Mining Bugzilla datasets with new Increasing Failure Rate Software Reliability models

Néstor Ruben Barraza
Engineering Department
UNTREF
Caseros, Argentina
Email: nbarraza@untref.edu.ar

***Abstract*—A data mining of several Bugzilla datasets using Software Reliability models is presented. We analyzed Bugzilla reports from the Xfce, Firefox, Eclipse and Tomcat projects for a long period of time of several thousand of days. In all the cases, an increasing failure rate have been found. Increasing failure rates are usually modeled by the S-shaped model in the literature. We propose to use some models where the failure rate depends not just on time, but also on the number of failures previously reported. These models are little modifications of the Yule and Polya stochastic processes. The comparison of the actual data with results from simulations is presented. Those results show that many Open Source bugs datasets can be fit by the proposed models.**

***Index Terms*—Software Reliability; Jelinski-Moranda model; Pure-birth processes; Yule process; Polya process; Bugzilla;**

I. Introduction

Software Reliability has become a very important issue in Software Engineering in the last decades. A lot of models have been proposed since the beginning of the specialty in the 70's. There are so many Software Reliability models that it is impossible to group them even in a short list. The most disseminated models are surely those based on non homogeneous Poisson processes. Whatever the model, finding software failures datasets in order to test Software Reliability models is one of the main challenges for researchers. This is mainly due to that many firms are reluctant to deliver their failures reports for the sake of their secrets and security. That is why old datasets are still taken into account in the literature. Despite the analysis of those datasets are still relevant since they capture important characteristics of failures like Reliability Growth or S-shaped, it would be also important to have available recent failures reports detected or introduced in modern software developing and testing environments. These environments should take into account modern techniques introduced in Software Engineering, as opposed to the waterfall model under most of the old datasets where collected. Contrarily to common software projects in the industry, open source projects are not developed or even tested in controlled environments, however, because of their high availability, their bugs repositories, can be used to test Software Reliability models, as it was performed several times in the literature. In this work, we analyze several Bugzilla bugs reports and evaluate how several Software Reliability models fit the bugs curves over the calendar time.

978-1-5386-3057-0/17/$31.00

As related approaches we can mention the following: A similar approach modelling Open Source bugs with non homogeneous Poisson processes has been done in [1] and [2]. Another bug miner model based on support vector machines applied to Open Source projects was proposed in [3]. Also, an analysis of Eclipse bugs reports was presented in [4]. A defect proneses modules prediction on the Eclipse and Windows Vista projects was presented in [5]. A Software Reliability analysis on Open Source projects based on non-parametric approaches was presented in [6]. Bug characteristics in Open Source Software was analyzed in [7].

With our approach, we will show common characteristics and demonstrate that increasing failure rate models achieve to fit most of the analyzed datasets. Since those and many other Software Reliability models including the well known based on the non homogeneous Poisson process are particular cases of Pure Birth stochastic processes, we evaluate those models through a Pure Birth process simulation with different birth (failure) rates. Historically, Software Reliability models were intended to follow a Reliability Growth, it means a decreasing failure rate as a consequence of the failure fixing process, as expected to be obtained in a waterfall, a software development method very popular decades ago, when the first models like those based on non homogeneous Poisson processes were proposed. As opposed to that, our models are intended to follow an increasing failure rate, instead of a Reliability Growth.

Increasing failure rate Software Reliability models can be applied to different cases, like at the very first testing stage, or when new code is added, or in modern software development methodologies where developing is performed simultaneously as testing, as it happens in agile or TDD (Test Driven Development). This opens an interesting research issue in Software Reliability. Increasing failure rate datasets are usually modeled with the S-shaped or Gompertz models, see for example [8, pp. 393-402] and [9]. We show that the new increasing failure rate modified Yule and modified Polya contagious stochastic processes proposed models, describe well the cumulative number of failures and the mean time between failures for the Xfce, Firefox, Eclipse and Tomcat projects. This work is organized as follows: Pure birth processes, their applications to Software

Reliability and the new Pure Birth Process proposed model are described in section II, the simulation process, datasets and results of applications are presented in section III, interesting concluding remarks are discussed in section IV, a validity threats analysis is presented in section V, final conclusions are presented in section VI.

