Activity' and 'Effort' as Quantitative Parameters of Software Projects

EKBAL RASHID
Department of CSE, Rtcit, Ranchi, INDIA

Abstract: - GitHub in 2023 has listed the top ten repositories in terms of number of contributors. The authors of the present paper felt it necessary to take a look at these repositories on the basis of some quantitative parameters to gather some insight as to why these projects have been successful. In this paper there have been attempts to define some novel parameters as 'activity' and 'effort' of different kinds. These parameters have also been calculated for the top ten projects. Based on these data, some inferences have been made relating quantitative study of software projects to the quality of these projects and there is also the scope of some future work in this field. The significance of study is sufficiently interesting and has been adequately explained.

Key-Words : - GitHub, repositories, activity, effort.

Received: April 27, 2022. Revised: August 16, 2023. Accepted: September 22, 2023. Published: October 4, 2023.

1 Introduction

GitHub is an important place to look for data if we want to understand the quantitative growth of software projects. The GitHub repositories remain an interesting place to explore and draw conclusions in this matter. 2020 has been a time of extraordinary developments so far as the world of software development is concerned. With governments urging citizens to remain at home and work from home, GitHub has seen a huge surge of activity inspiring developers from all over the globe to collaborate and get involved in innovation in the search for connections and the attempts to dig for solutions to problems. The GitHub publishes the 'State of Octoverse' each year outlining interesting facts and figures. As per the document presented for the year 2020[1], there are more than fifty-six million developers with more than 60 million repositories created just in the year 2020. The GitHub has lots of data about these repositories. We can extract historical and current data using suitable methods of data collection. There is also the Google Big Query where much data about GitHub repositories are present in the public domain for researchers to explore and draw conclusions. The research about all that data will form the basis of some future work, but at present this paper holds some data about some of the top projects in GitHub listed on the basis of number of contributors to the project. Now, if we can discuss what data we can find about these projects, then the answer to this question is that lots and lots of data can be found. We can find when a particular project was started, for how long it worked, how it grew, how people collaborated in the project, how lines of code got into it. How there were forks, pull requests and commits. How branches came into existence for different projects and how they were integrated into the master branch, all these and many more things can be found on delving into GitHub data. And how do we do it? Well, GitHub has excellent documentation [2] on how we can search data on this platform and also about what type of data we can look for. In this paper the authors have mainly dealt with some ratios. There has been an attempt to define some new terms regarding the *activity* of a particular project and also regarding the *effort* of a particular project. These two terms have been used in other situations by various authors and there is every possibility that some better words may be coined. The authors of the present paper feel that there is every possibility that new words and more appropriate ones may be coined by peers in the future. In this paper two types of activities namely – issue activity and pull request activity have been defined. Now, these quantities are basically ratios which may look similar for small data and big data as well. Ratios are many times misleading numbers because they may show the same value for diverse volumes. Hence this paper suggests some ways to avoid this trap. Then there is also the attempt to define three kinds of efforts involved in software projects namely – commit effort, fork effort and branch effort. What care should be taken so as to get the maximum out of these parameters is also an area of discussion in this paper. The paper not only defines these values but also calculates these values for the top software projects in GitHub so as to get an idea of what these values may be for top projects. The authors have tried to compare these values for the software projects listed here and have deduced inferences from this objective analysis. It has been seen that almost all the projects have similar patterns and this makes the present research all the more interesting. This work also highlights the significance of such a kind of study and hints at the possibility of relating the quantitative study to the qualitative study of software projects. It makes special remarks on the possibility of betterment of engineering techniques using such a study. After inferences have been drawn, the paper discusses the

future scope of this research and points out the areas that could help to enrich the understanding of parameters defined here.

