iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IV **Month of publication:** April 2017

DOI: http://doi.org/10.22214/ijraset.2017.4180

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Brief Review of Software Reliability Prediction Models

Boby John[1], Rajeshwar S Kadadevaramath[2], Immanuel A Edinbarough[3]
[1]SQC & OR Unit, Indian Statistical Institute, Bangalore
[2]Department of Industrial Engineering & Management, Siddaganga Institute of Technology, Tumkur
[3]Department of Manufacturing and Industrial Engineering, University of Texas, USA

Abstract: ***Software plays an important role in every field of human activity today varying from medical diagnosis to remote controlling spacecraft. Hence it is important for the software to provide failure-free performance whenever needed. The Information technology industry has witnessed rapid growth in the recent past. The competition among the firms also increased. The software organization in the developing countries like India can no longer survive on cost advantage alone. The software companies need to deliver reliable and quality software on time. A lot of research has been carried out on software quality management and reliability estimation. The objective of this paper is to provide a brief review of the major research contribution in the field of software reliability and identify the future research areas in software reliability estimation and prediction***

Keywords: software reliability growth models, nonhomogeneous Poisson process models, s-shaped models, imperfect debugging

I. INTRODUCTION

Many organizations utilize information technology (IT) to improve productivity, enhance operational efficiency, responsiveness, etc [1] As a result, the IT industry has witnessed tremendous growth in the past few decades. As the number of information technology companies increased, the competition among them also increased. The software organization in the developing countries like India can no longer survive or grow based on cost advantage alone. But delivering reliable and quality software on time within budgeted cost is a challenge for many organizations [2], [3]. Many times the companies would compromise on software testing and release the software with residual defects. This would make the software unreliable.

The software reliability is defined as the probability of failure-free operation of a software system for a specified time in a specified environment [4]. The failure of the software during operations can lead to customer dissatisfaction, loss of market share, etc. The failure of a software used in the medical device or that used in air traffic control system can have a disastrous effect on the individual as well as society. Hence it is imperative for the software firms to ensure their product is sufficiently reliable before releasing the software for usage. This paper is a brief review of the important developments happened in the field of software reliability and identifies the future research areas.

The remaining part of this article is arranged as follows: the session II describes the literature review methodology, the literature review analysis is given in session III and the conclusion are discussed in session IV.

II. LITERATURE REVIEW METHODOLOGY

A lot of articles have been presented at conferences, published in journals and books have been written in the last few decades on software reliability estimation and prediction. The aim of this paper is to provide a brief review of the important researches carried on developing software reliability models. The process started with searching for relevant published articles. The scope of the review is limited to the published books and papers published in journals and important conference proceedings. The databases searched are IEEE explore, Science direct, Google scholar and research gate. Two hundred and nine papers are identified for review. After reading the abstract, ninety-seven papers are shortlisted for review. Another twenty-nine papers are later dropped as the content is not directly related to the focus area of the review. Finally, sixty-eight papers are included in the review. The details are given in fig 1.

