Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 62 (2015) 360 – 369

Procedia Computer Science

The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

A Parametric Empirical Bayes Model to predict Software Reliability Growth

Néstor R. Barraza*

Department of Engineering, Universidad Nacional de Tres de Febrero, Caseros, Argentina

Abstract

A new software reliability model based on the empirical Bayes estimate is developed. The number of failures estimated up to a given time is used in order to estimate the probability of failure appearance during the next time interval. Instead of a non homogeneous in time failure rate as it is usually used to model reliability growth, a failure rate depending non linearly on the previous number of failures is obtained from our model. The estimate is obtained from a mixed Poisson model where the mixing probability density function models the reliability growth. The model can be used either to simulate the cumulative failures curve or to estimate the time between failures. Data of a similar project can be used to estimate the parameters of a given project. Results of simulations and estimated mean time between failures comparing well with experimental data are also shown.

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)

Keywords: Software Reliability; Empirical Bayes; Simulation of software failures; Probability of failure

1. Introduction

The interest in Software Reliability has been increasing since the first models were proposed in the 70's. Several statistical models involve Non-homogeneous Poisson, Compound Poisson, clusters and Markov chains among others, see References [1,2,3,4]. There were also analyzed particular methods of estimation. These researches show that Software Reliability has become an important application of statistics.

Software Reliability models are generally stochastic processes that intend to introduce metrics on the failure detection process like MTBF and MTTF or the number of remaining failures at any time. Those metrics allow to evaluate the software development and testing processes in order to assign testing resources or to predict the release time. The characteristic of the failure detection process as a function of time shows a decreasing rate as testing time progresses. In order to follow this behavior, several software reliability models were proposed in the literature, the so called software reliability growth models. Non homogeneous and Compound Poisson processes among others were proposed decades ago as software reliability growth models.

* Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: nbarraza@untref.edu.ar

Peer-review under responsibility of organizing committee of The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015)
doi:10.1016/j.procs.2015.08.416

On the other hand, besides statistical software reliability models, simulations of the stochastic software failures detection process is an important research issue related to testing and debugging, see References [5,6,7].

The Bayes estimation framework has also been an important statistical method of estimation. Bayesian estimation has been largely applied to Software Reliability since the pioneer paper listed in Reference [8]. Estimation of the mean time between failures using several priors distribution for the parameters were proposed in Reference [9,10]. Other approaches involve the use of combinatorial optimization algorithms like Metropolis and Gibbs sampler in order to calculate the posterior distribution, see References [9,11]. Since the Bayes method lies on the number of previous reported failures, this procedure leads to the Empirical Bayes estimate.

The empirical Bayes estimate is obtained from the Bayes rule applied to the conditional expectation of the unknown parameter given the samples. The conditional expectation minimizes the mean square error function as it is well known. Several applications can be found in different areas of engineering, specially in speech recognition and word processing, Refs [12,13]. Many simple forms of this estimate were proposed decades ago.

In this work, a new software reliability growth model to predict the failure cumulative curve and the mean time between failures behavior using information at the very start of the project is proposed. We propose to estimate the probability of failure in the next interval time given r failures are detected before from the Empirical Bayes method. Our model allows on one hand to simulate the stochastic failure detection process and to estimate the mean time between failures based on the number of previous reported failures on the other. From a Montecarlo simulation of the estimated probability, our model allows to simulate the stochastic failure process. Prediction of Software Reliability at an early stage is quite important in order to adjust testing resources or to estimate the release time.

Since a nonlinear dependence on r is required for software failures in order to follow the software failure cumulative curve, we use the expression introduced in Reference [14]. Parameters of the model are estimated in two ways, from previous reported data of the same project and using a mixing probability density function that fits well the cumulative failures curve of a similar project. The use of software reliability information of similar projects in order to estimate the reliability of a given project is a common practice, see References [15,16,17,18,19].

This paper is organized as follows: The Empirical Bayes estimate with a nonlinear characteristic is shown in section (2), the theoretical foundations that supports our main motivation are presented in section (3), discussions on choosing the mixing distribution are presented in section (4), a simulation and comparison with experimental data is shown in section (5), an application of our model to predict the mean time between failures is presented in section (6), conclusions are presented in section (7).

