MODELING AND ANALYSIS OF FAULT DETECTION AND CORRECTION PROCESS FOR MULTI RELEASE SOFTWARE SYSTEM

Adarsh Anand*, Deepika*[1], Jagvinder Singh** and Ompal Singh*
*Department of Operational Research, University of Delhi, Delhi 110007, India
**University School of Management and Entrepreneurship, DTU, India

ABSTRACT
With ever changing market demands, software organizations in today's scenario have to improve regularly by coming up with new functionality and gain over their rival counterpart. For software developers, fault detection and fault correction are important activities making the software qualitative. In the present framework, we have used convolution theory for analysis of fault detection and correction processes. Different types of distributions have been inculcated in the modeling. The proposed reliability growth models for software are validated on real life software data set. Further, we have also discussed normalized criteria distance method, to rank and choose superlative release from between four releases based on a set of criteria.

KEYWORDS: Distribution function, Fault detection process (FDP), Fault correction process (FCP), Multi Release, Normalized criteria distance method (NCDM), Software reliability growth models (SRGMs).

MSC: 12H20, 62EXX, 62H10

RESUMEN
Con las siempre cambiantes demandas del mercado, las organizaciones de software en el actual escenario tienen que mejorar regularmente actualizándolos con las nuevas funcionalidades y ganarle a sus contrapartes rivales. Para los desarrolladores de software, la detección de fallas y su corrección son importantes actividades para mejorar la calidad del software. En el presente marco de trabajo, hemos usado la teoría de convolución para el análisis de los procesos de detección de fallas y de corrección. Diferentes tipos de distribuciones han sido inculcados en el modelado. La propuesta de modelo para el incremento de la fiabilidad modelos para softwares son validados usando conjuntos de datos de la vida real. Además, también discutimos criterios normalizados basados en métodos de distancia para rankear las selecciones de aprobación superlativos entre 4 aprobaciones basadas en un conjunto de criterios.

PALABRAS CLAVE: Función de distribución, procesos de detección de fallos (FDP), Procesos de corrección de fallos (FCP), Multi aprobación , Criterios normalizados de distancia (NCDM), Criterios del crecimiento de la fiabilidad de software (SRGMs).

1. INTRODUCTION

Current Market surroundings do not allow the developer to spend too much time to build the software. "Due to the rapid changing technology, the fact that software might get obsolete also dread the developers. Developers prefer to follow a multi release policy by introducing multiple upgraded versions of software instead of delivering the entire product at one go. During the lifespan of large software systems, iterative development procedure is commonly adopted with continuously incremental software versions released to the market (Kapur et. al 2011a). Without the loss of generality, a specific iterative software development scenario is considered for our current study, where a software development team develops, tests and releases software version by version." Continuous up- gradation has become a requirement for the industries. "The term upgrade refers to the replacement of a product with a modified version of the same product". In one progressive cycle, software companies do not endeavour to convey an absolute and ideal product due to resource restriction and time. They enhance the performance of successive system by eliminating the errors from existing software. In present time, firms are re-releasing their software by improving the existing functionality adding new features and so on. "One such example is versions of Android like: Alpha (1.0), Beta (1.1), Cupcake (1.5), Donut (1.6), Eclair (2.0–2.1), Froyo (2.2–2.2.3), Ginger bread (2.3–2.3.7), Honey

deepika.sre@gmail.com

comb (3.0–3.2.6), Ice Cream Sandwich (4.0–4.0.4), Jelly Bean (4.1–4.3.1), Kit Kat (4.4–4.4.4), Lollipop (5.0–5.0.1), Marshmallow (6.0 - 6.0.1), Nougat (7.0 -7.1.2) and Oreo (8.0-8.1)"(Anand et al., 2015).