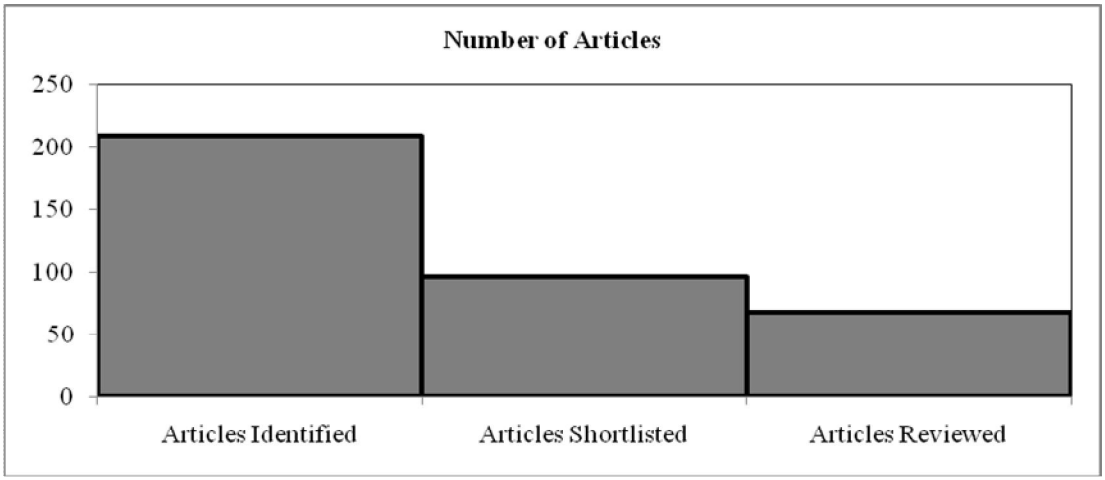


Fig 1. Details of articles shortlisted for review

III. LITERATURE REVIEW ANALYSIS

Most of these research resulted in developing software reliability prediction models. These models are based on the data from the software testing phase. Most of the aforementioned models are mathematical functions expressing the relationship between the number of faults detected and the effort spent on testing. The testing effort can be measured using calendar time, execution time or the number of test cases executed. The models can be used for predicting the software reliability as well as the number of faults remaining undetected (residual defects) in the software. The models can thus help on deciding when to stop testing and release the software. The basic assumption underlying software reliability estimation is that as testing progresses, more and more faults get detected. As a result, the number of residuals defects decreases and the reliability increases. Hence these models are called software reliability growth models (SRGM).

The widely popular class of software reliability growth models are the nonhomogeneous Poisson process (NHPP) models. The NHPP models try to identify a nonhomogeneous Poisson process model which best fits the fault detection pattern as software testing progresses and use the best fit model to estimate the software reliability or residual defects. Most of these models have a parameter indicating the expected total number of faults in the software [5]. The reliability growth models provides a mathematical equation to estimate the number of defects detected m(t) at time t

$$m(t) = aF(t) \quad (1)$$

where time t is often measured in terms of testing effort and a is the parameter representing the expected total number of faults. Whenever the cumulative number of faults detected during testing is equal or close to the expected total number of faults, it is an indication that more or less all the faults injected to the software are detected and the software can be released. The difference between the number of faults detected and the number of faults predicted by the model at the time of software release gives the number of residual faults still present in the software. Hence the software reliability models can also be used for determining when to stop testing and release the software.

The software reliability growth models are broadly classified into two groups namely nonhomogeneous Poisson process models and stochastic process models. Out of the two classes of models, the NHPP models are more popular and widely used. The NHPP models are further classified based on a variety of criteria. The important among them are

A. The classification based on test effort measurement as testing time based and the number of test cases based models.

B. Models based on the shape of the reliability or mean value function *m(t)* as exponential and s-shaped models. The s-shaped models can be further divided into delayed s-shaped and inflation s-shaped models

C. Models based on the type of assumption on debugging as models incorporating perfect debugging and those with imperfect debugging.

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

The classification of software reliability growth models is given in fig 2.

