International Journal of Control and Automation
Vol. 11, No. 6 (2018), pp.99-108
http//dx.doi.org/10.14257/ijca.2018.11.6.10

Enhancing the Quality of Software Project Bug Repositories by Early Prediction of Average and Instantaneous Bug Closing Rates

Ekbal Rashid[1] and Mohan Prakash[2]

[1]Associate Professor,
Department of Computer Science and Engineering
Aurora's Technological and Research Institute, Parvathapur, Hyderabad, India
[2]Research Scholar, Jharkhand Rai University, Ranchi, India
[1]ekbalrashid2004@yahoo.com, [2]mpduty@gmail.com

Abstract

The present paper discusses the average and instantaneous rates of bug closing with respect to the bugs being reported. Quadratic regression functions and derivative as rate measurer are used to model the above ideas. The work has been done on primary data from the bug repositories of three major software projects namely Linux Kernel, Fedora and Mozilla. These rates may help understand the health of the software projects and also may help the project team take vital decisions.

Keywords: *bugs, rates, regression, derivates*

1. Introduction

Bug detection, reporting and closing are important for a software project. The quality of a project can be maintained if the number of bugs being closed is at pace with the number of bugs being reported. Otherwise the software would be too buggy and would lose relevance. It may be important to estimate the possible rate of bugs being closed with respect to the bugs being reported. This is important so that the project leaders might be able to decide whether the project is at par with the necessary quality standards that they have set for resolving bugs. The status and the parameters related to bug triaging differ slightly for different projects. Some projects use the term 'fixed' and others use 'closed'. Some use both 'fixed' as well as 'closed'. Some have also used the term 'resolved'. There are also terms like 'rejected'. Nevertheless, the authors in this paper have confined themselves to a common term 'closed' which means the bug that was reported has reached the state it could ideally achieve. This state may be a fix by changing the code, or by even rejecting the bug report on the grounds that it was wrongly considered to be a bug. Estimating what would be the rate of bug fixing at some future point of time with an estimate of the number of bugs being reported then would be interesting. It might be something directly related to the quality of the software and would reflect the health of the software project. In the present paper, the authors have attempted to model the process through which such an estimate can be performed by using polynomial regression and derivative as rate measurer. Parameters like average rate of bug closing with respect to bug reporting and instantaneous rate of bug closing with respect to bug reporting have been defined and illustrated. The data used is primary data mined from Bugzilla interface. Research has been performed on bugs of three different repositories namely Linux kernel project, Fedora project and Mozilla project. All three are big projects with active community involvement. The study is not exhaustive but the three projects may be considered to be representative in nature given their size and influence in the world of information technology.

Received (January 28, 2018), Review Result (April 30, 2018), Accepted (May 14, 2018)

