameter for Normal Distribution
$F(t)$	Cumulative Distribution Function
$f(t)$	Probability Density Function

3.2 Assumptions

(a) Potential Entropy is constant.
(b) At original time *t=0*, there is no change in file and CCC is zero.

Using the above set of assumptions to capture the uncertainty, we propose the unified models based on CCC to be diffused in software over a period of time. Stochastic differential equations have an inbuilt tendency to cater to irregular fluctuations that can be represented as an ordinary differential equation. This equation describes the Brownian motion, which can be solved by using the $\hat{I}to$ integral. In line with what is available in software reliability literature, the following linear differential equation contributed to model the diffusion rate of CCC (Kapur et al., 2011b):

$$\frac{dH(t)}{dt} = k(t)(a - H(t)) \tag{1}$$

where, $k(t)$ is the relative diffusion rate of errors being detected/removed. Under certain situations, $k(t)$ might not be known in its exact sense; also it may happen that the fluctuations are because of some environmental factor. In that case, it is important to account for such uncertainty that can be modeled by considering associated "noise" term as follows (Singhal et al., 2019):

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

$$\left.\begin{aligned} k(t) &= \frac{f(t)}{1-F(t)} + noise \\ &= h(t) + noise \end{aligned}\right\} \tag{2}$$

where, $h(t)$ is the hazard rate or random external factor. The exact behaviour of noise is difficult to understand and only the distribution function that it might follow can be identified in advance, the function $h(t) = \frac{f(t)}{1-F(t)}$ is time dependent and non-random in nature.

$$k(t) = \frac{f(t)}{1-F(t)} + s\lambda(t) \tag{3}$$

where, $\lambda(t)$ is the factor that portrays the Gaussian white noise and s accounts for the magnitude of irregular fluctuations, that is, $dZ(t)/dt = \lambda(t)$. Before we proceed, it is important to understand the basic definition and properties of the Wiener process (Brownian motion), which are as follows:

"The Wiener process $Z(t)$ is in essence a series of normally distributed random variables and some time points". "The variances of these normally distributed random variables increase to reflect that it is more uncertain to predict the value of the process after a longer period of time".

The properties of the Wiener process $\{ Z(t) \}$ for $t \geq 0$

- $Z(t) - Z(q) \sim N(0, t-s)$ {Normal increments}.
- $Z(t) - Z(q)$ and $Z(u)$ are independent, for $u \leq q < t$ {Independence of increments}.
- $Z(t)$ is a continuous function of t.

Eq^{n}. (3) when substitute in Eq^{n}. (1) results in uncertainty differential equation as follows:

$$\frac{dH(t)}{dt} = \left(\frac{f(t)}{1-F(t)} + s\lambda(t) \right)(a - H(t)) \tag{4}$$

Eq^{n}. (4) can be converted to the subsequent Ito's type DE (based on Taylor series) (Yamada et al., 2003; Oksendal, 2013):

$$dH(t) = \left(\frac{f(t)}{1-F(t)} - \frac{1}{2}s^2\right)(a - H(t))dt + s(a - H(t))dZ(t) \tag{5}$$

Further, solve the above DE with integration

$$\int dH(t) = \int \left(\frac{f(t)}{1-F(t)} - \frac{1}{2}s^2\right)(a - H(t))dt + s(a - H(t))dZ(t) \tag{6}$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

Random variable *H(t)* is presumed to be continuous and its expected value is given as *E(H(t))=G(t)*. Now taking expectation on equal sides

$$\int dG(t) = \int (\frac{f(t)}{1-F(t)} - \frac{1}{2}s^2)(a - G(t))dt + E\left[\int [s(a - H(t)dZ(t)]\right] \quad (7)$$

Using the ideology from Anand et al. (2018a, 2018b, 2019a, 2019b) of Ito Integral, the next component of the Eq^{n}. (7) has vanished, that is

$$E\left[\int s(a - H(t)dZ(t)\right] = 0$$

"which implies that the non-anticipating function will be statistically independent in the future time or mathematically we can say that if we take the expected value of any non-anticipating function then it vanishes the whole component" i.e.

$$E\left[\int_0^T J(t)dZ(t)\right] = 0, \text{ where, } J(t) = s(a - H(t)).$$

Therefore DE (7) can be modeled as:

$$G(t) = E(H(t)) = a\left[1 - \exp\{-\int_0^t \frac{f(n)}{1-F(n)}dn - sZ(t)\}\right] \quad (8)$$

