Journal of Software Engineering and Applications, 2020, 13, 179-190
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Assessment and Prediction of Software Reliability in Mobile Applications

Osama Barack, Liguo Huang

Department of Computer Science, Southern Methodist University, Dallas, TX, USA
Email: obarack@mail.smu.edu, lghuang@lyle.smu.edu

How to cite this paper: Barack, O. and Huang, L.G. (2020) Assessment and Prediction of Software Reliability in Mobile Applications. *Journal of Software Engineering and Applications*, **13**, 179-190.
https://doi.org/10.4236/jsea.2020.139012

Received: August 14, 2020
Accepted: September 18, 2020
Published: September 21, 2020

Abstract

Software reliability is an important quality attribute, and software reliability models are frequently used to measure and predict software maturity. The nature of mobile environments differs from that of PC and server environments due to many factors, such as the network, energy, battery, and compatibility. Evaluating and predicting mobile application reliability are real challenges because of the diversity of the mobile environments in which the applications are used, and the lack of publicly available defect data. In addition, bug reports are optionally submitted by end-users. In this paper, we propose assessing and predicting the reliability of a mobile application using known software reliability growth models (SRGMs). Four software reliability models are used to evaluate the reliability of an open-source mobile application through analyzing bug reports. Our experiment proves it is possible to use SRGMs with defect data acquired from bug reports to assess and predict the software reliability in mobile applications. The results of our work enable software developers and testers to assess and predict the reliability of mobile software applications.

Keywords

Software Reliability Modeling, Mobile Application, Mobile Environment, Bug Report

1. Introduction

Software reliability is a measure for controlling and maintaining the processes of the software development life cycle (SDLC) to develop reliable software. This measure is used during the testing process until the process's exit criteria are met. In addition, software reliability helps to maintain and predict the correctness of the software [1]. Software reliability engineering was introduced to aid in analyzing and measuring the quality of software applications. It presents the quality of the software running without producing defects [2] [3]. Researchers and practitioners have been improving software reliability models to assess the reliability of different types of software.

Measuring and predicting the reliability of a mobile application are real challenges due to the following reasons. First, the nature of mobile environments is different from that of PC and server environments. Second, in mobile environments, new functionalities and features are introduced, such as energy, network, incompatibility, modified and limited Graphical User Interface (GUI), interruption, and notification, which produce new types of defects [4]. Third, mobile operating systems and devices are divers. Fourth, the high demand for mobile applications from users has made the development process fast and the functionality of mobile application more complex [5]. Finally, after a mobile application is released, failures occur in mobile devices. In addition to testing, software developers partially rely on bug reports, which are optionally submitted by end-users.

To assess software reliability in mobile applications, researchers are required to spend more time and effort to evaluate the efficacy of software reliability. Considering the characteristics of mobile applications while measuring their software reliability will produce more accurate results and analyses.

Predicting mobile application failures is as important to software developers as to companies and research organizations. Therefore, we propose measuring the reliability of mobile applications and producing more accurate results based on defects that are extracted from bug reports.

The remainder of this paper is structured as follows: Section 2 presents related work. Section 3 describes the proposed method for measuring and predicting the reliability of a mobile application. Section 4 provides a case study applying several SRGMs to the data sets of an open-source mobile application. Section 5 discusses and analyzes the evaluation and prediction in the reliability of the selected data sets. Finally, Section 6 contains the conclusions and future work.

2. Related Work

Due to the high demand of complex heterogeneous software, software reliability models have become more useful to assess and predict the correctness of the software. Lyu [6] presented software reliability models in practice to help researchers and practitioners quantitatively address the characteristics of the SDLC. In addition, these models guide developers and testers to understand and apply software reliability techniques.

Tian *et al.* [7] evaluated the reliability of web applications after determining their defects and usage. In addition, the possibility of enhancing web application reliability was inspected. The author used the characteristics of web applications as a base to classify web defects. The website workload was measured and characterized at different levels and perspectives, and combined with the failure information about the website to evaluate the operational reliability. The experiment results indicate the efficacy and benefits of the authors' approach.

Many SRGMs for estimating and predicting the reliability of software have been developed and introduced. However, some of these models show inaccurate results, such as delayed S-shape and the exponential type, which indicates that these models may not fit when spending effort that is not constant on testing to detect faults. Therefore, Huang *et al.* [8] reviewed the logistic testing effort function that can be used to describe the amount of testing effort spent on software testing. In addition, the author proposed how to integrate the logistic testing-effort function into software reliability models. The proposed models show more accurate results compared to the traditional SRGMs.

