International Journal of Software Engineering and Its Applications
Vol. 9, No. 12 (2015), pp. 337-348
http://dx.doi.org/10.14257/ijseia.2015.9.12.30

Assessing Quality of Open Source Software Based on Community Metrics

Malanga Kennedy Ndenga[1*,2], Mehat Jean[1], Ivaylo Ganchev[1] and Wabwoba Franklin[3]

[1]*Laboratoire d'informatique avancée de Saint-Denis (LIASD), University of Paris 8, Paris, France.*
[2]*Department of Computer Science, Dedan Kimathi University of Technology (DeKUT), Nyeri, Kenya.*
[3]*School of Computing and Informatics Kibabii University College, Bungoma, Kenya.*
[2]*malangalanga@dkut.ac.ke*

Abstract

The purpose of this study is to analyze data from Open Source Software (OSS) community with an objective of identifying community metrics that can predict quality of OSS projects. We experimented with data from Apache OfBiz and Apache httpd-2 server OSS projects. We applied linear regression technique to the dataset to assess the strength of possible relationships of variables and also examined possible trends amongst variables. From the analysis, we found out that the size of user mailing list has a correlation with number of reported bugs. We concluded that the size of user mailing list community may not be an accurate representation of the entire user community that adopted the project basing on quality. However Backlog Management Index was found to be a better metric for assessing how projects manage issues reported by users.

Keywords: *Open Source Software, Quality, Bug density, Backlog Management Index, Community metrics*

1. Introduction

Quality is a critical factor that must be considered when selecting software amongst similar OSS solutions. IEEE Standard 1061-1998(R2004) defines software quality as the degree to which a software possesses a desired combination of attributes [1]. Software quality consists of two levels: intrinsic product quality - which is a measure of the functional defects in the software, and customer satisfaction - which is a measure of problems customers encounter when using the product [2]. To determine quality of a particular software product, software quality metrics should be applied on the software so as to establish the extent to which it satisfies specific quality attributes. A software quality metric is a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality [1].

Stol and Babar [3] argue that evaluation and selection of OSS products is a challenge users of OSS experience. According to Malanga, Mehat, Ganchev, Wandeto and Shikali [4], OSS product selection process is a challenge since tools for assessing OSS quality are still evolving thus not very reliable. Herraiz, Robles and Barahona [5] add that it is challenging to understand OSS development considering that its development nature is different from classical in-house software development model. They note that two key

* Corresponding Author

ISSN: 1738-9984 IJSEIA

differences exist between these two models; the first one is that for OSS, developers and other human resources are generally a mixture of a few hired developers and many volunteers. The second difference is that unlike in classical model, OSS development involves self-selected volunteers who perform crucial tasks such as support, bug reporting, minor software enhancements, and eventually form a part of the developing core group [5].

Several OSS quality evaluation models for example Qualipso OMM, depend heavily on project development process documentation and final product documentation [4]. However in reality, developers working on OSS projects may not have sufficient resources to dedicate to proper documentation. They may tend to focus more on the end product rather than documentation. Apart from documentation, existing evaluation models may also require interaction with the development team for interviews. However, availability of the developer team may not be guaranteed considering the dynamic nature of OSS development. For example due to code contribution, there could be no distinct owner of the project, or the developers may have stopped supporting the project. These two factors reduce the reliability of such evaluation tools. There is a need to define evaluation methods that can provide reliable data about OSS quality.

2. Related Work

2.1. Previous Studies on Software Quality and Community Metrics

Several researchers have studied quality of software in relation to its community in different perspectives. Some perspectives that have been pursued for software quality include; defect density, quality of modules and failure rates. While for software community's perspective, ownership, process maturity, organization structure and developer network are areas of concern.

Bird, Naggapan, Murphy, Gall and Devanbu [6] examined the effect of code ownership of software to the overall code quality. They found out that high levels of ownership of software components are associated with fewer defects. Lee, Hee and Gupta [7] developed an OSS success model. Using this model, they demonstrated that software quality and community service quality have a significant effect on user satisfaction. In a study of Windows Server 2003, Nagappan and Ball [8] presented a technique for early prediction of system defect density. This technique used a set of relative code churn measures that relate the amount of code churn to other variables such as component size and the temporal extent of churn. They showed that absolute measures of code churn are poor predictors of defect density while relative measures of code churn are highly predictive of defect density. Like our study, this study used defect density as a proxy for software quality.

