stochastically but surely not violating causality. There has always been debates in the philosophical domain between the deterministic scholar and the stochastic theoretician and the matter never seems to settle out. Nevertheless, stochastic methods have been proved to be decisive, and are being seriously used in many forms of epistemology. Hence the very idea of looking at development of software projects from the stochastic point of view may be interesting. Now that might mean a lot of technical and rigorous effort which we aim to target somewhere later in our course of research. At present, we are discussing something that is much simple to understand. And for that we have decided to take the analogy of the world in which we live in. The question before us is quality and what does that finally come down to? The authors believe that quality finally comes down to the ability to exist in a particular domain. Looking at the example of evolution of life forms, we see that so many kinds of species have evolved over millions of years, but why has it been possible for the human race to exist amidst so many hurdles, odds and perishable situations? This is quality. The rise in quality makes such an evolution possible. Taking up the analogy, what has made projects such as Linux, Apache, Mozilla, grow and reach its present stature? Definitely it is quality. Existence demands quality although sorrowfully it may not be the other way out. We may easily find examples of projects that have been high on quality standards but that have failed to exist in the stochastic run. Many have wondered why, and have discussed the issue whether it is enough for us to just keep track of classical quality parameters and exist substantially, meaningfully, popularly, usefully in our domain. The authors have presented their thoughts on this very subject in this paper and have felt that there is more to just following the trail of classical software development parameters. What has emerged out as a result of such continuous thinking and pondering is that popularity of software projects may be considered to be a parameter of quality. Again, the authors have felt that this is important because popularity may be easily measured from a number of obvious things related to the software project. This paper aims to highlight and define the ways to measure popularity of the project and to try to establish how popularity may be considered a parameter of quality of the software project itself. This paper takes a look at some of important projects hosted at GitHub because GitHub is the host to the most important software projects in the open-source world. GitHub is the host to Linux, Fedora, Mozilla, Apache, Facebook-React, Flutter, React-Native, Node, JQuery and what not. If someone has to look for data in the open-source world then it can not be anywhere else other than GitHub. So the authors have mined different repositories and have tried to establish the importance of popularity and the different aspects that go into the measure of popularity.

2. Motivation

Recently, the use of AI technique has proven the great practical value in solving a variety of software engineering problems including software quality parameters such as popularity as a quality parameter.The whitepaper published by Altexsoft [2] deals with two levels of software quality namely functional and non-functional. In the functional type they have given due stress on the ‘practical use of the software’ from the ‘point of view of the user’ along with things like features, performance and absence of defects. However, this was not the original normal as most researchers had written to show that there is no evidence to accept that popularity actually denotes quality of the software. Then again, they were looking at quality simply from the classical viewpoint. One such work is of Sajnani et. al. [3] where there has been evaluation of maven components on the basis of such classical quality parameters such as efferent coupling, afferent coupling, lack of cohesion, depth of inheritance, ratio of derived to base interface, etc. Simultaneously there has been a discussion about popularity parameters such as projects using the component, files using the component, and such things. The conclusion drawn here is that there is no evidence from such an empirical study that could lead to the conclusion that there is indeed a relation between popularity of the software and its quality. Another work by Siavvas et. al. [4] discusses about the relation between the popularity of software projects and the security of the project. In this paper the authors have used a parameter called Static Analysis Vulnerability Density (SAVD) to get the number of vulnerabilities per thousand lines of code and then used this parameter to determine the security of the software. The Spearman’s rank correlation value turns out to be negative in this statistical analysis and that makes the authors of this paper arrive at the conclusion that greater the popularity, more the vulnerability. However, authors of [4] further state that this is a contradiction of the thresholds proposed by Cohen in [5] where the understanding goes such that widely used software products are more likely to be secure and so on the basis of this the conclusion should have been that ‘popular software are more secure’. Now security can definitely be considered to be a quality parameter. This is because more secure a software is, better is its quality. The very fact that the analysis of data does not satisfy the expected Cohen thresholds is a motivation to work more in this field. Similarly, another work by Carvalho et. al. [10] discusses about several popular software projects and the different vulnerability issues they face. The authors of this academic article have shown that there are indeed quite some software projects that are very popular but have serious security threats. This boils down to the situation that if indeed we do consider security to be a quality parameter, then