II. Pure Birth Processes in Software Reliability

A. Pure Birth Processes

The application of Pure Birth processes in Software Reliability is not new, recent applications and review can be seen in [10] and [11]. The Pure Birth stochastic differential equation is given by (see also [12]):.

$$P_r'(t) = -\lambda_r(t)P_r(t) + \lambda_{r-1}(t)P_{r-1}(t) \tag{1}$$

where $P_r(t)$ is the probability of having r failures at instant t, and $\lambda_r(t)$ is the failure rate. The well known exponential waiting time arises from (1), where the probability of having no failures in a given time interval $t-s$ given that r failures were detected by the time s is given by:

$$P(\text{no failures in } t \geq t-s) = e^{-\int_s^t \lambda_r(t)dt} \tag{2}$$

This exponential waiting time is what we used in order to simulate the stochastic failure reporting process. The non homogeneous Poisson process based Software Reliability models are special cases of pure birth processes where the failure rate $\lambda_r(t)$ is a non linear function of time. We focus on Software Reliability models based on Pure Birth processes where the failure rate is not just a function of t but also depends on r, or just on r, as it will be explained in the following section.

B. Software Reliability models based on Pure Birth processes

There are a lot of Software Reliability models that can be obtained from a Pure Birth process by choosing different failure rates. Most of them were considered in the literature, like the well known non homogeneous Poisson processes, see for example the Non Homogeneous Continuous Time Markov Chain simulations as performed in [13], the Polya process proposed in [11] or the Jelinski-Moranda geometric de-Eutrophication model analyzed in [14]. Another pure birth stochastic process we can consider as a Software Reliability model is the Yule process, with a birth rate given by $\lambda_r = a\ r$ as described in [12].

Following the probability (2), the mean time between failures when r failures were detected by time s is given by:

$$MTBF(s,r) = \int_s^\infty \lambda_r(t)\ t\ e^{-\int_s^t \lambda_r(t)dt}\ dt \tag{3}$$

Then, the failure creation or detection stochastic processes can be modeled in the same way as the increase of individuals in a given population. The challenge in this case is to find the appropriate failure rate function.

C. The Modified Yule and Polya processes Software Reliability models

As explained before, we consider failure rates that can have a general dependency on the number of previously detected failures. Among several failure rates we tested, we found that a modified Yule process given by:

$$\lambda_r = a\ r^b \tag{4}$$

and a three parameter modified Polya process with a failure rate given by:

$$\lambda_r(t) = \frac{a}{c}\ \frac{1+b\ r}{1+a\ t} \tag{5}$$

fit the best most of all the Bugzilla datasets, as it will be shown.

Besides the Polya stochastic process that comes from the Polya urn model for contagion, see [12] and [11], the Yule process, though it is not truly contagious, has also an increasing failure rate due to an increase in software failures finding, instead of an increase in the probability of failure detection, see [12] and [15] for discussions.

III. Experiments

In order to test the models we perform simulations of the stochastic failure reporting process as given by the Pure birth process equation. The simulation gives a failure after a random waiting time with probability given by (2). At any stage of time t and cumulative number of failures r, we generate a random time interval with the exponential waiting time, we consider that a failure is reported after that random time interval. Then we update the failure rate with the current time and number of failures for the next stage and repeat the process. Getting this way a cumulative number of failures curve. The parameter of the models considered here are chosen as those to fit the actual curves. The simulations were performed using Wolfram Mathematica 8.