2 Literature Review

A number of authors have written on the subject of software repositories of GitHub. Many have highlighted the tremendous potential of this platform. This platform was first developed for collaborative development. Later version control also was incorporated and this made GitHub extremely popular and useful specially for open-source software development. The most significant work that needs mention is the study by Padhey et. al. [3] in which there has been a discussion about the participation in the development process of open-source software projects. This paper has provided the basic motivation to explore Github projects and has also shown that such data is extensively available on Github. This paper has discussed the fork and pull model and has suggested the number of forks and pulls can serve as the basis of quantitative analysis of the growth of software. The only thing that can enhance the study is the historical data of forks and pulls that can lead to a possible analysis of the different projects with respect to their growth in quantity. It is this paper that has provided the motivation to study the ratio between the number of open issues and number of closed issues. Another parameter that has been discussed in the present paper is also motivated from Padhey et. al. [3] and that is about the ratio between the pull requests open to the pull requests closed. The current paper discusses the manner in which this data can be interpreted. Similarly, another parameter that has been discussed here is the average rate of closing of pull requests. It must be said that the chief idea behind using pull requests of Github is something which has been borrowed from the work of Padhey et. al. [3] although the interpretation by the authors of the current paper is relatively novel and looks to update the state of art.

It goes without saying that understanding the behavior of the community and the involvement of people in the projects can help understand the health of the project and that has been acknowledged in the paper published by Hata et. al. [4] which stresses on the fact that there are studies that show that ‘sustainable open-source communities’ have to be explored in order to better understand the organi9sational structure of open source software projects to drive them forward.

Tamburi et. al. [5] has demonstrated that there can be quite a lot of research in the best practices of organization, research of social network and also by considering different kinds of models, different types of theories, the characteristics of different open source software projects. There also has been stress laid upon ‘social aspects’ of software projects such as member activity (Gamaleilsson et. al. [6] and Schweik et. al. [7]). These works show that there was a time when people had a kind of feeling that there should be a method of understanding and measuring this activity. The reason for this is quite obvious. There was an underlying concept building up and that was what the authors of the present paper have often stated with confidence that the growth of software projects and more so, the growth of open-source software projects have to be viewed as linked to the quality of the projects.

Many engineers and analysts felt that this socialization did not attach much importance to the overall quality of the software project. But this idea came out to face serious challenges because there have been the gradual collapse of open source software projects such as Softonic due to the lack of increase of volunteers who could steer the project and maybe that could be determined by measuring the activity as the authors have attempted to do in the present paper.

Tamburi et. al. [8] has made an attempt to present an automated tool called the Yoshi which stands for yielding open-source health information. This tool is able to measure what may be called the open-source community health status. It also “associates a community pattern of organizational structure types” [8]. The state of art however fails to highlight the activity of the organization and as the authors of the present paper believe without the quantitative measurement of activity of those involved and the analysis of the same, very little can be achieved in terms of actually realizing in the direction of quantitative growth. As a result, the present paper focuses with greater details on the submission of patches and number of commits as a measurement of one form of activity. Nevertheless, there are other forms such as bug fixing and documentation. And these activities too in no way contribute any less to the enhancement of the quality of the software project quality, the authors have decided to make these parameters areas of future study and research.

3 Methodology

A study of the document published by GitHub named ‘The State of the Octoverse’ [9], we find many interesting things that can help study the quantitative growth of software projects in GitHub repositories.

GitHub publishes this document every year. Going through the document published in 2019 we have a list of the twenty most popular software projects in terms of the number of contributors. We find ten projects having more than twenty thousand contributors. This clearly shows that the number of contributors has really a lot to do with the success of such software projects. The authors decided to take up this list and look into the GitHub site itself and dig up more interesting data about these software repositories. GitHub itself is a rich source of data about many repositories and there are well documented techniques about how all this data can be recovered from the site. Hence GitHub's documentation worked as a motivation for more data collection and further analysis.

