Software Project Quality Parameters Can Be Expressed as Functions of Software Project Growth Parameters

EKBAL RASHID[1], NIKOS E. MASTORAKIS[2]

[1]Technical University of Sofia, Sofia, BULGARIA

[2]Sector of Electrical Engineering and Computer Science, Hellenic Naval Academy, Piraeus, GREECE
and English Language Faculty of Engineering,
Technical University of Sofia, Sofia, BULGARIA

Abstract: The tremendous success of software projects like Linux Foundation, Python Foundation, Apache, Gnome, Fedora, Ubuntu, have inspired researchers to study the different aspects of growth and quality of such software projects. This area of research is finding importance in industry and academics. This has compelled companies like Microsoft and Apple, to look towards community ways of software development, and that is paying them good dividends. In this backdrop, the present work aims to correlate the growth parameters of such kinds of collaborative community-based projects to their quality parameters. It aims to identify those growth parameters that would directly affect quality of the software projects.

Keywords: Software projects, Software Engineering, Collaborative Software, Predict

Received: March 7, 2024. Revised: August 9, 2024. Accepted: September 12, 2024. Published: October 22, 2024.

1. Introduction

As complex software continues to play important roles in our lives, we face challenges of getting involved in very large software projects [1] [2][3]. Such projects involve the efforts of a significant number of developers, designers, programmers, testers, maintainers, and of course users. Such software projects evolve over time with continuous and collaborative efforts. These collaborations may be planned and may involve definite and pre-determined developers most of whom are often employees of some organization and are paid to work for the growth and maintenance of the project [4][5]. Otherwise, developers may also belong to some community or constitute just temporary part time involvements that collaborate in such big projects. Such developers may not be paid or may not even be recruited in a planned manner. Their entries and exits into the project may be totally arbitrary and non deterministic [6]. Moreover, the quality of developers getting involved in such projects is not the same [7]. The time that the developers spend in coding or other development work is also not uniform. There are often peak times when the software development work is going at full throttle and there are times when no work is done for several weeks. Quite naturally, the quality of the software project in such cases is difficult to determine and more difficult to predict [8].

In spite of all this a considerable amount of time and energy is being used in recent times to actually understand the process of software development in such collaborative projects [9]. The chief reason for this is the tremendous success of many such gigantic projects such as Linux distributions, Apache, Open Office,

Gnome etc. Even proprietary organizations are considering involving communities in their endeavors to harness the benefit of open collaborative work. And when such attempts are being made, it becomes quite incumbent to develop software engineering models for such kinds of projects [10]. Certainly, such projects will work on models quite different from the traditional software models [11]. At the same time their growth and quality determination would be more complex and would require different kinds of metrics [12].

There have been attempts in the past to understand patterns in collaborative efforts. A huge amount of data is available in software repositories of such collaborative software projects and that has been mined and various types of inferences have been drawn. However, the author feels that there is a need to review and if necessary suggest improvements in the methods of data mining used to mine software repositories of such kinds. Moreover, the author hasn't found any concrete suggestions in software engineering practices for such kinds of projects. Over and above, the author has felt that enhancements can be made in the domain of metrics for understanding the growth of such projects as well as for determining the quality of such projects. There is a scope of applying mathematical concepts and tools to concretely understand and use such metrics in real work.

There also remains an area of concern as to why study about such collaborative software remains confined to only academic activities and are seldom absorbed in practical fields. A number of important international conferences are organized by significantly important organizations to understand the trends developing in this field. An example of such a conference is the International Conference on Mining Software Repositories organized by IEEE TCSE and ACM SIGSOFT [13][14]. Such conferences have been publishing rich literature over several years in the field of data mining software repositories to uncover interesting knowledge. However there is very little knowledge about any of such discovered facts presented in actual scenarios for uplifting quality standards. The author feels that there needs to be a mechanism to actually study such important developments and incorporate the same to practical fields in collaborative software development. If this is achieved, the quality of software being developed will rise to greater levels and will also usher in newer areas of research and progress.