2. The Empirical Bayes estimate using mixed distributions

A well known non parametric estimate was originally proposed in Reference [20] as an alternative to the maximum likelihood, the so called Good-Turing estimate. An Empirical Bayes form of the non parametric Good-Turing estimate based on a mixed binomial distribution was proposed in Reference [12]. From a general expression of the Empirical Bayes estimate using mixed distributions, either mixed binomial or mixed Poisson, that method was later generalized in [14] in order to get a given family of estimates.

In order to get the estimate, we start to consider the distribution of r statistically independent single events in n outcomes. This distribution is given by the binomial.

$$P(r) = \binom{n}{r} \theta^r \, (1-\theta)^{n-r} \tag{1}$$

From the Poisson approximation valid for large values of n and $r \ll n$, we obtain a Poisson distribution with parameter:

$$\lambda = \theta \, n \tag{2}$$

Thus, using (2), we can get an estimate for θ from an estimate for λ. Being $S(\lambda)$ the prior probability density function of λ, we get for r a mixed Poisson distribution:

$$P(r) = \int \frac{\lambda^r}{r!} \exp(-\lambda) \, dS(\lambda) \, . \tag{3}$$

The λ estimate is obtained from an Empirical Bayes framework as:

$$\hat{\lambda} = E[\lambda|r]. \tag{4}$$

From (2), (3) and (4), and by applying the Bayes rule, the θ estimate results:

$$\hat{\theta} = \frac{1}{n} \frac{\int \lambda \frac{\lambda^r}{r!} \exp(-\lambda)\, dS(\lambda)}{\int \frac{\lambda^r}{r!} \exp(-\lambda)\, dS(\lambda)} \,. \tag{5}$$

Some estimates well known in the literature of natural language processing can be obtained from (5) by choosing properly $S(\lambda)$, (see Reference[14] for details). Despite the last approximation was obtained for a binary probability θ valid for $r \ll n$, it can be considered as a general definition of a given estimate.

The last equation can be written as:

$$\hat{\theta} = \frac{r+1}{n} \frac{P(r+1)}{P(r)} \,. \tag{6}$$

The expression $(r+1)\,\frac{P(r+1)}{P(r)}$ for the estimate of the Poisson parameter λ is well known in statistics, see Reference[21]. Our novel proposal is to use this expression in order to obtain an estimate for the binary probability θ.

We model the failure detection process using the probability (6). We estimate the probability of a failure detection in the next interval time given r failures were detected up to the current discrete time n. From (6), the introduction of the smoothing nonlinear factor $\frac{P(r+1)}{P(r)}$ must be noted. The effect and necessity of the smoothing factor in software reliability is analyzed in the next section. Despite we can estimate θ directly from the binomial (1), we prefer the Poisson approximation since simple mathematical forms are obtained for complex mixing distributions $S(\lambda)$, and the mixed Poisson distribution has been widely studied in statistics, see References[21,22]. On the other hand, Poisson usually arises to count failures from considering that inter-failure times are exponentially distributed. The condition $r \ll n$ is usually accomplished in software reliability as it will be discussed next. Instead of considering the mixed Poisson, the application of the binomial and also non parametric techniques in order to get an estimate of the probability of failures, could also be an interesting matter of study.

3. Software failures prediction using the Empirical Bayes estimate

The cumulative number of software failures collected during the testing phase of a software product has several characteristics. We expect an increasing number of failures per time unit at first, and a low rate of failures at the end, after some corrections were introduced. We expect a decreasing number of remaining failures. The release time will be determined by a sufficiently long mean time to failure. Well known software reliability growth models based on the non homogeneous Poisson process were proposed in the 1970s, see a revision in References[1,23,24], a detailed analysis and revision is presented in Reference[25]. There were also other models proposed like that based on a Compound Poisson process, this model was originally proposed in Reference[26] and reviewed in References[27,28]. All of these models try to follow the cumulative number of failures curve by estimating the number of remaining failures from the number of failures previously reported. A different approach is proposed in this work, instead of predicting the total number of remaining failures or the time to next failure, we propose to estimate the probability of a failure arrival during the next time interval[1]. This time interval could be either, calendar time, execution time or testing stage.