Working according to the methods available on the documentation of GitHub about how data can be collected about various software repositories, the authors proceeded to collect data about the ten software projects that were mentioned in the document titled State of Octoverse published for the year 2019. Data about the following parameters were collected while studying these software projects – open and closed issues, open and closed pull requests, total number of months the software project is working for, total number of commits, total number of forks, total number of branches, etc. On the basis of all this collected data, the following ratios were formulated and derived:

1. Ratio between the issues closed to the total number of issues for a particular project
2. Ratio between the pull requests closed to the total number of pull requests for each project
3. Ratio between the number of forks to the number of contributors in each project
4. Ratio between the number of branches to the number of contributors in each project
5. Ratio between the number of commits to the number of contributors in each project

It is necessary to take a look at the interpretation about these different quantities mentioned above. The ratios can be interpreted in the following manner:

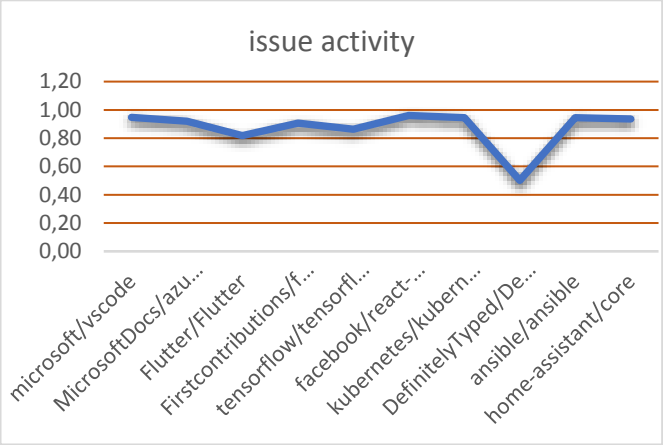
1. The ratio between the issues closed to the total number of issues submitted gives an idea about the working of the software project with respect to the end users. We know that the issues submitted are mostly done by the end users of the software. It is not that the developers do not submit issues. But mostly the developers would tend to work on the code itself and submit patches. Hence it would not be seriously wrong to assume that the matter related to issues submitted in a software project is largely related to the activity of the end users. Now it is quite normal that a very active project would have a high number of end users and that would also be a very good cause behind the popularity of the project. But more the number of users, more is the number of suggestions that come in. Of course, all issues are not bugs, and many of the issues would get closed without the need to work upon. Still, it goes without saying that a good project would have people in it readily attending to all the issues filed and responding to them as soon as possible. Hence we can say that this ratio is in a way an indication of the activity of the software project with relation to its end users. We may call this parameter as *issue activity.*
2. The next ratio is between the number of pull requests closed to the total number of pull requests submitted. Now, the pull requests are not much of a thing for the end users to get involved into. Rather, pull requests are more or less concerned with developers. It is through pull requests that the developer or the contributor in the project actually asks for code or patch or documentation or anything else to be merged with the project. The workflow is such that contributors open pull requests and people in the project who are authorized to review and merge these pull requests will go through the details of the content in the pull request and after adequate review will decide to either merge, or reject or communicate for further clarifications with the concerned developer. We can say that pull requests are the heart of GitHub workflow and they indicate the activity of development work or contributions to the project. Hence this ratio is different from the first and of a separate indicative measure. This ratio also indicates activity about the project but this is an activity not related to the end users but more or less related to the contributors of the project. Hence, we may call this parameter as *pull request activity.*
3. The third ratio is the indication of how much work is done by an average contributor. Actually, the authors of the present paper haven't differentiated between the different types of contributors to the project. There may be official contributors and other contributors. Hence the number of contributions needs to be judged from various angles. The first one is the ratio between the number of commits to the total number of contributors in the projects. A commit is a kind of timestamp for the project, when the contributor feels that a part of the work is done. So these are tiny bits and pieces of work and may indicate how

much effort is being spent in the project as a whole. The number of commits per contributor may not be a very good authenticated data, but it does give an idea about the activity of the project as such with respect to each contributor in the project irrespective of whether the contributor is a person within the project or whether the contributor is from outside the project. As such we may call this parameter as *commit effort of the project*.