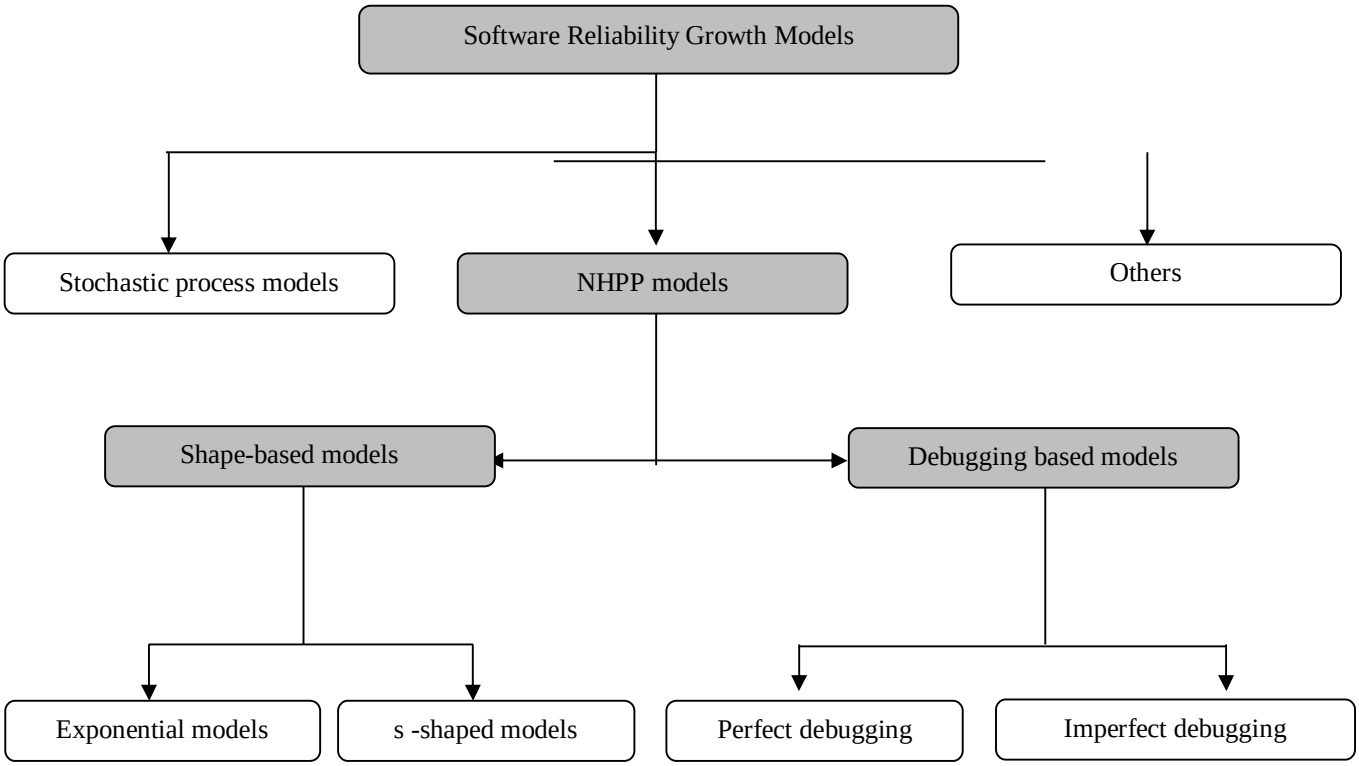


Fig 2: Classification of Software Reliability Growth Models(SRGMs)

The NHPP model assumes that the cumulative number of failures detected till time *t*, *[N(t), t* ≥ 0] as a nonhomogeneous Poisson process with failure rate or intensity function *λ (t)*. Then the probability or chance that *N(t) = k*, where *k* is an integer value is [6], [7] given by

$$P[N(t)=k]=\frac{[m(t)^{k}]}{k!}e^{-m(t)}, k = 0, 1, 2, - - - \quad (2)$$

where *m(t)* is the mean value function representing the expected number of failures till time *t* and is given by

$$m(t)=\int_{0}^{t}\lambda(s)ds \quad (3)$$

Similarly, the intensity function *λ(t)* can be computed from *m(t)* as

$$\lambda(t)=\frac{dm(t)}{dt} \quad (4)$$

In the case of stochastic process models, the assumption is that a fixed unknown number n_0 faults are injected to the software during development, which need to be detected and fixed to make the software perfectly reliable. As testing progresses, these faults will get detected and fixed. According to Jelinski and Moranda [8], the expected number of faults at time *t* or mean value function *m(t)* is given by

$$m(t)=n_0[1-e^{-\phi t}] \quad (5)$$

where ϕ is the constant hazard rate. The corresponding failure intensity function is given by

$$\frac{dm(t)}{dt} = \phi[n_0 - m(t)] \quad (6)$$

The important papers discussing the stochastic process models are given in table 1.

TABLE 1. List of Important articles on stochastic process models

SL No.	Articles
1	Jelinski and Moranda [8]
2	Schick and Wolverton [9]
3	Shanthi Kumar [10]
4	Goel [11]
5	Littlewood [12]

The NHPP models develop functions to estimate the expected number of faults at time *t* or mean value function *m(t)* in terms of testing effort. The test effort can be measured using calendar time, execution time or the number of test cases executed. The models based on the number of test cases executed are called discrete time models. The important articles discussing the discrete time models are given in table 2.