Software as a Service (SaaS) is a software distribution model that is provided through cloud computing. Alannsary and Tian [9] proposed a method for measuring and predicting the reliability of SaaS. The authors analyzed web server log files to extract failure data. The input domain reliability model was used to measure the operational reliability. SRGMs were used to measure the growth in SaaS reliability.

The Application Programming Interface (API) is an interface, which is used to allow clients and servers to interact. Bokhary and Tian [10] proposed a framework for measuring the reliability of APIs. The authors followed a three-stage approach to collect available failure data, and then the API reliability was estimated. In addition, the authors introduced a case study based on Google Map APIs and showed the effectiveness and success of the proposed framework.

New technologies have been added to mobile phones due to the high demand of end-users. Consequently, the predicted field failure rate has deviated from the actual rate. Therefore, developing new methods for predicting the failure rate before the release has become a challenge for researchers and practitioners. Perera [11] presented a reliability prediction method to overcome the inapplicable traditional reliability prediction methods and deliver more accurate results.

The number of lines of code of software applications in mobile devices has increased to millions. Development organizations must produce predictable fault-free software products. Almering *et al.* [12] presented an empirical study to assess the reliability of software and validate SRGMs during the integration and test phases. In addition, the capability of the prediction model was compared to predictions by experts. Moreover, obtaining solid reliability assessment and prediction before software release using SRGMs was shown to be possible.

Researchers have proposed studies to assess the software reliability of mobile operating systems and applications. Ivanov *et al.* [13] presented a comparison between the reliability of three operating systems in mobile environments by applying SRGMs. In addition, Meskini [14] evaluated the reliability of three mobile applications by applying SRGMs to failure data extracted from mobile devices. However, to successfully assess and predict software reliability, the characteristics of mobile environments must be considered. Therefore, to achieve more accurate results in this work, we propose applying software reliability growth models to defect data extracted from bug reports.

3. Methodology

The evolution of mobile software development started with the development of applications for mobile devices with a few thousand lines of code. Today, mobile applications have become more complex due to the high demand from end-users. Software engineering and software quality are involved to ensure the accuracy of the new functionalities and features of mobile applications. This introduces new requirements which need to be considered to improve the stability and reliability of mobile applications.

The nature of mobile environments is different from that of PC and server environments. In addition, mobile application developers rarely share application defect data generated during the testing phase. Therefore, due to the lack of failure data for mobile applications, we propose using bug reports to analyze defect data and measure, assess, and predict application reliability. The proposed method consists of the following:

- Phase 1: Extract mobile application defects from the bug report repository.
- Phase 2: Analyze the bug reports found to discard those that are not related to software reliability, such as defects that originate from the mobile operating system or hardware.
- Phase 3: Weigh each bug report based on its classification as shown in Table 1.
- Phase 4: Relate the date of each bug report to the total number of days since the release day of the mobile application.
- Phase 5: Assess and evaluate the reliability of the selected mobile application and predict its future reliability using SRGMs.
- Phase 6: Use the purification rate and the standard error of the estimate for assurance.

4. Case Study

Measuring and predicting reliability in mobile applications are carried out through the testing stage, collecting defect data from the bug report repository, and applying the most commonly used SRGMs. In this work, we select an open-source mobile application as a case study because its failure data is available for developers and end-users to present the adequacy of the selected SRGMs in mobile applications.

Table 1. Suggested bug report weight.

Importance	*Weight*
Critical	1
High	
Medium	
Low	
Wishlist	0
Undecided	Defect validity ratio

4.1. Defect Data Set

The bug reports are extracted from a web application called Launchpad [15]. This web application provides end-user developers the ability to upload, develop, and maintain open-source software applications. The Launchpad repository has 42,947 projects and 1,779,680 bug reports. Launchpad classifies bug reports based on importance and status. For importance, the bug reports are classified as *critical*, *high*, *medium*, *low*, *wishlist*, *and undecided*. For status, the bug reports are classified as *new*, *incomplete*, *invalid*, *confirmed*, *in progress*, *fix committed*, *fix released*, *under consideration for removal*, *triaged*, *and won't fix*. For this study, we chose an open-source mobile application called Telegram [16] [17] which is a cloud-based instant messaging and voice-over IP service. Telegram in Launchpad has two versions. Version 1 has 114 bug reports from 5 September 2014 to 7 April 2016 and version 2 has 136 bug reports from 8 April 2016 to 4 December 2019.