from this study we have to use caution to call popularity as a quality parameter. Then the authors of this paper came across another work by Alsmadi et. al. [6] where the authors have clearly stated that popularity increases with increase of several quality parameters. The authors have tested the quality of several Java based open-source software projects from GitHub on the basis of sixty-five quality parameters using several tools and then have done the necessary analysis work to come to this conclusion. Since we have a concrete discussion here about the direct relationship between popularity of software projects and their quality, the authors of the present paper were motivated to look further into the issue whether popularity can itself be considered to be a quality parameter as it is being directly affected by the other quality parameters that are more or less popularly used. Another work of Borges et. al. needs to be mentioned here. The authors in this paper [7] have measured popularity of software projects from the number of stars they received. Then they have tried to relate the popularity of these software projects to the growth pattern of some other parameters such as number of forks etc. and in this way, they have tried to show that the popularity increases as these growth parameter increases. In this way, the authors here have spoken in favor of popularity being proportional to several parameters which may affect the quality of the software projects. In this discussion [7] the authors have also shown that apps that are highly rated show specific patterns related to seventeen parameters which may show the quality of the software project. Another study by Bavota et. al. [8] discusses the popularity of Android apps and related the same to the use of types of APIs. The gist of their discussion is that apps that tend to use fault free APIs or stable APIs tend to be more highly rated than those apps that use comparatively fault prone APIs and those that are often subject to changes. Although this may sound highly intuitive, we believe that this too speaks in favor of considering popularity as a software quality parameter. Another recently and popularly used parameter for estimating popularity specially in mobile applications is customer churn rate which is defined as the percentage of difference between customers joining and leaving against the total number of customers joining. In a simplified manner we can put it like this:

$$\text{churn rate} = \frac{\text{no. of customers leaving} - \text{no. of customers joining}}{\text{no. of customers joining}}$$

There is a work by Guerrouj et. al. in [9] which tries to relate this churn of mobile applications with the success of the app. The findings led to the conclusion that mobile apps with larger number of churns seem to be less successful. This discussion also calls for different kinds of relative studies between discussions on Stack Overflow forums about the code of a specific application and the changes that have been made to the classes or other APIs by developers in the real development scenario. Worth mentioning is the approach taken by some important industry players while considering the quality of software. Now, we know that the software development process is important in determining quality. The white paper [11] discusses the need to come out of traditional methods and processes and opt for other ways that may not be of the text book type but are more suited to the needs of business and market. This seems to be taking the side of popularity of the software project in the market compared to the other quality parameters that are usually considered. There is another interesting discussion by Bissyandé et. al. [12] in which the authors have made a survey of a hundred thousand projects from GitHub and seen which is the most popular language that is being used. It can be seen from the discussion in this paper [12] that the most successful projects are using the most popular programming languages. Although, this is not something to do with the popularity of the software projects as such, but still, there seems to be a link between the popularity of programming languages and the success of software projects. The authors in [12] have also mentioned that since the development of web application languages like JavaScript have flourished. They have also shown that Object C has gained popularity with the success of Apple. Another discussion by Kwan et. al. [13] is about socio technical congruence. The authors after a thorough study have decided to conclude that an increase in socio technical congruence may not improve the software development process. However, after stating so they have gone further to state that socio technical congruence will help understand the relationship between socio and technical areas and the area of software development. To us, this remark seems to be somewhat contradictory. An increase in socio technical congruence can only mean that the popularity of the software is increasing. Without increase in popularity how can people socialize within the ecosystem of a particular software development work? Hence, if it does help in understanding relation between socio-technical areas and software development, the authors in [13] perhaps are agreeing to the fact that there is indeed some relation between the two which needs to be studied and this motivates us further for the present study. Betz et. al. in [14] has worked on Conway's laws which assumes a strong association between system's architecture and the system's communication structure. Right back in 1968 Conway had suggested that "any organization which designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure" [15]. The work by Betz. et. al. [14] has made an extensive survey of literature and has concluded that there are significant differences between how one