TABLE 2. List of articles on discrete time models

SL No.	Articles
1	Brooks and Motle [13]
2	Yamada and Osaki [14]
3	Kapur et al. [15] - [17]

One of the ways used to group the NHPP model is based on the shape of the mean value function or fault detection curve. Many researchers assumed the fault detection curve will be an exponential curve and developed exponential NHPP models to predict software reliability. Later on, researchers found that during the bug fixing phase of software development life cycle, the programmer's knowledge of the software product improves and as a result the fault detection curve will be most likely an s-shaped rather than exponential. Those researchers developed a group of NHPP models called s-shaped models. The important research work on exponential and s-shaped NHPP models are given in table 3

. TABLE 3. Shape-based models

SL No	Shape	Articles
1	Exponential	Musa [18], Schneidewind [19], Goel and Okumoto [20], Yamada et al. [21], Hossain [22], Xie and Zhao [23]
2	s-shaped	Yamada et al. [24], Ohba [25], Ohba and Yamada [26], Ohba [27], Bittanti et al. [28], Kareer et al. [29], Kapur and Garg [30], Yamada et al. [31], Kimura et al. [32], Pham [33], Lee et al [34]

Another popular way of classifying the NHPP models is based on the assumption on debugging. Many researchers assume that as soon as a fault is detected, the work on fixing it will commence. The fix will be perfect and no new fault will be introduced while fixing the bugs. The models with such assumption on perfect debugging are classified as perfect debugging models. But many situations the debugging will not be perfect. There is the possibility of introducing new faults while fixing the bugs. Many models incorporate this scenario also. Such models are classified as imperfect debugging models. The important research papers published on perfect and imperfect debugging assumption are given in table 4.

TABLE 4: Models classified on debugging assumption

SL No	Debugging Assumption	Articles
1.	Perfect Debugging	Dalal and Mallows [35], Schneidewind [36] - [39]
2	Imperfect Debugging	Ohba and Chou [40], Kapur and Garg [41], Pham [42], Zeephongsekul [43], Gokhale et al.[44], Pham et al. [45], Yamada and Sera [46], Gokhale et al. [47], [48]

A lot of research work on NHPP models which will not fall strictly under the aforementioned classifications. The important among them are given in table 5.

TABLE 5: Other Important articles on NHPP models

SL No.	Article
1	Musa and Okumoto [49]
2	Xie [50]
3	Kenney [51]
4	Leemis [52]
5	Farr [53]
6	Xie and Wohlin [54]
7	Mullen [55]
8	Gokhale and Trivedi [56]
9	Bishop and Bloomfield [57]
10	Kapur et al [58]

Recently many researchers have attempted to develop software reliability prediction models using machine learning techniques also. The important among them are given in table 6.

TABLE 6: Software reliability models using machine learning techniques

SL No.	Technique Used	Article
1	Fuzzy logic	Utkin et al. [59]
2	Artificial Neural Networks	Cai et al. [60], Tian and Noore [61], Kapur et al [62], Zheng [63],
3	Genetic Programming	Afzal and Torkar [64]

Since a large number of software reliability models have been proposed, the research on comparison and unification of the models started from 1980's itself. The major attempts on comparing the different software models are given in table 7.

TABLE 7: Articles on comparison and unification of the software reliability models

SL No.	Articles
1	Yamada and Osaki [65]
2	Langberg and Singapurwalla [66]
3	Abdel_Ghaly et al. [67]
4	Miller [68]
5	Huang et al [69]
6	Gokhale [70]
7	Kapur et al [71]

Even though a lot of models are proposed, no single model is suitable for predicting the reliability for all types of software cases. Moreover, identification of the best-suited model from the available models is also a challenge. Even different models may come up

as the best model based on the criteria used to select the best model.

IV. CONCLUSION

This paper provided a brief review of the important research happened in the field of software reliability estimation and prediction. Since software development is a human activity, it is almost impossible to eliminate the injection of faults during software development. Hence it is required to detect and fix as many faults as possible before releasing the software for use. The purpose of software testing is to detect as many faults as possible. As more and more faults are getting detected and fixed through testing, the number of faults remaining in the software decreases and software reliability increases. A lot of software reliability growth models have been developed in the past to estimate the software reliability. These models can also be used for deciding when to stop testing and release the software.

The development of a large number of software reliability models itself is an indication that there is no single model suitable for estimating the reliability of all types software. So one of the future areas of research can be developing a single model suitable for all kinds of scenarios or development of a unified or common model which can take different forms suitable for different types of the software.