2. Literature Review

The subject of bug study from various angles has been carried out in the past and is being actively carried forward today. Saha *et. al.*, [1] have described how severities mentioned in the bug reports vary across different software projects. The above work has conducted a survey of several bug reports and has tried to infer whether the severity mentioned in those reports have objective basis. The authors have also tried to discuss about the bugs that have been classified as normal but are actually not normal. This work has tried to look deep into understanding the misclassifications and the reasons behind these misclassifications. Keeping this in view the authors have selected three reputed and big software projects (Linux, Fedora and Mozilla) for the present study where chances of misclassifications are relatively less. D'Ambros *et al.*, [2] in their work have discussed about different techniques of bug prediction and have made a comparative study of the same. Their work shows how the studies of bugs lead ultimately to understanding of software quality and defect analysis. A parameter called 'entropy of changes' which is a measure of the complexity of code changes has been introduced in this work. Mainly two versions – change log approach and single version approach have been compared. Other approaches have also been discussed and an optimum solution is sought. Although the present study does not go as far as relating study of bugs to software quality, the authors of this study too believe similar to [2] that such a relationship can definitely be drawn. Another work similarly related to bug prediction and which involves the study of bugs from software repositories followed by suggestions to avoid GUI bugs is outlined by Michail A. *et al.*, [3]. Here the authors have mentioned about the Linux kernel repository which has also been used in the present study and have explicitly mentioned that the average lifespan of bugs in this repository is 1.8 years with a median of 1.25 years. A tool based approach has been suggested in contrast to the manual approach of bug detection and avoidance. Giger E. *et al.*, [4] in their works have pointed out the importance of the rectification of bugs and have highlighted the difference between change metrics and code metrics. There is the method level approach, four different types of classifiers and three types of models for each classifier. The authors have pointed out that change metrics would significantly outperform code metrics, the experiments have been performed on bug repositories similar to the present study. Rainsberger J.B. in his article [5] has propounded the need for good quality testing work for software projects. He has explained what should be the approach of the developer once a bug has been reported. Going through the article one may understand the methodology most of the developers go through to sort out the bugs that have been reported. Rainsberger has elucidated the importance of learning to write test cases and stresses upon the fact that all developers should write test cases may be in any language. Authors Wang J. *et al.*, have tried to measure the defect fixing performance of a project [6] by using historical data. Similar kind of work has been done in the present study. The authors of [6] have argued that the defect state transition can be studied by using Markov like models. Mathematical models to measure defect fixing performance stability have been drawn up. BugZilla has been mined similar to the present study. One of the most important works that have been used as a motivation and basis of the present study is the article by Zou W., *et al.*, [7] which has discussed about the rate of bug fixing in open source software repositories. Two open source software repositories have been considered for study in [7], they are Mozilla and Eclipse. It should be mentioned that the authors of the present study feel that some of the data published in [7] is not consistent. There is no status such as fixed for the bugs in Mozilla project. Moreover, there have been studies about the bias in bug fixing on grounds of severity and also on grounds of experience of developers such as in [8] by Bird *et al.*, and also about bias in terms of tagging the particular bug as in the works of Antoniol [9] *et al.*, Extending these concepts further Nguyun *et al.*, in their work [10] have explored the reasons behind such bias and have addressed the point whether this bias is due to the process of software development in general or whether the reason lies in some faulty of heuristics. The paper concludes that the tagging bias does not affect studies of bugs that are based on defect counts. There are many bugs which are not often considered to be bugs at all. Zaineb G., *et al.*, in their article [11] have discussed about the reasons for rejection of these bugs. In the present study, the authors have considered such bugs to be closed bugs because some of the repositories are actually treating such cases as 'closed'. When considered from the view point of quality testing, such bugs which are reported but have been rejected indicate the level of involvement of the organisation or the community involved in the software project. Whether this indicates a higher quality of the project or whether this may be considered to be a quality parameter might be a subject of a separate discussion. Rastkar S. *et al.*, have delved into the area of bugs from the standpoint of trying to make things easy for developers. They have [12] pointed out that it would be convenient to summarize the bug report so that it saves time and energy for the developer. This is also an attempt to study the bugs from software repositories of open source software project like KDE and has some similarity to the present study in terms of methods used and the procedures followed to draw inferences. They have elucidated the difference between conversation based generators and random generators for automating the process of creating summaries for effective work. There is the mention of strange case of bugs being introduced when bugs are fixed in the works of Sullivan M. *et al.*, [13] which again suggests that the rate of bug fixing can again get affected which is one of the subjects of discussion in the present paper. The data used in [13] is from IBM field service database called *Retain*. This is not similar to the present study but the issue of bugs in [13] is interesting nevertheless. The bug that may be generated during some other bug fix may not even be viewed from the developers system but may surface in the users system. Thung F. *et al.*, speak about bug reporting latency in their work [14] which is a term to describe the time gap between the actual time of insertion of the bug and the time when the bug was first reported. The paper adequately suggests that the developers have to make a optimum decision about what bugs to fix and which to be allowed so that fixing a particular bug does not interfere with the normal and proper working of the other parts of the system. There is a prediction model in this paper which clearly outlines the parameters used in predicting bug reporting latency. Herzig K., *et al.*, in their academic paper [15] discusses how misclassification can lead to wrong bug reports and thus resulting in improper handling of the bug. Sometimes a particular feature of software is made to look like a bug. Only after detailed testing can one confirm the fact that the reported thing is certainly a bug. Such improper reporting and misclassification invariably leads to loss of time and energy from the point of view of the developer. Moreover, such a bug is supposed to be closed by the developer and adds no value to the quality of the project neither it adds any value to the growth parameters. The existence of such bugs is substantially large and should definitely be dealt with adequately.