We weigh each bug report based on its validity. Valid bug reports whose importance levels range from *critical* to *low* are weighted 1. Wishlist bug reports are weighted 0 because they are not considered valid defects. Undecided bug reports could be valid 1 or wishlist 0. Therefore, we calculate the weight of an undecided bug report based on the ratio of the valid bug reports to the total of valid and wishlist bug reports as illustrated in **Table 2** and **Table 3**.

The following equation is used to determine the undecided bug report weight:

$$U_{\text{weight}} = \frac{C+H+M+L}{C+H+M+L+W}, \tag{1}$$

where C is critical, H is high, M is medium, L is low, W is wishlist, and U is undecided.

Table 2. Weighted bug reports: Version 1.

Importance	*Number of bug reports*	*Weight*
Critical	2	1
High	13	1
Medium	22	1
Low	9	1
Wishlist	25	0
Undecided	43	0.65

Table 3. Weighted bug reports: Version 2.

Importance	*Number of bug reports*	*Weight*
Critical	3	1
High	8	1
Medium	11	1
Low	0	1
Wishlist	3	0
Undecided	111	0.88

Applying Equation (1), we get the following ratio of the undecided bug report for version 1 of the Telegram application:

$$\frac{2+13+22+9}{71}=0.65. \tag{2}$$

The ratio of the undecided bug report for version 2 of The Telegram application is calculated as follows:

$$\frac{3+8+11}{25}=0.88. \tag{3}$$

4.2. Modeling SRGMs

The main goal of introducing many software reliability growth models is to assess and analyze reliability growth through software testing and related defect arrival and removal. Non-homogeneous Poisson process (NHPP) software reliability models were developed to overcome the inconsistency of failure occurrence intervals. The NHPP models assume defects that are discovered during the testing phase are removed without introducing new defects, and the mobile application used in the field environment is the same as that used during the testing phase. In this case study, we use the following three commonly used SRGMs and Song's newly proposed model *et al.* [18]:

- The Goel-Okumoto model by Goel and Okumoto [19] is one of the most frequently used of the NHPP models and is defined as:

$$m(t)=N\left(1-\mathrm{e}^{-bt}\right), \tag{4}$$

where N is the estimated total defects, and b is a constant.

- The S-shaped model by Yamada *et al.* [20]) is also an NHPP model, which predicts the cumulative defects in each given time (t) with constants $b>0$ and $N>0$ and is defined as:

$$m(t)=N\left(1-(1+bt)\mathrm{e}^{-bt}\right), \tag{5}$$

where b and N can be estimated from observation data.

- The Musa-Okumoto model by Musa *et al.* [3] is a logarithmic execution time model. This model is a different type of NHPP model and is defined as:

$$m(r)=\frac{1}{\phi}\log\left(\lambda_0\phi_r+1\right), \tag{6}$$

where r is the measurement of the CPU-time execution, λ_0 is the intensity of the initial failure, and ϕ is a model parameter.

- A newly proposed model, which was presented by Song *et al.* [18] to measure software reliability while considering the uncertainty of operating systems and learn-curve in the fault detection rate function, is as follows:

$$m(t)=N\left(1-\frac{\beta}{\beta+\ln\left(\frac{a+\mathrm{e}^{bt}}{1+a}\right)}\right)^{\alpha}, \tag{7}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are constant.

5. Discussion and Analysis

After the selected SRGMs are applied to the weighted defect data sets, the reliability assessment and prediction of the case study are analyzed and discussed.

5.1. Reliability Assessment

After we weight each bug report for a valid defect, we plot the cumulative weighted defects over calendar time for Telegram version 1 and version 2. In the plot, the y-axis represents the cumulative number of defects, and x-axis represents the cumulative arrival day for each defect. Figure 1 shows the distribution of all bug reports, and Figure 2 shows the valid and weighted bug reports over time for both versions of the Telegram application. Version 1 contains 114 bug reports with a total of 76.6 cumulative defects. Version 2 contains 136 bug reports with a total of 119.68 cumulative defects. Figure 3 shows the fitted Goel-Okumoto, S-shaped, Musa-Okumoto, and Song's model on the number of defects over time. Table 4 and Table 5 show the SRGM equation for the total number of defects over time.