Forty successful and forty unsuccessful projects from SourceForge were studied by Zielirinski and Szmuc [10] to investigate the relationship between process maturity and success of OSS. They showed that maturity of some processes is linked to the success of an OSS project. Sarkar, Rama and Kak [9] presented a set of metrics that measure the quality of modularization of non-object-oriented software systems by characterizing the software in perspectives of structural, architectural and similarity of purpose and goals. Their study confirmed that these metrics were able to detect improvement in modularization. Nagappan, Ball and Zeller [11] studied post-release defect history of five Microsoft software systems and found that failure-prone software entities are statistically correlated with code complexity measures. They concluded that complexity metrics can successfully predict post-release defects.

Meneely, willams and Snipes [12] examined structure of developer collaboration with developer network derived from code churn information that can predict failures at file level. They showed that developer networks are useful for failure prediction. Finally,

Naggappan, Murphy and Basili [13] presented a metric scheme to quantify organizational complexity, in relation to product development process so as to identify if these metrics impact failure-proneness. They used these organizational measures to quantify and study the effect that an organization structure would have on software quality. They realized that organizational measures predict failure-proneness in Windows Vista with significant precision.

The studies described above have one common aspect. Most of them investigated some aspect of software quality with regard to some aspect of developers' community. However our study investigates an aspect of software quality with regard to users' community.

2.2. Mining Data from OSS Development Repositories

A lot of static data corresponding to OSS products exist in software engineering repositories on the Internet. Information stored in these repositories represents a group memory for software projects [14]. Such repositories include: source control repositories, bug repositories, archived communications, deployment logs, and code repositories. Mining data from software repositories facilitates analyzing and cross-linking data available in these repositories to uncover interesting and actionable information about software systems [14]. Xie [15] points out that software engineering data from repositories can be used to: a) gain empirically-based understanding of software development, b) predict, plan, and understand various aspects of a project and c) support future development and project management activities. Robles, Barahona, Cortazar and Herraiz [16] add that mining and analyzing these data sources offers an ample amount of possibilities that surpass or complement other intrusive data-acquiring methodologies such as surveys and interviews. Hassan [14] notes that studying quality of source code, mining of data captured by project monitoring and tracking infrastructures as well as customer support records can be used to determine the expected quality of a software product.

2.3. OSS Community Metrics

Data found in OSS development tracking repositories is generated from an interaction between the community of this project and components of the project. OSS project tracking systems like source code management systems, mailing lists and bug tracking systems keep data of such interactions. An analysis of this data, results in measures of some aspects of the community involved in the project. Such measures lead to quantities that can be described as OSS Community Metrics. The Community metrics under focus in this study are number of messages on user mailing list and number of bug reports posted by users on bug tracking system. If OSS Community metrics are well understood, they can be used as pointers towards quality of Open Source projects. Fenton argues that classical software metrics for example size and static complexity metrics were inherently poor indicators of defects in software systems since they were not well understood [17]. As such, for OSS Community metrics to guide organizations in making better decisions on OSS, they need to be well understood.

3. Methodology

In this study, we employed experimental research strategy. The purpose of an experiment is to study cause effect relationships amongst variables *i.e.*, whether a change in one independent variable produces a change in another dependent variable [18]. Number of bugs and bug density were independent variables while number of messages on user mailing list was the dependent variable. In the first experiment we tested whether a change in size of user mailing list messages is linked to an increase in bug reports in issue tracking system. In the second experiment we tested whether a change in size of

user mailing list messages is related to a change in software quality *i.e.*, changes in bug density.

Using *mlstats* [19] and *bicho* [20] tools, we retrieved data from user mailing list and issue tracking system respectively from Apache OfBiz [21] and Apache httpd-2 server [22] projects. For Apache httpd-2 project, only bug reports associated with Linux operating system were considered since bug density was calculated using source code for Linux platform. Exploratory Data Analysis (EDA) approach was our initial step of data analysis. According to Saunders *et. al.*, [23], EDA approach allows one the flexibility to introduce previously unplanned analyses to respond to new findings thus formalizing the common practice of looking for other relationships in data, which the research was not initially designed to test. EDA emphasizes use of diagrams to explore and understand data. As a consequence, to gain initial understanding of our dataset, we explored it using scatter graphs. We performed quantitative analyses on the data by applying linear regression technique to assess the strength of possible relationships amongst data variables. Finally we examined possible trends amongst variables in the datasets.

