PAPER

olivier-oss-myth

Olivier NOURRY ***and*** **m-kondo,**

Question (Myth) Question: Do OSS project contributors frequently change?

Description (background) Throughout their lifetime, open-source software systems will naturally attract new contributors and lose existing contributors. Not all OSS contributors are equal, however, as some contributors within a project possess significant knowledge and expertise of the codebase (i.e., core developers). When investigating a project's ability to attract new contributors and how often a project loses contributors, it is therefore important to take into account the expertise of the contributor. **Fact lead (One-sentence summary, similar to a topic sentence)** Since core developers are vital to a project's longevity, we therefore aim to find out: can OSS projects attract new core developers and how often do OSS projects lose core developers? **Fact data (Results)** To investigate core developer contribution patterns, we calculate the truck factor (or bus factor) of thousands of OSS projects calculate how often TF developers join or abandon OSS projects. We find that 66% of our studied projects have experienced losing their entire core development team. Our results also show that in 55.59% of cases, this project abandonment happens within the first 4 years of a project's life. We also find that more than half of projects that were abandoned had 5 or more core developers prior to the abandonment. Finally, we find that only 27.53% of projects that were abandoned were able to attract at least one new TF developer. **Insight (Discussion)** Our analysis shows that it is not uncommon for OSS projects to lose their initial core development team. We also find that the first four years of development is when projects are most at risk of losing their core developers. We also find that even projects that have several core developers are at risk of losing their development team. Finally, our results indicate that most projects that lose their initial core development team will not be able to attract new core developers to continue development activities.

1. Introduction

The Open-source software ecosystem has become one of the most important pillar of software development over time. Today, almost every software company in the world uses open-source software to some extent. Open-source projects such as Linux, kubernetes, Docker, Tensorflow, Apache HTTP Server have revolutionized the way software development is conducted and are used by the entire software development world. To maintain software development activities, these critical open-source projects all depend on open-source contributors to keep the project active and add new features. Specifically, these projects often tend to rely on a few core developers which have been actively working on these projects for years and are very knowledgeable about the codebase. Consequently, the loss of expertise incurred by the loss of core developers (or turnover) can have significant impact on a project's development and the overall productivity of the development team. To get a better understanding of open-source project abandonment, researchers have therefore tried to conduct studies to study open-source core developers' development patterns [**?**].

One common metric to identify these core developers in open-source projects is called the truck factor (or bus factor). The truck factor metric refers to the amount of developers that can stop contributing (or get hit by a truck) before a project is at risk of dying. When all truck factor developers (or core developers) quit a project we refer to this event as a Truck Factor Developer Detachment (TFDD). Conversely, if a project has experienced a TFDD and is currently inactive or at risk of dying but is able to attract a new core developer, we define this event as a project survival. Using these metrics, researchers have been able to study the development activity of core developers in open-source projects [1]–[3]. Due to the heavy computational cost of calculating the truck factor, most studies so far have been conducted with less than 50 open-source projects. To the best of our knowledge, as of 2024, only one study (led by Avelino et al. [3]) has used over 1,000 projects to study developers' software development activities. In this study, Avelino et al. compute the truck factor in 2,000 to study the abandonment of open-source projects by open-source contributors. While 2,000 projects is a significant leap over previous studies that used the truck factor metric, 2,000 projects is still too few projects to get an overview of the entire ecosystem and truly understand how common often core developers abandon open-source projects.

To address this limitation, we therefore decided to

DOI: 10.1587/trans.E0.??.1

conduct the first large scale empirical study using the truck factor by replicating Avelino et al.'s study using 5000 projects. In this work, we therefore aim to address the following research questions.

- RQ1) How common are TFDDs in GitHub projects?
- RQ2) How often do open-source projects survive a TFDD?
- RQ3) How do surviving projects differ from non-surviving ones?

2. Background and Related Work

2.1 Studies on developers' contribution patterns

Due to how critical open-source projects are to software development, some work has already been conducted to study aspects of project sustainability and developer activity in open-source projects.

Ferreira et al. [4] investigated the turnover of core developers in 174 open-source projects and found that there was significant developer turnover in the studied projects. From their analysis, they found that larger projects and projects that were owned by an organization both showed high rates of developer turnovers. Their results also show that projects with higher turnover tend to be slower at fixing bugs and addressing issues.

Lin et al. [5] also studied developer turnover in 5 large industrial projects. Their results show that developers with higher ownership of the codebase tend to be more likely to stay than developers that mostly work on files created by other developers. They also find that developers that work on the source code tend to be part of a project for longer than developers that work mostly on documentation.

Other aspects of open-source contributors' development activities have also been studied. Qiu et al. [6]interviewed 15 open-source contributors to understand how open-source developers choose a project to contribute. From these interviews, they then quantitatively measure 11 factors in 9,977 projects and show that open-source developerse are less likely to contribute to projects that have strict contribution guidelines.

2.2 Truck factor studies

The concept of truck factor (or bus factor) was first used in the context of software engineering at the start of the millennium and was defined as the number of developers that need to stop contributing (or get hit by a truck/bus) for a project to be at risk of dying[?]. Over time, several implementations and algorithms have been proposed to calculate the truck factor [2], [7]–[9]. As of 2024, multiple studies have used this metric to investigate the activities of core developers in open-source projects.

In 2010, Ricca et al. proposed one of the earliest implementation of the truck factor in the context of software engineering. Using their tool, they calculated the truck factor of 20 open-source projects using different threshold and found that projects typically rely on few truck factor developers to keep development activities going. Torchiano et al. [1] also measured the truck factor in 20 open-source project in their 2011 study where they tried to calculate the theoretical maximum truck factor value. Their analysis show similar patterns as Ricca et al.'s results where projects seem to rely on very few core developers to maintain development activities.

Calefato et al. used the truck factor to study the abandonment of open-source projects by developers. In their study, they proposed a method to detect which developers have abandoned open-source projects and validated their approach with real open-source developers. Using their approach, they then studied developer abandonment in 18 open-source projects. Their results show that all open-source core developers take at least one break from open-source contributions and that 45% of them will completely disengage from contributing to an open-source project for at least one year. Their study also shows that developers have between 35% and 55% chance of returning to an open-source project after abandoning the project.

For our study, since we aiming to conduct a large scale empirical study, we needed an implementation that was reliable but also that could scale well with large projects that have dozens or hundreds of contributors. We therefore decided to use Avelino et al.'s [2] implementation because since it proved to be able to handle the analysis of 2,000 repositories in Avelino et al.'s 2019 study [3]. To ensure that our truck factor measurements were reliable and reproducible, we also decided to use an openly available (on GitHub[†]) implementation of Avelino et al.'s truck factor algorithm rather than re-implementing our own version of the algorithm.

3. Methodology

Dataset selection and filtering. To find open-source projects, we first used the publicly available libraries.IO [?] dataset which contained the names of over 37.7 million open-source source projects along with other metrics such as where the project repositories are hosted, when the projects were created, how many stars each project has, and several other metrics. From this large dataset, we then applied a set of filters with the goal of keeping as many projects as possible while minimizing the chances of investigating toy projects. Additionally,

[†] https://github.com/aserg-ufmg/truck-factor

because the truck factor is calculated on a yearly basis, our filters needed to ensure that the remaining projects had enough development history to calculate the truck factor. Our filtering criteria were therefore as follow: each project had to have a minimum of 20 stars, 10 contributors, could not be a fork, had to be hosted on GitHub, and needed at the very least two years of development experience (i.e., a project created in 2024 was not elligible). After applying these filters, 5000 projects remained and were used for our study.

Data mining. To calculate the yearly truck factor in each project, we first extracted the creation date of all repositories in our dataset. From that initial creation date, we used the *git checkout* command to jumped ahead in each project's development history one year at a time. During each jump, we executed the truck factor calculation tool[†] which would calculate the commit and file information of a project, determine the main programming language of the project then calculate the number of truck factor developers. Following the original paper's methodology we also data mined the name and emails of all contributors in each project to find similar names or email addresses and map them to a single entity/developer. This process was done in order to avoid cases where a developer had multiple accounts or would contribute to a GitHub repository using a different account.

To compare repositories that survived TFDDs and those that did not (RQ3), we used the official GitHub API to mine the number of commits, the number of contributors, the number of files and the age of the studied repositories. For this part, we also used the GitHub API to find out the name of each repository's main branch since we only wanted to calculate the truck factor based on the contributions pushed to the main branch (not to development branches).

Data analysis. After the data mining process, we then aimed to identify instances of Truck Factor Developer Detachment (TFDD) in our studied projects. To find TFDDs, we once again started from the creation date of a repository and jumped one year a time to find the date of the last commit of each developer during that year. For a given year, if a developer had not contributed (had no commits) for at least a year, we considered that this developer had abandoned the project. To identify truck factor developer detachment, we therefore used our truck factor data to identify truck factor developers then looked at the state (active/abandon) of each of these developers and flagged a project as TFDD whenever, all truck factor developers had abandoned the project.

Using the truck factor data, the TFDD data, and the repository data, we then proceeded with the analysis to answer our research questions. To answer RQ1, we first calculated the number of projects that experienced a TFDD and how many TFDD each project experienced. To understand when open-source projects are most at risk of dying, we then calculated during which year TFDDs happened in our projects that experienced a TFDD. Additionally, we also summed up the number of TFDDs each year to calculate the cumulative percentage of TFDD year after year. Finally, we calculated how many truck factor developers our projects have to better understand how fragile (i.e., a single core developer) or robust (i.e., many core developers) open-source projects are in real world scenarios.

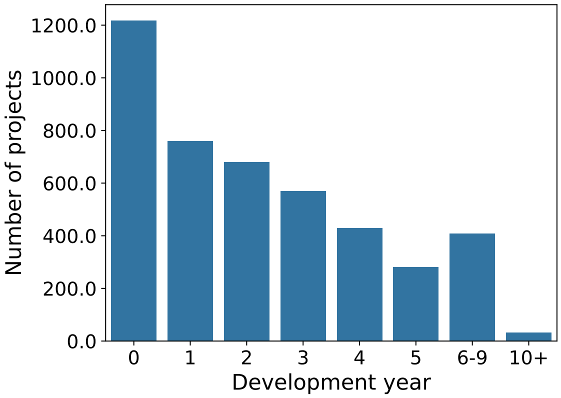
Next, we calculated how many of the projects that experienced a TFDD were able to survive (i.e., attract a new core developer). Additionally, we also calculated how many developers were involved with the survival of the studied projects (i.e., how many new core developers were involved with reviving the project).

Lastly, we calculated the number of commits, the number of files, the number of contributors and the age of each project (in days) to visualize the difference between projects that survive a TFDD and those that do not.

4. Results

4.1 RQ1) How common are TFDDs in GitHub projects?

From our 5,000 studied projects, we find that 3,682 (73.65%) projects have faced at least one TFDD throughout their lifetime. Calculating the number of TFDD that each project experienced, we then find that 3,052 projects experienced only a single TFDD, 569 projects experienced two TFDDs, 59 projects experienced three TFDDs, and two projects experienced four TFDDs during development.

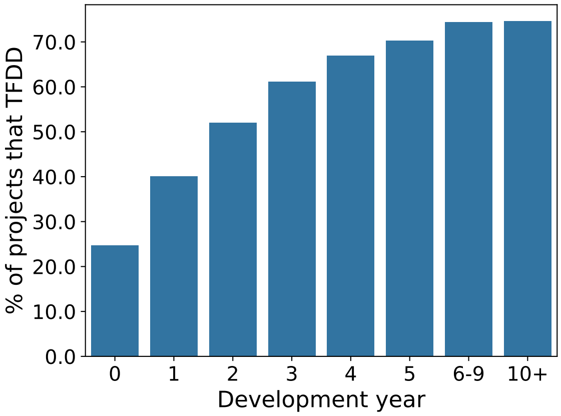


Most TFDDs happen within the first year of development. Figure 4.1 shows during what year our studied projects experienced their first TFDD. As Figure 4.1 shows, there are significantly more TFDDs

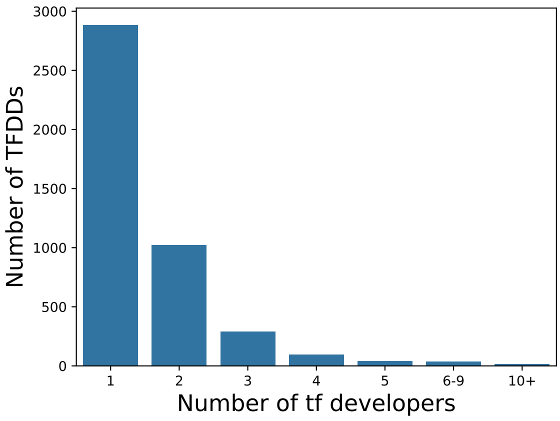
[†] https://github.com/aserg-ufmg/truck-factor

in the first year of development with a gradual decrease during each subsequent year.