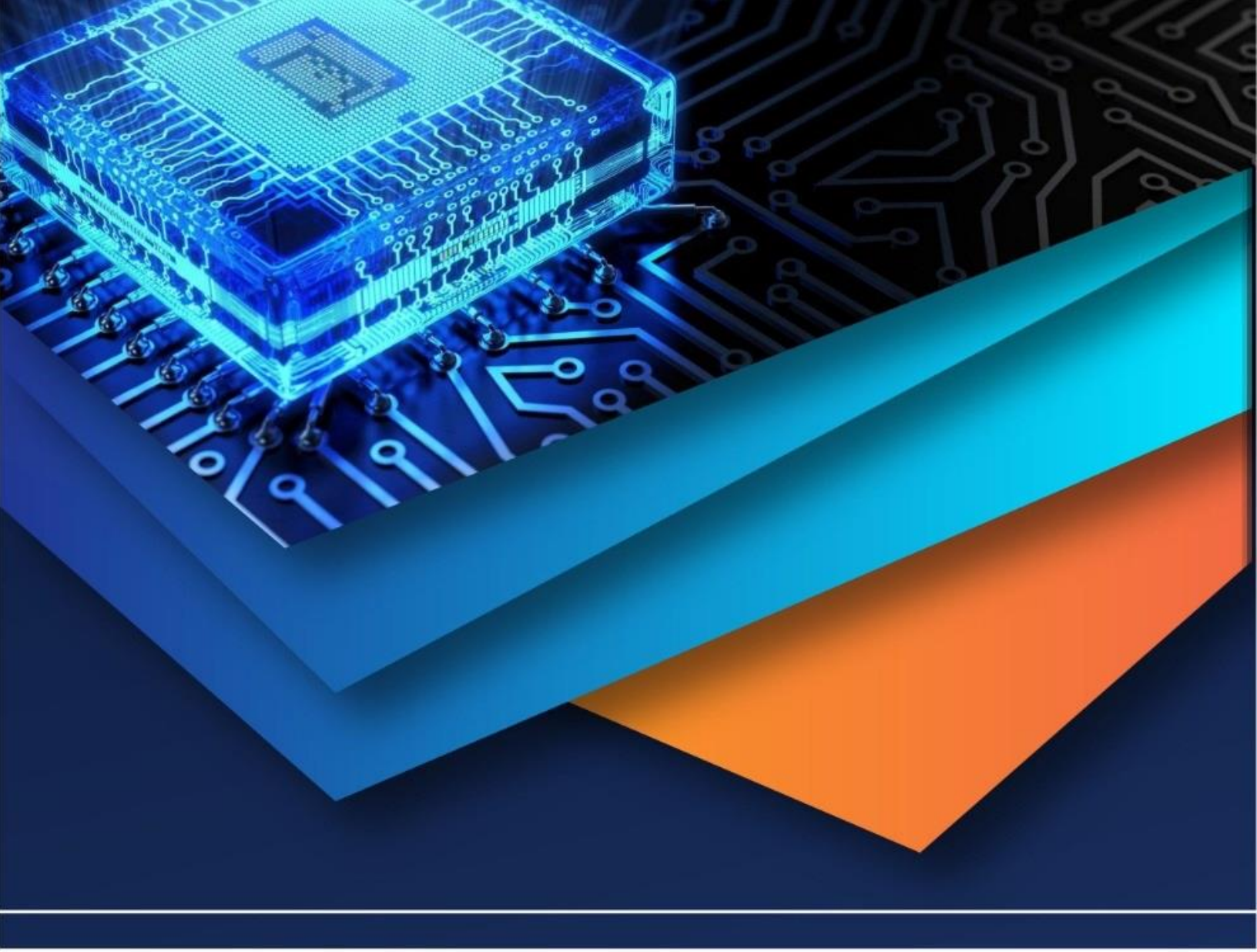
Another important research area can be identifying the best-suited model for a given situation. Multiple evaluation criteria have been suggested for identifying the best fit model. Many times, different criteria may suggest different models as the best-fit model for a given situation. Hence the development of a methodology to identify the best model suited for a given case based on multiple criteria can be another area of future research.