This is the distribution based mean value function using SDE. The complexity of the software system influenced the developer to deviate the testing phenomenon from the NHPP based modeling behaviour to a more realistic environment that accounts for various uncertain factors during the testing process. "The behaviour of the testing process, that is, majorly uncertain in nature is influenced by various factors like, testing efforts, testing skills, efficiency and methods. In order to capture these uncertain aspects regulating the testing process, various researchers have considered one-dimensional wiener process but changes in software source codes are unavoidable and the source code of the software is frequently modified to meet the user's huge requirements (Kapur et al., 2011a). Code change process means to study the patterns of source code Modifications. These changes are occurring due to bug repair (BR), features enhancement (FE) and addition of new features (NF)". In present work, these uncertainties have been represented by S_1 (BR), S_2 (FE) and S_3 (NF) respectively (Singh and Sharma, 2014).

Bug repair	BR is the changes or modifications made in the code due to fixing of a bug.
Features enhancement	FE is the changes made due to some cosmetic changes like formatting, alignment, justification, comments etc.
Addition of new features	NF is the changes made in the code due incorporation of new features or new components.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

Hence, Eq[n]. (9) can be modeled that reflect the three-dimensional Wiener process with three types of fluctuations; we get the mathematical structure as (Tamura and Yamada, 2014):

$$\left.\begin{aligned} dH(t) = \{&\frac{f(t)}{1-F(t)} - \frac{1}{2}(s_1^2 + s_2^2 + s_3^2)(a - H(t)\}dt \\ &+ s_1\{a - H(t)\}dZ_1(t)) \\ &+ s_2\{a - H(t)\}dZ_2(t)) \\ &+ s_3(a - H(t)\}dZ_3(t)) \end{aligned}\right\} \tag{9}$$

The solution of Eq[n]. (9), under the seed value $H(0) = 0$ as follow:

$$G(t) = E(H(t)) = a[1 - \exp\{-\int_0^t \frac{f(n)}{1-F(n)}dn - s_1Z_1(t) - s_2Z_2(t) - s_3Z_3(t)\}] \tag{10}$$

$$G(t) = a\left[1 - (1 - F(t))\,\mathrm{e}^{\frac{1}{2}(s_1^2+s_2^2+s_3^2)t}\right] \tag{11}$$

Now, using the solution process $G(t)$ in Eq[n]. (10), several entropy predictions models can be derived for different distributions. It is to note that as per our knowledge, this is the first attempt to model the complexity of code changes with irregularity with a unified approach. There have been proposals in the past that have studied about these two aspects separately and in different fields, but in unification; they have been studied for the first time (which is obtained by unified approach)

3.3 Proposed Model I

In this proposed model, it is assumed the distribution of diffusion for entropy follows Weibull distribution. It is much used in reliability engineering. Due to its versatile nature, it can take the characteristics of other types of distributions. Thus, we can say that generalization of the exponential distribution is Weibull distribution because of its flexible nature. Hence, in expression (11), *F(t)* follows the Weibull distribution

$$F(t) \sim (1 - e^{-bt^{\gamma}}) \quad ;\gamma > 0$$

where γ is shape parameter (or slope) and *b* is the rate at which the complexity of code changes is diffused in the code. Now putting the value of *F(t)* in Eq[n]. (11), thus the proposed model for entropy prediction is given as

$$G_1(t) = E(H_1(t)) = a(1 - \exp(-bt^{\gamma} + \frac{1}{2}(s_1^2 + s_2^2 + s_3^2)t)) \tag{12}$$

The preceding equation represents the expected number of CCC at any given time *t*.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

3.4 Proposed Model II

It is considered that *F(t)* follows the Rayleigh distribution in the given Eq^{n}. (11). This distribution is the measure of a two-dimensional random vector whose coordinates are distributed identically.

$$F(t) \sim (1-e^{-bt^2}).$$

Now substituting the value of *F(t)* in Eq^{n}. (11), so the entropy prediction can be modeled as

$$G_2(t) = E(H_2(t)) = a(1-\exp(-bt^2 + \frac{1}{2}(s_1^2 + s_2^2 + s_3^2)t)) \quad (13)$$

The above Eq^{n}. (13) describes the amount of entropy prediction based on the unified approach.