3. Methodology

1. Knowing the system of bug triaging from different projects:

Bugzilla is a place where bugs are filed for most open source projects. In this study we have used primary data from three projects. One project is the Linux kernel project. The upstream Linux kernel maintainers fix bugs only for latest stable versions of the kernel [16]. If any kernel version has reached EOL, it means the maintainers are not going to fix any bug for that kernel. In that case the advice for the user is to use the latest version of the kernel and try to reproduce the same bug on that kernel. If it can be reproduced, then a bug report may be filed. The upstream engineers are busy people and at times bugs are not attended to for long intervals of time. The person who has filed the bug may keep in contact with the community to receive updates from time to time. The other project that is used for collecting primary data is Fedora which is a flavor of Linux based operating system and whose community is very active in maintaining and releasing latest OS versions. The project wiki pages have a clear cut and detailed guide line for contributors who wish to get involved in bug triaging. It [17] also outlines the bug life cycle, knowledge about which is of prime importance while working with bugs. Mozilla also follows similar patterns as it is also an open source project.

2. Collecting Data:

The data related to Linux kernel project, Fedora project and Mozilla project are primary data and have been collected by mining Bugzilla. For getting the data for Linux kernel project, the authors have used the interface available at [18]. The query.cgi interface was suitably set to obtain the required numbers. On the vertical axis ‘status’ was selected. All the options in the status list were selected. In the category ‘search by change history’, ‘bug creation’ was selected and in the date field titled ‘between’ the first and the last dates of years from 2007 to 2016 were entered one by one to get the required tables. The remaining options were left default. In this manner the data of all bugs created each year was obtained. The status shown in the tables of the Bugzilla interface indicate the current status of the bug at present. For instance, the current status of a particular bug may be ‘new’, ‘closed’ or whatever but if it was created in a particular year then it was counted and recorded as a bug created in the corresponding year. Clicking each link corresponding to the number of bugs of each type for each year, the total list of bugs could be found which was used to check whether the listed bugs were definitely created in the year that was entered. A review of the list of bugs shown always validated that the number of bugs shown against each of the current status, but those which were created on some particular year was correct. The data was tabulated at one place for further study. The process was by and large similar for collecting data for Fedora bugs. However, the mining was done at [19]. An interface much similar to [18] is available and tables related to bugs from the Fedora project were recorded. There is one difference between the tagging of bugs between Fedora and Linux kernel projects. While in Fedora the bugs that have been dealt with (finalized) are all marked ‘closed’, in Linux kernel there are three tags, namely - ‘resolved’, ‘closed’ and ‘rejected’. The authors have assumed that all these three tags ultimately burn down to the understanding that these bugs are closed. Primary data related to the Mozilla project from Bugzilla, was done from [20]. The interface for Mozilla project in Bugzilla was similar to the interface for Linux kernel project and Fedora project.

3. Using polynomial regression model

The scatter diagrams constructed from the data collected clearly indicate that it would be appropriate to use a quadratic regression model. The best fit lines also point out towards similar considerations. Moreover, a linear relationship between bugs reported and bugs closed seemed to be a super idealized situation. The authors have used the online tool available in [22] to obtain regression polynomials for the data collected from all the three projects. The polynomials obtained have been validated using the data again. The error is less than 10%. The best fit curves are parabolic which again speak in favor of the choice of quadratic regression model.

4. Results

The following table gives the data of the bugs reported and bugs closed for the Linux kernel project for a period of 10 years wise. The Linux kernel project has three status parameters for its bugs, namely resolved, closed and rejected. We have considered the sum of these three parameters as closed in our present study.

Table 1

Year	reported	Closed
2007	1911	1998
2008	2671	2937
2009	2591	2209
2010	2530	1775
2011	1614	988
2012	1686	3489
2013	1912	1942
2014	2254	1144
2015	1966	1079
2016	2189	1013

The regression function that best suits the above data is the following:

$$\mathbf{y = 3.964828356 \cdot 10^{-3}\, x^2 - 16.80889199\, x + 19168.64851}$$

Residual Sum of Squares: **rss = 4779224.156**
Coefficient of Determination: $\mathbf{R^2 = 2.724253658 \cdot 10^{-1}}$

The trend line is parabolic as expected from the above function.