REFERENCES

[1] J. G. Mooney, V. Gurbaxani, and K. L. Kraemer, "A process oriented framework for assessing the business value of information technology," ACM SIGMIS Database, Vol. 27, No. 2, pp : 68-81, 1996.
[2] M. J. Pearson, C. S. McCahon, and R. T. Hihgtower, "Total quality management: are information systems managers ready?" Information and Management, Vol. 29, No. 5, pp: 251 – 263. 1995.
[3] D. D. Phan, J. F. George, and D. R.Vogel, "Managing software quality in a very large development project", Information & Management, Vol. 29, No. 5, pp: 277-283. 1995.
[4] J. Pan, Software Reliability. Carnegie Mellon University, available at: https://users.ece.cmu.edu/~koopman/des_s99/sw_reliability/. 1999.
[5] A. Wood, "Software Reliability Growth Models", Tandem Technical Report, Vol. 96, No.130056, 1996.
[6] H. Pham, and X. Shang, "NHPP software reliability and cost models with testing coverage", European Journal of Operational Research, Vol. 145, pp: 443 - 454, 2003.
[7] R. Lai, and M. Garg, "A detailed study of NHPP software reliability models", Journal of Software, Vo. 7, No. 6, pp. 1296 - 1306, 2012.
[8] Z. Jelinski, and P. B. Moranda, "Software reliability research", Computer Performance Evaluation. (ed,) Freiberger, pp. 465 - 484, 1972.
[9] G. J. Schick, and R, W. Wovlerton, "An analysis of competing software reliability models", IEEE Transactions on Software Engineering, Vol. 4, No. 2, pp: 104 - 120, 1978.
[10] J. G. Shanthikumar, "Software reliability models: A review", Microelectronics Reliability, Vol. 23, pp: 903 - 949, 1983.
[11] A. L. Goel, "Software reliability models: Assumptions, limitations, and applicability", IEEE Transactions on Software Engineering, Vol. 12, pp:1411 1423, 1985.
[12] B. Littlewood, Software Reliability Modelling and Identification. Springer-Verlag, pp: 141 - 209, 1987.
[13] W. D. Brooks, and R. W. Motley, Analysis of Discrete Software Reliability Models. IBM federal systems div Gaithersburg MD, 1980.
[14] S. Yamada, and S. Osaki, " Discrete software reliability growth models", Applied Stochastic Models and Data Analysis, Vol. 1, pp: 65 - 77, 1985.
[15] P. K. Kapur, M. Xie, R. B. Garg, and A. K. Jha, "A discrete software reliability growth model with testing effort", 1st International Conference on Software Testing, Reliability and Quality Assurance, 1994.
[16] P. K. Kapur, S. Younnes, and S. Agarwala, "A general discrete software reliability growth model", International Journal of Modelling and Simulation, Vol. 18, No. 1, pp: 60 - 65, 1998.
[17] P. K. Kapur, R. B. Garg, and S. Kumar, Contributions to Hardware and Software Reliability. World Scientific, Singapore, 1999.
[18] J. D. Musa, "A theory of software reliability and its applications", IEEE Transactions on Software Engineering, Vol. 1, No.3, pp: 312 - 327, 1975.
[19] N. F. Schneidewind, "Analysis of error processing in computer Software", Sigplan Notices, Vol. 10, pp: 337 - 346, 1975.
[20] A. L. Goel, and K. Okumoto, "Time-dependent error detection rate model for software reliability and other performance measures", IEEE Transactions on Reliability, Vol. 28, No. 3, pp: 206 - 211, 1979.
[21] S. Yamada,, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing-effort", IEEE Transactions on Reliability, Vol. 35, No.1, pp: 19-23, 1986.
[22] S. A. Hossain, and R. C. Dahiya, "Estimating the parameters of a non-homogeneous Poisson-process model for software reliability", IEEE Transactions on Reliability, Vol. 42, No.4, pp: 604-612, 1993.