3.5 Proposed Model III

In proposed model III, we have assumed that the cumulative distribution function follows the normal distribution. The normal distribution is a type of continuous probability distribution for a real value random variable. It is a bell-shaped curve and also called Gaussian distribution. The general form of its probability density function

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2}.$$

The cumulative distribution function (CDF) of the normal distribution, usually denoted with the Greek letter ϕ, is the integral

$$\phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-x^2/2} dx.$$

The related error function *erf (t)* gives the probability of a random variable with the normal distribution of mean 0 and variance ½ falling in the range [*-t,t*]; that is

$$erf(t) = \frac{2}{\sqrt{\pi}} \int_{0}^{t} e^{-x^2} dx.$$

The above two functions closely related, namely

$$\phi(t) = \frac{1}{2}\left[1 + erf\left(\frac{t}{\sqrt{2}}\right)\right].$$

For a generic normal distribution with density *f*, mean μ and deviation σ, the cumulative distribution function is

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

$$F(t) = \phi\left(\frac{t-\mu}{\sigma}\right) = \frac{1}{2}\left(1 + erf\left(\frac{t-\mu}{\sigma\sqrt{2}}\right)\right).$$

The parameter μ is the mean or expectation of the distribution (and also its median and mode); and σ is its standard deviation. A variance of the distribution is σ^2.

Now substitute the value of *F(t)* in Eq^n. (11), we get,

$$G_3(t) = E(H_3(t)) = a\left[1-(1-\frac{1}{2}\{1+erf\,(\frac{t-\mu}{\sigma\sqrt{2}})\}\,e^{\frac{1}{2}(s_1^2+s_2^2+s_3^2)t}\right] \quad (14)$$

In the above Eq^n. (14), we have described the expected number of entropy predictions for normal distribution.

4. Result and Discussion

4.1 Data Description and Analysis

For validation of the above proposal, we have carried out the data analysis on Apache open source software data sets. Data Set- 1(Abbreviated DS-1) consists of 62.68 entropy that have been predicted in 17 months from Avro Project. Similarly, Data Set-2 (Abbreviated DS-2) and Data Set-3 (Abbreviated DS-3) consist of 68.58 entropy and 56.79 entropy that has been predicted in 18 months and 15 months from Hive and Pig projects respectively. All three data sets can be seen in Table 1. The parameters of the models have been estimated using Statistical Analytical Software (SAS)/ETS user's guide 9.1 (2004) based on the nonlinear regression method. "The performance of models is judged by their capability to fit the data (goodness of fit) and predicting the future performance of the entropy (as shown in Figures 1, 2 and 3). The parameter estimation and comparison criteria result for DS-1, DS-2 and DS-3 of all models under consideration can be viewed through Table 2, Table 3 and Table 4 respectively.".

Table 1. Monthly data of apache open source software.

Avro Data Set		Hive Data Set		Pig Data Set	
Time (Monthly)	Non-Cumulative Entropy	Time (Monthly)	Non-Cumulative Entropy	Time (Monthly)	Non-Cumulative Entropy
1	0.988798	1	3.918735	1	1.9815
2	3.877407	2	3.876313	2	3.937522
3	3.855519	3	3.85041	3	3.88785
4	3.823077	4	3.853447	4	3.918368
5	3.858738	5	3.892308	5	3.901588
6	3.939776	6	3.882447	6	3.886225
7	3.896395	7	3.84032	7	3.876901
8	3.940347	8	3.810116	8	3.90468
9	3.941543	9	3.872064	9	3.941262
10	3.963296	10	3.885398	10	3.927332
11	3.926444	11	3.84444	11	3.921351
12	3.948949	12	3.874096	12	3.941777
13	3.923868	13	3.814615	13	3.912891
14	3.915305	14	3.897128	14	3.936509
15	3.97836	15	3.878384	15	3.923166
16	3.906676	16	3.843008		
17	3	17	3.843319		
		18	2.910245		

Source: DS-I, DS-II and DS-III: (Anand et al. 2019a).

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

Table 2. Estimation of model parameters for DS -1.