In order to apply the Empirical Bayes estimate to software failures, we propose to estimate the probability of a failure arrival during the next time interval given that r failures have appeared before. In this approach, no more than one failure per time interval is allowed. Since time is usually recorded in seconds, this requirement is accomplished. In case that failures are reported in days or weeks, they can be distributed in seconds along the time interval in order to apply our model. We summarize next the assumptions of the model which can be compared with those proposed in Reference[10].

[1] The probability of failure gives also as a consequence an estimate of the MTBF, as it will be shown in section 6.

General assumptions

(a) Failures are randomly distributed over discrete time intervals.
(b) In each time interval a failure can occur with probability θ dependent on time.
(c) Each time interval has just one failure or no failure, no more than one failure can occur in each time interval.
(d) Given the probability of failure at any time interval, failures at each time interval are statistically independent.
(e) The probability of failure at each time interval is a random variable generated by some specified parametric distribution.
(f) Given the parameters of the distribution in (e), the failure probabilities at each time interval are statistically independent.

As described by the general assumptions, this process is a special case of a pure birth process were the system is in state r at time n if r failures have occurred up to time n, with transition probabilities given by (6), see Reference[29]. Pure birth processes commonly arise in simulations of reliability studies, see for example the non homogeneous continuous time Markov chain process analyzed in Reference[5].

Then, using (6) we estimate the probability of failure in the next time interval given that r failures were reported in the previous n intervals. The smoothing factor introduced in (6) plays an important role in software failures, since for an unimodal probability $P(r)$ we have an increasing probability of failure for values of r lower than the mode as $P(r+1) > P(r)$, and, conversely, we have a decreasing probability of failure for values of r above the mode since $P(r+1) < P(r)$. Modeling this way the reliability growth property of software failures. This behavior can be controlled by choosing properly the mixing probability density function $S(\lambda)$.

Related approaches were presented in References[9,10,11], where the empirical Bayes is applied to estimate the mean time between failures modeled by a mixed exponential distribution. Our concern is to estimate directly the probability of failure given by the expression (5), i.e., we propose to estimate the binary probability of failure occurrence from an empirical Bayes framework.

4. Choosing the mixing distribution

Since we are using the parametric form of $P(r)$ in (6), we have to choose the mixing distribution $S(\lambda)$ and to estimate its parameters in order to perform the prediction model mentioned above. An alternative could be to choose arbitrarily or estimate the pdf $S(\lambda)$ and proceed to estimate the parameters as long as failures are reported. Since this could be a complex task depending on the form of $S(\lambda)$, we propose to choose the mixing pdf as that which fits the predicted failure cumulative curve of a similar project. Then, we consider that similar projects share the mixing pdf but they have different starting points, it means, different failures rate at the start of the corresponding phase. Since the prior mixing pdf models the failures rate behavior, like the point when this rate starts to decrease, our assumption seems to be reasonable since this characteristic can be shared by similar projects. A gamma prior for the failures rate was chosen in Reference[9]. Other criteria for choosing priors for non homogeneous Poisson processes were proposed in References[30,31].

In our experiments, the Generalized Inverse Gaussian probability function used in insurance and queuing models, given by:

$$S(\lambda) = \frac{\mu^{-\alpha}\ \lambda^{\alpha-1}\ \exp(-(\lambda^2+\mu^2)/2\beta\lambda)}{2K_\alpha(\mu\,\beta^{-1})}\ , \tag{7}$$

was arbitrarily chosen as the mixing distribution. In (7) μ, α and β are parameters, and K_α is the Modified Bessel Function of the third Kind, see Reference[22] for details on the Poisson mixed by an Inverse Gaussian distribution.