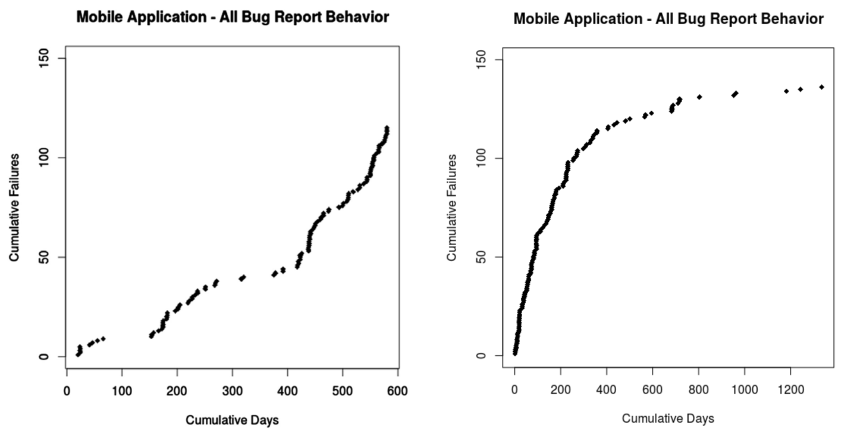

Figure 1. Bug reports distribution over time for version 1 (Left) and version 2 (Right).

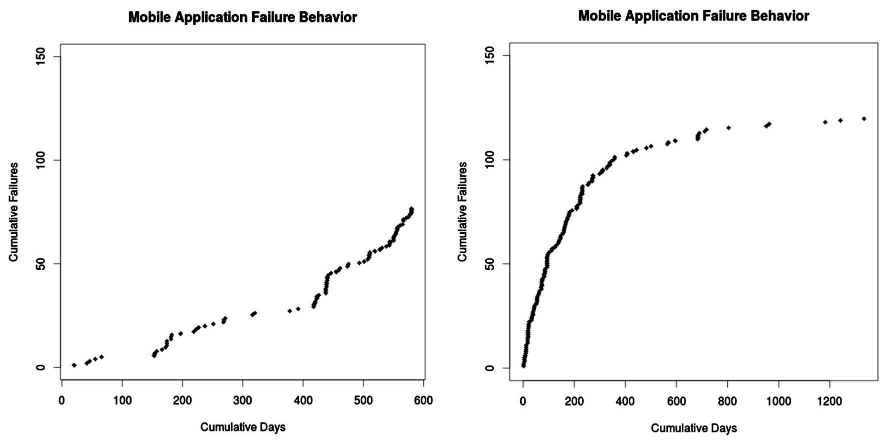

Figure 2. Valid bug reports distribution for version 1 (Left) and version 2 (Right).

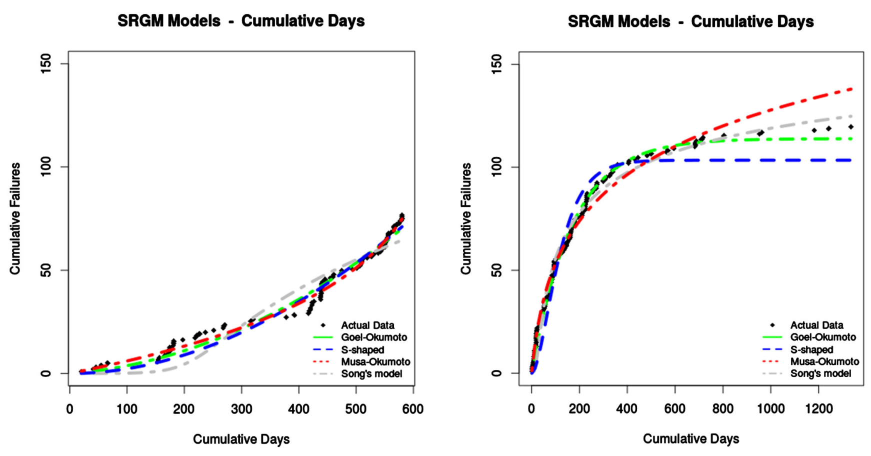

Figure 3. SRGMs fitted on valid failures for version 1 (Left) and version 2 (Right).