[23] M. Xie, and M. Zhao, "On some reliability growth models with simple graphical interpretations", Microelectronics Reliability, Vol. 33, No. 2, pp: 149 - 167, 1993.
[24] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection", IEEE Transactions on Reliability, Vol. 32, No. 5), pp: 475-484, 1983.
[25] M. Ohba, "Software reliability analysis models", IBM Journal of Research and Development, Vol. 28, No.4, pp: 428-443, 1984.
[26] M. Ohba, and S. Yamada, "S-shaped software reliability growth models", In 4th International Colloquium on Reliability and Maintainability, Tregastel, France (pp: 430-436), 1984.
[27] M. Ohba, "Inflection S-shaped software reliability growth model". In Stochastic Models in Reliability Theory (pp. 144-162). Springer Berlin Heidelberg, 1984.
[28] S. Bittanti, P. Bolzern, E. Pedrotti, M. Pozzi, and R. Scattolini, "A flexible modelling approach for software reliability growth", Software Reliability Modelling and Identification, pp.101-140, 1988.
[29] N. Kareer, P. K. Kapur, and P. S. Grover, "An S-shaped software reliability growth model with two types of errors", Microelectronics Reliability, Vol. 30, No.6, pp:1085-1090, 1990.
[30] P. K. Kapur, and R. B. Garg "A software reliability growth model for an error-removal phenomenon", Software Engineering Journal, Vol. 7, No.4, pp: 291-294, 1992.
[31] S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment", International Journal of Systems Science, Vol. 23, No. 12, pp: 2241-2252, 1992.
[32] M. Kimura, S. Yamada, and S. Osaki, "Software reliability assessment for an exponential-S-shaped reliability growth phenomenon", Computers & Mathematics with Applications, Vol. 24, No.1-2, pp: 71-78, 1992.
[33] H. Pham, "Recent studies in software reliability engineering", In Handbook of Reliability Engineering (pp. 285-302). Springer London, 2003.
[34] C. H. Lee, Y. T. Kim, and D, H. Park, "S-shaped software reliability growth models derived from stochastic differential equations",. IIE Transactions, Vol. 36, No. 12, pp:1193-1199, 2004.
[35] S. R. Dalal, and C. L. Mallows, "Some graphical aids for deciding when to stop testing software", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 2, pp:169-175, 1990.
[36] N. F. Schneidewind, "Software reliability model with optimal selection of failure data", IEEE Transactions on Software Engineering, Vol. 19, No. 11, pp: 1095-1104, 1993.
[37] N. F. Schneidewind, " Modelling the fault correction process", In Proceedings of 12th IEEE International Symposium on Software Reliability Engineering, ISSRE 2001, pp: 185-190, November, 2001.
[38] N. F. Schneidewind, "An integrated failure detection and fault correction model", In Proceedings of IEEE International Conference on Software Maintenance, pp: 238-241, 2002.
[39] N. F. Schneidewind, "Assessing reliability risk using fault correction profiles", In Proceedings of Eighth IEEE International Symposium on High Assurance Systems Engineering, pp: 139-148, March, 2004.
[40] M. Ohba, and X. M. Chou, "Does imperfect debugging affect software reliability growth?", In Proceedings of the 11th ACM International Conference on Software Engineering, pp. 237-244, May, 1989.
[41] P. K. Kapur, and R. B. Garg, "Optimal software release policies for software growth model under imperfect debugging", Operations Research (RAIRO), Vol. 24, pp: 295–305, 1990.
[42] H. Pham, "Software reliability assessment: imperfect debugging and multiple failure types in software development", EGandG-RAAM-10737, Idaho National Engineering Laboratory, 1993.
[43] P. Zeephongsekul, G. Xia, and S. Kumar, "Software-reliability growth model: primary-failures generate secondary-faults under imperfect debugging", IEEE Transactions on Reliability, Vol. 43, No.3, pp: 408-413, 1994.
[44] S. S. Gokhale, P. N. Marinos, M. R. Lyn, and K. S. Trivedi, "Effect of repair policies on software reliability", In Proceedings of the 12th IEEE Annual Conference on Computer Assuranc, (COMPASS'97), pp: 105-116, June, 1997.
[45] H. Pham, L. Nordmann, and Z. Zhang, "A general imperfect-software-debugging model with S-shaped fault-detection rate", IEEE Transactions on Reliability, Vol. 48, No. 2, pp:169-175, 1999.
[46] S. Yamada, and K. Sera, "Imperfect debugging models with two kinds of software hazard rate and their Bayesian formulation", Electronics and Communications in Japan (Part III: Fundamental Electronic Science), Vol. 84, No. 3, pp: 12-20, 2001.
[47] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi, "Analysis of software fault removal policies using a non-homogeneous continuous time Markov chain", Software Quality Journal, Vol. 12, No. 3, pp: 211-230, 2004.
[48] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi, "Incorporating fault debugging activities into software reliability models: A simulation approach", IEEE Transactions on Reliability, Vol. 55, No.2, pp.281-292, 2006.
[49] J.D. Musa, and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", In Proceedings of the 7th IEEE international conference on Software engineering. Pp: 230-238, March, 1984.
[50] M. Xie, Software Reliability Modeling. World Scientific, Singapore, 1991.
[51] G. Q. Kenny, "Estimating defects in commercial software during operational use", IEEE Transactions on Reliability, Vol. 42, No.1, pp: 107-115, 1993
[52] L. M. Leemis, Reliability: probabilistic models and statistical methods. Prentice-Hall, Inc, 1995.
[53] W. Farr, "Software reliability modeling survey", Handbook of software reliability engineering, pp.71-117, 1996.
[54] M. Xie, and C. Wohlin, "A study of the exponential smoothing technique in software reliability growth prediction", Quality Control and Applied Statistics, Vol. 43, pp: 587-588, 1998.
[55] R. E. Mullen, " The lognormal distribution of software failure rates: origin and evidence", In Proceedings of the Ninth IEEE International Symposium on