5. Simulation of the Stochastic Failure Process

5.1. Simulation Procedure

As it was previously explained, we proceed to simulate failures production from a starting point obtained experimentally. Thus, taking the time interval up to the, for example, first two reported failures, we estimate the probability θ of failure arrival during the next time interval. From eq. (6) we get for the starting point:

$$\hat{\theta}_0 = \frac{r_0 + 1}{s_o} \frac{P(r_0 + 1)}{P(r_0)} , \tag{8}$$

where $r_0 = 2$ and s_o is the time up to the first two failures arrival. From this starting point, we proceed to perform a Montecarlo simulation of the probability (6). Then, the simulation progresses by generating a failure with probability θ in the next time interval, if a failure is produced, the value of r is increased by one, else, keeps its value. Whatever the result of the failure generation, the loop progresses to a new time interval with an updated value of θ given by (6). The time n is always taken in seconds in order to perform the simulation according to two requirements: no more than one failure per time interval is allowed on one hand, and a considerable number of iterations must be performed in order to produce failures according to their probability on the other. A pseudo code of the simulation procedure is shown next.

Algorithm 1 Simulation of the stochastic failure process

```
procedure FAILURES_SIMULATION
    r ← r_0                                          ▷ initial number of failures
    s ← s_0                                          ▷ Time to the r_0th failure
    for i ← s to the total testing time do
        p = ((r + 1)/i) (P(r + 1)/P(r))              ▷ Probability of failure
        if FAILURE_ARRIVAL(p) = True then
            r ← r + 1
        end if
        print arrived failures r and elapsed time i
    end for
end procedure
function FAILURE_ARRIVAL(p)
    x ← a uniform random number between 0 and 1
    if x < p then                                    ▷ Montecarlo
        return True
    else
        return False
    end if
end function
```

We propose then to simulate a single realization of the stochastic failure process. Our procedure is also an interesting alternative to rate-based simulation techniques, see References[6,7] for details.

5.2. Application to Real Data

In order to test the goodness of fit of the proposed model we present two simulations based on actual data for comparison. Failures data based on execution time were collected by Prof. John D. Musa for several projects, they are available in Reference[32]. Despite these data were saved decades ago when the interest in Software Reliability began, they are still relevant in order to see the software failures behavior and to test models since they show the main characteristic in software reliability, i.e. the reliability growth. Because of this, these datasets keep useful and

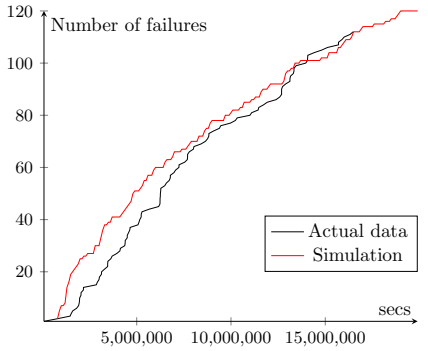

Fig. 1. Actual and simulated cumulative failures curve for the SS1A project.

are still used in recent publications (as an example, DS1 and DS2 data were used in Reference[7]). An interesting discussion on datasets and the use of software failures reports available on the web for Open Source projects is presented in Reference[33]. New software failures data collections and adaptation of the SR models to modern software development methodologies, are needed in order to improve software quality metrics, see for example the analysis presented in Reference[34].

Results of simulations are shown in Fig. 1 for project SS1A from the mentioned dataset. Data were taken during the operational phase and corresponds to an operating systems of hundred of thousands of instructions. We have chosen these data since they show the software reliability growth characteristic. Software reliability growth with an inflection point is usually modeled with the S-shaped stochastic model. The parameters of the Inverse Gaussian mixing distribution were also arbitrarily chosen as those which fit better the actual data: $\beta = 0.5$, $\mu = 25$, $\alpha = 1$.

From Fig. 1, we can see that the simulated curve fits well the actual data all the time and follows the curvature.

Since the only necessary testing information of the project is such at the first two failures, we conclude that failures production can be modeled well enough by information at the start of the project. It seems to indicate that data at the start of the project together with the mean value of the mixing distribution $S(\lambda)$ in (5) have a lot of information in order to predict the future behavior. All simulations have been performed using Mathematica v.8. Despite the complex form of the Inverse Gaussian distribution, the whole simulation takes several minutes in an Intel© I3 processor. Then, we can expect that whatever the mixing distribution, this procedure run fast enough.