Thus, "up-gradation is a process of adding new features, defects fixes and patches to an application in the form of installer or additions or patch. It is essential to know the content of faults in the software before debugging them". Kapur et al. (2010b) gave model related to multiple releases, considering that cumulative faults in each generation depend on all previous releases and assumed that fault is removed with certainty. Later, Kapur et al. (2011c) has given a multi release software reliability growth model in which they identified the faults left in the software when it is in operational phase during the testing of the new code incorporating that the software includes different types of faults. Singh et al. (2012) has delivered that overall fault removal of the new release depends on the reported faults from the just previous release of the software and on the faults generated due to the addition of some new functionality to the existing software system. They developed two SRGMs using Logistic distribution and Normal distribution. Anand et al. (2014) incorporated the generalised framework for faults in new release due to up-gradation of the features and undetected faults from operational phase of preceding releases and different distributions have used for fault removal phenomenon". Researchers have also worked on concept of testing effort, imperfect debugging, change point, uncertainty and release time problem, Multi attribute utility theory (Kapur et. al 2010a, 2010b; Singh et. al 2011; Kapur et. al 2015). Later, Anand et al. (2015) proposed "fault severity based multi up-gradation modeling considering testing and operational phase". These proposed models have the postulation that the overall fault removal of the new release depends on the reported bugs from the just previous release of the software.

As testing progresses, the dormant bug is diagnosed, terminated and the number of faults remaining in the software system thus gradually decreases. One of the main objectives of software testing is detection and correction of faults before the release of the software in the market. Generally, whenever a failure is identified the fault correction team requires a period of time to locate the fault and modify some codes accordingly to remove them. Thus, the time lag between detection and correction is a common experience in software testing (Kapur et al., 2011b). This time lag is the time delay between the fault detection and correction processes. The removal time of a fault depends on various factors such as complexity of the faults, number of the detected faults etc. Some faults which are detected but not corrected still remain in the software. These latent faults are caused by the correction lag and reflect the relationship between fault detection and correction processes (Kapur et al., 2011a).

Various SRGMs have been proposed to calculate approximately essential manner such as leftover faults, failure time etc. Most of these SRGMs are based on NHPP (Goel et al., 1979, Musa et al., 1987, Yamada et al., 2003, Pham 2006, Kapur et al., 2010a, Kapur et al., 2011a) and are useful to illustrate behaviour of software testing method that includes FDP & FCP. Each fault correction process is connected to a detection process as fault can only be corrected if they are detected. The testing process as a two stage process in which all observed/detected faults are corrected after a constant delay of time has been discussed by Schneidewind (1975) and Yamada et al. (1984). Lo and Huang (2006) have proposed a general framework where some existing NHPP models are re-evaluated from the viewpoint of correction process. Further Xie et al. (2007) emphasized on fault correction process described by delayed detection process with a random or deterministic delay. Therefore, a convolution methodology with different distributions (i.e Exponential and Erlang distribution) can be used in SRGMs which cater to softwares coming in multi versions.

Rest of the manuscript is prearranged as follow: Firstly, assumptions about SRGMs and notations have been comprised. Then we have discussed the modeling framework. The successive sections enlighten us about the multi release modeling with convolution probability function and parameter estimates in each release of all SRGMs. Further, a segment describes the ranks by normalized criteria distance approach (Pham, 2014) of all SRGMs. At last, conclusion is followed by references.

2. ASSUMPTION

Some fundamental presumptions of the framework are as given below:

a) The FDP and FCP are based on NHPP.
b) At any time, the amount of bugs detected is directly proportional to leftover quantity of bugs.
c) There is mutual independence between the faults.

3. NOTATIONS

$m(t)$	Expected number of faults removed by time *t*
$f(t)$	Probability Density Function (PDF) of fault removal process
$F_{di}(t)$	Cumulative Distribution Function (CDF) for time up to *'t'* for detection process used in i^{th} release
$F_{ci}(t)$	Cumulative Distribution Function (CDF) for time up to'*t*' for correction process used in i^{th} release
t_{i-1}	Time for i^{th} release $(i = 1\ to\ 4)$
a_i	Fault count for i^{th} release $(i = 1\ to\ 4)$
b_i	Constant parameter for i^{th} release (*i*=1 *to* 4)
$\otimes$	Steiltjes convolution

4. GENERAL FRAMEWORK FOR MULTI-RELEASE MODEL

In dynamic scenario, primary release of the piece is foremost foundation of firms so company have to pay more attention on it. Testing team has to detect and remove faults as much as possible and minimize the risk of errors in future. After first release company has to plan for next version with new update. They will test and analyze the reported bugs of just previous release during testing phase of the current release (Kapur et. al 2011a; Singh et. al 2012).