Table 4. SRGM equations fitted on valid failures for version 1.

SRGM	Equation
Goel-Okumoto	$\mu(t) = 0.0205675\left[1-\mathrm{e}^{0.0001734t}\right]$
S-shaped	$\mu(t) = 8016\left(1-(1+0.0002404t)\mathrm{e}^{-0.0002404t}\right)$
Musa-Okumoto	$\mu(\tau) = 1/-0.026714\ln\left((0.05578*-0.026714t)+1\right)$
Song's model	$\mu(t) = 1926\left(1-\frac{9}{9+\ln\left(\frac{1926+\mathrm{e}^{0.03675t}}{1+1926}\right)}\right)^{0.03675}$

Table 5. SRGM equations fitted on valid failures for version 2.

SRGM	Equation
Goel-Okumoto	$\mu(t) = 113.9\left(1-\mathrm{e}^{-0.005897t}\right)$
S-shaped	$\mu(t) = 103.43603\left(1-(1+0.01591t)\mathrm{e}^{-0.01591t}\right)$
Musa-Okumoto	$\mu(\tau) = 1/0.02788\ln\left((0.02788*1.23059t)+1\right)$
Song's model	$\mu(t) = 197.211\left(1-\frac{9}{9+\ln\left(\frac{197.211+\mathrm{e}^{0.2705t}}{1+197.211}\right)}\right)^{0.2705}$

As SRGMs are used to measure the growth in software reliability, there is a need to understand and evaluate the change in reliability. This is achieved through calculating the purification level ρ: The closer ρ is to 1, the more reliability growth in the application. When all failures are removed, ρ will become 1. This implies that the greater the ρ value, the more reliability growth we will have [21]:

$$\rho = \frac{\lambda_0 - \lambda_\tau}{\lambda_0} = 1 - \frac{\lambda_\tau}{\lambda_0}, \tag{8}$$

where the initial or peak failure rate for the models is represented by λ_0. The final failure rate is represented by λ_τ.

For further investigation and assurance of the results, we calculate the standard error of estimate (SEOE) to measure the accuracy of the SRGM predictions. The SEOE is defined as follows:

$$\sigma_{est} = \sqrt{\frac{\sum (Y - Y')^2}{N}}, \tag{9}$$

where σ_{est} is the standard error of the estimate, Y is an actual defect, Y' is a predicted defect, and N is the total number of defects. The numerator is the sum of the squared differences between the actual defects and the predicted defects.

Table 6 lists the ρ values of the selected SRGMs for both versions of the mobile application, representing the potential reliability improvement estimated by the SRGMs.

Continuous testing and fault removal will lead to a decrease in the failure rate, or what is referred to as improvement in potential reliability, to be between 89.4% and 98.6% in version 1, which is not that significant. However, the decrease in the failure rate for version 2 is between 93.8% and 99.4%.

Table 6 also shows the SEOE for the SRGMs in version 1 is between 0.28 and 0.775, and version 2 is between 0.288 and 0.735. This indicates the prediction is close to the real data based on the calculated small values as shown in the results.

5.2. Reliability Prediction

To test the prediction of the selected SRGMs, we use the undecided bug report weight for each version as a percentage to select the number of failures. For version 1, we use first 65% of the defects to predict the last 35%. For version 2, we use the first 88% of the defects to predict the last 12%. Figure 4 shows the prediction of the selected SRGMs where the vertical line separates the selected failures from the predicted failures. In addition, Table 7 and Table 8 show the fitted model equations for the selected SRGMs. The results show that the models' predictions are not far from the actual failures. Therefore, developers can use known SRGMs to evaluate and predict the reliability of mobile applications.

Table 6. Purification and SEOE values of selected SRGMs.