interprets and applies this law. Now, we find this study relevant because communication system in an open-source software project is not limited to within the system but rather is also open in nature. Communications in open-source software projects also involve the naïve and the dexterous. The extent of communications cannot be seen as isolated from the popularity of the software project. Any intuitive assumption would be that greater the popularity, more is the degree of communication in the project ecosystem. The work by Betz et. al. [14] has highlighted the existence of extensive literature that has validated Conway's law and that means one has to say that in the case of open-source software projects, the quality of the software (at least the structure) has indeed deep association with the communication system, which in turn is associated with the popularity of the project. A white paper on the standards of modelling software development [16] deals about selection of programming language for a particular software project in terms of its popularity. It enlists the popular software programming languages in different fields and solicits looking at popular programming languages while making such selections. Now, why should one select a popular programming language if it is not of good quality. Thus, a practical method in the industry is towards looking at things from the viewpoint of popularity to make an estimate of its quality. Numerous industry white papers like the ones listed from [17] to [21] are highlighting the importance of user satisfaction, user experience and end-user management in their development systems. [21] has in fact prepared a check list to ensure that the goals for usability are properly fulfilled. All this and much more thus serve material for the authors of the present work to consider popularity as a parameter of software quality.

3. Scope of Study and Methodology

While the issue of considering popularity as a software quality parameter is a broad and debatable issue which may be approached from various angles, the authors of the present work have decided to limit their discussion to the trends in the modern development world, especially in the domain of open-source software projects. Most of the works cited here are from this domain. There are few that are not from the domain of open-source software projects. But they too seem to have followed an approach similar in nature. Open-source software projects like the ones hosted on GitHub are increasingly getting more and more popular and are overtaking other software projects. The authors have attempted to discuss whether their popularity is due to their enhanced qualitative standards. To achieve this, the authors have decided to go through several rounds of research. This work may be considered to be the first work in this direction. The first round is the publication of a survey result and the possible directions evident from it. For this, as stated earlier, a number of academic papers and industry white papers have been consulted and screened. It is not that academic papers have spoken against the consideration of popularity as a quality measure. However, the existence of a number of academic papers in support of the same makes the study all the more interesting. Besides, recent white papers from the world of industry seem to unequivocally suggest that popularity is an important metric that needs to be achieved and more than often stress is laid upon this metric rather than other traditional quality parameters.

4. Significance of Research

As far as AI is concerned in software engineering, software engineers wanted to incorporate artificial intelligence (AI) techniques into their work, so that by the help of AI technique they can fixed the errors automatically without time consuming. The significance of this research lies in the fact that popularity is a very easily measurable metric and that many works have already tried to define it earlier. If we do consider popularity to be a quality parameter, we can study the pattern of quantitative changes in popular software projects and emulate them as a system of software development model. The quantitative parameters of open-source software projects are easily measurable, and their threshold values can be planned to be achieved by setting suitable time-bound targets. So in this way it would be possible to control the quantitative parameters suitably in order to achieve high quality and in that manner popularity.

5. Discussion and Conclusion

The following conclusions may be drawn from this study:

As the AI technology is accepted globally and users or software engineers can expect to see even more innovative uses of AI in software engineering. Right From automation of testing to creating new software quality parameter. AI has the huge potential to transform and maintain the accuracy, reliability, correctness of software. A particular software or software project which is of high quality may not enjoy popularity among users. However, software or software projects that enjoy popularity are definitely having a qualitative standard and hence looking at the popularity of a software we may safely assume that the software or the software project is of high quality.