Parameter	Model-I	Model-II	Model-III
a	96.43	64.90	78.98
b	0.018	0.0087	-
γ	1.427	-	-
s_1	0.0032	0.0021	0.030
s_2	0.00052	0.0012	0.215
s_3	0.0042	0.001	0.109
μ	-	-	15.90
σ	-	-	11.29

Table 3. Estimation of model parameters for DS –II.

Parameter	Model-I	Model-II	Model-III
a	79.06	67.46	86.43
b	0.024	0.0093	-
γ	1.470	-	-
s_1	0.012	0.011	0.0031
s_2	0.0032	0.0033	0.113
s_3	0.00013	0.00012	0.110
μ	-	-	13.42
σ	-	-	9.98

Table 4. Estimation of model parameters for DS –III.

Parameter	Model-I	Model-II	Model-III
a	87.02	57.71	61.23
b	0.024	0.0146	-
γ	1.39	-	-
s_1	0.0112	0.011	0.055
s_2	0.0032	0.0032	0.215
s_3	0.0124	0.0124	0.109
μ	-	-	11.88
σ	-	-	9.01

4.2 Comparison Criteria for Proposed Models

The following table gives the results of comparison criteria for developed models which are calculated by Sum of Square errors, Mean Squared Error, Root Mean Square Errors and Coefficient of Determination.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

Table 5. Comparison criteria for all the proposed models.

Model →	Model-I	Model-II	Model-III
DS-1			
SSE	6.592	96.49	241.1
MSE	0.439	6.433	18.54
Root MSE	0.662	2.536	4.306
R-Square	0.99	0.98	0.96
DS-2			
SSE	67.083	240.7	124.1
MSE	4.19	15.04	8.274
Root MSE	2.047	3.885	2.87
R-Square	0.99	0.97	0.98
DS-3			
SSE	7.647	91.08	117.4
MSE	0.588	7.006	10.67
Root MSE	0.767	2.647	3.266
R-Square	0.99	0.98	0.97

On the basis of analysis performed in above Table 5, it can be clearly seen that the values of comparison criteria are quite significant, that is the obtained values are in acceptable range as the value of *R*-square obtained in approaching 1 which means the proposed Model-I are able to cater the entropy prediction in the good sense. Figures 1-3 also provide a way to graphically analyze the predictive capability of models which show that computed values are close correspondence to open source software data.

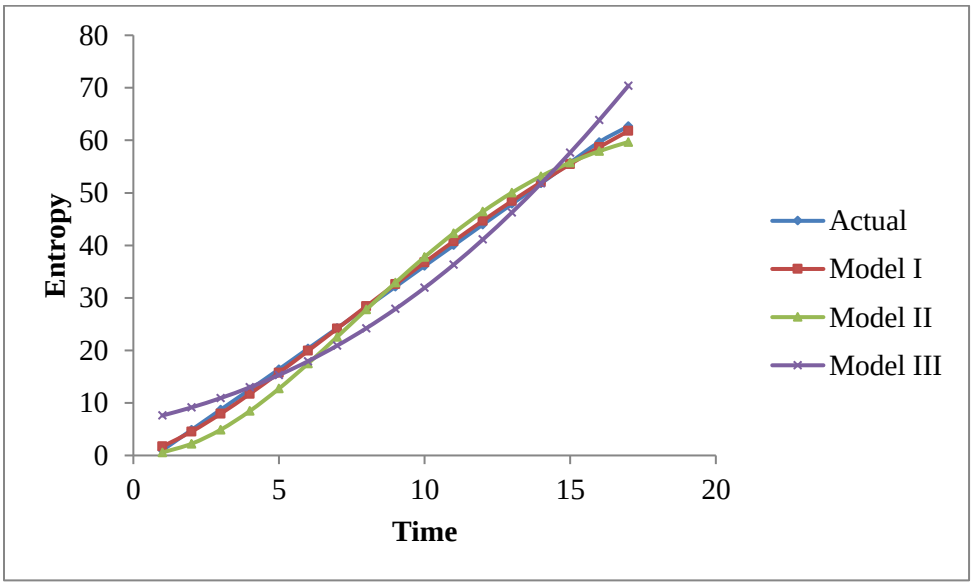

Figure 1. Goodness fit curve for Data Set-I.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 207-222, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.013