As it was explained before, we analyzed several Bugzilla bug reports from Open Source projects because of their advantages for Software Reliability researches. All datasets correspond to a log period of time of thousands of days. We have chosen some projects that were analyzed in the literature before in some cases, others were chosen at random. These datasets and the failure rate of the Modified Yule and Polya processes that fit the corresponding data are analyzed in the following sections. In all the considered data sets, all bug status were taken into account, "fixed", "open", "closed", etc., and we have considered just the creation date in the analysis. We have also considered all reports corresponding to all versions, even those with unspecified versions, and all the Operating Systems. The collected datasets and predicted data were uploaded to *figshare.com*, they can be found and downloaded from [16].

The datasets and the models considered are analyzed in the following subsections. In all the cases we draw the actual and the simulated cumulative number of failures curve and the time between failures. In the last case, besides the actual and the

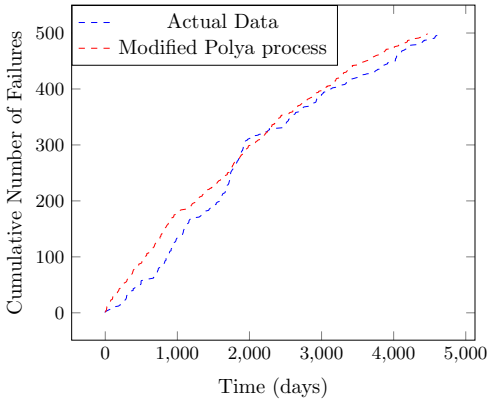

Fig. 1. Actual data and simulation results for Xfce.

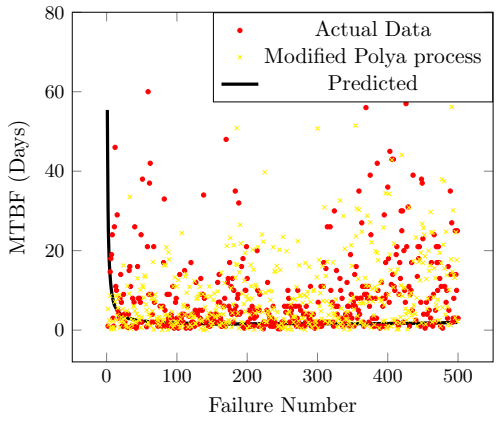

Fig. 2. Actual data and simulated time to failure for Xfce.

simulated points, we show the predicted mean time between failures continuous curve as given by (3)[1].

A. Xfce project

1) Data analysis: Bugs for several components of the Xfce project has been analyzed for a period of one year from Nov 2003 to Nov 2004 in [17]. We analyzed a long period of calendar time since Feb 2004 through July 2016. The actual data curve is almost s-shape, with an increasing failure rate for the first 2000 days and a Reliability Growth for the 3000 rest of failures. The time to failure data shown in fig. 4 do not show any Reliability Growth at any time, the time to failure data are mainly under 20 days and shows a constant dispersion between 20 and 60 days. The time between failures is specially low between failures number 200 and 300, which is shown as a greater than 10 days TBF gap in fig. 4.

2) Proposed model: The cumulative number of failures simulated curve with a Modified Polya process with parameters: $a = 0.003$ days^{-1}, $b = 0.003$ and $c = 0.2$ is shown in fig. 1. It can be seen that despite the simulated curve does not have an s-shape, it follows the curvature and predicts a smooth Reliability Growth at the end. The time between failures curve shown in fig. 2 shows a lot of dispersion, though with points concentrated at time between failures of few days. The model follows the dispersion and concentration points and predicts a mean time between failures of few days, as shown by the continuous curve.

3) Goodness of fit: The cumulative number of failures curves and the time between failures points show that the model fits reasonable well the actual data, we can see that the simulated data follows the curvature. That can be evaluated from the time between failure points shown in fig. 2. From a statistical analysis, we can say that the best quantile[2] corresponding to the 37 % of the total points gives a reliability of 99 % in a χ^2 test. The predicted mean time between failures curve predicts too low values, however, the simulated points show a similar dispersion to the actual data. Taking into account the quite long period of time we are trying to adjust the model, we can say that the goodness of fit of the proposed modified Polya process is reasonable well.