Software and software projects that are intended for the common man user strive for end user satisfaction. This may not be the case with scientific automated software which may be used only by technical people. For software projects with common people as the user base, there would be no real use of all the hard work even if the project is of good quality but fails to satisfy the users at large. Hence for such software projects, quality actually means popularity.

Software companies are working round the clock to improve user satisfaction and popularity and for that matter, they are improving the software from the user experience point of view. Other things for such projects seem to acquire lesser importance.

Many software firms have started speaking in terms of two types of quality. One is the code quality and the other is usability. If the product is meant to be marketed, the code quality is tested with a standard benchmark and passed. However, the usability remains continuously on the road to improvement. It is more than evident that what the firms target the most is the popularity of their product.

There have been instances that higher versions tend to freeze while lower versions continue to thrive because they are more popular. Then the so-called higher versions go out of commission, but the lower ones continue to roll. Can we consider the on-the-path-to-becoming-extinct software to be of higher quality compared to the ones that may be one version below but are liked and successfully used by the people at large?

The concept of quality of a software often becomes a victim of metaphysical approach, in which the software is studied in isolation and not in relation to other things that are associated with it (use of hardware, level of users, etc.). The authors believe that the quality of the software should be considered not in isolation but in relation to other things that are associated with it.

Another tendency is to understand the concept of quality of software project from the viewpoint of an absolutist where quality is something that once defined may not change and all engineers are supposed to adhere to the same, whatever may be the objective reality. On the contrary, the users believe that the concept of software quality should be viewed from the standpoint of something that is continuously evolving with time and that different ideas need to emerge and wither away as per the objective situation.

In the wake of the above considerations and conclusions the authors propose the following metrics for the measure of software projects as hosted on GitHub:

Stars: This is the random rating given by users and non-users who happen to pass by the project and happen to audit it due to some reason or the other. This may be just the casual passing glance of the onlooker, or the analytical view of the developer. However, a five-star rating definitely means that the project has earned some respect and in that sense some popularity.

Forks: Although the number of forks may be considered to be a quantitative parameter, it does indicate popularity. This is because only a person who is in some way interested with the software project actually will fork the project. Now, this forking may not be from any development point of view. It may be just a reuse of the codebase in some form. Or, it may be just for an audit of the codebase. Whatever may be the case, forks indicate interest in the project and hence definitely show popularity.

Contributors: An increase in the number of contributors indicates popularity. In open-source projects, most of the contributors are volunteers and are not paid. Hence, a person joins as a contributor only out of interest of some kind and hence the authors feel that the number of contributors are in a way an indicator of the popularity of the software.

Comments: Comments are also an indicator of popularity because more the number of comments, more is the involvement of people in the project and more is the rate of communication within it. It is here where Conway's law comes into reckoning.

Likes: In some software projects there is a feature of 'likes' much similar to that of social networking sites. This obviously becomes a metric for measuring popularity.

6. Future Scope

Measurement of the different metrics of popularity and comparing them to other metrics of software quality is something that needs to go on. But the authors feel that if some traditional software quality metric does not confirm to the popularity test, then that metric should be question marked. This may be done at least for those

categories of software which are intended for the common people. For such categories, traditional quality metrics may have little value which do not improve the user experience and involvement. Similarly newer quality metrics may be improvised that augment the popularity of the software by improving the user experience and overall performance. A study may be conducted by taking into consideration the software in relation to all other factors associated with it such as hardware and others. The study of quality in isolation needs to be done away with.