There can actually be many areas of study in this particular domain. Some of them are: building models for development processes in collaborative software projects, analysis of ecosystems and growth of collaborative software projects, data mining of software repositories to analyze and predict future quality of software, software engineering models for collaborative projects based on historical data from repositories, richer algorithms for mining software repositories, developing tools for data mining of software repositories, mining data related to bugs and bug fixes in software repositories, programming language specific data mining of software repositories, building infrastructure for sharing data mined from software repositories and many more [15][16].

This research is inspired from the recent developments in these fields and the tremendous importance that it has in the realm of research and other related work. Data mining of software repositories is relatively a new area of research and there remains much to be studied and accomplished in this direction. Moreover, this field has the potential of influencing other epistemological areas. Research methodologies are being influenced by data over the past years. The large amount of data available in practically all domains of scientific and social activities has made this possible. Knowledge discovery from software repositories can pave the way for a similar sort of outlook in other fields too, leading to a more objective study of the universe as a philosophical category.

2. Scope of the Problem

The research mainly covers the problem of finding a relation of collaborative growth with quality of the software. The pertinent questions in this regard are:
Does the size of a software project have to do anything with the quality of the software? Does it enhance the quality or does it degrade the quality of the software project. Or is the quality of the software project independent of the growth of the software? Can the growth of the software have different aspects? Can we relate the growth in the number of people collaborating in the project to the quality of the software? Can we co relate the increase in the number of lines of code to the enhancement of quality? There may be a possibility that the number of people collaborating is increasing but the quality of software projects is not. Or maybe the number of collaborators has no effect on the quality. Or it may be so that the number of collaborators and the number of lines of code both have got some role to play in the quality enhancement. It may be that something else like the rate of growth is somehow related to the quality of the software project. Can we devise data mining methods to mine the data available in the software repositories to actually discover interesting facts related to growth analysis, growth rates and quality estimation of software projects? Closely linked to the software project quality is the issue of security. Obviously any software can't be of high quality if it is not secure. Now it is being widely recognized that collaborative and more so open source software are more secure than non open source ones. This is because there are more chances of bugs being detected and once detected there are better chances of getting the fixes as there are a large number of people collaborating. However it would be interesting to find if there is some kind of a mathematical relation between the number of collaborators and the security of the software. Similarly with the volume of the project, it would be interesting to try and study whether there is any relation between the growth of the software and the issues of security. Maybe the project is growing because people feel it is more secure. Maybe people do not like to use things that are less secure and so less secure projects are not likely to grow at increasing rates. Another problem that needs to be addressed in this regard is that are the present methods and algorithms of data mining sufficiently capable to unearth the interesting things that the author mentions in this problem statement? Or are there other ways and means to actually understand and delve into the software repositories to study the growth analysis. The problem also lies in having a better understanding of what actual quality of software means so far as large and collaborative software is concerned. Are the current metrics defined sufficient to understand software quality in the present perspective? Or is there a need of further refining the metrics and redefining some so that software quality can be studied and understood in the perspective that is being discussed herein. It may be the case that only some metrics are related to growth and growth rates. Other kinds of metrics describing software quality may not at all be related to such issues. Which metrics and in what way their relations exist are an issue of interest in the current problem solving effort. Will the data mining activities point to pertinent issues in this regard?

3. Significance of the research

1. A relation between growth parameters and quality parameters will help to make the task of estimating software quality more objective. This is because the growth and growth rates are bound to be very objective and measurable quantities. Once growth, growth rates and other aspects related to growth of software projects get linked to quality parameters, the latter also becomes measurable or at least capable of being calculated from directly measurable quantities.

2. This implies that people will start measuring the growth or growth related parameters of software projects and from those measurements they can easily estimate software quality metrics.
3. This may also open the doors for future research in the field of relating growth and growth rates of software projects to its quality and the estimation of quality from these parameters.
4. This research will pave the way for understanding software engineering principles in a better manner for such collaborative projects. There may be a possibility of better software engineering models coming up on the basis of this research work.
5. Communities can be developed in schools, colleges, universities, offices, workplaces in a scientific way to help produce quality software.