4.1. First Release (R1)

Let us presume that first version of the software is released at time $t = t_1$. It is a fact that correcting all the bugs during R1 of the software is practically infeasible i.e. some of the faults of the previous release have to be removed in the successive releases. Modeling of first release for two stage detection-correction process is given as:

$$m_1(t) = a_1[(F_{d1}\ \text{Ä}\ F_{c1})(t)] \qquad ;0\ \text{£}\ t < t_1 \tag{1}$$

4.2. Second Release (R2)

Considering the time for introduction of R2 is t_2 and testing interval for second release $[t_1, t_2)$ will be operational phase for just previous release. In this period when there are two versions of the software $a_1[1-(F_{d1} \otimes F_{c1})(t_1)]$, the leftover fault content of the first version interacts with new debugging rate. As a result of these interactions a fraction of faults which were not removed during the testing of first version of the product gets removed (Anand et al., 2014). In accumulation, fault are generated due to the enhancement of the features are also removed during the testing with new detection-correction proportion i.e. $(F_{d2} \otimes F_{c2})(t-t_1)$.

Hence mathematical expression of R2 during $[t_1, t_2)$ can be structured as:

$$m_2(t) = a_2[(F_{d2} \otimes F_{c2})(t-t_1)] + a_1[1-((F_{d1} \otimes F_{c1})(t_1))][(F_{d2} \otimes F_{c2})(t-t_1)] \tag{2}$$

$$= [a_2 + a_1\{1-(F_{d1}\otimes F_{c1})(t_1)\}][(F_{d2}\otimes F_{c2})(t-t_1)] \qquad ; t_1 \le t < t_2$$

4.3. Third Release (R3)

Correspondingly, for (R3), we assume faults generated in third release due to the new lines of code and new functionalities and remaining number of faults from the just previous (R2) release. Time for introduction of third release R3 is t_3 and $\left[t_2, t_3\right)$ is the testing period for third release. In this interval, $a_2[1-(F_{d2}\otimes F_{c2})(t_2-t_1)]$ the left over faults of second edition interacts with changed fault detection-correction rate and faults related to new functionalities are corrected with new detection-correction proportion $(F_{d3}\otimes F_{c3})(t-t_2)$. The mathematical expression for R3 is as follow:

$$\begin{aligned} m_3(t) &= a_3[(F_{d3}\otimes F_{c3})(t-t_2)] + a_2[1-\{(F_{d2}\otimes F_{c2})(t_2-t_1)\}][(F_{d3}\otimes F_{c3})(t-t_2)] \qquad (3)\\ &= \left[a_3 + a_2[1-\{(F_{d2}\otimes F_{c2})(t_2-t_1)\}]\right][(F_{d3}\otimes F_{c3})(t-t_2)] \qquad ; t_2 \le t < t_3 \end{aligned}$$

4.4. Fourth Release (R4)

Similarly, mathematical form for (R4) can be written as:

$$\begin{aligned} m_4(t) &= a_4[(F_{d4}\otimes F_{c4})(t-t_3)] + a_3[1-((F_{d3}\otimes F_{c3})(t_3-t_2))][F_{d4}\otimes F_{c4})(t-t_3)] \qquad (4)\\ &= \left[a_4 + a_3[1-\{(F_{d3}\otimes F_{c3})(t_3-t_2)\}]\right][F_{d4}\otimes F_{c4})(t-t_3)] \qquad ; t_3 \le t < t_4 \end{aligned}$$

Similarly we can express the mathematical equation for $(n-1)^{th}$ and n^{th} release.

$$\begin{aligned} m_{n-1}(t) &= a_{n-1}[(F_{d(n-1)}\otimes F_{c(n-1)})(t-t_{n-2})]\\ &\quad + a_{n-2}[1-((F_{d(n-2)}\otimes F_{c(n-2)})(t_{n-2}-t_{n-3}))][F_{d(n-1)}\otimes F_{c(n-1)})(t-t_{n-2})] \qquad (5)\\ &= \left[a_{n-1} + a_{n-2}[1-\{(F_{d(n-2)}\otimes F_{c(n-2)})(t_{n-2}-t_{n-3})\}]\right][F_{d(n-1)}\otimes F_{c(n-1)})(t-t_{(n-2)})]\\ &\qquad ; t_{n-2} \le t < t_{n-1} \end{aligned}$$

and

$$\begin{aligned} m_n(t) &= a_n[(F_{dn}\otimes F_{cn})(t-t_{n-1})]\\ &\quad + a_{n-1}[1-((F_{d(n-1)}\otimes F_{c(n-1)})(t_{n-1}-t_{n-2}))][F_{dn}\otimes F_{cn})(t-t_{n-1})] \qquad (6)\\ &= \left[a_n + a_{n-1}[1-\{(F_{d(n-1)}\otimes F_{c(n-1)})(t_{n-1}-t_{n-2})\}]\right][F_{dn}\otimes F_{cn})(t-t_{n-1})] \qquad ; t_{n-1} \le t < t_n \end{aligned}$$