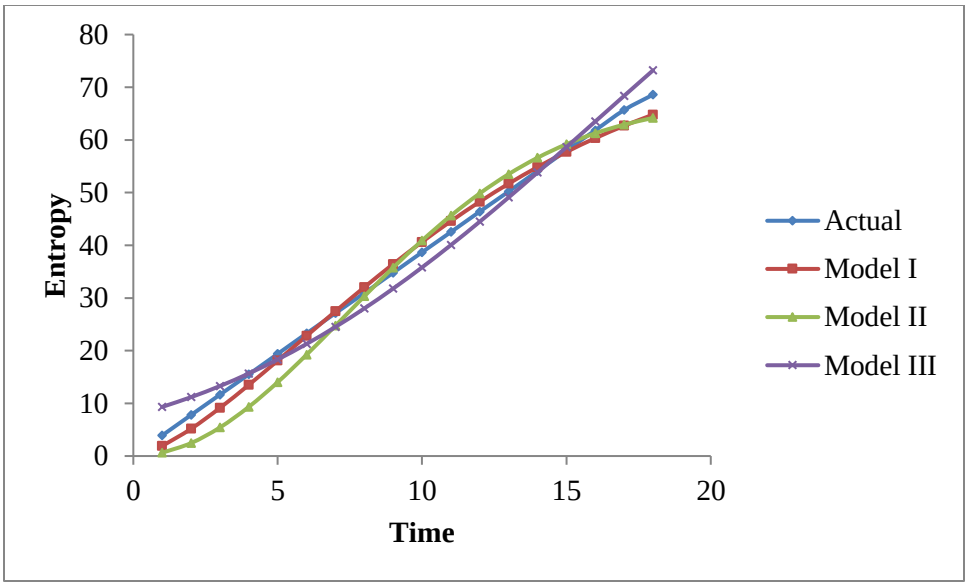

Figure 2. Goodness fit curve for Data Set-II.

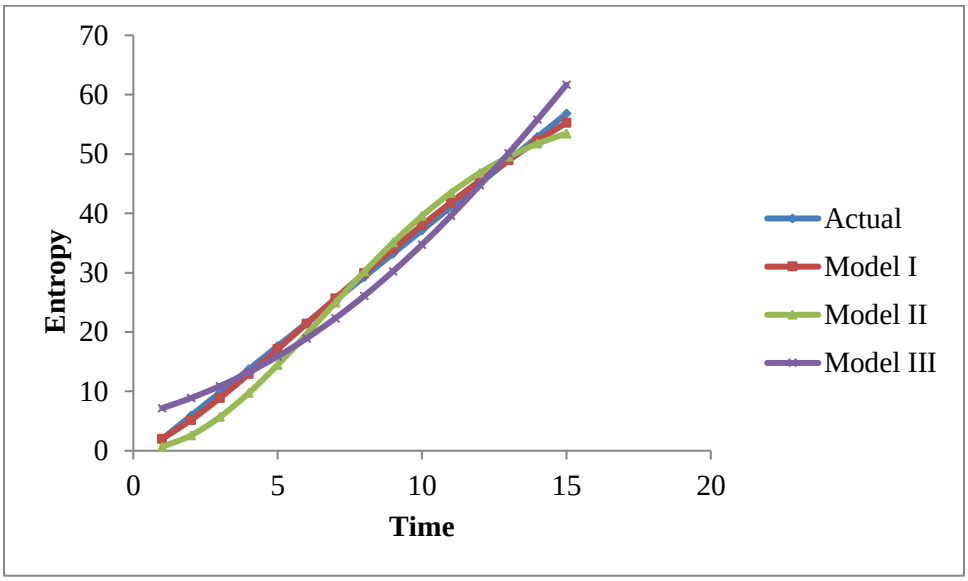

Figure 3. Goodness fit curve for Data Set-III.

5. Conclusion

In the present work, a generalized framework for deriving several entropy prediction models based on SDE of Ito's type has been presented. Three basic reasons that; bug fixing, features improvement and new feature introduction have been studied for the diffusion process which can bring out changes in the code structure. The proposed framework is capable of handling any general distribution function and takes care of three-dimensional Wiener processes for inculcating

the irregularity. The proposed set of models has been validated on three open source data sets and Model- I turns out to be the best fit in all data sets under consideration.

Conflict of Interest
The authors confirm that there is no conflict of interest to declare for the publication.