4. Similarly, we have considered the ratio between the number of forks to the number of contributors. Now, forking is a very important activity. A person may fork a particular project for a wide number of reasons. It may be for personal or separate use, it may be to study the code and the project itself or may be to work upon it, hack through it and contribute to it. Whatever may be the case, forking is mostly done by people who are not in a way officially attached to the project. Now arguably, forking may not lead to anything very substantial and many times there are forks that do not matter much. Still, one thing is very certain. A person would fork a project only when the person is in some way or the other attached to some interest with the project. And hence forking indicates the general interest of not so novice people about the project. We may say that this ratio actually is a benchmark of interests about the project. As such, we may call this parameter as *fork effort of the project.*
5. Lastly, we have the ratio between the number of branches of the project to the total number of contributors to the project. Work on a branch is mostly done by people who are officially related to the project. This is the main reason why the number of branches has to be considered separately from the number of forks. The number of branches of the project shows the activity of the project's internal contributors. The workflow is that there is a master branch, and whenever a new thing has to be tried out, it is not done in the master branch, rater, a new branch is created and all the modification is done in that branch. GitHub workflow urges not to work directly on the main branch or the master branch even in the case of forks. So we may call the number of branches divided by the number of contributors in the project as a measure of *branch effort of the project.*

Having followed the following methods and dealt with the parameters in detail, it is time to take a look at the data available through this exercise and try to come to some proper conclusions.

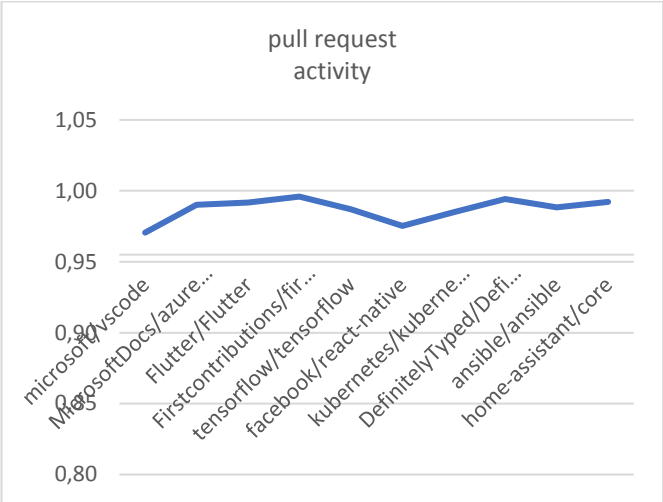
Projects	issues		issue activity
	closed	open	
microsoft/vscode	91403	5222	0.95
MicrosoftDocs/azure-docs	39041	3410	0.92
Flutter/Flutter	34948	7765	0.82
Firstcontributions/first-contributions	215	22	0.91
tensorflow/tensorflow	23710	3751	0.86
facebook/react-native	19256	793	0.96
kubernetes/kubernetes	33958	2025	0.94
DefinitelyTyped/DefinitelyTyped	3423	3400	0.50
ansible/ansible	26470	1531	0.95
home-assistant/core	15059	1016	0.94



The issue activity clearly shows a threshold value of 0.8 for most of the projects. Only for one project namely definitely typed the issue activity falls below the threshold value. We can see that the value is about 0.5 for this particular project. This can be related to the nature of this software project. Since it contains TypeScript type definitions, is there little scope for issues to surface?