As it was pointed out, we can use the mixing distribution that fits well the cumulative failures curve of a similar project to predict the actual curve. Following this idea, we simulate the cumulative failures curve using the mixing distribution of the SS1A project but starting from different time to the first two failures s_o. Results are shown in Fig. 2, where some interesting characteristics must be noted. We can see that the mixing pdf determines the shape of the curve. Despite the starting points are quite similar, the curves separate more as testing progresses. Another interesting remark is to note that the curves exhibit inflection points or jumps. These jumps are more notorious for more reliable projects, those with time to the first two failures $s_o = 1000000$ and $s_o = 1500000$.

6. Mean Time Between Failures Prediction

The Time Between Failures can be estimated from the distribution of no failures run length of a Bernoulli variable with probability (6), which results in a geometric distribution of parameter θ. As it is well known, the geometric distribution is the discrete counterpart of the continuous exponential distribution. Then, the probability of having time between failures of length k is given by:

$$P(k) = (1-\theta)^k\,\theta \quad k = 0, 1, 2, \cdots \tag{9}$$

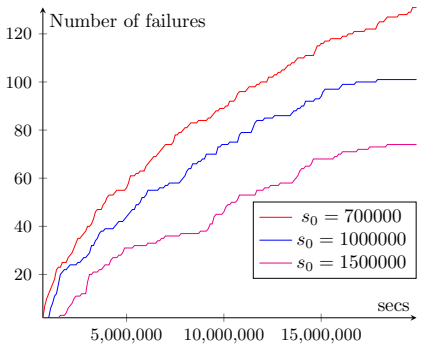

Fig. 2. Simulated curves using the mixing pdf of SS1A project with different times to the first two failures s_o.

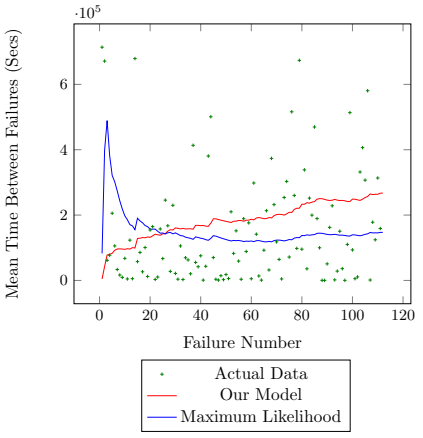

Fig. 3. Mean Time Between Failures curves for the SS1A project.

with mean value:

$$E[k] = \frac{1-\theta}{\theta} \tag{10}$$

From (6) and (10) and taken into account that $\theta \ll 1$, having reported r failures at time n, the mean time between failures estimate results:

$$\widehat{mtbf} = \frac{n}{r+1}\,\frac{P(r)}{P(r+1)} \tag{11}$$

A more exact calculation should take into account the adaptation of probabilities as no failures are obtained in the run. However, because of the decreasing law $\frac{1}{n}$ of θ, the average run length calculated this way diverges. Then, the estimate (11) must be considered as a lower bound.

We show in Fig. 3 the estimated MTBF for the SS1A project. In order to see the effect of the smoothing factor, the maximum likelihood estimate is also shown. It is seen that the smoothing factor rises at first and attenuates at the end the probability of failure (conversely for the mean time between failures). The transition point is controlled by the mixing pdf.