4. Methodology

The method has evolved out of extensive literature study and first hand experience in contributing to open source software projects. There are several quality parameters such as bug fixing rates, number of security features, enhanced lines of code, increase in number of packages that can be linked to software quality. An intuitive understanding that these growth parameters may enhance software quality is quite agreeable. Besides, extensive research work is being conducted and large amounts of primary and secondary data have been collected to study and analyze these parameters. Quality is also an extensive research issue for software engineers. There is every possibility of drawing Mathematical models for relating the two aspects of software projects – namely, growth and quality. Hence the method applied is justified in the present work.

5. Results Analysis

The research aims to discover concrete and effective steps to engineer collaborative project involvements, and their growth so as to estimate, predict and enhance the quality of such software projects in a systematic manner. These findings can shape community projects at various places starting from schools, colleges, institutes, offices, industries with software engineering models to control growth parameters and produce quality software everywhere. Results can be seen in fig1,fig2,fig3 ,fig4 and fig5.

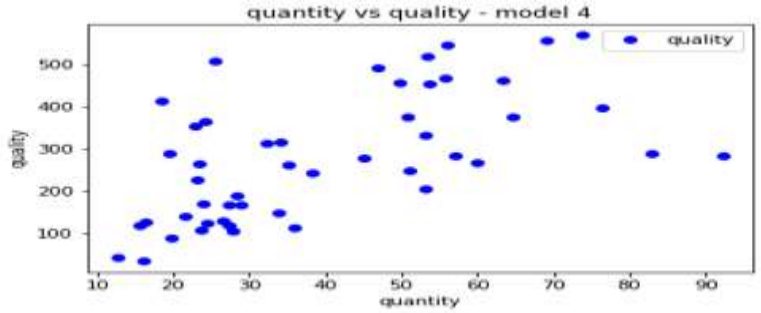

Fig1: Quantity vs Quality

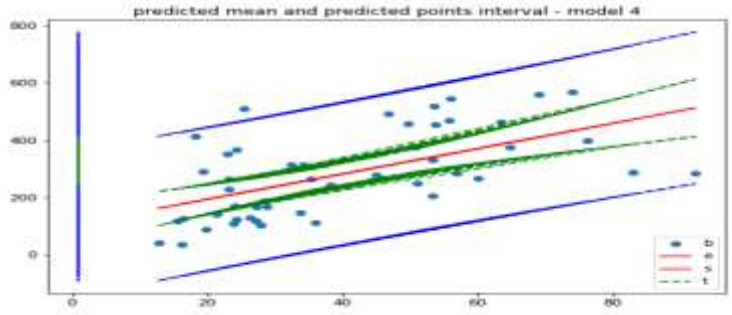

Fig2: Predicted mean and predicted points interval

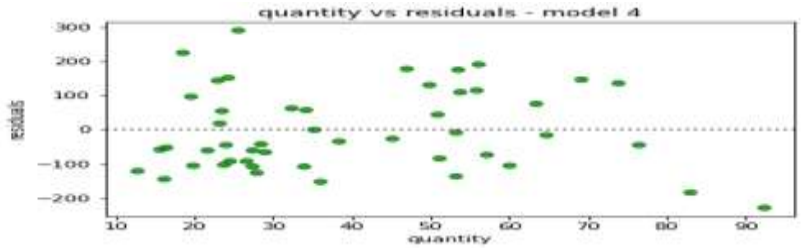

Fig3: Quantity vs residuals

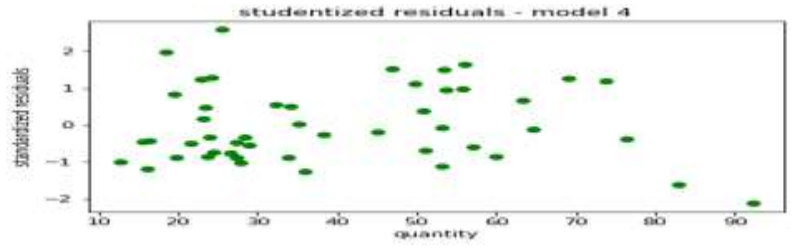

Fig 4: Studentized residuals

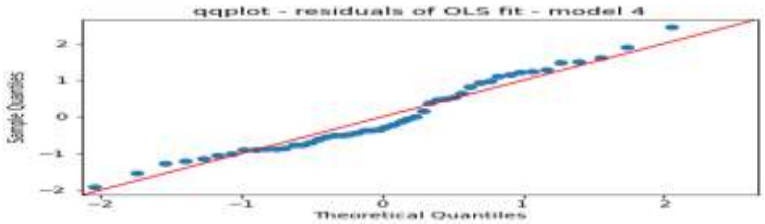

Fig 5: qqplot residuals

References

[1] Padhye, Rohan & Kumarasamy Mani, Senthil Kumar & Sinha, Vibha. (2014) *A study of external community contribution to open-source projects on GitHub*. 10.1145/2597073.2597113. (ResearchGate)
[2] Hata H, Todo T, Onoue S, Matsumoto K (2015) Characteristics of sustainable oss projects: A theoretical and empirical study. In: Proceedings of the 8th international workshop on cooperative and human aspects of software engineering, CHASE '15. IEEE Press, Piscataway, pp 15–21.
[3] Tamburri, D.A., Palomba, F., Serebrenik, A. et al. Discovering community patterns in open-source: a systematic approach and its evaluation. Empire Software Eng 24, 1369–1417 (2019). https://doi.org/10.1007/s10664-018-9659-9 (Springer)