Projects	pull requests		pull request activity
	closed	open	
microsoft/vscode	7614	231	0.97
MicrosoftDocs/azure-docs	19335	190	0.99
Flutter/Flutter	22178	183	0.99
Firstcontributions/first-contributions	29993	122	1.00
tensorflow/tensorflow	15089	199	0.99

facebook/react-native	9353	237	0.98
kubernetes/kubernetes	57647	878	0.98
DefinitelyTyped/DefinitelyTyped	40207	237	0.99
ansible/ansible	42935	509	0.99
home-assistant/core	23221	185	0.99

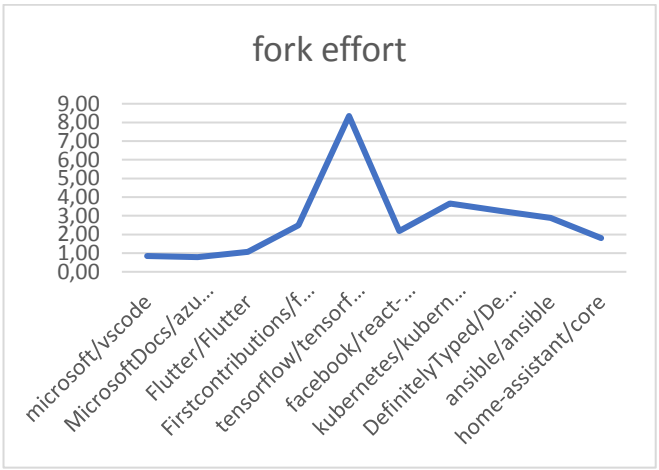


The pull request activity shows a threshold value of 0.95 to 1 for almost all the projects. This is a very interesting trend because it points towards the fact that good projects should have similar trends in pull requests. Pull requests being an important stage of collaborative software development, and that too very much of the open source type, this data is really very significant.

Projects	commits	Contributors in 1000s	commit effort
microsoft/vscode	70260	19.1	3.68
MicrosoftDocs/azure-docs	628475	14	44.89
Flutter/Flutter	20346	13	1.57
Firstcontributions/first-contributions	6838	11.6	0.59
tensorflow/tensorflow	94848	9.9	9.58
facebook/react-native	20905	9.1	2.30
kubernetes/kubernetes	93927	6.9	13.61
DefinitelyTyped/DefinitelyTyped	70952	6.9	10.28
ansible/ansible	50626	6.8	7.45
home-assistant/core	29244	6.3	4.64

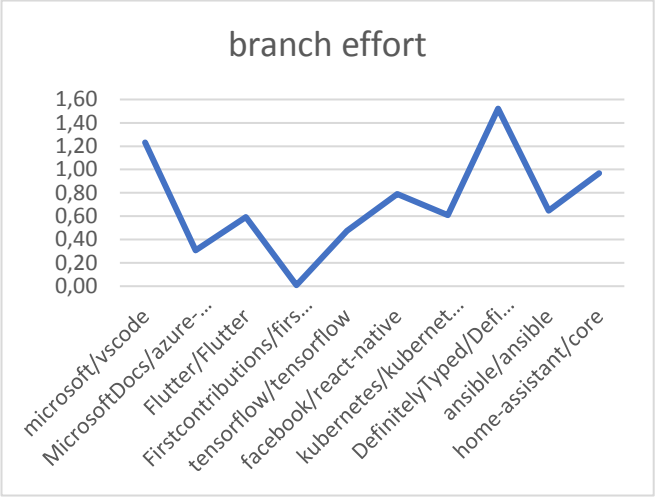
If we remove the second entry from the list which is a project about documentation really, we may take a rough threshold value of about 1.5 for the remaining projects. Commits for documentations are bound to be more because documentations undergo updating more frequently compared to the project itself.

Projects	forks in 1000s	contributors in 1000s	fork effort
microsoft/vscode	16.2	19.1	0.85
MicrosoftDocs/azure-docs	11	14	0.79
Flutter/Flutter	13.9	13	1.07
Firstcontributions/first-contributions	28.9	11.6	2.49
tensorflow/tensorflow	82.6	9.9	8.34
facebook/react-native	19.9	9.1	2.19
kubernetes/kubernetes	25.2	6.9	3.65
DefinitelyTyped/DefinitelyTyped	22.5	6.9	3.26
ansible/ansible	19.6	6.8	2.88
home-assistant/core	11.4	6.3	1.81



Barring one project the threshold value may be taken as near to unity. In a few cases it is just below 1 (about 0.8). In other cases, it is a little over 1. The data for Tensorflow shows a sharp spike upwards. Its value stands oddly different. However this is understandable because this project is mainly for machine learning and from the recent trends in the area of machine learning, we find that people have opted to fork this project in order to use it in the main.

Projects	branches	contributors in 1000s	normalised branch effort
microsoft/vscode	235	19.1	1.23
MicrosoftDocs/azure-docs	43	14	0.31
Flutter/Flutter	77	13	0.59
Firstcontributions/first-contributions	1	11.6	0.01
tensorflow/tensorflow	47	9.9	0.47
facebook/react-native	72	9.1	0.79
kubernetes/kubernetes	42	6.9	0.61
DefinitelyTyped/DefinitelyTyped	105	6.9	1.52
ansible/ansible	44	6.8	0.65
home-assistant/core	61	6.3	0.97



This parameter looks somewhat out of sorts and does not seem to be a suitable thing to consider for quantitative growth of software projects as it does not point towards a possible threshold value. This seems to be the only quantity defined in this paper that does not seem to show any definite interesting trend. Otherwise the rest of the parameters have strikingly similar values.

4 Conclusions

What are the conclusions that we may draw from the above set of data? We may discuss the possible conclusions point by point as they have been elucidated above. Before doing so, it is necessary to highlight certain areas that could lead to problems if care is not taken while using such parameters. An example would help to understand this point in greater detail. Suppose we have a software project named A that has a total of 80,000 pull requests and out of these 60,000 have been closed. Now the pull request activity would be calculated as the ratio between the number of pull requests closed to the total number of pull requests. For this project named A, the pull request activity would be 0.75. Now, suppose we have another software project named B and, in that project, we have a total of 10 pull requests and out of that 8 have been closed. The pull request activity for this project named B would be 0.8. Looking only at this data, that is, pull request activity of A is 0.75 and pull request activity of B is 0.8, one may be inclined to believe that the project B is in a better position than project A. However, the pull request activity ratio for B has been calculated using far a smaller number of pull requests as compared to A and so the conclusion that B is doing better than A is a misjudgment. To avoid such problems, it would be better to look at a project and analyze its pull request activity or issue activity only when it has crossed a certain number of pull requests or a certain number of issues. In this paper the authors have dealt with those projects which have a large number of pull requests and issues. Most of the projects discussed here have crossed ten thousand issues or ten thousand pull requests. There are a few that do not fit these criteria. Still, they have been discussed because they feature in the list of GitHub's top ten software projects of 2019.

1. The first data is about issue activity. As we have discussed to some extent in the methodology section, the issue activity is something to do with the users in general. We may take this parameter as an indicator of how common people both technical and non-technical are getting interested about the project in general. We have seen from the table and the following graph that the values for issue activity for almost all the projects, better to say, for all projects except one is above 0.8. Whether this value can be taken as a threshold value is something that is worth debating. Also, it is worth debating about the significance of this parameter which we keep for some future discourse.
2. The second parameter that has been discussed here is the pull request activity. This indicates activities of the developers. The value of this parameter is quite high for all the projects listed

here. It is almost 0.95 and above. Hence, we may suggest that the sign of a quantitatively steady project is one having pull request activity around a value of 0.95. Moreover, we have discussed this point before, that the projects listed here are the best so far as the number of contributors are concerned. As already discussed, most of the projects mentioned here have crossed the benchmark of ten thousand pull requests that we have set over here. There are also some projects that have crossed much bigger numbers of pull requests than only ten thousand. Hence, the value 0.95 is a good value to believe in.