4. Data and Analysis

Context of analysis for this work is focused on Apache OfBiz and Apache httpd-2 server projects for the period between 2009 and March 2015. We mentioned in the introduction that software quality involves intrinsic software product quality and user satisfaction of the software product. And that these two parameters are established by measuring the functional defects in the software and also measuring the problems users encounter when using the product respectively. A defect is an error in the source code while a failure is an observable error in the program behavior [11]. Establishment of all functional defects in a software product can be very tricky since it requires testing the software in all possible usage scenarios. Therefore not all defects lead to failure of the software system. Some defects may not lead to undesirable behavior of the software unless specific rare conditions are met by the usage scenario. To simplify the differences between the two, we considered bug reports recorded in the issue-tracking systems as functional defects from the user's perspective.

4.1. Relationship of Size of User Mailing List Community and Number of Reported Bugs.

The size of user mailing list community for a particular OSS product serves as an important aspect from where potential adopters of the product can roughly paint a picture on the size of the entire user community. A project with a large user community size in comparison with similar projects can be interpreted to mean that this project is of good quality. The number of people on the users' mailing list can be perceived as being a representative sample of the entire user community for a particular software project. We expect that as the number of bugs reduces *i.e.*, quality for a particular software product increase, the number of users should also show a remarkable increase. This increase in users should subsequently lead to a growth of user community size in the user mailing list and eventually an increase in number of messages in this list. The scatter graph in Figure 1 shows a strong positive linear relationship between number of major bugs reported by users and the number of messages posted by users for Apache OfBiz project.

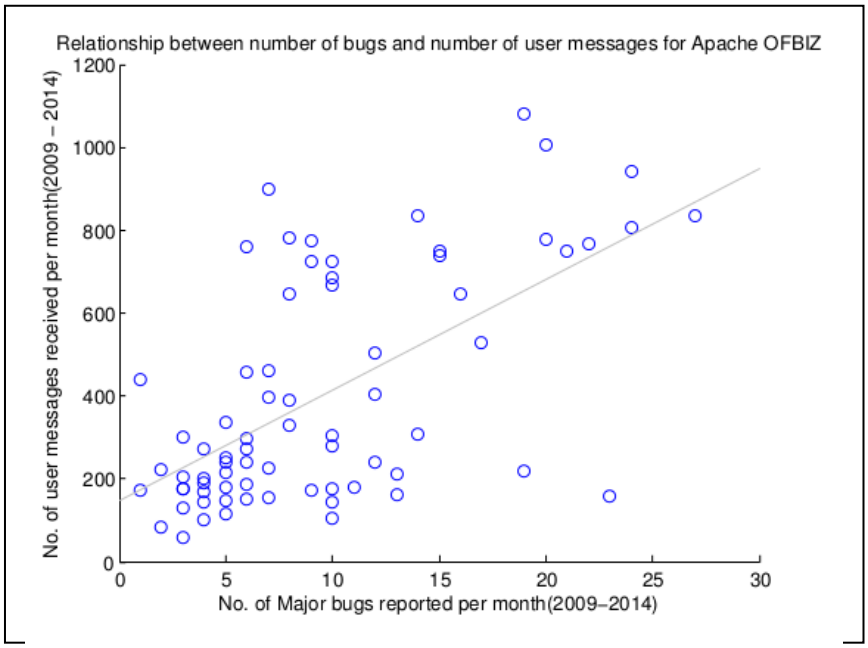

Figure 1. A Graph Showing Linear Relationship between Number of Reported Bugs versus Number of Messages Send to Users' Mailing List for Apache OfBiz Project.