Software Reliability Engineering, pp: 124-133, November, 1998.
[56] S. S. Gokhale, and K. S. Trivedi, "A time/structure based software reliability model". Annals of Software Engineering, Vol. 8, No. (1-4), pp: 85-121, 1999.
[57] P. G. Bishop, and R. E. Bloomfield, "Using a log-normal failure rate distribution for worst case bound reliability prediction", In 14th IEEE International Symposium on Software Reliability Engineering (ISSRE 2003), pp: 237-245, November, 2003.
[58] P. K. Kapur, A. Kumar, K. Yadav, and S. K. Khatri, "Software reliability growth modelling for errors of different severity using change point", International Journal of Reliability, Quality and Safety Engineering, Vol. 14, No. 0), pp: 311-326, 2007.
[59] L. V. Utkin, S. V. Gurov, and M. I. Shubinsky, "A fuzzy software reliability model with multiple-error introduction and removal", International Journal of Reliability, Quality and Safety Engineering, Vol. 9, No. 03, pp: 215-227, 2002.
[60] K. Y. Cai, L. Cai, W. D. Wang, Z. Y. Yu, and D. Zhang, "On the neural network approach in software reliability modeling", Journal of Systems and Software, Vol. 58, No. 1, pp: 47-62, 2001.
[61] L. Tian, and A. Noore, "Software reliability prediction using recurrent neural network with Bayesian regularization", International Journal of Neural Systems, Vol. 14, No. 3, pp: 165-174, 2004.
[62] P. K. Kapur, S. K. Khatri, and M. Basirzadeh, "Software reliability assessment using artificial neural network based flexible model incorporating faults of different complexity", International Journal of Reliability, Quality and Safety Engineering, Vol. 15, No. 2, pp: 113-127, 2008.
[63] J. Zheng, "Predicting software reliability with neural network ensembles", Expert Systems with Applications, Vol. 36, No. 2, pp: 2116-2122, 2009.
[64] W. Afzal, and R. Torkar, "On the application of genetic programming for software engineering predictive modeling: A systematic review", Expert Systems with Applications, Vol. 38, No. 9, pp:11984-11997, 2011.
[65] S. Yamada, and S. Osaki, "Software reliability growth modeling: Models and applications", IEEE Transactions on Software Engineering, Vol. 12, pp: 1431-1437, 1985.
[66] N. Langberg, and N. D. Singpurwalla, "A unification of some software reliability models", SIAM journal on scientific and statistical computing, Vol. 6, No. 3, pp: 781-790, 1985.
[67] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions", IEEE Transactions on Software Engineering, Vol. 9, pp: 950-967, 1986.
[68] D. R. Miller, "Exponential order statistic models of software reliability growth", IEEE Transactions on Software Engineering, Vol. 1, pp:12-24, 1986.
[69] C. Y. Huang, S. Y. Kuo, M. R. Lyu, and J. H. Lo, "Quantitative software reliability modeling from testing to operation", In Proceedings of 11th IEE International Symposium on Software Reliability Engineering(ISSRE 2000), pp: 72-82, 2000.
[70] S. S. Gokhale, "Architecture-based software reliability analysis: Overview and limitations", IEEE Transactions on Dependable and Secure Computing, Vol. 4, No. 1, 2007.
[71] P. K. Kapur, H. Pham, S. Anand, and K. Yadav, "A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation", IEEE Transactions on Reliability, Vol. 60, No.1, pp: 331-340, 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY