Effort Estimation for Free Open Source Software Projects

Bharath Nalluri
bn448@nau.edu
Northern Arizona University
Flagstaff, Arizona, USA

Venkata Vaishnavi Jalireddygari
vj275@nau.edu
Northern Arizona University
Flagstaff, Arizona, USA

Saikiran Veerla
sv742@nau.edu
Northern Arizona University
Flagstaff, Arizona, USA

Sai Kiran Potru
sp2584@nau.edu
Northern Arizona University
Flagstaff, Arizona, USA

Abstract

Software effort estimation is considered the process for the prediction of the amount of effort to maintain or develop software. Free Open Source Software Projects (FOSS Projects) have been gaining popularity since the late 1990s and this paper helps to find out the effort estimation for FOSS projects. When we have an open source repository for a project (in our case its GitHub), we consider three factors to calculate effort: the total number of unique users working on the project, the total number of unique working days worked by the users and the total number of commits made. Multiplying these three factors will give us an effort estimation for a specific repository. We took some keywords and looped through GitHub (using python) until we got repository names that matches the selected keywords. After getting all the repository links, we mined through them and calculated the effort for each repository. We also plotted graphs to show which repository has the highest calculated effort among the selected repositories.

Keywords: software development, effort estimation, FOSS, free open source software repositories, github

ACM Reference Format:
Bharath Nalluri, Venkata Vaishnavi Jalireddygari, Saikiran Veerla, and Sai Kiran Potru. 2022. Effort Estimation for Free Open Source Software Projects. In *Flagstaff'22: [Effort Estimation for Free Open Source Software Projects], December 12, 2022, Flagstaff, AZ.* ACM, New York, NY, USA, 5 pages. https://doi.org/12345.67890

, December 12, 2022, Flagstaff, AZ

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/12345.67890

1 Introduction

Software effort estimation is considered the process for the prediction of the amount of effort to maintain or develop software. Effort can be said in terms of the number of persons involved to the number of hours spent on the software or the total money spent on the software. If the effort estimation is done right for any project, it can be used as an initial variable to plan the budget for the project or to change or modify project plans for more efficiency and time complexity. There are several effort estimation approaches from various authors and researchers in practice now, however, there is no such model that accurately predicts the effort, because effort completely depends on person to hours and varies depending on the project [4].

Open Source Software projects (shortly called FOSS Projects) came into light in the late 1990s. Since then, these have been gaining popularity, and many developers and researchers have been using these for their analysis and research. For a FOSS project, some volunteers work on it and the effort can be calculated according to the users and hours, but the problem is volunteers keep moving in and out of the project. There can be additional people joining the project if they were paid [3]. In this case, calculating effort is complex because there is no exact information on who joined or left the project. There is a case study from a paper [3] that tried to maintain and study records of open stack authors and commits, this paper's authors also made an online survey for the time commitment of the open stack developers but the results are not much accurate comparing experimental and survey results. There are other approaches explained in the Related work section for the effort calculation.

To fix the issue, in this paper, we propose a new metric for effort calculation. To calculate effort, we mainly focus on three factors for a project repository: the total number of unique active users, the total number of unique active days, and the total number of commits. To do this, we chose

GitHub, an open-source data repository platform that neatly maintains commits, users, and other important things for a project. We accessed some repositories (we explained more in detail about data collection in 3.1.Data Gathering Section) from GitHub and tracked the above three factors using python selenium and scraping. With this metric, we can calculate effort with more ease and accuracy, since we are dependent on three main factors.

RQ: What is the highest estimated effort for FOSS project repositories?

To get results for the research question, it is important to get data from open-source repositories (in our case it is GitHub). We looped through each GitHub repository and got the number of users, commits, and other data. From there we used our metric to get effort. We will explain the methodology more in detail in the coming sections.

This paper is organized into several sections. Section 1 is the introduction. Section 2 is the related work. Section 3 and 4 is the methodology and the results. Section 5 is Discussions. Section 6 is Threats to validity. And finally, section 7 is the conclusion and future work.

2 Related Work

After a lot of related work research on effort estimation and its techniques, we found some variables and other factors that will affect the estimation. We have referred to other publications and their methods, from the reference[4], we can categorize estimation into these approaches: Expert estimation, Formal estimation model, and Combination-based estimation. Some of the popular implementation approaches for formal estimation methods are COCOMO[8], SLIM[9], Weight Micro Function Point[10], Function Point Analysis, Use Case Points, Object Points[11], and Agile software development[12]. Some popular approaches for Expert estimation are Project management software[13], Planning Poker[14], and Wideband delphi[15]. Some of the combination-based estimation approaches are Expert judgment based on estimates from a parametric model and group estimation, and the Average of an analogy-based and a Work breakdown structure-based effort estimate[16]. Among all the models, empirical estimation models are considered to be better accurate models. SLIM, FP analysis schemes, and COCOMO are the best models so for in this category[6][7].

Our approach for effort estimation is using a python scraper. For this, we chose three main factors that are responsible for the effort. Similarly, many other papers chose factors that they thought were important[9]. Our estimation method and metric are simple to use and get the effort. So among all the methods, we have discussed above, our model, 'effort estimation using python scraping', is simple to use as long as you have the repository data. We can mine this repository data and get the effort estimate easily. Some factors might affect our results and those were discussed in the Threats to Validity section.

3 Methodology

3.1 Data Gathering

For our analysis, we chose GitHub as our source because it is free and has millions of open source repositories and they organize commits, and user details (which are the main considered factors for our method), in the best possible way, compared to other open source repositories. We did python scraping to get data from GitHub repositories.

Selection Criteria for this paper, we considered these four keywords- Python, C++, Ruby, and Java. Using python, we looped through GitHub repositories to get 8 repository links for each keyword. Now we will have 8 repositories for python, 8 for C++, 8 for Ruby, and 8 for Java. We saved the repository links to a text file as well. We looped through each repository from the text file and created a data frame in pandas that will have these columns – Date, User, and Repository. We got all the commits that were made between the years 2018 – 2022. This is how our data frame will look after the gathering.

	Date	User	Repository
0	Nov 10 2022	itsAkshayDubey	TheAlgorithms/Python
1	Nov 10 2022	pre-commit-ci[bot]	TheAlgorithms/Python
2	Nov 10 2022	cclauss	TheAlgorithms/Python
3	Nov 10 2022	github-actions	TheAlgorithms/Python
4	Nov 10 2022	cclauss	TheAlgorithms/Python
...	...	...	...
6502	Nov 4 2022	veblush	grpc/grpc
6503	Nov 4 2022	github-actions[bot]	grpc/grpc
6504	Nov 4 2022	Vignesh2208	grpc/grpc
6505	Nov 4 2022	markdroth	grpc/grpc
6506	Nov 4 2022	Vignesh2208	grpc/grpc

Figure 1. Final data frame having three columns. We can see, we have all the dates that commits were made, the name of the users who made the commits, and the repository names.

3.2 Approach

As a reminder, our research question is effort estimation for FOSS projects and now that we have our data set ready(which are GitHub repositories and its data), we are ready for effort estimation. We propose a new metric for the estimation. From the data frame we have, we get the number of days, number of users, and number of commits they made. But this will encounter a problem. Sometimes a user might create

more than one commit per day. This will impact (potentially increase) the final effort estimate. So, to avoid this problem, we consider unique information from the data frame. So, our factors from the data frame will be modified to the number of unique users that made the commits, the number of unique dates the commits were made, and the number of unique commits.

Now we have the number of unique dates(D), the number of unique authors(A), and the number of unique commits(C). We multiply these three factors to get effort. In brief, our formula would be: E = A*C*D. Where E is the effort for each repository, A is the number of unique active users, C is the number of unique commits, D is the number of unique users. Our final data frame after the above formula will be as follows

4 Results

Our resultant data frame after applying our approach will be as figure 2.

	Repository	Effort
0	TheAlgorithms/Python	3046042
1	geekcomputers/Python	56700
2	walter201230/Python	4032
3	injetlee/Python	30
4	kubernetes-client/python	88200
5	Show-Me-the-Code/python	157500
6	xxg1413/python	103950
7	grpc/grpc	111720

Figure 2. Estimated effort for the repositories. We can see, we have all the dates that commits were made, the name of the users who made the commits, and the repository names.