Acknowledgment
The authors would like to thank the editor in chief and reviewers for giving valuable inputs in improvising the overall quality of the paper.

References

Anand, A., Bharmoria, S., & Ram, M. (2019a). Characterizing the complexity of code changes in open source software. In: Anand, A., Ram, M. (eds) *Recent Advancements in Software Reliability Assurance*. pp. 1-14, CRC Press- A Taylor & France Group, Boca Raton.

Anand, A., Deepika, & Singh, O. (2019b). Formulation of error generation-based srgms under the influence of irregular fluctuations. In: Kapur, P., Klochkov, Y., Verma, A., Singh, G. (eds) *System Performance and Management Analytics*. Asset Analytics (Performance and Safety Management). pp. 103-117, Springer, Singapore.

Anand, A., Deepika, Verma, A.K., & Ram, M. (2018a). Revisiting error generation and stochastic differential equation-based software reliability growth models. In: Anand, A., Ram, M. (eds) *System Reliability Management: Solutions and Technologies*. pp. 65-78, CRC Press- A Taylor & France Group, Boca Raton.

Anand, A., Singhal, S., & Singh, O. (2018b). Revisiting dynamic potential adopter diffusion models under the influence of irregular fluctuations in adoption rate. In *Handbook of Research on Promoting Business Process Improvement Through Inventory Control Techniques*. pp. 499-519, IGI Global, USA.

Arisholm, E., & Briand, L.C. (2006, September). Predicting fault-prone components in a java legacy system. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (pp. 8-17). Rio de Janeiro, Brazil. DOI: 10.1145/1159733.1159738.

Arora, H.D., Kumar, V., & Sahni, R. (2014, October). Study of bug prediction modeling using various entropy measures-a theoretical approach. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization* (pp. 1-5). IEEE. Noida, India.

Bhatt, N., Anand, A., Yadavalli, V.S.S., & Kumar, V. (2017). Modeling and characterizing software vulnerabilities. *International Journal of Mathematical, Engineering and Management Sciences, 2*(4), 288-299.

Chatterjee, S., & Shukla, A. (2016). Modeling and analysis of software fault detection and correction process through Weibull-type fault reduction factor, change point and imperfect debugging. *Arabian Journal for Science and Engineering*, *41*(12), 5009-5025.

Chaturvedi, K.K., Bedi, P., Misra, S., & Singh, V.B. (2013, December). An empirical validation of the complexity of code changes and bugs in predicting the release time of open source software. In *2013 IEEE 16th International Conference on Computational Science and Engineering* (pp. 1201-1206). IEEE. Sydney, Australia.

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.

D'Ambros, M., Lanza, M., & Robbes, R. (2012). Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empirical Software Engineering*, *17*(4-5), 531-577.

D'Ambros, M., Lanza, M., & Robbes, R. (2010, May). An extensive comparison of bug prediction approaches. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)* (pp. 31-41). IEEE. Cape Town, South Africa.

Deepika, Singh, O., Anand, A., & Singh, J.N.P. (2017). Testing domain dependent software reliability growth models. *International Journal of Mathematical, Engineering and Management Sciences*, *2*(3), 140-149.

Fenton, N.E., & Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, *25*(5), 675-689.

Graves, T.L., Karr, A.F., Marron, J.S., & Siy, H. (2000). Predicting fault incidence using software change history. *IEEE Transactions on Software Engineering*, *26*(7), 653-661.

Hassan, A.E. (2009, May). Predicting faults using the complexity of code changes. In *2009 IEEE 31st International Conference on Software Engineering* (pp. 78-88). IEEE. Vancouver, Canada.

Jeon, C.K., Byun, C., Kim, N.H., & In, H. (2014). An entropy-based method for defect prediction in software product lines. *International Journal of Multimedia and Ubiquitous Engineering*, *9*(3), 375-377.

Kafura, D., & Reddy, G.R. (1987). The use of software complexity metrics in software maintenance. *IEEE Transactions on Software Engineering*, *13* (3), 335-343.

Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K.I., Adams, B., & Hassan, A.E. (2010, September). Revisiting common bug prediction findings using effort-aware models. In *2010 IEEE International Conference on Software Maintenance* (pp. 1-10). IEEE. Timisoara, Romania.

Kapur, P.K., Pham, H., Gupta, A., & Jha, P.C. (2011a). *Software reliability assessment with OR applications* (p. 364). Springer, London.