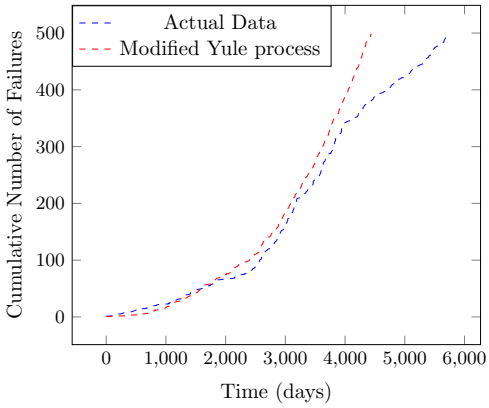

Fig. 3. Actual data and simulation results for Firefox.

B. Firefox

1) Data analysis: The data corresponds to a long period of time from May 2001 to August 2016. Actual and simulated data are shown in fig. 3. We can see again that the cumulative number of failures does not follow a Reliability Growth, but an interesting S shape. Looking at the curve, we can think of applying the S-shaped model, however, it is hard to estimate the parameters in order to get a well goodness of fit because of the big number of failures and the long period of time. The software Reliability model we propose to use in this case is the modified Yule process as explained in the following subsection. The time to failure data shown in fig. 4 show an increasing failure rate from 1/50 to less than 1/10 days^{-1} failures.

[1]Despite the MTBF is a function of two variables, our simulation gives the number of failures as a function of time. Then, we can depict MTBF(r,t(r))

[2]Here and thereafter, best quantile means the set of points having the lowest difference with the actual data.

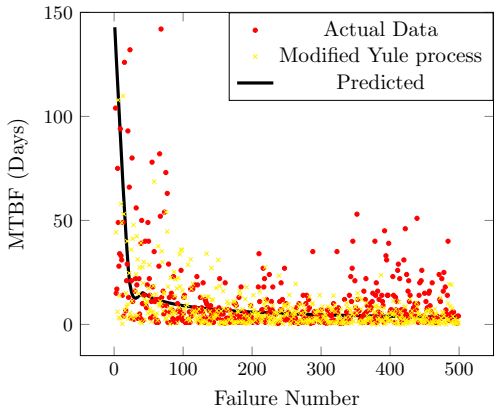

Fig. 4. Actual data and simulation time to failure for Firefox.

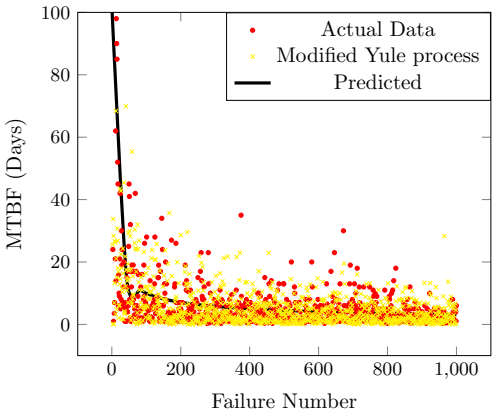

Fig. 6. Actual data and simulation time to failure for Eclipse.

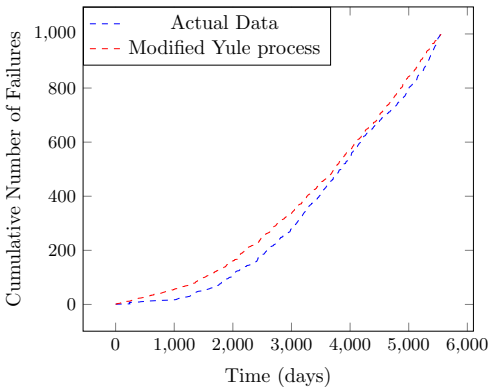

Fig. 5. Actual data and simulation results for Eclipse.