From Figure 2, we can see the estimated effort corresponding to each repository. The highest calculated effort is for the repository 'TheAlgorithms/Python' and the least calculated effort is 'Injetlee/python'. Since the effort value is too high, we scaled this effort value using Min-Max Scaling. After scaling, our final output will be figure 3.

4.1 Additional Results

For the effort estimation values we got for the repositories, we did some additional data analysis. Using a bar plot, we

	Repository	Effort
0	TheAlgorithms/Python	1.0
1	geekcomputers/Python	0.018605
2	walter201230/Python	0.001314
3	injetlee/Python	0.0
4	kubernetes-client/python	0.028946
5	Show-Me-the-Code/python	0.051697
6	xxg1413/python	0.034117
7	grpc/grpc	0.036668

Figure 3. Effort after min-max scaling. The repository 'TheAlgorithms/Python' has the highest effort so it is scaled to the maximum value (i.e. 1) and the other repository's effort was scaled accordingly.

plotted graphs for the results. Figure 4 plots the graph between the x-axis being effort value and the y-axis being repositories

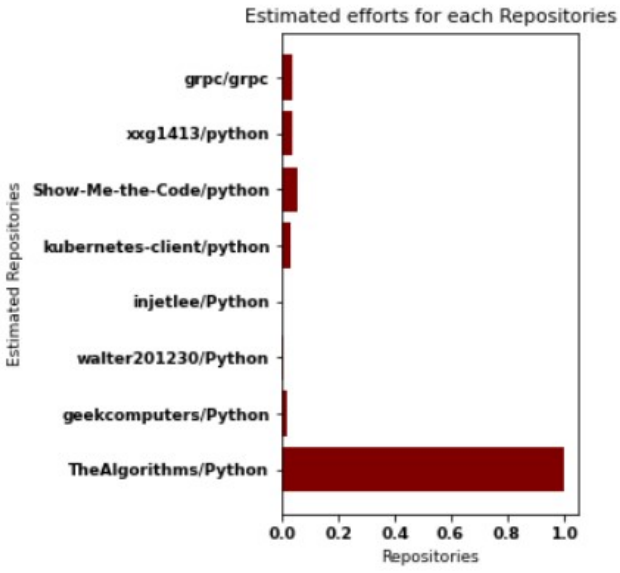

Figure 4. Estimated efforts for each repository

Figure 5 is a plot between the number of unique days and repositories.

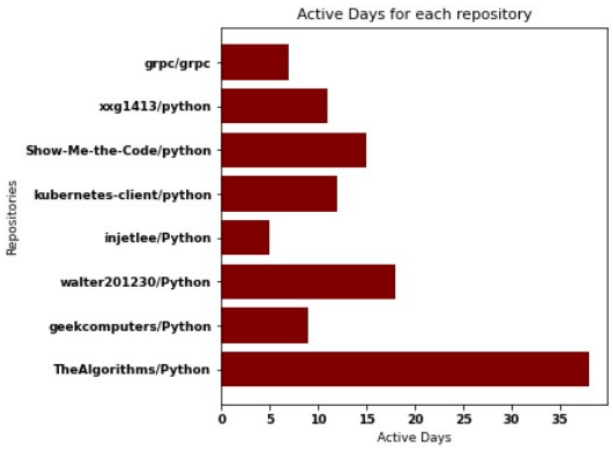

Figure 5. Active days for each repository

5 Discussions

Unlike other papers, this paper mainly focused on two things. They are selecting keywords and selecting factors from those repositories. Keywords selection is completely user choice. Anyone can select any keyword, and from the selected keywords we can get repositories from a certain period(first eight repositories in our case) from GitHub. Once we have all the repositories ready, we will now select factors to calculate effort, which in our paper are unique days, unique users, and commits made.

This metric is very useful for open-source project effort estimation. From our results, effort estimation for other projects from open-source repositories can also be predicted. This will help volunteer researchers who want to know the approximate effort estimate before starting a project.

6 Threats to validity

Almost all empirical studies will have some threats to validity that affects the project and studies' results and conclusions. For our study, we found the following as our threats to validity.

Construct Validity: For our analysis, we selected parameters- the total number of unique active users, the total number of active hours spent by users, and the total number of commits for a repository. Considering these parameters, we calculated effort. This effort estimation might change completely when we use another parameter for the calculation. For example, if we consider commits that include coding or if we consider the length of commits to be a particular length, our results will change completely.