Kapur, P.K., Singh, O., & Singh, J. (2011b). Stochastic differential equation-based software reliability growth modeling with change point and two types of imperfect debugging. *In Proceedings of 5th National Conference on Computing for Nation Development*, (pp. 605-12). INDIACom, New Delhi.

Khoshgoftaar, T.M., Allen, E.B., Jones, W.D., & Hudepohl, J.P. (1999). Data mining for predictors of software quality. *International Journal of Software Engineering and Knowledge Engineering*, *9*(05), 547-563.

Leszak, M., Perry, D.E., & Stoll, D. (2002). Classification and evaluation of defects in a project retrospective. *Journal of Systems and Software*, *61*(3), 173-187.

Li, Q., & Pham, H. (2017). A testing-coverage software reliability model considering fault removal efficiency and error generation. *PloS one*, *12*(7), e0181524-e0181524.

Li, X., Li, Y.F., Xie, M., & Ng, S.H. (2011). Reliability analysis and optimal version-updating for open source software. *Information and Software Technology*, *53*(9), 929-936.

Moser, R., Pedrycz, W., & Succi, G. (2008, May). A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the 30th International Conference on Software Engineering* (pp. 181-190). ACM. DOI: 10.1145/1368088.1368114.

Nagappan, N., & Ball, T. (2005, May). Use of relative code churn measures to predict system defect density. In *Proceedings of the 27th International Conference on Software Engineering* (pp. 284-292). ACM. Doi:10.1145/1062455.1062514.

Oksendal, B. (2013). *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media. London.

Rahmani, C., Azadmanesh, A.H., & Najjar, L. (2010). A Comparative analysis of open source software reliability. *Journal of Software*, *5*(12), 1384-1394.

Rahmani, C., Siy, H., & Azadmanesh, A. (2009). An experimental analysis of open source software reliability. *Department of Defense/Air Force Office of Scientific Research*.

SAS Institute Inc. (2004). SAS/ETS User's Guide Version 9.1. Cary, NC: SAS Institute Inc.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379-423.

Shannon, C.E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, *30*(1), 50-64.

Singh, V.B., Chaturvedi, K.K., Khatri, S.K., & Kumar, V. (2015). Bug prediction modeling using complexity of code changes. *International Journal of System Assurance Engineering and Management*, *6*(1), 44-60.

Singh, V.B., & Sharma, M. (2014, November). Prediction of the complexity of code changes based on number of open bugs, new feature and feature improvement. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops* (pp. 478-483). IEEE. Naples, Italy.

Singh, V.B., & Chaturvedi, K.K. (2013, June). Improving the quality of software by quantifying the code change metric and predicting the bugs. In *International Conference on Computational Science and Its Applications* (pp. 408-426). Springer, Berlin, Heidelberg.

Singh, V.B., & Chaturvedi, K.K. (2012, November). Entropy based bug prediction using support vector regression. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 746-751). IEEE. Kochi, India.

Singhal, S., Anand, A., & Singh, O. (2019). SDE based generalized innovation diffusion modeling. *International Journal of Mathematical, Engineering and Management Sciences*, *4*(3), 697-707.

Tamura, Y., & Yamada, S. (2014, October). 3D software tool for reliability assessment based on three dimensional wiener process model considering big data on cloud computing. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization* (pp. 33-37). IEEE. Noida, India.

Tamura, Y., & Yamada, S. (2008). A component-oriented reliability assessment method for open source software. *International Journal of Reliability, Quality and Safety Engineering*, *15*(01), 33-53.

Yamada, S., Nishigaki, A., & Kimura, M. (2003). A stochastic differential equation model for software reliability assessment and its goodness-of-fit. *International Journal of Reliability and Applications*, *4*(1), 1-12.

Yang, J., Liu, Y., Xie, M., & Zhao, M. (2016). Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes. *Journal of Systems and Software*, *115*, 102-110.

Zhou, Y., & Davis, J. (2005). Open source software reliability model: an empirical approach. *ACM SIGSOFT Software Engineering Notes*, *30*(4), 1-6.

Zhu, M., & Pham, H. (2018). A multi-release software reliability modeling for open source software incorporating dependent fault detection process. *Annals of Operations Research*, *269*(1-2), 773-790.