We used linear regression model shown in equation (1) to test whether there is a significant relationship between number of bugs reported and number of messages posted by users.

$$y = \beta_0 + \beta_1 x + \varepsilon \quad (1)$$

From equation (1), y corresponds to a typical value of dependent variable for a particular value of independent variable in the dataset. Parameters β_0 and β_1 are population regression coefficients. Geometrically, β_0 and β_1 represent the y-intercept and slope, respectively, of the line on which all of the means are assumed to lie. The error term ε shows the amount by which y deviates from the mean of the population of dependent variable values [24]. To perform significance test we defined two statistical hypotheses H_0 and $H_{a;}$

H_0 - No relationship exists between number of bugs reported and number of user messages: $\beta_1 = 0$ *i.e.*, false

H_a - Relationship exists between number of bugs reported and number of user messages: $\beta_1 \neq 0$ *i.e.*, true

We observed a test statistic of 6.4436 with a p-value of 0.000000000126987 and a mean square error of 4.8390e+04. Similar results were realized when this experiment was repeated on Apache httpd-2 server project. The scatter graph in Figure 2 also shows a positive linear relationship between number of bugs reported per month and number of user messages received per month for Apache httpd-2 server project. For this project, a test statistic of 3.2985 with a p-value of 0.0015 and a mean square error of 2.5009e+04 was observed. Daniel [24] describes a test statistic as a statistic that is computed from a data sample serving as a decision maker to reject or not to reject a null hypothesis, while

p-value as the smallest value of level of significance for which a null hypothesis can be rejected.

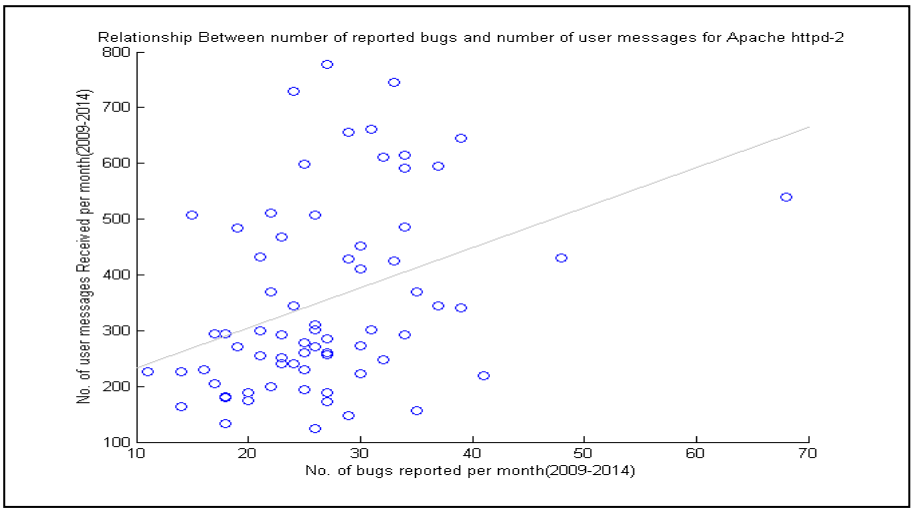

Figure 2. A Graph Showing Linear Relationship between Number of Reported Bugs Versus Number of Messages Send to Users' Mailing List for Apache httpd-2 Project

Saunders *et. al.,* [23], point out that a p-value of less than 0.05 leads to rejecting a null hypothesis since it shows that the probability of a test statistic having occurred by chance is very low and thus there exist a statistically significant relationship. Whereas a p-value higher than 0.05 leads to accepting a null hypothesis since it indicates that the probability of obtaining a test statistic by chance is very high thus the relationship is not statistically significant [23]. Therefore we rejected the null hypothesis and concluded that there is a significant relationship between number of reported major bugs and number of messages from users. This reveals that messages send by users on Apache OfBiz and Apache httpd-2 server user mailing lists are related directly to bug reports recorded in issue tracking systems of these projects. Therefore users posting messages in user mailing lists can be perceived to be users facing some trouble with the system.

Another important quality factor for software is bug density per Kilo Shipped Source Instructions (KSSI) for a given release. Using *SourceMonitor* [25] tool, we counted logical Lines of Code for nine releases of Apache OfBiz project and calculated bug density per KSSI for each release. Consider Table 1 that shows data for nine Apache OfBiz releases that were released for use between January 2011 and March 2015.