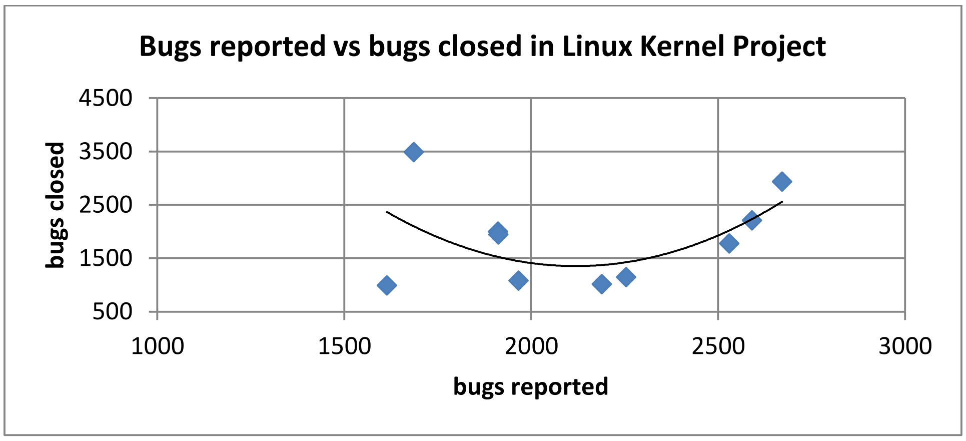

Figure 1. Bugs Reported vs Bugs Closed in Linux Kernel Project

The validation of the function modeled in the above discussion gives an error having a mean of 531.6117548.

The Fedora project has the following data of bugs reported and bugs closed. Fedora project considers all bugs resolved, fixed or rejected as closed so we are presenting the data as is from the Bugzilla repository.

Table 2

year	reported	closed
2007	25938	22703
2008	28180	30354
2009	39730	37457
2010	61485	55939
2011	42709	37671
2012	42895	44007
2013	49311	49690
2014	36308	35200
2015	38169	46590

The regression function that best suits this data is as below:

$$\mathbf{y = -1.142995717 \cdot 10^{-5}\, x^2 + 1.841444889\, x - 14412.76944}$$

Residual Sum of Squares: **rss = 124642897**
Coefficient of Determination: $\mathbf{R^2 = 8.513482806 \cdot 10^{-1}}$

As expected the trend line is parabolic when we plot the data

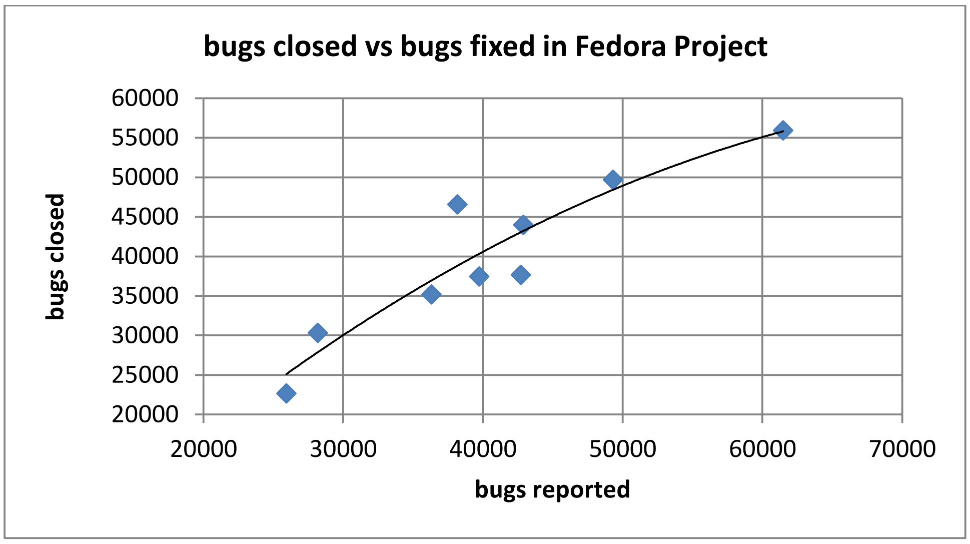

Figure 2. Bugs Closed vs Bugs Fixed in Fedora Project

The testing of the data with 10 values gives an error with a mean of 2819.792 which is about 7.09%

The data set for Mozilla is as given below. Mozilla does not have any status called fixed. So we have taken the status as closed.