The other parameters, that is, effort parameters are basically things calculated with respect to unit contributors and they may not pose such a serious a problem whether the project is small or large. This is because even if the project is small the effort of the contributors cannot be undermined. Say for example a certain project named X has 5000 forks and 2500 contributors. The fork effort for this project is the number of forks per unit contributor and that would amount to 2. Now, suppose there is another project named Y and it has 30 forks and 10 contributors. For this project Y the fork effort is 3. Can we undermine the effort of the contributors in Y simply because it has a smaller number of forks and a smaller number of contributors? The authors feel that the total number of contributors or the number of commits, branches and forks may not be taken into consideration while evaluating the effort parameters. Nevertheless, in the present scenario, we have taken large projects, not because it was essential from the sampling point of view, but because these are the top projects in GitHub and it was necessary to study their quantitative features. Based on the data of efforts from the top GitHub repositories we may draw the following inferences:

1. The commit effort will give an idea about how effectively the contributors are getting involved in the project so far as developing the content of the project is concerned. It is not necessary that all patches or other content committed are for improving the bugs or other defects. A big number of commits will be for adding features or improving the state of art of the project. Whatever may be the case, commits aim to improve the quality of the software project as a whole. Hence the effort of the developer in commits is an important thing to watch out for in a particular project. Looking at the data in this paper about the commit efforts of the top ten GitHub projects of 2019, we find that this value is almost one for all the projects, It is definitely more than 0.95. We may say that for each contributor in these projects there is at least one commit. We may take this as a sign of good health of the software project and may take it in this manner that in order to become a good rated software project we have to keep this value to at least one . Here, we have to point out a word of caution. That is, a good project should have a commit effort of at least one value equal to one. But it may not be true the other way round. That is, if any project has commit-effort value equal to 1, it does not automatically mean that it will be a good project. Like, suppose there is some project P having 2 contributors and 2 commits. Here also the commit-effort is 1, but only that does not take it to the category of the top ten projects mentioned in this paper. Because, the number of commits and the number of contributors is very small and insignificant. What may be the threshold value of number of contributors or number of commits that would be sufficient to decide whether commit-effort of 1 is suitable to judge a particular project is a matter of discussion and we leave that to future scope of research.
2. Looking at the data of fork-effort we find that this is also almost equivalent for the top ten projects. Only one project has a different value. The value of this parameter for the remaining nine is around and above 1. As we have already discussed earlier, this parameter may be an indication about the interest of people in the software project. If the project is hosting some code, then this interest is definitely in the code of the project. Because forking it ultimately means creating a clone to work upon. Many would like to fork simply for the purpose of studying the code. Yet that also is a kind of interest. The authors feel that there should be a parameter for measuring this interest in the project and for that the fork effort may be an option that can be considered. Now, the fork-effort of 1 means there is at least one fork per contributor. Again similar to the situation of commit-effort, the fork-effort has to be around 1 for a good project, but the fork-effort value being 1 simply does not imply that the project is a good one. Here too, there should be a minimum number of contributors to decide whether fork-effort can decide the level of quantitative growth or further even the quality of the project. The authors feel that this issue is beyond the scope of this paper and have thus decided to pass over this to some future scope of study.

3. The data for the branch-effort is different from all that we have discussed so far. Branches are official copies of the repositories that are created to work upon something new. We find that the values of branch-efforts of different projects are significantly different from each other. This suggests that the branch-effort is largely dependent on the nature of the project while the other parameters namely – commit-effort and fork-effort are largely independent of the nature of the project. Since branch-effort is to a great extent dependent on the nature of the project, we may not use it as a yardstick to determine the quantitative growth or for that matter the quality of the software project. This then takes us to the important discussion about whether this parameter may be of any use at all. The authors feel that if the software projects are grouped into categories, then, this parameter may be more meaningful and may throw better insight about the quantitative character of the software project. This too may be covered in some future discussion.