[4] Gamalielsson J, Lundell B (2013) Sustainability of open source software communities beyond a fork: how and why has the libreoffice project evolved? J Syst Softw 3(11):128–145. https://doi.org/10.1016/j.jss. 2013.11.1077
[5] Schweik CM (2013) Sustainability in open source software commons: lessons learned from an empirical study of sourceforge projects. Technol Innov Manag Rev 3:13–19. http://timreview.ca/article/645
[6] Tamburri DA, Lago P, van Vliet H (2013a) Organizational social structures for software engineering. ACM Comput Surv,46(1):3,1–3,35. https://doi.org/10.1145/2522968.2522971
[7] D. Nemer. *IMPEX: An Approach to Analyse Source Code Changes on Software Run Behaviour.* Journal of Software Engineering and Applications, 2013, vol. 6, no. 4
[8] S. Butler, M. Wermelinger, Y. Yu and H. Sharp. *Exploring the Influence of Identifier Names on Code Quality: An Empirical Study.* 14th European Conference on Software maintenance and Reengineering, Madrid, Spain, 2010.
[9] B. D. Bois, T. Mens. *Describing the impact of refractoring on internal program quality.* ELISA workshop, Amstredam, Netherlands, 2003.
[10] C. Gerlec, M. Hericko. *Analysing Structural Software Changes: A Case Study. BCI-LOCAL* 2012, Novi Sad, Serbia, 2012.
[11] Aggarwal K., Hindle A., Stroulia E. *Co-evolution of project documentation and popularity within GitHub* in proceedings of MSR 2014, ACM pp.361-362. ISBN: 978-1-4503-2863-0
[12] T. F. Bisisyande. *Got Issues? Who cares about it? A large-scale investigation of issue trackers from GitHub.* IEEE 21st international Symposium on Software Reliability Engineering (ISSRE). Nov. 2013 pp. 189-196
[13] Jarczyk O. and others in *GitHub Projects. Quality Analysis of Open-Source Software.* Social Informatics Springer, Cham, Nov. 2014 pp. 81-89
[14] Peterson Kevin on *Mining GitHub: Why Commit Stops Exploring the Relationship between Developer's Commit Pattern and File Version Evolution*. 20TH Asia Pacific Software Engineering Conference Vol. 2 Dec 2013, pp. 164-170
[15] Kalliamvakou E. and others. *The Promises and Perils of Mining GitHub.* Proceedings of the 11th Working Conference on Mining Software Repositories MSR 2014 ACM pp. 92-101
[16] Weicheng Y., Beijun S., Ben X. *Mining GitHub: Why Commit Stops Exploring the Relationship between Developer's Commit Pattern and File Version Evolution.* 20th Asia Pacific Software Engineering Conference Vol. 2 Dec. 2013