As another interesting illustration, we show the predicted mean time between failures for data first reported in Reference[35] and used in References[9,10] and recently in Reference[11]. These data correspond to one module of the Naval Tactical Data System. As in the previous simulations, we proceed to simulate the failures production process

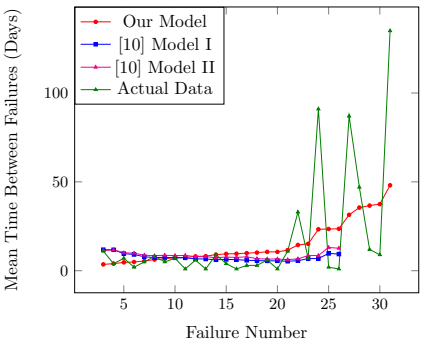

Fig. 4. Mean Time Between Failures for the Naval Tactical Data System dataset.

from a starting point corresponding to the first two failures. The parameters of the mixing Inverse Gaussian distribution were chosen as those which fit well the total number of failures: $\beta = 0.1$, $\mu = 9$, $\alpha = 1$. Results of this simulation are shown in Fig. 4. Actual data and those reported in Reference [10] are also shown for comparison. Since the original data are reported in days, we convert the time unit to seconds in order to perform our simulation.

The values in Fig. 4 are also reproduced in table A.1. We can see that for these type of data having dispersion in the time between failures, our model overestimates the reliability in some cases, though coincides for failures 10 and 21. Our simulation also follows the increasing reliability given by the big jumps in the actual data for failures 22, 24, 27 and 31. Since our model intends to simulate reliability growth, a monotonous increasing mean time between failures must be noted. Data in Reference [10] were reported just for the production phase corresponding to the first 26 failures, the last 5 failures correspond to the test phase. We remark again that as opposed to the other models, our simulation uses only the reported failures at the start of the project instead of performing a progressive estimation based on past reported failures.

It must be pointed out that due to dispersion in the given datasets, the application of metrics like perplexity in order to compare estimates does not produce relevant differences. However, the aim and performance of our estimate are clearly seen in Figs. 3 and 4.

7. Conclusion

A new model in order to predict software reliability has been proposed. This method consists in estimate directly the probability of failure occurrence based on the Empirical Bayes estimate. In order to model reliability growth, a nonlinear smoothing factor was introduced. This factor is based on a mixed Poisson distribution. Our model allows to simulate the failures production curve and the mean time between failures behavior at the very start of the project. A simulation of the cumulative failures curve, using a Poisson mixed by a Generalized Inverse Gaussian probability density function has been performed with good agreement with reported data. Some criteria of choosing the mixing distribution and methods to estimate its parameters will be reported in a future work. Our method allows also to introduce other suitable expressions for the estimate of the probability of failure from the empirical Bayes literature in statistics.

Acknowledgments

This work was supported by Universidad Nacional de Tres de Febrero, Caseros, under Grant No. 62 “Análisis de Modelos Estadísticos aplicados a Confiabilidad, Programacin en la nube, Compresión y Codificación de Datos”.

Appendix A. Data Comparison

Table A.1. Actual and Predicted Time Between Failures

Failure Number	Actual Data	Simulation	[10] Model I	[10] Model II
1	9	-	-	-
2	12	-	-	-
3	11	3.5751	11,84	11,36
4	4	3.98176	11,79	11,77
5	7	4.70847	9,64	10,09
6	2	4.92747	9,14	9,87
7	5	5.47497	7,85	8,74
8	8	6.28757	7,44	8,45
9	5	6.76152	7,55	8,71
10	7	7.36402	7,27	8,5
11	1	7.46922	7,27	8,61
12	6	8.10042	6,66	7,92
13	1	8.20562	6,62	7,93
14	9	9.06154	6,16	7,35
15	4	9.47816	6,39	7,7
16	1	9.58232	6,23	7,5
17	3	9.89478	5,89	7,83
18	3	10.2072	5,71	6,78
19	6	10.6186	5,56	6,55
20	1	10.6148	5,59	6,61
21	11	11.6111	5,35	6,23
22	33	14.4789	5,64	6,68
23	7	15.1591	6,84	8,52
24	91	23.3057	6,86	8,57
25	2	23.4944	9,73	13,1
26	1	23.5887	9,39	12,66
27	87	31.4886	-	-
28	47	35.5323	-	-
29	12	36.6427	-	-
30	9	37.4755	-	-
31	135	48.07	-	-