5. CONVOLUTION METHODOLOGY AS A TOOL FOR MODELING VARIOUS RELEASES

In this segment, we use convolution methodology for deriving the SRGMs with different distribution in each release (release 1 to 4). To analyze the performance of complete software system joint distribution is very useful. One of the most important concepts in Fourier theory is that of a convolution. Mathematically, a Steiltjes Convolution is defined as the integral over all space of one function at z times another function at $w-z$. The integration is taken over the variable z typically from minus infinity to infinity over all the dimensions. So the convolution is a function of a new variable w, as shown by following equations. Convolution operation is commutative in nature (Randy, 2009).

$$C(w) = f(z)\otimes g(z) = \int_{space} f(z)g(w-z)dz \qquad (7)$$

This design gives us an idea about how we can think about the convolution, as giving a weighted sum of shifted replica of one function: the weights are given by the function value of the second function at the shift vector. In time horizon $[0,t]$, using the method of convolution probability function which is mathematically represented as:

$$(F \otimes G)t = \int_0^t F(t-z)\, g(z) dz \tag{8}$$

As a matter of fact, it is considered that exponential distribution has been taken for fault correction process (with distinct rate for all the versions) but nature of detection process requires some discussion. Initially we assume that it follows a constant pattern. But once, foremost version is released and when it is in its operational phase then we are technically in testing phase for the successive version (i.e. Release-2 here). So the new detection process shall follow what has been jointly calculated for earlier version of software. i.e. the new detection process shall follow exponential pattern of debugging (obviously with different rates). Likewise the resultant of second release acts as the new detection process during the testing phase of third release and similarly the resultant of joint distribution from third release would be treated as detection process during the testing of forth version of software.
Therefore, entire process can be summarized as follows:

5.1. Release-I

Assuming that FDP as a constant and FCP as exponential *i.e.* $F_{d1}(t):\ 1(t)$ and $F_{c1}(t):\ \exp(b_1)$ using the above equation (1) with convolution methodology, we have mathematical function for R1.

$$m_1(t) = a_1[1-\{1-(F_{d1} \otimes F_{c1})(t)\}] \tag{9}$$

$$\text{where } (F_{d1} \otimes F_{c1})(t) = (1-e^{-b_1 t})$$

5.2. Release-II

Using convolution methodology, we assume that FDP and FCP both are following exponential distribution *i.e.* $(F_{d2}(t) = (1-e^{-b_2 t})$ and $F_{c2}(t) = (1-e^{-b_2 t}))$. Thus mathematical formulation for R2 is represented in given eq. (10).

$$m_2(t) = [a_2 + a_1\{1-(F_{d1} \otimes F_{c1})(t)\}][(F_{d2} \otimes F_{c2})(t)] \tag{10}$$

$$\text{where } (F_{d2} \otimes F_{c2})(t) = (1-(1+b_2 t)e^{-b_2 t})$$

5.3. Release-III

In release 3, we assume that FDP follows Erlangian 2- stage distribution and FCP follows exponential distribution i.e. $F_{d3}(t) = \{1-(1+b_3 t)e^{-b_3 t}\}$ *and* $F_{c3}(t) = (1-e^{-b_3 t}))$. With the concept of Convolution, mathematical formation for third release can be structured as:

$$m_3(t) = \left[a_3 + a_2[1-\{(F_{d2} \otimes F_{c2})(t)\}]\right][(F_{d3} \otimes F_{c3})(t)] \tag{11}$$

$$\text{where } (F_{d3} \otimes F_{c3})(t) = (1-(1+b_3 t + \frac{b_3^2 t^2}{2!})e^{-b_3 t})$$

5.4. Release-IV

Similarly, for release-4, it is assuming that FDP follows Erlang 3-stage and FCP follows exponential distribution i.e.