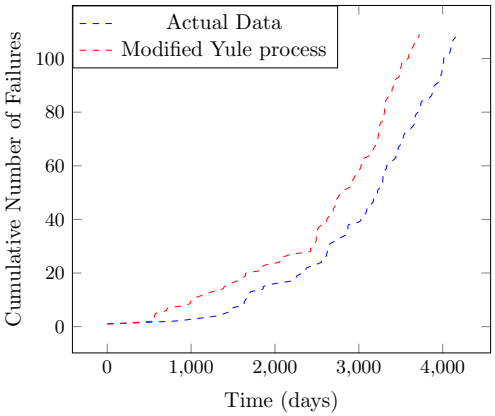

Fig. 7. Actual data and simulation results for Tomcat.

2) Proposed model: We propose to apply in this case the modified Yule process with parameters $a = 0.007$ days^{-1} and $b = 0.6$. Results of the simulation are shown in fig. 3. The model follows the curve until the failure number 400, after that, the model predicts an excessive increasing failure rate. This effect can be also seen from the time between failures points shown in fig. 4.

3) Goodness of fit: Similarly as for the data analyzed before, taking the best 48 % quantile for the MTBF, we get a 97.5 % of χ^2 level of confidence. This fact is due to the proposed model adjusts pretty good the positive curvature of the actual data trough the first 4000 days. Our fit can be compared with that reported in ([1], fig. 2), where Firefox bugs were modeled using non homogeneous Poisson processes during a shorter period of time, our approach gives better results in the increasing failure rate stage.

C. Eclipse

1) Data analysis: This data corresponds to the Eclipse project from October 2001 through December 2016. As shown in fig. 5, we can see again a convex cumulative number of failures curve, with an increasing failure rate. The time to failure picture show in fig. 6, shows an increasing failure rate from 1/60 to less than 1/20 days^{-1}.

2) Proposed Model: A good model for this data appears to be again the modified Yule process with parameters $a = 0.01$ days^{-1} and $b = 0.5$. The model appears to fit well both, the cumulative number of failures curve as well as the time between failures points. Then, we can consider that the predicted time between failures continuous curve shown in fig. 6 gives reliable values, as analyzed in the following subsection.

3) Goodness of fit: Like in the previous dataset, we find that we can get a good level of significance for the time to failure for the best 50 % quantile giving a 90 % of χ^2 level of confidence.

D. Tomcat

1) Data analysis: Bugs reports for the Tomcat projects from May 2005 through December 2016 are shown in fig. 7. The time between failure shown in fig. 8 shows a dispersion starting around 200 and concentrated in almost 80 days.

2) Proposed Model: The parameters for the modified Yule process that fit the best the actual cumulative number of

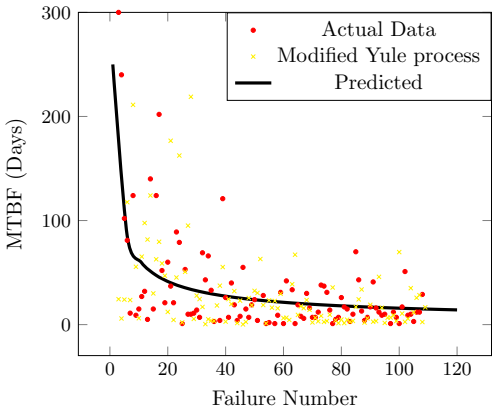

Fig. 8. Actual data and simulation time to failure for Tomcat.

failures results: $a = 0.004 \ \ \text{days}^{-1}$ and $b = 0.6$. The model follows the cumulative number of failures curvature though the simulated curve departs a little from the actual one.

3) Goodness of fit: The model achieves a 90 % of χ^2 level of confidence for the best 30 % quantile.

IV. Concluding Remarks

The analysis performed before raises interesting conclusions.

1) Most of the Open Source projects show an increasing failure rate or an s-shape over a long period of time of thousands of days.
2) They can be modeled with contagious stochastic processes described by Pure Birth Processes.
3) The projects have similar time between failures, which gives similar values of the parameters of all the considered models.