Table 3

year	reported	closed
2007	42143	37323
2008	57017	49954
2009	61151	55646
2010	78484	71371
2011	79449	74094
2012	96158	81938
2013	115420	100518
2014	147612	127190
2015	107304	96900
2016	79467	81914

The regression function for this data set is as follows.

$$\mathbf{y = -1.632993866 \cdot 10^{-6}\, x^2 + 1.147035796\, x - 7819.373668}$$

Residual Sum of Squares: **rss = 123384431.8**
Coefficient of Determination: $\mathbf{R^2 = 9.804581566 \cdot 10^{-1}}$

As expected the trend line for the above data is parabolic.

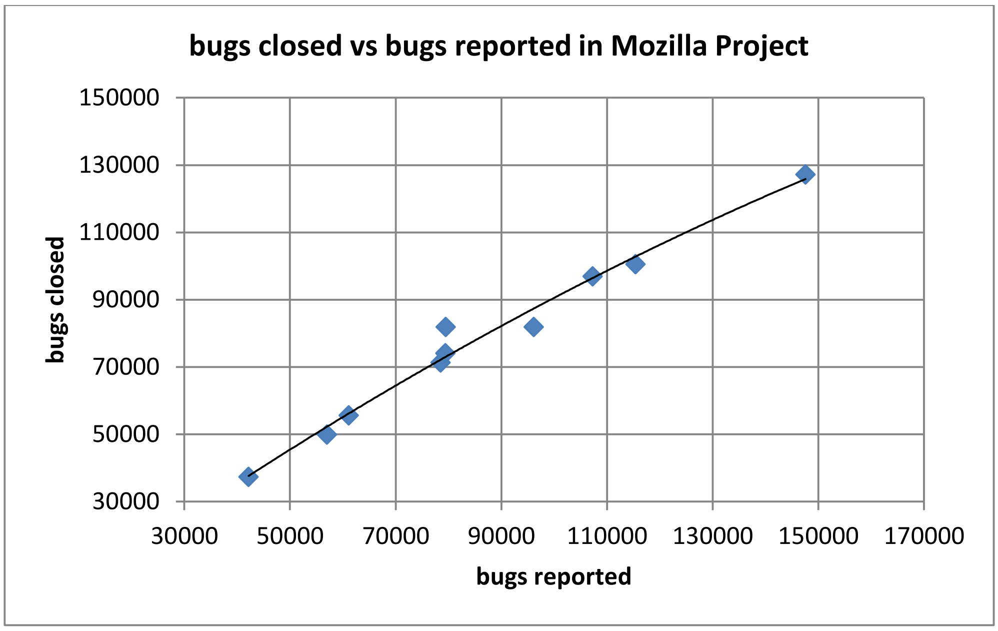

Figure 3. Bugs Closed vs Bugs Reported in Mozilla Project

Average and Instantaneous bug closing rates:

As shown in the above cases, we may take a regression function and estimate number of bugs that may be closed for a given number of reported bugs. We may consider the rate of bug fixing between two intervals of time as the ratio between the number of bugs closed in that interval of time to the number of bugs reported in that interval of time. The expression may be given as below:

Average rate of bug closing = $\frac{number\ of\ bugs\ closed\ in\ the\ interval\ of\ time}{number\ of\ bugs\ reported\ in\ the\ interval\ of\ time}$

Let the number of bugs closed in an interval of time be Δy

And the number of bugs reported in the same interval of time be Δx

Then average rate of bug closing is $\bar{v} = \frac{\Delta y}{\Delta x}$

Clearly this is the slope of the line joining the two points in the regression function. Then we may consider the instantaneous rate of bug closing with respect to bug reporting as the limiting value of the ratio $\frac{\Delta y}{\Delta x}$ when the quantity Δx approaches zero. This limit turns out to be the first derivative of the regression function with respect to the number of bugs reported. The instantaneous rate of bug closing with respect to bug reporting thus becomes:

$$v = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \frac{dy}{dx} \quad \text{at } x = a$$

Where a is the exact number of bugs reported when we want to estimate the rate of bugs being closed at that very instance of time when number of bugs being reported is a.