$$F_{d4}(t) = \{1-(1+b_4 t + \frac{b_4^2 t^2}{2!})e^{-b_4 t}\} \text{ and } F_{c4}(t) = \{1-e^{-b_4 t}\}\ .$$

$$m_4(t) = \left[a_4 + a_3[1 - \{(F_{d3} \otimes F_{c3})(t)\}]\right][(F_{d4} \otimes F_{c4})(t)] \tag{12}$$

where $(F_{d4} \otimes F_{c4})(t) = (1 - (1 + b_4 t + \frac{b_4^2 t^2}{2!} + \frac{b_4^3 t^3}{3!})e^{-b_4 t})$

6. STATISTICAL ANALYSIS AND MODEL JUSTIFICATION

6.1. Data Explanation

The presentation of proposed model has been analysed by using real data. For the validation we have employed data from a real software project. The project manager can type the "report" command in the MR system and get the monthly report summary as shown in below Table-1. In the total epoch, data comprises of 16 months in which 592 faults has been detected in first release and 443 faults, 361 faults, 428 faults has been detected in second, third and fourth release respectively (Sun, 2002).

Table 1: Monthly data of four software releases (Sun, 2002)

Time	R1	R2	R3	R4
1	10	9	12	5
2	58	95	15	23
3	93	178	99	131
4	167	229	180	214
5	234	270	281	257
6	310	309	302	312
7	409	346	322	379
8	455	388	345	402
9	486	414	359	426
10	515	427	361	427
11	555	437		428
12	576	442		
13	586	442		
14	589	442		
15	589	443		
16	592			

6.2. Performance Analysis

There are numerous techniques to estimate the parameters of SRGMs. The parameters of the proposed models have been estimated via nonlinear regression using the data analysis software package known as SAS (SAS/ETS User's Guide, 2004). The parameter estimation and comparison criteria results for data set of the models under consideration can be viewed through Table-2 and Table -3 respectively.

Table 2: Estimates of Model Parameters

Parameter	R1	R2	R3	R4
a_i	1030.71	411.05	388.54	432.18
b_i	0.06	0.41	0.65	0.85

Table 3: Comparison Criteria for proposed Models

Releases under consideration	MSE	Bias	Variation	RMSPE	R^2
Release-I	2184.27	-9.45	47.27	48.2	0.95
Release-II	101.4	-0.63	10.4	10.42	0.99
Release-III	371.36	-3.55	19.96	20.27	0.98
Release-IV	156.02	-0.45	13.09	13.09	0.99