5 Significance of study

This study is related to the quantitative nature of software projects. The quantitative study is significant for several reasons:

1. It helps to understand the pattern of growth of software projects. In collaborative projects, collaborators come together to share their skills and develop projects. The project maintainers had a manner of indifference to who is getting involved and to what extent the person contributed to the project. However as more and more professionalism go into these projects and they come out of the ambit of simply the playground of some interested people, as more and more such projects start playing important roles in the market, engineers are confronted with the obvious question – what should be the engineering techniques, methods, models for such projects. At such a juncture, study of such parameters are the key to understanding whether things are going along the desired path, and if not then decisions may be taken to bring things under control.
2. The authors are of the belief that quantitative changes lead to qualitative changes. So, the changes in quantity however imperceptible they may seem, in the right direction may affect the quality of the software. This seems to be intuitively evident from what we have seen from the data presented in the present paper. We were discussing the top ten projects of GitHub in the year 2019. There can be little doubt about the fact that the success of these projects is largely due to their better quality and that means these are arguably the best quality software projects around for that particular span of time. We are of course speaking of quality in general without referring to any sort of particular metric of quality. Popular software is objectively better in quality, else why should they be popular? So we may safely assume that the list of software projects mentioned in the 2019 edition of 'The State of the Octoverse' is popular and hence of good quality. Now that being a common thing with all of them, we can relate this feature to the quantitative parameters that we have stated here. Most of these projects have similar values of issue activity, pull request activity, commit-effort and fork-effort. So the significance of the present research paper lies in the indication towards the fact that quantitative parameters may be in some way related to the quality of software projects and the two may be studies in their dialectical relationship.
3. The authors have published primary data related to important software projects which may be used in the future for other kinds of study and research.
4. Any research paper that defines new metrics ought to be used seriously for further study. New metrics are important because they help to enrich the state of art and of the perception about the subject in general. It also does enrich the literature related to software engineering, to system analysis and design. It has the potential to open up a plethora of future scope of studies. What this paper has in store for the future researcher is highlighted in the subsequent section.

6 Future Scope

Although a comprehensive discussion has been presented about some parameters on the activity and effort in software projects, there are many things that have to be dealt with in the near future to fully appreciate the discussion that has been conducted in this paper. For a more thorough and in depth understanding of the parameters discussed here, and also for their greater applicability future research options are necessary. The authors have therefore decided to outline some such points that may help in such a kind of study. They are as follows:

1. We have seen that the parameters such as commit-effort and fork-effort are one sided. That is, a good project may have a commit-effort and fork-effort of value equal to 1 but this value by itself doesn't guarantee that the project is good. We have seen that there is the need for determining a threshold

value of the number of contributors so that these parameters may actually work as suitable benchmarks. By this we mean that if a project has a certain minimum number of contributors then commit-effort and fork-effort values may help us decide whether a project is quantitatively on the right track, or even if the project is a good one.

2. We have seen that the data of branch-effort is not such that it gives us a value independent of the nature of the software project. It does not give us a value which will simply help us to determine whether the project is up to the mark or not. In this direction, what may be done is that the software projects may be divided into separate categories and then comparison of branch-effort may be performed and comparisons made in each category to see whether there is any specific pattern of interest or not. The authors feel the need of a separate paper to look into this matter. There is every possibility that the same category of software projects will have the same type of branch-effort values.
3. So far, the ratios haven't taken the time factor into consideration. Hence, we are not in a position to differentiate between two projects having similar values of activity and effort but are working for different time intervals. Say, we have two projects A and B having the same value of pull request activity or issue activity. Even though they might have the same values of commit-effort and fork-effort. This may make one conclude that the software projects are of the same quantitative nature, or even of the same quality to a particular extent. But if these two projects have been in operation for different intervals of time. Suppose project A has been around for 30 months and project B has been around for 60 months. Then, should we still treat them as of similar quantitative nature? The authors feel that there should be some parameter that works around the concept of time and this again can be something for future study.

References:

[1] https://octoverse.github.com/

[2] https://docs.github.com/en/github

[3] 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)

[4] 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.

[5] 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)

[6] 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

[7] 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

[8] 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

[9] https://octoverse.github.com/2019/