Clearly it is the slope of the tangent at x = a. Let the following illustration explain the situation:

For Mozilla project, in 2010 number of bugs reported is 78484 and number of bugs closed is 71371. In 2015 the number of bugs reported is 107304 and number of bugs closed is 96900. Hence the average rate of bug closing with respect to bug reporting in the time interval from 2010 to 2015 is

$$\bar{v} = \frac{\text{no. of bugs closed in 2015} - \text{no. of bugs closed in 2010}}{\text{no. of bugs reported in 2015} - \text{no. of bugs reported in 2010}}$$

$$\bar{v} = \frac{96900 - 71371}{107304 - 78484} = 88\%\ bugs\ closed\ with\ respect\ to\ bugs\ reported.$$

Now suppose we wish to calculate the rate of bugs being closed with respect to bugs being reported for a particular year say 2012. We take the regression function for Mozilla project which is:

$$\mathbf{y = -1.632993866 \cdot 10^{-6}\, x^2 + 1.147035796\, x - 7819.373668}$$

To find the instantaneous rate of bug closing with respect to bugs being reported at a given instance of time, we differentiate the above function with respect to x and get:

$$\frac{dy}{dx} = -3.265987732 \times 10^{-6} x + 1.147035796$$

This is the function to estimate the instantaneous rate of bug closing which we can now easily do by plugging in the value of x which is the number of bugs reported in that instance of time (2012 in this illustration) at which we wish to calculate the rate. Putting x = 96158 as available from the data we have the rate of bug closing with respect to the rate of bug fixing in the year 2010 as:

$$v = 83.3\%\ bugs\ closed\ with\ respect\ to\ bugs\ reported\ in\ 2012.$$

The ratio of bugs closed to the bugs reported for 2012 is 85.2% which gives a difference of 2.1%

5. Conclusions

1. Bug closing can be taken as the general parameter instead of bug fixing.
2. The average rate of bug closing with respect to the bugs being reported over an interval of time may be calculated as the ratio of number of bugs closed in that interval of time to the number of bugs reported in that interval of time.
3. The instantaneous rate of bug closing with respect to the bug reporting may be calculated as the first derivative of the function that represents the data of bugs being closed and bugs being reported over a period of time.
4. The instantaneous rate of bug closing with respect to bug reporting may be taken as a quality parameter for any software project.
5. The instantaneous rate of bug closing with respect to bug reporting may be used to estimate such rates at some future point of time.

References

[1] R. K. Saha, J. Lawall, K. Sarfraz and D. E. Perry, "Are Bugs Really Normal?", Proceedings of 12th Working Conference on Mining Software Repositories (MSR), IEEE, **(2015)**, pp. 258-268.

[2] M. D'Ambros, M. Lanza and R. Robbes, "An Extensive Comparison of Bug Prediction Approaches", Proceedings of 7th Working Conference on Mining Software Repositories (MSR), IEEE, **(2010)**, pp. 32-41.

[3] A. Michail and X. Tao, "Helping Users Avoid Bugs in GUI applications", Proceedings of 27th International Conference on Software Engineering, IEEE, **(2005)**, pp. 108-116.

[4] E. Giger, M. D'Ambros, M. Pinzger and H. C. Gall, "Method Level Bug Prediction", published in the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), **(2012)**, pp. 171-180.

[5] J. B. Rainsberger, "Avoiding Defects", IEEE Software, vol. 24, no. 2, **(2007)** March-April, pp. 14-15.

[6] J. Wang and H. Zang, “Predicting Defect Numbers Based on Defect State Transition Models”, ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), **(2012)**, pp. 191-200.

[7] W. Zou, X. Xia, W. Zhang, Z. Chen and D. Lo, “An Empirical Study of Bug Fixing Rate”, Proceedings of 39th Annual Computer Software and Applications Conference (COMPSAC), **(2015)**.

[8] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov and P. Devanbu, “Fair and balanced?:bias in bug-fix datasets”, Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (FSE 09). Amsterdam, The Netherlands: ACM, **(2009)**, pp. 121-130.