References

1. Musa, J.D., Iannino, A., Okumoto, K.. *Software reliability - measurement, prediction, application.* McGraw-Hill series in software engineering and technology. McGraw-Hill; 1987. ISBN 978-0-07-044093-7.
2. Sahinoglu, M.. *Trustworthy Computing: Analytical and Quantitative Engineering Evaluation*. Wiley; 2007. ISBN 9780470127865. URL: `http://books.google.com.ar/books?id=AI2sEWj3mw0C`.
3. Tian, J.. Better reliability assessment and prediction through data clustering. *IEEE Trans Software Eng* 2002;**28**(10):997–1007. doi:http://www.computer.org:80/tse/ts2002/e0997abs.htm.
4. Goseva-Popstojanova, K., Trivedi, K.. Failure correlation in software reliability models. In: *Proceedings of the 10th International Symposium on Software Reliability Engineering*; ISSRE '99. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-0443-4; 1999, p. 232–. URL: `http://dl.acm.org/citation.cfm?id=851020.856185`.
5. Gokhale, S.S., Lyu, M.R.. A simulation approach to structure-based software reliability analysis. *IEEE Trans Software Eng* 2005;**31**(8):643–656. URL: `http://dblp.uni-trier.de/db/journals/tse/tse31.html#GokhaleL05`.
6. Lin, C.T., Huang, C.Y., Sue, C.C.. Measuring and assessing software reliability growth through simulation-based approaches. In: *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*; vol. 1. 2007, p. 439–448. doi:10.1109/COMPSAC.2007.141.
7. Lin, C.T., Huang, C.Y.. Staffing level and cost analyses for software debugging activities through rate-based simulation approaches. *IEEE Transactions on Reliability* 2009;**58**(4):711–724. URL: `http://dblp.uni-trier.de/db/journals/tr/tr58.html#LinH09`.
8. Littlewood, B., Verrall, J.. A bayesian reliability growth model for computer software. *Applied statistics* 1973;:332–346.
9. Lynn, K., Tae, Y.Y.. Bayesian computation for nonhomogeneous poisson processes in software reliability. *Journal of The American Statistical Association* 1996;**91**(434):763–773.
10. Mazzuchi, T.A., Soyer, R.. A Bayes Empirical-Bayes Model for Software Reliability. *IEEE Trans on Reliability* 1988;**37**:248–254.
11. Aktekin, T., Caglar, T.. Imperfect debugging in software reliability: A bayesian approach. *European Journal of Operational Research* 2013;**227**(1):112 – 121. URL: `http://www.sciencedirect.com/science/article/pii/S0377221712009083`.