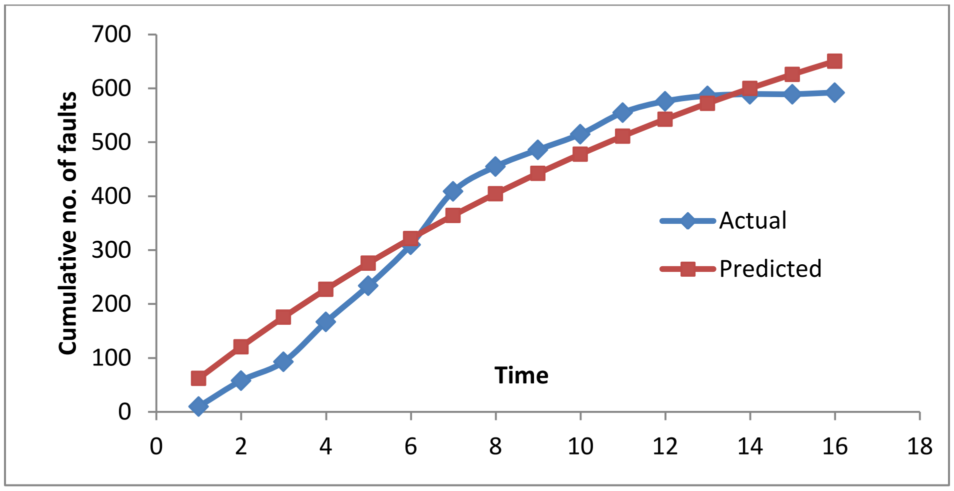

Figure 1: Curve of Goodness of fit

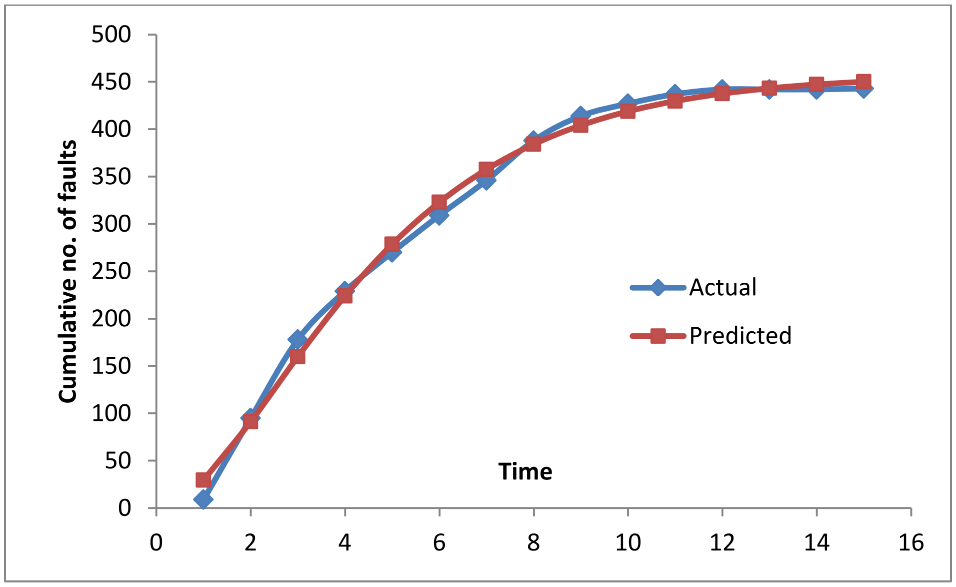

Figure 2: Curve of Goodness of fit

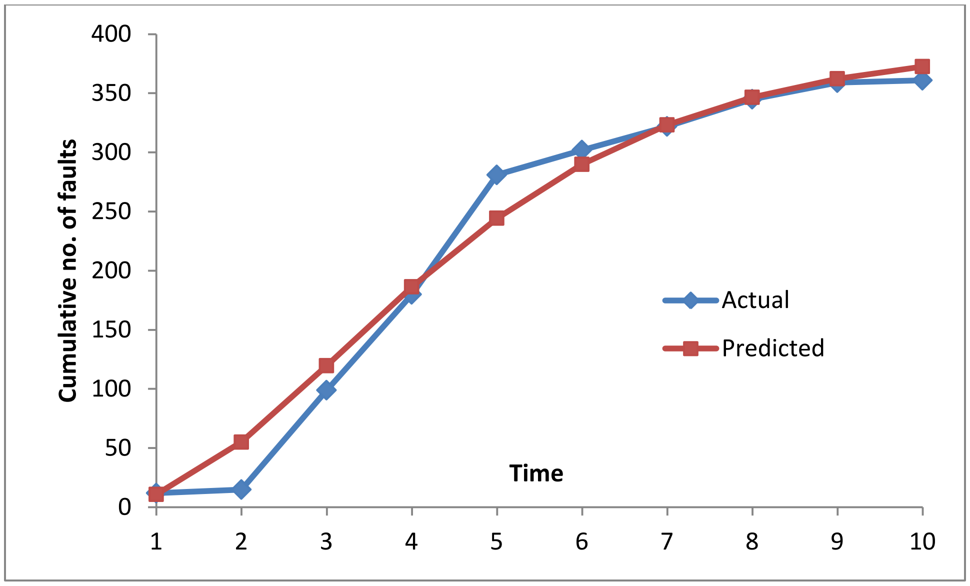

Figure 3: Curve of Goodness of fit

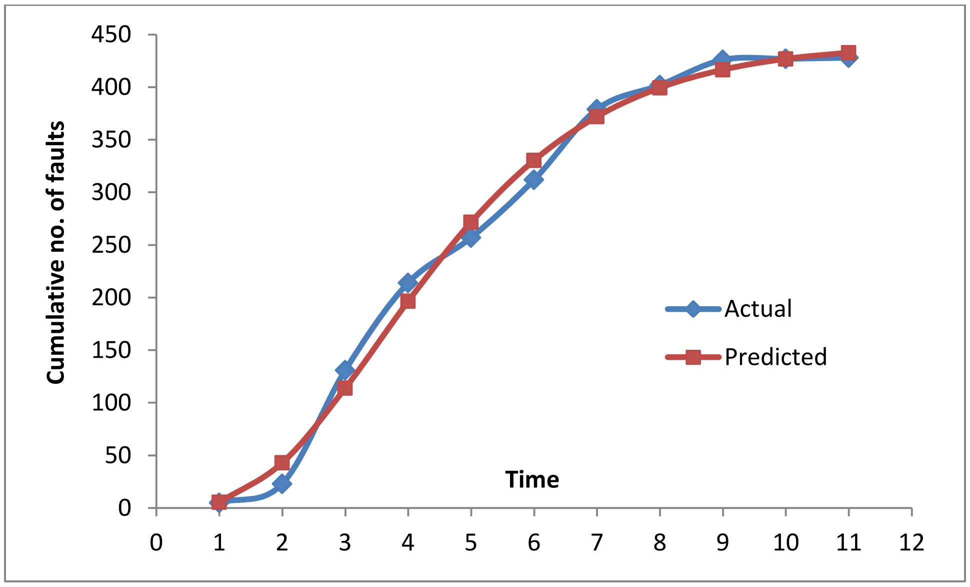

Figure 4: Curve of Goodness of fit

Above shown graphs (figure-1 to 4) correspond to better fit for all the four releases on proposed SRGMs. As it can be seen from Table-3, it is not clear that which is performing best among of the four releases. Thus, there was the need for an approach to quantify proposed SRGM on the basis of some ranking approach for four releases. To supplement, we have used normalized criteria approach for ranking the SRGMs on the basis of five comparison attributes. Making use of Normalized Criteria Approach as described in Pham, (2014), the appropriate ranking for the proposed SRGMs is obtained. The evaluated weights, their distance, thus obtained are presented in Table-4.

7. NORMALIZED CRITERIA DISTANCE METHOD

In this segment, "we discuss a method called NCD, for ranking and selecting the best model from among SRGMs based on a set of criteria taken all together with considerations of criteria weight" $\psi_1, \psi_2 \ldots\ldots \psi_d$.

Let α denotes the number of software reliability models with δ criteria, and σ_{ij} represents the criteria value of ith model of jth criteria where i =1, 2,..., σ and j =1,2...., δ . The NCD value, D_k , measures the distance of the normalized criteria from the origin for kth model" (Pham, 2014)

$$D_k = \sqrt{\left(\sum_{j=1}^{\delta}\left(\left(\frac{\sigma_{kj}}{\sum_{i=1}^{\sigma}\sigma_{ij}}\right)^2\right)\alpha_j\right)} \quad ; k = 1, 2, ..., \sigma$$

σ *and* δ are overall digit of models and sum numeral of criteria, respectively, and α_j indicate weight of the measure j where $j = 1, 2, ..., \delta$.

So, "the smaller NCD value, D_k , it represents the better rank as compare to higher NCD value. In proposed modeling, we use four comparison criteria such as Mean Square Error, Bias, variation, Root Mean square Prediction Error, to illustrate the NCD method.

Table 4: Model ranking

Model	MSE	BIAS	Variation	RMSPE	Weights	Distance	Rank
R-I	0.60291729	0.45078225	0.27149705	0.27460426	0.39995	0.632416	4
R-II	0.00129933	0.00197812	0.01314197	0.0128336	0.007313	0.085518	1
R-III	0.01742751	0.06361505	0.04840776	0.0485647	0.044504	0.210959	3
R-IV	0.00307614	0.00101764	0.02081964	0.02025315	0.011292	0.106262	2

The above table clearly portrays that modeling for Release II (convolution of two exponential distributions) has attained first rank on implementing the normalized criteria distance approach. Hence, out of the four releases, Release-II performs much better as compared to other releases.

8. CONCLUSION

In delivering the high quality product reliability plays a pivotal role. Multi release of software has proven to increment its reliability. In our proposed models, total fault elimination of recent release is evaluated on the support of informative bugs of prior one. While modeling the successive releases of the software, we have considered the interaction between the errors remaining in the just prior release and the present one. In this paper, we have used detection-correction phenomenon for joint analysis in different releases. So as to differentiate these two categories we have used convolution of probability distribution function. Further, different type of distributions has been inculcated in the fault removal process. We have reviewed standard distributions such as Exponential and Erlang 2-stage for detection and correction behaviours. The proposed models have produced reliable parameter estimates and goodness fit curve has also been calculated. For further clarity the optimal rank has been calculated using the normalized criteria approach. Release-2 is significantly superior to other three releases.

ACKNOWLEDGEMENT

The present work has been made possible with the grant provided to first and fourth author through DST Purse Phase II, Scheme, Department of Science and Technology, India.