Table 1. Defect Density per KSSI for Apache OfBiz Project

Release	Average user messages received per month	Reported bugs per release	Logical LOC	Bug density per KSSI
A	26	10	279725	0.03575
B	21	14	348689	0.04015
C	21	2	348892	0.00573
D	17	4	356115	0.01123
E	21	1	348774	0.00287
F	21	11	356214	0.03088
G	26	1	348774	0.00287
H	26	7	356219	0.01965
I	25	14	298175	0.04695

The scatter graph in Figure 3 shows the relationship between bug density per KSSI and messages received from mailing list for Apache OfBiz project.

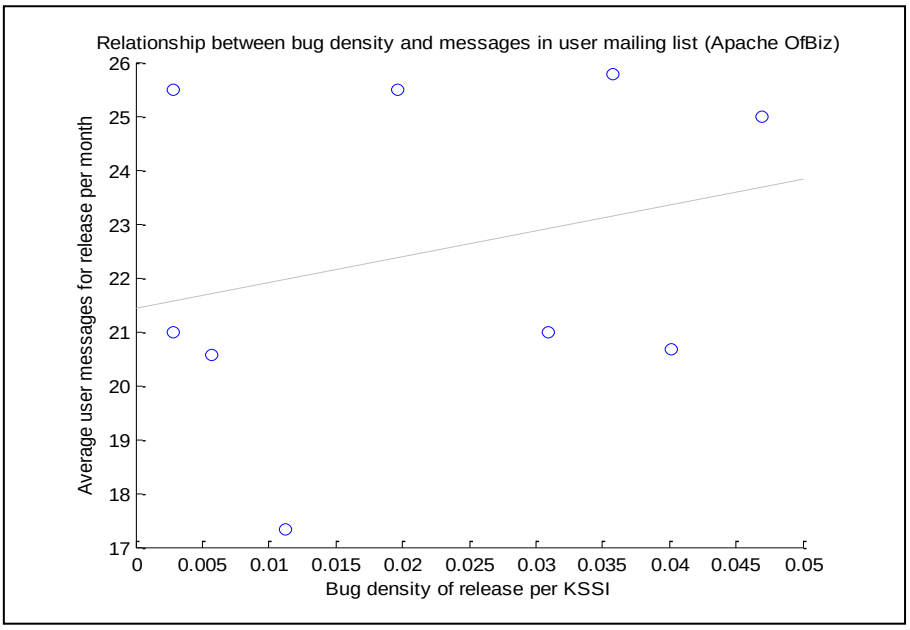

Figure 3. A Graph Showing Relationship between Bug Density and Number of Messages Send to Users' Mailing List for Apache OfBiz Project

We also tested this relationship by applying linear regression equation. We defined two statistical hypotheses H_1 and H_b for this test;

H_1 - No relationship exists between bug density of release per KSSI and number of user messages per month: $\beta_1 = 0$ *i.e.*, false

H_b - Relationship exists between bug density of release per KSSI and number of user messages per month: $\beta_1 \neq 0$ *i.e.*, true.

The relationship in Figure 3 is linear but weak. Test statistic of 1.0589 with a p-value of 0.3494 and a mean square error of 1.7185e-04 was observed. The probability that release bug density per KSSI of Apache OfBiz project has no significant relationship with user messages received is high *i.e.*, 0.3494. This experiment was also repeated on Apache httpd-2 server project. For this case an observed test statistic of 0.42117 with a p-value of 0.6792 and a mean square error of 2.501e+04 was observed. Like Apache OfBiz project, the probability of release bug density per KSSI of Apache httpd-2 server project not being related to user messages is also high *i.e.*, 0.6792. Thus we adopted the null hypotheses for both projects and concluded that there is no significant relationship between bug density of each release and user messages received for Apache OfBiz and Apache httpd-2 server project.

These two analyses show that the size of user mailing list community of Apache OfBiz and Apache httpd-2 server does not necessarily represent the entire user community which has adopted the product basing on good quality. If size of user mailing list was an indicator of adoption based on good quality of software product, we could expect a negative linear relationship where number of messages could increase with a reduction in number of major bugs or bug density per KSSI.

4.2. Backlog Management Index (BMI)

One way in which users can assess maintenance reliability of an OSS project is by factoring in the way the project handles fixing of reported bugs. This can be achieved by establishing the evolution of closed and open bugs per month for the project under assessment. The time series in Figure 4 shows how Opened and Closed bugs evolved per month for Apache OfBiz project.