The first remark can be justified due to an increasing failure rate seems to be reasonable since Open Source projects involve a lot of developers and testers, then, new code is constantly added to the project, and a lot of testing is performed. On the other hand, modeling with contagious processes could reveal a contagion phenomenon in the failure creation process. This fact is hard to be considered since developers and testers are not generally connected people located all over the world, though, perhaps, we can consider a contagion phenomenon regarding that some modules of software are concurrently being updated, or reported failures that lead to similar analysis and bugs discovering. This consideration is related to previous analysis of interactions between developers and reporters, see for example [18]. However, it is interesting to analyze which, when and why Open Source projects achieves a Reliability Growth, as it seems to be the case for the three projects (Firefox, Open Office and OpenSuse), well modeled by non homogeneous Poisson processes Software Reliability Growth models, as demonstrated in [1].

In the long period of time we have analyzed, three of the four considered projects show a time between failures of around 20 or 30 days, except the Tomcat project that has a little bigger time between failures of around 40 days. Despite the time between failures is far to be considered constant, it exhibits a dispersion of just 10 to 20 days. This fact could be useful in order to consider some other models for mean time between failures prediction. It is also important to be noticed that excepting the Xfce project, the other three projects start with a time between failures of around of 100 days that suddenly decreases.

TABLE I
Datasets and Predicted failure rates as a function of the number of previously reported bugs r and time t in days.

Project	Failure Rate [days^{-1}]	Time Window
Xfce	$1.5\ 10^{-2}\ \frac{1+0.003\ r}{1+0.003\ t}$	2010-01-20 to 2016-06-15
Firefox	$0.007\ r^{0.6}$	2001-05-01 to 2016-12-08
Eclipse	$0.01\ r^{0.5}$	2001-10-10 to 2016-12-18
Tomcat	$0.004\ r^{0.6}$	2005-08-01 to 2016-12-16

Our findings are summarized in table I. More Open Source projects will be analyzed in future works.

V. Validity Threats

As an empirical research in Software Engineering, Software Reliability should include a validity threats analysis. Regarding the study presented in this work, according to the analysis presented in [19], we can point out the following:

Conclusion validity: Since we have taken a long period of time, little variations of our model from the actual data results in a significant statistical dispersion. Then, we can achieve a statistical significant goodness of fit just for a fraction of the total samples. However, a more accurate model should have a bigger number of parameters and the present study intends to introduce increasing failure rate models in a simple way.

Internal validity: The increasing failure rate might arise from sources other than a contagious phenomenon. Our analysis should be seen as a first approach and an extended study on more datasets should be done as to achieve a stronger conclusion in that sense.

Construct validity: This item is close related to the previous ones, since in order to be sure that the increasing failure rate cases in Open Source projects are well modeled by the proposed contagious processes, we need both, statistical significant goodness of fit models and a great number of datasets.

External validity: Any extrapolation of our conclusions to many Open Source projects mus be supported by mitigation of the previous points, then, our study should be considered as an initial and interesting approach.

Mitigation of the previous mentioned threats can be achieved by extensive studies on more datasets and several adjusting of the proposed failure rate functions, by eventually adding new parameters.

VI. Conclusion

New Software Reliability models with increasing failure rate has been applied to four Bugzilla projects, Xfce, Firefox, Eclipse and Tomcat for a long period of time of thousand of days. We have shown that some of those Open Source bugs reports are well described by a Pure Birth process with the appropriate failure rate that depends just on the number of previously reported bugs. There are also some cases where the failure rate depends also on the calendar time. Contrarily to a Reliability Growth, all the analyzed data show an increasing failure rate where the S-shaped model is usually applied. We have shown that different Open Source projects are well described by a modified Yule process with similar values in the model parameters. A modified Polya contagion process was also introduced. Since these processes can be considered as contagious processes, a contagion phenomena in the failure reporting process seems interesting to be considered, as it was discussed.

Acknowledgment

The author would like to thank Universidad Nacional de Tres de Febrero for support under grant no. 32/15 201.