RECEIVED: NOVEMBER, 2019.
REVISED: JUNE, 2020

REFERENCES

[1] ANAND, A., SINGH, J., KAPUR, P.K. and DAS, S. (2014) : Modeling conjoint effect of faults testified from operational phase for successive software releases. In **Proceedings of the 5th international Conference on Life Cycle Engineering and Management (ICDQM),** 83-94.

[2] ANAND, A., SINGH, O. and DAS, S. (2015) : Fault severity based multi up-gradation modeling considering testing and operational profile. **International Journal of Computer Applications**, 124.

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

[4] KAPUR, P. K., SINGH, O., GARMABAKI, A. S. and SINGH, J. (2010a) : Multi up-gradation software reliability growth model with imperfect debugging. **International Journal of System Assurance Engineering and Management**, 1, 299-306

[5] KAPUR, P.K., TANDON, A. and KAUR, G. (2010b) : Multi up-gradation software reliability model. In **Reliability, Safety and Hazard (ICRESH),** 468-474, I**EEE**.

[6] KAPUR, P. K., PHAM, H., GUPTA, A. and JHA, P.C. (2011a) : **Software reliability assessment with OR applications**. Springer, London.

[7] KAPUR, P.K. and KUMAR, V. (2011b) : Testing resource allocation for fault detection and correction processes under dynamic environment. In **The Proceedings of National Conference on Computing for Nation Development** (**INDIACOM-2011, New Delhi**), 331-336.

[8] KAPUR, P.K., ANAND, A. and SINGH, O. (2011c) : Modeling successive software up-gradations with faults of different severity. In **Proceedings of the 5th national conference, INDIACom** , 351-356.

[9] KAPUR, P. K., SINGH, J. N., and SINGH, O. (2015) : Application of multi attribute utility theory in multiple releases of software. **International Journal of System Assurance Engineering and Management,** 6, 61-70.

[10] LO, J.H. and HUANG, C.Y. (2006). : An integration of fault detection and correction processes in software reliability analysis. **Journal of Systems and Software**, 79, 1312-1323.

[11] MUSA, J.D., IANNINO, A. and OKUMOTO, K. (1987) : **Software Reliability: Measurement, Prediction, Application**.

[12] PHAM, H., 2006. Software Reliability Modeling. In **System Software Reliability**, 153-177.

[13] PHAM, H. (2014) : Loglog fault-detection rate and testing coverage software reliability models subject to random environments. **Vietnam Journal of Computer Science,** 1, 39-45.

[14] RANDY, J.R. (2009) : "What is a Convolution?" **http://www.structmed.cimr.cam.ac.uk/Course/ Convolution /convolution.html.**

[15] SAS, S. (2004) : **STAT user guide, version 9.1. 2**. SAS Institute Inc, Cary, NC.

[16] SCHNEIDEWIND, N.F. (1975) : Analysis of error processes in computer software. **In ACM Sigplan Notices**, 10, 337-346. **ACM**.

[17] SINGH, O., KAPUR, P.K. and ANAND, A. (2011) : A stochastic formulation of successive software releases with faults severity. **In Industrial Engineering and Engineering Management (IEEM), International Conference**, 136-140. **IEEE**.

[18] SINGH, O., KAPUR, P.K., KHATRI, S.K. and SINGH, J.N.P. (2012) : Software Reliability Growth Modeling for Successive Releases. **Proceeding of 4th International Conference on Quality. Reliability and Infocom Technology (ICQRIT),** 77-87.

[19] SUN, H.W., (2002) : Analysis of Costs and Delivery Intervals for Multiple-release Software (Doctoral dissertation, **New Jersey Institute of Technology, Department of Industrial and Manufacturing Engineering.**

[20] XIE, M., HU, Q.P., WU, Y.P. and NG, S.H. (2007) : A study of the modeling and analysis of software fault detection and fault correction processes. **Quality and Reliability Engineering International,** 23, 459-470.

[21] YAMADA, S., OHBA, M. and OSAKI, S. (1984) : S-shaped software reliability growth models and their applications. **IEEE Transactions on Reliability**, 33, 289-292.

[22] YAMADA, S., NISHIGAKI, A. and KIMURA, M. (2003) : A stochastic differential equation model for software reliability assessment and its goodness-of-fit. **International Journal of Reliability and Applications,** 4, 1-11.