SRGM	Version 1		Version 2	
	ρ	*SEOE*	ρ	*SEOE*
Goel-Okumoto	0.972	0.4	0.994	0.228
S-shaped	0.967	0.43	0.938	0.735
Musa-Okumoto	0.986	0.28	0.98	0.423
Song's model	0.894	0.775	0.975	0.469

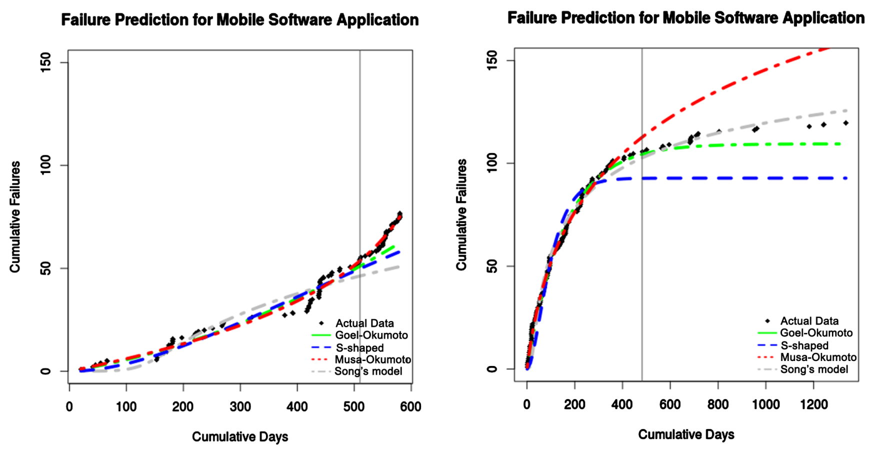

Figure 4. SRGMs fitted on partial valid failures for version 1 (Left) and version 2 (Right).

Table 7. SRGM equations fitted on partial valid failures for version 1.

SRGM	Equation
Goel-Okumoto	$\mu(t) = 0.0455790\left(1 - e^{0.0001071t}\right)$
S-shaped	$\mu(t) = 135.9\left(1 - (1 + 0.002512t)e^{-0.002512t}\right)$
Musa-Okumoto	$\mu(\tau) = 1/-0.02648\ln\left((-0.02648 * 0.05618t) + 1\right)$
Song's model	$\mu(t) = 789.9226\left(1 - \frac{9}{9 + \ln\left(\frac{789.9226 + e^{0.0519t}}{1 + 789.9226}\right)}\right)^{0.0519}$

Table 8. SRGM equations fitted on partial valid failures for version 2.

SRGM	Equation
Goel-Okumoto	$\mu(t) = 109.46932\left(1 - e^{-0.00636t}\right)$
S-shaped	$\mu(t) = 92.79993\left(1 - (1 + 0.01911t)e^{-0.01911t}\right)$
Musa-Okumoto	$\mu(\tau) = 1/0.02134\ln\left((0.02134 * 0.92742t) + 1\right)$
Song's model	$\mu(t) = 200.4941\left(1 - \frac{9}{9 + \ln\left(\frac{200.4941 + e^{0.2669t}}{1 + 200.4941}\right)}\right)^{0.2669}$

6. Conclusions

Software reliability is a measure for controlling and maintaining the development processes with the goal of developing reliable software. Researchers and practitioners have been improving software reliability models to assess the reliability of different types of software. Measuring and predicting the reliability of a mobile application are real challenges due to the differences between the nature of mobile environments, and PC and server environments.

Predicting mobile application failures is as important to software developers as to companies and research organizations. Therefore, to attain an accurate evaluation of software reliability for mobile applications, their characteristics should be considered.

In this work, we proposed measuring the reliability of mobile applications based on defects extracted from bug reports. The proposed process is composed of six steps. First, extract and characterize the bug reports for an open-source mobile application. Second, analyze the bug reports to discard ones that are not related to software reliability. Third, weight the bug reports based on their classification. Fourth, relate the date of each bug report to the total number of days since the release day of the mobile application. Fifth, assess and evaluate the reliability of the selected mobile application and predict its future reliability using SRGMs. Finally, use the purification rate and the SEOE for assurance and provide the fitted SRGM equations.

The results demonstrated that the reliability of mobile applications can be evaluated and predicted using SRGMs through defect data extracted from bug reports. This enables developers to evaluate and predict the reliability of mobile applications.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Shanmugam, L. and Florence, L. (2012) An Overview of Software Reliability Models. *International Journal of Advanced Research in Computer Science and Software Engineering*, **2**, 10.

[2] Muss, J.D., Iannino, A. and Okumoto, K. (1987) Software Reliability: Measurement, Prediction, Application. McGraw-Hill, Inc., Pennsylvania Plaza, New York City.