Internal Validity: We have analyzed repositories with limited data and commits. We have analyzed around 8 repositories per each keyword. Many papers have analyzed thousands of repositories with a huge amount of data(commits). In such a case our model might be inaccurate.

External Validity: Since we chose repositories that used languages Python, C++, Ruby, and Java, the effort might vary based on languages used in other repositories. We can't assure that if this language is used, this would be the effort. It varies depending on the keywords we selected. For example, if we have chosen keywords like JavaScript, PHP, C-Sharp, or any other languages, our data frame will be completely different and so the results.

7 Conclusion and Future work

As we understood how important is effort estimation in software engineering, In this paper we made an approach for FOSS projects that is clear and understandable. The parameters we considered can be called representatives for the project. But still, there are many other factors to be considered depending on the project's needs. If the project needs more coding-related things, include a parameter for calculation that keeps track of commit type i.e. either a code commit or text commit. If it is a code commit, you can count that, or else you can skip the commit from counting.

We would also like to develop this metric in a way that if the user or researcher selects hundreds of keywords and the repositories for the selected keywords has lakhs and lakhs of commits, we should be able to get the effort with minimal time and space complexity.

Since the approach of this paper is calculating effort for an open source repository, in the future we could consider developing this approach in such a way that, if a company or group of volunteers have rough data about people working on the project and the time required for each person to do a specific task, using these parameters we could generate rough effort estimation before the starting the project. Again, the parameters for effort estimation have to be specified by the company so that we can create a model accordingly or we can create a model in a way that researchers can select the parameters.

References

[1] Briand, L. C. and Wieczorek, I. (2002). Resource estimation in software engineering. Encyclopedia of software engineering. J. J. Marcinak. New York, John Wiley & Sons: 1160-1196.

[2] Robles, Gregorio, et al. "Development effort estimation in free/open source software from activity in version control systems." Empirical Software Engineering 27.6 (2022): 1-37.

[3] Robles, Gregorio, et al. "Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack." Proceedings of the 11th Working Conference on Mining Software Repositories. 2014.

[4] Software development effort estimation. (2022, December 5).In Wikipedia. https://en.wikipedia.org/wiki/Software_development_effort_estimation.

[5] Finnie, Gavin R., Gerhard E. Wittig, and Jean-Marc Desharnais. "A comparison of software effort estimation techniques: Using function

points with neural networks, case-based reasoning and regression models." Journal of systems and software 39.3 (1997): 281-289.

[6] M. van Genuchten and H. Koolen, "On the Use of Software Cost Models," Information & Management, vol. 21, pp. 37-44, 1991

[7] T. K. Abdel-Hamid, "Adapting, Correcting, and Perfecting softwareestimates: Amaintenance metaphor " in Computer, vol. 26, pp. 20-29, 199.

[8] Barry, Boehm. "Software engineering economics." New York 197 (1981).

[9] Putnam, Lawrence H.; Ware Myers (September 2003). Five core metrics: the intelligence behind successful software management. Dorset House Publishing. ISBN 0-932633-55-2.

[10] Ahn, Yeon-S. "An Enhanced Function Point Model for Software Size Estimation: Micro-FP Model." Journal of the Korea Society of Computer and Information 14.12 (2009): 225-232.

[11] Finnie, Gavin R., Gerhard E. Wittig, and Jean-Marc Desharnais. "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models." Journal of systems and software 39.3 (1997): 281-289.

[12] Usman, Muhammad, et al. "Effort estimation in agile software development: a systematic literature review." Proceedings of the 10th international conference on predictive models in software engineering. 2014.

[13] Project management software. (2022, November 29). In Wikipedia. https://en.wikipedia.org/wiki/Project_management_software

[14] Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. Hawthorn Woods: Renaissance Software Consulting, 3, 22-23.

[15] Gandomani, Taghi Javdani, Koh Tieng Wei, and Abdulelah Khaled Binhamid. "A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique." International Journal of Software Engineering and its applications 8.11 (2014): 173-182.

[16] Jørgensen, Magne, Barry Boehm, and Stan Rifkin. "Software development effort estimation: Formal models or expert judgment?." IEEE software 26.2 (2009): 14-19.