doi:http://dx.doi.org/10.1016/j.ejor.2012.11.056.
12. Nadas, A.. On Turing's formula for word probabilities. *IEEE Trans Acoust Speech and Signal Processing* 1985;**33**:1414–1416.
13. Vaithyanathan, S., Mao, J., Dom, B.. Hierarchical bayes for text classification. In: *PRICAI Workshop on Text and Web Mining*. 2000, p. 36–43.
14. Barraza, N.R.. The empirical bayes estimator and mixed distributions. *AIP Conference Proceedings* 2008;**1073**(1):103–110. URL: http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.3038987. doi:http://dx.doi.org/10.1063/1.3038987.
15. Xie, M., Hong, G., Wohlin, C.. Software reliability prediction incorporating information from a similar project. *Journal of Systems and Software* 1999;**49**(1):43 – 48. URL: http://www.sciencedirect.com/science/article/pii/S0164121299000655. doi:http://dx.doi.org/10.1016/S0164-1212(99)00065-5.
16. Malaiya, Y.K.. Software reliability and security. *Encyclopedia of Library and Information Science* 2005;.
17. Sharma, I., ParveenBano, M.. A combined approach of software metrics and software fault analysis to estimate software reliability. *IOSR Journal of Computer Engineering* 2013;**11**(6):1 – 14.
18. Xie, M., Hong, G.Y., Wohlin, C.. A practical method for the estimation of software reliability growth in the early stage of testing. In: *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*. 1997, p. 116–123. doi:10.1109/ISSRE.1997.630856.
19. Wu, H.L., Zhong, Y., Chen, Y.. A software reliability prediction model based on benchmark measurement. In: *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*. 2010, p. 131–134. doi:10.1109/ICIME.2010.5478245.
20. Good, I.J.. The population frequencies of species and the estimation of population parameters. *Biometrika* 1953;**40**(3-4):237–264. URL: http://biomet.oxfordjournals.org/content/40/3-4/237.abstract. doi:10.1093/biomet/40.3-4.237.
21. Robbins, H.. An empirical bayes approach to statistics. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif.: University of California Press; 1956, p. 157–163. URL: http://projecteuclid.org/euclid.bsmsp/1200501653.
22. Willmot, G.. Mixed Compound Poisson Distributions. *Astin Bulletin* 1986;**16**:59–79.
23. Almering, V., van Genuchten, M., Cloudt, G., Sonnemans, P.J.. Using software reliability growth models in practice. *IEEE Software* 2007;**24**(6):82–88. doi:http://doi.ieeecomputersociety.org/10.1109/MS.2007.182.
24. Sharma, K., Garg, R., Nagpal, C.K., Garg, R.K.. Selection of optimal software reliability growth models using a distance based approach. *IEEE Transactions on Reliability* 2010;**59**(2):266–276. URL: http://dblp.uni-trier.de/db/journals/tr/tr59.html#SharmaGNG10.
25. Stringfellow, C., Andrews, A.A.. An empirical method for selecting software reliability growth models. *Empirical Softw Engg* 2002;**7**(4):319–343. URL: http://dx.doi.org/10.1023/A:1020515105175. doi:10.1023/A:1020515105175.
26. Sahinoglu, M.. Compound-poisson software reliability model. *IEEE Trans Software Eng* 1992;**18**(7):624–630. URL: http://doi.ieeecomputersociety.org/10.1109/32.148480. doi:10.1109/32.148480.
27. Barraza, N.R., Pfefferman, J.D., Cernuschi-Frías, B., Cernuschi, F.. An application of the chains-of-rare-events model to software development failure prediction. In: Keller, H.B., Plödereder, E., editors. *Reliable Software Technologies - Ada-Europe 2000, 5th Ada-Europe International Conference, Potsdam, Germany, June 26-30, 2000, Proceedings*; vol. 1845 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-67669-4; 2000, p. 185–195. URL: http://dx.doi.org/10.1007/10722060_18. doi:10.1007/10722060_18.
28. Barraza, N.R.. Parameter estimation for the compound poisson software reliability model. *International Journal of Software Engineering and its Applications* 2013;**7**(1):137–148.
29. Feller, W.. *An Introduction to Probability Theory and Its Applications, Vol. 1*. Wiley; 3rd ed.; 1968. ISBN 0471257087.
30. Campodónico, S., Singpurwalla, N.D.. A bayesian analysis of the logarithmic-poisson execution time model based on expert opinion and failure data. *IEEE Trans Software Eng* 1994;**20**(9):677–683. URL: http://doi.ieeecomputersociety.org/10.1109/32.317426. doi:10.1109/32.317426.
31. Campodónico, S., Singpurwalla, N.D.. Inference and predictions from poisson point processes incorporating expert knowledge. *JASA Journal of the American Statistical Association* 1995;**90**(429):220–226. doi:10.2307/2291146.
32. Musa, J.D.. Cyber security & information systems. information analysis center. https://sw.thecsiac.com/databases/sled/swrel.php; 1970.
33. Gandy, A., Jensen, U.. A non-parametric approach to software reliability. *Applied Stochastic Models in Business and Industry* 2004;**20**(1):3–15. URL: http://dx.doi.org/10.1002/asmb.510. doi:10.1002/asmb.510.
34. Far, B.. Software reliability engineering for agile software development. In: *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*. 2007, p. 694–697. doi:10.1109/CCECE.2007.178.
35. Jelinski, Z., Moranda, P.. Software reliability research. In: Freiberger, W., editor. *Statistical Computer Performance Evaluation*. New York: Academic Press; 1972, p. 465–484.