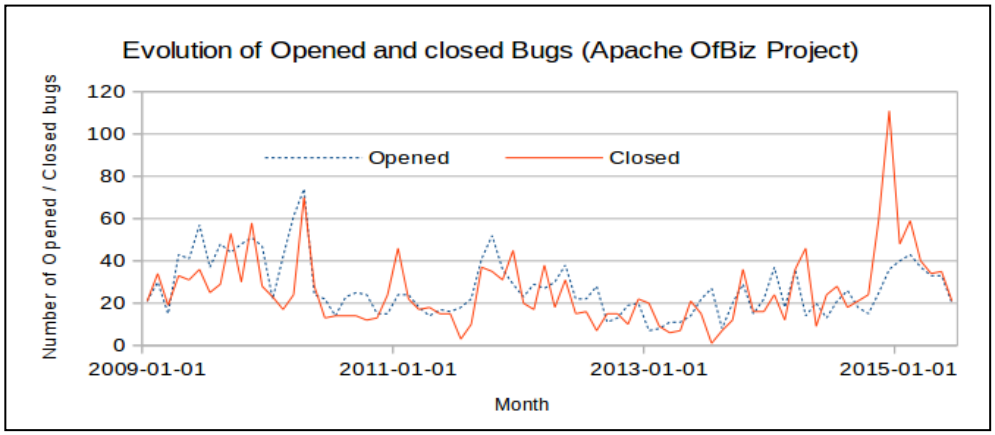

Figure 4. A Graph Depicting Evolution of Opened and Closed Bugs Per Month for Apache OfBiz Project

Qualitatively it is clear from this time series that bug tickets were opened and subsequently closed during the period of study. However this is not sufficient evidence to show that the backlog of bugs is under control and that it is on a reducing trend. Theoretically, a perfect or ideal software product should have no bugs. Thus if the rate of closing bugs is higher than the rate at which new bugs are reported, it means that the software product is moving towards perfection since bugs are reducing with time.

To measure this rate, we need to determine Backlog Management Index (BMI) for the project. BMI is a ratio of number of closed issues to the number of new issues reported during a month. A BMI larger than 100% indicates a reduction in backlog of issues while a BMI of less than 100% shows an increase in backlog of issues [2]. Figure 5 shows fluctuations of Apache OfBiz BMI of bugs. The BMI mean is 100.66%. This shows that Apache OfBiz bugs are under control and on a reducing trend. BMI is a software management metric that indicates how a particular software project supports its product.

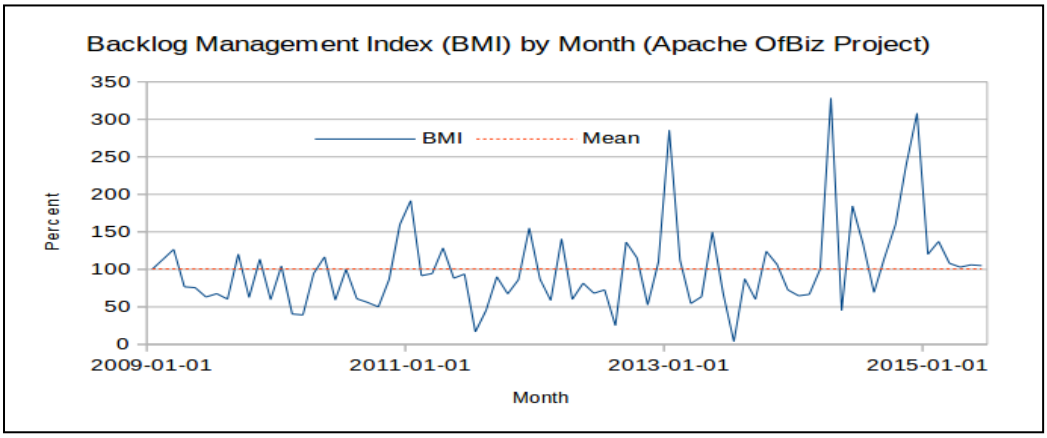

Figure 5. A Graph Showing Backlog Management Index (BMI) Per Month for Apache OfBiz Project