Figure 4.1 shows the cumulative percentage of projects that have experienced a TFDD each year. Our results show that for projects who do experience a TFDD, 61% of them will TFDD within the first three years, 66% within the first four years, and 70% with the first five years.



Finally, Figure 4.1 shows the number of truck factor developers involved with our studied projects at the time of TFDD. As our results show, open-source projects relying on a single core developer to keep development activities active seem to be a common situation in the GitHub ecosystem.

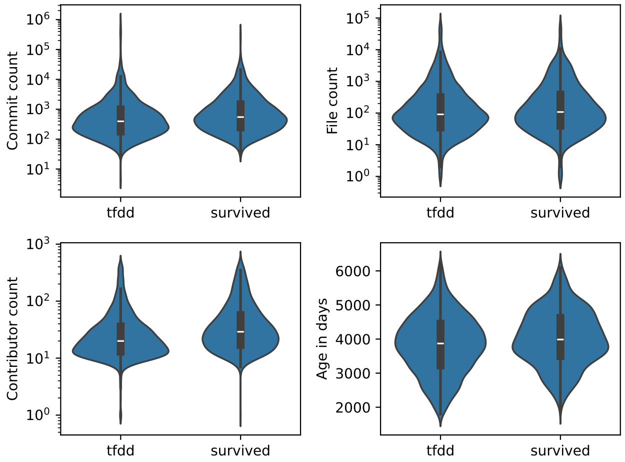


4.2 RQ2) How often do open-source projects survive a TFDD?

Most projects do not survive a TFDD. Calculating the number of projects that survived a TFDD, we find that only 1,042 (28,30%) of projects that faced a TFDD were able to survive and attract new core developers to continue development activities. For these 1,042 TFDD survivals, we then calculated the number of developers involved with each survival. Our results shows that for 925 of the survivals, only one developer was involved, two developers were involved for 112 survivals and in 5 cases three developers took part in a project's TFDD survival.

4.3 RQ3) How do surviving projects differ from non-surviving ones?

Surviving projects show more development activity than non-surviving projects across all studied metrics. Figure 4.3 shows the results obtained from calculating the studied metrics across all projects. From Figure 4.3, we find that projects that survive TFDDs have more commits and files than non-surviving projects. We also find that surviving projects tend to have more contributors and a longer lifespan than projects that do not survive. To ensure the statistical significance of our results, we then conduct a Mann-whitney test and find that all four studied metrics have a p-value < 0.05.



5. Conclusion

In this study, we investigated the activity of core open-source developers. Our results show that open-source projects are most at risk of getting abandoned at the start of the project's lifetime. Additionally, we also find that OSS projects often rely on a single developer to maintain development activities.

References

[1] M. Torchiano, F. Ricca, and A. Marchetto, “Is my project's truck factor low? theoretical and empirical considerations about the truck factor threshold,” Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics, p.12–18, Association for Computing Machinery, 2011.

[2] G. Avelino, L.T. Passos, A.C. Hora, and M.T. Valente, “A novel approach for estimating truck factors,” CoRR, vol.abs/1604.06766, 2016.

[3] G. Avelino, E. Constantinou, M.T. Valente, and A. Serebrenik, “On the abandonment and survival of open source projects: An empirical investigation,” CoRR, vol.abs/1906.08058, 2019.

[4] F. Ferreira, L.L. Silva, and M.T. Valente, “Turnover in open-source projects: The case of core developers,” Proceedings of the XXXIV Brazilian Symposium on Software Engineering, p.447–456, Association for Computing Machinery, 2020.

[5] B. Lin, G. Robles, and A. Serebrenik, “Developer turnover in global, industrial open source projects: insights from applying survival analysis,” Proceedings of the 12th International Conference on Global Software Engineering, p.66–75, IEEE, 2017.

[6] H.S. Qiu, Y.L. Li, S. Padala, A. Sarma, and B. Vasilescu, “The signals that potential contributors look for when choosing open-source projects,” Proc. ACM Hum.-Comput. Interact., vol.3, 2019.

[7] M. Ferreira, M.T. Valente, and K. Ferreira, “A comparison of three algorithms for computing truck factors,” 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC), pp.207–217, 2017.

[8] F. Ricca and A. Marchetto, “Are heroes common in floss projects?,” Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Association for Computing Machinery, 2010.

[9] E. Jabrayilzade, M. Evtikhiev, E. Tüzün, and V. Kovalenko, “Bus factor in practice,” Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice, p.97–106, Association for Computing Machinery, 2022.