References

[1] B. Rossi, B. Russo, and G. Succi, "Modelling failures occurrences of open source software with reliability growth," in *Open Source Software: New Horizons - 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010, Notre Dame, IN, USA, May 30 - June 2, 2010. Proceedings*, ser. IFIP Advances in Information and Communication Technology, P. J. Ågerfalk, C. Boldyreff, J. M. González-Barahona, G. R. Madey, and J. Noll, Eds., vol. 319. Springer, 2010, pp. 268–280. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13244-5_21

[2] Y. Tamura and S. Yamada, "Comparison of software reliability assessment methods for open source software," in *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, vol. 2, July 2005, pp. 488–492.

[3] L. Wu, B. Xie, G. E. Kaiser, and R. J. Passonneau, "Bugminer: Software reliability analysis via data mining of bug reports." in *SEKE*. Knowledge Systems Institute Graduate School, 2011, pp. 95–100. [Online]. Available: http://dblp.uni-trier.de/db/conf/seke/seke2011.html#WuXKP11

[4] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, ser. PROMISE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 9–. [Online]. Available: http://dx.doi.org/10.1109/PROMISE.2007.10

[5] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change bursts as defect predictors," in *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, November 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/change-bursts-as-defect-predictors/

[6] A. Gandy and U. Jensen, "A non-parametric approach to software reliability," *Applied Stochastic Models in Business and Industry*, vol. 20, no. 1, pp. 3–15, 2004. [Online]. Available: http://dx.doi.org/10.1002/asmb.510

[7] L. Tan, C. Liu, Z. Li, X. Wang, Y. Zhou, and C. Zhai, "Bug characteristics in open source software," *Empirical Softw. Engg.*, vol. 19, no. 6, pp. 1665–1705, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10664-013-9258-8

[8] W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30 – July 4, 2014, Brunów, Poland*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2014. [Online]. Available: https://books.google.com.ar/books?id=NToqBAAAQBAJ

[9] N. Ahmad and M. Z. Imam, "Article: Software reliability growth models with log-logistic testing-effort function: A comparative study," *International Journal of Computer Applications*, vol. 75, no. 12, pp. 6–11, August 2013, full text available.

[10] N. R. Barraza, "A new homogeneous pure birth process based software reliability model," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 710–712. [Online]. Available: http://doi.acm.org/10.1145/2889160.2892645

[11] ——, "Software reliability modeled on contagion," in *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct 2016, pp. 49–50.

[12] W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 1*, 3rd ed. Wiley, Jan. 1968.

[13] S. S. Gokhale and M. R. Lyu, "A simulation approach to structure-based software reliability analysis." *IEEE Trans. Software Eng.*, vol. 31, no. 8, pp. 643–656, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/tse/tse31.html#GokhaleL05

[14] T. Vasanthi and G. Arulmozhi, "Reliability computation of Moranda's geometric software reliability model," *Economic Quality Control*, vol. 22, no. 2, pp. 261–272, 2007. [Online]. Available: http://EconPapers.repec.org/RePEc:bpj:ecqcon:v:22:y:2007:i:2:p:261-272:n:9

[15] J. Douglas, *Analysis with standard contagious distributions*, ser. Statistical distributions in scientific work series. International Co-operative Pub. House, 1980. [Online]. Available: https://books.google.com.ar/books?id=BQvvAAAAMAAJ

[16] N. R. Barraza, "show_bug.cgi-xxxx.xml," 1 2017. [Online]. Available: https://figshare.com/articles/show_bug_cgi-eclipse_xml/4592446

[17] S. Yamada and Y. Tamura, *OSS Reliability Measurement and Assessment*, ser. Springer Series in Reliability Engineering. Springer International Publishing, 2016. [Online]. Available: https://books.google.com.ar/books?id=uq3WCwAAQBAJ

[18] M. Habayeb, A. Miranskyy, S. S. Murtaza, L. Buchanan, and A. Bener, "The firefox temporal defect dataset," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, May 2015, pp. 498–501.

[19] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.