5. Discussion

Size of user mailing list: The size of user mailing list community can be thought of as being a random sample of the entire user community of an OSS product. From the analyses above, the growth of the user mailing list community is directly related to the increase in number of reported bugs. This finding is against the expectation that the number of users is expected to increase as the quality of software increase (reduction in number of reported bugs) – of course keeping all other variables constant. The direct relation of user messages and number of reported bugs implies that users post messages on mailing list when they are seeking help to solve issues related to usage of the software system. Some questions may arise from this situation. Assume we have a 'perfect' or 'ideal' OSS product where users will not encounter any problem. Will they have a reason to post anything on its user mailing list? Will it be in order to estimate the size of user-community for this software basing on the size of its mailing list community? Such an ideal software project will probably have an extremely small community present on its user mailing list and perhaps a large invisible user community. Therefore it may not be correct to select software basing on the size of the community and the number of messages visible on the user mailing list.

Bug density per KSSI: A measure of software bug density per KSSI is actually a good measure of intrinsic quality of software especially for types of software that change less often. In theory bug density is expected to reduce as the quality of the software increase. The bug density for most software is not steady since these software experience changes quite often. For example many software products have got multiple releases that are unique from each other in terms of size and features. Therefore bug density of future releases depend on changes in lines of code and changes in product features. A newer release with a higher bug density - due to introduction of a new feature - may not necessarily be of inferior quality as compared to an older release with a lower bug density.

Backlog Management Index (BMI) is a reliable metric that can be used to evaluate the efficiency of issue resolution by a project.

6. Threats to Validity

Construct validity of this work has been maintained since data collection from the OSS community is done automatically by issue-tracking and mailing-list tools. The external validity of these results is threatened by the fact that data was collected and analyzed for only Apache OfBiz and Apache httpd-2 server projects. Therefore these results cannot be generalized on all Open Source software. On the other hand, internal validity of this work has been maintained since we collected data non-intrusively without involving participants from Apache OfBiz and Apache httpd-2 projects. As a result, developers of these projects did not have an opportunity to influence these results.

7. Conclusion

Evaluating quality of OSS objectively by users before deployment of the products has been a difficult task. This difficulty is mainly attributed to lack of proper project documentation. Analyzing data from OSS development tracking systems like mailing-lists and issue-tracking systems gives reliable data which can mitigate this problem. Since success of an OSS project depends on its community, measuring the performance of the community could be helpful in predicting quality, thus success of the project. As a result, OSS developers should be encouraged to use development tracking systems so that they can make data about their community's involvement publicly available.

Results of this work cannot be generalized since they apply only to Apache OfBiz ERP and Apache httpd-2 server projects. There is need to repeat this study on many other OSS

projects so as to assess whether the results can be generalized. Finally, this study contributes toward theoretical advancement of OSS quality prediction in addition to shading more light on simplifying OSS selection process.

Future work: Our future work will entail validating findings from this research by studying a larger sample of OSS projects with a large user base. We also intend to perform sentimental analysis on user mailing lists and social media platforms of OSS projects so as to establish users' sentiments that can guide quality prediction.

Acknowledgments

We sincerely thank the Advanced Computing Laboratory of Saint-Denis (LIASD), France, for supporting this study.