[3] Musa, J.D., Iannino, A. and Okumoto, K. (1990) Software Reliability. *Advances in Computers*, **30**, 85-170. https://doi.org/10.1016/S0065-2458(08)60299-5

[4] Barack, O. and Huang, L. (2019) Adaptation of Orthogonal Defect Classification for Mobile Applications. *Proceedings of the 28th International Conference on Software Engineering and Data Engineering*, **64**, 119-128.

[5] Vithani, T. and Kumar, A. (2014) Modeling the Mobile Application Development Lifecycle. *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, Vol. 1, Hong Kong, 12-14 March 2014.

[6] Lyu, M.R., *et al.* (1996) Handbook of Software Reliability Engineering. Vol. 222, IEEE Computer Society Press, Washington DC.

[7] Tian, J., Rudraraju, S. and Li, Z. (2004) Evaluating Web Software Reliability Based on Workload and Failure Data Extracted from Server Logs. *IEEE Transactions on Software Engineering*, **30**, 754-769. https://doi.org/10.1109/TSE.2004.87

[8] Huang, C.-Y., Kuo, S.-Y. and Lyu, M.R. (2007) An Assessment of Testing-Effort Dependent Software Reliability Growth Models. *IEEE Transactions on Reliability*, **56**, 198-211. https://doi.org/10.1109/TR.2007.895301

[9] Alannsary, M.O. and Tian, J. (2016) Measurement and Prediction of SaaS Reliability in the Cloud. 2016 *IEEE International Conference on Software Quality, Reliability and Security Companion* (*QRS-C*), Vienna, Austria, 1-3 August 2016, 123-130. https://doi.org/10.1109/QRS-C.2016.20

[10] Bokhary, A. (2017) Measuring Cloud Service Reliability by Weighted Defects over the Number of Clients as a Proxy for Usage. *Proceedings of the 32nd International Conference on Computers and Their Applications* (*CATA*), Honolulu, HI, 20-22 March 2017, 63-70.

[11] Perera, U.D. (2006) Reliability Index—A Method to Predict Failure Rate and Monitor Maturity of Mobile Phones. *RAMS'06. Annual Reliability and Maintainability Symposium*, Newport Beach, CA, 23-26 January 2006, 234-238.

[12] Almering, V., van Genuchten, M., Cloudt, G. and Sonnemans, P.J.M. (2007) Using Software Reliability Growth Models in Practice. *IEEE Software*, **24**, 82-88. https://doi.org/10.1109/MS.2007.182

[13] Ivanov, V., Reznik, A. and Succi, G. (2018) Comparing the Reliability of Software Systems: A Case Study on Mobile Operating Systems. *Journal of Information Sciences*, **423**, 398-411. https://doi.org/10.1016/j.ins.2017.08.079

[14] Meskini, S., Nassif, A.B. and Capretz, L.F. (2013) Reliability Models Applied to Mobile Applications. 2013 *IEEE 7th International Conference on Software Security and Reliability Companion*, Gaithersburg, MD, 18-20 June 2013, 155-162. https://doi.org/10.1109/SERE-C.2013.30

[15] Canonical Ltd. https://launchpad.net/

[16] Herrmann, Tiago Telegram Application. https://launchpad.net/telegram-app

[17] Herrmann, Tiago. https://bugs.launchpad.net/telegram-app

[18] Song, K.Y, Chang, I.H. and Pham, H. (2019) NHPP Software Reliability Model with Inflection Factor of the Fault Detection Rate Considering the Uncertainty of Software Operating Environments and Predictive Analysis. *Symmetry*, **11**, 521. https://doi.org/10.3390/sym11040521

[19] Okumoto, K. and Goel, A.L. (1984) Optimum Release Time for Software Systems Based on Reliability and Cost Criteria. *Journal of Systems and Software*, **1**, 315-318. https://doi.org/10.1016/0164-1212(79)90033-5

[20] Yamada, S., Ohba, M. and Osaki, S. (1983) S-Shaped Reliability Growth Modeling for Software Error Detection. *IEEE Transactions on Reliability*, **32**, 475-484. https://doi.org/10.1109/TR.1983.5221735

[21] Tian, J. (1995) Integrating Time Domain and Input Domain Analyses of Software Reliability Using Tree-Based Models. *IEEE Transactions on Software Engineering*, **21**, 945-958. https://doi.org/10.1109/32.489071