References:

[1]. Stahl, B.C., Artificial intelligence for a better future: an ecosystem perspective on the ethics of AI and emerging digital technologies. 2021: Springer Nature.
[2]. https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/
[3]. Sajnani, Hitesh & Saini, Vaibhav & Ossher, Joel & Lopes, Cristina. (2014). Is Popularity a Measure of Quality? An Analysis of Maven Components. Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014. 231-240. 10.1109/ICSME.2014.45.
[4]. Siavvas, Miltiadis & Jankovic, Marija & Kehagias, Dionysios & Tzovaras, Dimitrios. (2018). Is popularity an indicator of software security? A preliminary study on Maven Repository. 10.13140/RG.2.2.21269.58085.
[5]. Cohen J., Statistical Power Analysis for the Behavioral Sciences, Hilsdale N.J., Earlbaum Associates, 1988
[6]. I. Alsmadi and I. Alazzam, "Software attributes that impact popularity," 2017 8th International Conference on Information Technology (ICIT), 2017, pp. 205-208, doi: 10.1109/ICITECH.2017.8080001.
[7]. Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the Characteristics of High-Rated Apps? A Case Study on Free Android Applications," in 31st International Conference on Software Maintenance and Evolution (ICSME), pp. 1–10, 2015.
[8]. G. Bavota, M. Linares-Vásquez, C. E. Bernal-Cárdenas, M. D. Penta, R. Oliveto and D. Poshyvanyk, "The Impact of API Change- and FaultProneness on the User Ratings of Android Apps," in IEEE Transactions on Software Engineering, vol. 41, no. 4, pp. 384-407, April 1 2015.
[9]. L. Guerrouj, S. Azad and P. C. Rigby, "The influence of App churn on App success and StackOverflow discussions," IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal, QC, 2015, pp. 321-330,2015.
[10]. M. Carvalho, J. DeMott, R. Ford and D. A. Wheeler, "Heartbleed 101," in IEEE Security & Privacy, vol. 12, no. 4, pp. 63-67, July-Aug. 2014, doi: 10.1109/MSP.2014.66.
[11]. https://www.trigent.com/resources/whitepaper/pitfalls-of-not-transitioning-from-waterfall-to-agile-devops.html
[12]. Bissyandé, Tegawendé & Thung, Ferdian & Lo, David & Jiang, Lingxiao & Reveillere, L.. (2013). Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects. Proceedings - International Computer Software and Applications Conference. 303-312. 10.1109/COMPSAC.2013.55.
[13]. Kwan, I., Schroter, A., & Damian, D. (2011). Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project. IEEE Transactions on Software Engineering, 37(3), 307–324. doi:10.1109/tse.2011.29
[14]. Betz, S., mite, D., Fricker, S., Moss, A., Afzal, W., Svahnberg, M., … Gorschek, T. (2013). An Evolutionary Perspective on Socio-Technical Congruence: The Rubber Band Effect. 2013 3rd International Workshop on Replication in Empirical Software Engineering Research. doi:10.1109/reser.2013.8
[15]. M. Conway, "How do Committees Invent?," Datamation, vol. 14, no. 4, pp. 28-31, 1968
[16]. Erich Wimmer, Volker Eyert, Kurt Stokbro, White paper for standards of modelling software development, EMMC-CSA – GA N°723867 (2019) https://emmc.info/wp-content/uploads/2018/04/EMMC-CSA-WP-StandardsMOD_SOFTW-DEV.pdf
[17]. White Paper on Why Master Data Management is Critical to the Modern Customer Data Platform, (2021), Deloitte, https://media.bitpipe.com/io_15x/io_155658/item_2336767/IO155658_Deloitte_2342337_EGuide_4.14.2021.pdf
[18]. White Paper on Your Ultimate Chat Solution Checklist (2021), Genesys, https://media.bitpipe.com/io_15x/io_154292/item_2264634/Your-ultimate-chat-solution-checklist-ebook-GB-pictures.pdf
[19]. White Paper on An Expert Guide to Desktop Virtualization Implementation (2019), Dell, http://viewer.media.bitpipe.com/1182112995_571/1263417896_281/DellsVirtualDesktopVirtEguide.pdf
[20]. White Paper on Usability as ERP Selection Criteria (2008), IFS, http://hosteddocs.ittoolbox.com/wpusability.pdf
[21]. White Paper on High Usability Check List, Outsystems, https://www.bitpipe.com/data/loadAsset.action?resId=1378207002_81