[9] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh and Y.-G. Guhneuc, “Is it a bug or an enhancement? a text-based approach to classify change requests”, Proceedings of the 2008 conference of the center for advanced studieson collaborative research: meeting of minds (CASCON 08). Ontario, Canada: ACM, **(2008)**, pp. 304-318.

[10] T. H. D. Nguyen, B. Adams and A. E. Hassan, “A Case Study of Bias in Bug-fix Datasets”, Proceedings of 17th Working Conference on Reverse Engineering (WCRE), **(2010)**.

[11] G. Zaineb, Dr. Irfan and I. Manarvi, “Identification And Analysis Of Causes For Software Bug Rejection With Their Impact Over Testing Efficiency”, International Journal of Software Engineering & Applications, vol. 2, **(2011)**, 10.5121/ijsea.2011.2407.

[12] S. Rastkar, G. C. Murphy and G. Murray, “Summarizing Software Artifacts: A Case Study of Bug Reports”, Proceedings of 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, **(2010)**, pp. 505-515.

[13] M. Sullivan and R. Chillarege, “Software Defects and Their Impact on System Availability: A Study of Field Failures in Operating Systems”, Proceedings of 21st International Symposium on Fault Tolerant Computing, **(1991)**.

[14] F. Thung, D. Lo, L. Jiang, R. F. Lucia and P. T. Devanbu, “When would this bug get reported”, Proceedings of 28th IEEE International Conference on Software Maintenance (ICSM), **(2012)**.

[15] K. Herzig, S. Just and A. Zeller, “Its not a Bug, its a Feature: How Misclassification Impacts Bug Prediction”, Proceedings of the 2013 International Conference on Software Engineering, **(2013)**, pp. 392-401.

[16] https://www.kernel.org/doc/linux/REPORTING-BUGS on 21 October 2017.

[17] https://fedoraproject.org/wiki/BugZappers/How_to_Triage on 24 October 2017.

[18] https://bugzilla.kernel.org/query.cgi?format=report-graph.

[19] https://bugzilla.redhat.com/query.cgi?format=advanced.

[20] https://bugzilla.mozilla.org/query.cgi.

[21] https://developer.mozilla.org/en-US/docs/Mozilla/QA/A_Bugs_Life =.

[22] http://www.xuru.org/.

Authors

Ekbal Rashid, he is working as an Associate Professor in the Deptt. of Computer Science and Engineering with Aurora’s Technological and Research Institute, Uppal, Hyderabad. He received his M.Tech in Computer Science in year 2009 from BIT, Mesra and his Ph.D in Computer Science and Engineering in May 2015 from Siksha ‘O’ Anusandhan University (Deemed University), Bhubaneswar, Orissa. Dr. Rashid has more than 30 International and National publications in journals and conference proceedings of repute including Springer, IEEE, Inderscience. His research area is software engineering, machine learning, data mining and artificial intelligence. Dr. Rashid has written two books on the topic of “Analogy-Based Software Cost Estimation” published by Advance Academic Publisher, India and “Enhancing Software Fault Prediction with Machine Learning: Emerging Research and Opportunities” that would be published by IGI Global, USA. He has more than fourteen years of teaching experience in various reputed Institutes/Universities across the country.

Mohan Prakash, is a research scholar and computer science and mathematics teacher at Ranchi, Jharkhand India. He is also an active contributor to FOSS projects specially Fedora and has been recognized as a Fedora ambassador in India. He is actively engaged in social service works aimed towards education of students from marginalized sections of the society and runs a trust named Peoples Education Trust which operates in different districts of Jharkhand. He runs charity schools and centers with money from self income and contribution from friends. He is also a member of Arasmin, a four star rated environmental organization accredited to United Nations with special consultative status. Mohan Prakash is also working with Opal Eduventure, a Mumbai based Software Company as technical head of its Jharkhand state office in Ranchi. He is the part of a team that has recently launched a start up named Linom Technologies pvt. Ltd. Mohan Prakash is actively involved in free lance software development work and data analysis. He is enrolled in a PhD program with Jharkhand Rai University, Ranchi under the guidance of Dr. Ekbal Rashid. He has attended many seminars and academic conferences in India and abroad and is a well known promoter of free and open source software in schools and colleges of Ranchi.