References

[1] Software Engineering Standards Committee, IEEE Standard for a software quality metrics methodology, Std. 1061-1998, Technical Report. **(1998)**, pp. 24-26.
[2] Software Quality Metrics Overview, Retrieved July 24, 2015, from Pearson Higher Education: http://www.pearsonhighered.com/samplechapter/0201729156.pdf, **(2014)**.
[3] K.-J. Stol and M. A. A Babar, "comparison framework for open source software evaluation methods", Open Source Software: New Horizon, Springer, **(2010)**, pp. 389-394.
[4] K. N. Malanga and J. M., "Evaluation of Open Source Software with QualiPSO OMM: a case for Bungeni and AT4AM for All", Free And Open Source Conference (FOSSC-15), Muscat, **(2015)**, pp. 41-46.
[5] I. Herraiz, G. Robles and J. M. Gonzalez-Barahona, "Towards predictor models for large libre software projects", ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, **(2005)**, pp. 1-6.
[6] C. Bird, N. Nagappan, B. Murphy, H. Gall and P. Devanbu, "Don't touch my code!: examining the effects of ownership on software quality", In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ACM, **(2011)** September, pp. 4-14.
[7] S.-Y. T. Lee, H.-W. Kim and S. Gupta, "Measuring open source software success", Omega, vol. 37, no. 2, **(2009)**, pp. 426–438.
[8] N. Nagappan and T. Ball, "May, Use of relative code churn measures to predict system defect density", In Software Engineering, 2005, ICSE 2005, Proceedings, 27th International Conference on, IEEE, **(2005)**, pp. 284-292.
[9] S. Sarkar, G. M. Rama and A. C. Kak, (2007), API-based and information-theoretic metrics for measuring the quality of software modularization. Software Engineering, IEEE Transactions on, vol. 33(1), 14-32.
[10] K. Zieliriski and T. Szmuc, "Software process maturity and the success of free software projects", Software Engineering: Evolution and Emerging Technologies, vol. 130, no. 3, **(2005)**.
[11] N. Nagappan, T. Ball and A. Zeller, "Mining metrics to predict component failures. In Proceedings of the 28th international conference on Software engineering", ACM, **(2006)** May, pp. 452-461.
[12] A. Meneely, L. Williams, W. Snipes and J. Osborne, "Predicting failures with developer networks and social network analysis", In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, ACM, **(2008)** November, pp. 13-23.
[13] N. Nagappan, B. Murphy abd V. Basili, "The influence of organizational structure on software quality: an empirical case study", In Proceedings of the 30th international conference on Software engineering, ACM, **(2008)** May, pp. 521-530.
[14] A. E. Hassan, "The road ahead for mining software repositories", In Frontiers of Software Maintenance, 2008. FoSM 2008, IEEE, **(2008)** September, pp. 48-57.
[15] T. Xie, J. Pei and A. E. Hassan, "Mining software engineering data", In Software Engineering-Companion, 2007. ICSE 2007 Companion. 29th International Conference on, IEEE. **(2007)** May, pp. 172-173.
[16] G. Robles, J. M. González-Barahona, D. Izquierdo-Cortazar and I. Herraiz, "Tools for the study of the usual data sources found in libre software projects", International Journal of Open Source Software and Processes (IJOSSP), vol. 1, no. 1, **(2009)**, pp. 24-45.
[17] N. E. Fenton and M. Neil, "Software metrics: roadmap", In Proceedings of the Conference on the Future of Software Engineering, **(2000)**, pp. 357–370.
[18] C. Hakim, "Research Design: Successful Designs for Social and Economic Research (Second Edition)", London: Routledge, **(2000)**.

[19] Github, MetricsGrimoire / MailingListStats, Retrieved May 10, 2015, from Github: https://github.com/MetricsGrimoire/MailingListStats, **(2015)**.
[20] Github, MetricsGrimoire / Bicho. Retrieved May 10, 2015, from Github: https://github.com/MetricsGrimoire/Bicho, **(2015)**.
[21] The Apache Software Foundation, Apache OfBiz, Retrieved May 10, 2015, from Ofbiz: https://ofbiz.apache.org/, **(2015)**.
[22] The Apache Software Foundation, Http Server Project, Retrieved August 28, 2015, from Apache: Http Server Project: http://httpd.apache.org/, **(2015)**.
[23] M. N. Saunders, M. Saunders, P. Lewis and A. Thornhill, Research Methods For Business Students, 5/e. Pearson Education India, **(2011)**.
[24] W. W. Daniel, Biostatistics: A Foundation for Analysis in the Health Sciences, Ninth Edition. United States of America: John Wiley & Sons, Inc., **(2009)**.
[25] Campwood, (n.d.). SourceMonitor Version 3.5. Retrieved July 30, 2015, from Campwood Software: http://www.campwoodsw.com/sourcemonitor.html.

Author

Malanga Kennedy Ndenga is a PhD student of University of Paris 8 (France) and a member of Laboratoire d'Informatique Avancée de Saint-Denis (LIASD). He is researching on prediction of fault proneness and evaluation of quality of Open Source Software. Currently, he works for Dedan Kimathi University (Kenya) as an assistant lecturer in the School of Computer Science and Information Technology. He can be contacted through malangalanga@dkut.ac.ke or +254735742905.