App Abandonment in Mobile Marketplaces: The Impact of Developer Dropout and User Feedback

Kazuya MATUSHITA†, Olivier NOURRY†, and Yoshiki HIGO†

† Faculty of Computer Science, Osaka University Yamadaoka, Suita-shi, Osaka, 565-0871 Japan

E-mail: †{kaz-mtst,nourry,higo}@ist.osaka-u.ac.jp

Abstract Core developers are fundamental for the sustainability of mobile applications. In this study, we investigate the abandonment of mobile application by their development teams from two perspectives: 1) the reliance of mobile applications on few core developers, and 2) the impact of user reviews. We calculate the Truck Factor (TF) in 831 Android applications released on the Google Play Store and investigate Truck Factor Developer Detachment (TFDD) events in mobile applications. We also conduct a large scale qualitative analysis to investigate how user reviews correlate with the sustainability of a mobile software project. Our results show that half of our studied apps have faced abandonment from their development teams. Finally, our results show that even popular apps are at risk of being abandoned by their core developers.

1. Introduction

Since the advent of the smartphone, mobile applications have become a part of everyday life for billions of people around the world. With millions of mobile applications released through various app stores, mobile development is now a core pillar of software development and constitutes a significant portion of the open-source ecosystem. As with any ecosystem, some parts of the mobile application ecosystem thrive, while others die (i.e., get abandoned by their developers or users) and get replaced by newer apps. While prior studies have investigated thriving parts of the ecosystem by exploring what makes an app successful or by predicting the adoption of mobile applications [1]–[3], little work has been done to understand the factors that lead to a mobile application's abandonment within the app store.

While mobile applications are continuously being released, their sustainability depends heavily on the involvement of their development teams. Prior studies on open-source software systems have shown that many projects rely on a small set of "core developers" who perform most of the development and maintenance work [4]. Research in OSS further indicates that the departure of these core developers can significantly affect a project's longevity [5]. For example, Nourry et al. analyzed over 36,000 OSS projects and found that 89% lost their core development team at least once, often during early development, and only 27% were able to attract new core developers afterward. These findings highlight the essential role of core developers in sustaining a software system.

Because prior work on the role of core developers has focused on generic OSS projects, a knowledge gap remains in the context of mobile software development. Specifically, the impact of losing core developers on the sustainability of mobile applications is largely unexplored. Moreover, unlike OSS projects, mobile apps are strongly shaped by non-developer end users through marketplace factors such as reviews, download counts, ratings, and store visibility. It is therefore unclear whether the abandonment patterns observed in OSS projects also apply to mobile software projects.

In this study, we aim to elucidate what factors (or characteristics) lead to mobile application abandonment in the app store by focusing on two main aspects: human factor (i.e., the development team) and external factors (i.e., publicly available metadata from the app store). Specifically, we analyze 831 mobile applications to investigate the frequency and the characteristics of mobile software projects abandonment.

For the human factor, we use the concept of *Truck Factor (TF)* (or bus factor) to conduct our analyses. The truck factor refers to the minimum number of developers that need to leave a project before it becomes unsustainable or can no longer be effectively maintained [6]. In this context, we refer to the core developers which software projects depend on to continue development as TF developers. Building upon the truck factor concept, Avelino *et al.* [7] proposed the notion of *Truck Factor Developer Detachment (TFDD)* which refers to an event where all TF developers of a project leave or stop contributing, jeopardizing the project ' s longevity. TFDD thus provides a concrete way for researchers to investigate at what point an application gets abandoned by its core development team.

We break down our goal of understanding the factors that lead to a mobile application's abandonment into the following three research questions:

- RQ1: To what extent do mobile apps experience TFDD, and how frequently are they abandoned?
- RQ2: How do abandoned apps differ from surviving apps??
- RQ3: How do user reviews relate to TFDD events in mobile applications?

By answering these questions, this study aims to deepen

our understanding of app development sustainability and provide insights that support the maintenance of a healthy mobile application ecosystem. In Section 2, we provide background on the core concepts of this study, in Section 3, we breakdown our dataset creation and our overall methodology, in Section 4, we show the results of our analyses, in Section 5, we discuss the implications and main takeaways of our findings, and finally we conclude the paper in Section 6.

2. Background and Related Work

2.1 Truck Factor and Core Developers

The *Truck Factor* (TF) is a widely used metric for assessing how vulnerable a software project is to the loss of its key developers. It is defined as the minimum number of core developers whose departure would cause the project to stall or substantially slow down [7].

Several prior studies have used the truck factor to investigate software development activities in open-source projects. Ricca *et al.* [8] calculated the truck factor for 20 open-source projects and found that most of them relied on very few developers to maintain development activities. Avelino *et al.* [4], [7] conducted multiple studies investigating core developers in open-source projects using the truck factor and observed the same development pattern where most projects rely on one or two core developers to take on most of the development workload. Ferreira *et al.* [9] and Calefato *et al.* [10] both investigated core developer development patterns in open-source projects and observed significant core developer turnover with Calefato *et al.* also finding that 45% of core developers will completely stop open-source contributions for extended periods of time. Nourry *et al.* [5], analyzed over 36,000 projects, and found that 89% experienced TFDD at least once, often in their early stages, and only 27% subsequently attracted new core developers.

Several tools have been proposed to calculate the truck factor [4], [8], [11], [12]. In this study, we employ the algorithm proposed by Avelino *et al.* [4] to calculate the truck factor. This algorithm estimates code ownership by tracing the change history of each source file and calculating each developer's Degree of Authorship (DOA). Developers whose combined ownership accounts for at least 50% of a project's code base are considered TF developers. To ensure consistency with prior work, we use the official tool[1] released by Avelino *et al.* to calculate the truck factor of OSS projects. Additionally, as Avelino *et al.* previously proposed, we also use a threshold of 1 year without contributions to consider a developer as inactive for the TFDD calculation process.

2.2 Mobile App Development and Software Project Sustainability

Unlike traditional Open-Source Software (OSS) projects, whose sustainability is largely shaped by contributor retention, mobile applications must also navigate marketplace-specific dynamics. Because apps are distributed through platforms such as Google Play, their survival depends on factors that directly affect visibility and user acquisition, including release timing, update frequency, installation counts, and user-generated signals such as review volume, sentiment, and review type [13]. App store algorithms heavily weigh ratings, reviews, and installation numbers when determining search ranking and recommendation likelihood [14], making user feedback a central indicator of an app ' s health [15]. Positive sentiment can enhance an app ' s visibility, whereas sustained negative sentiment or increasing problem-oriented reviews may reduce discoverability and threaten long-term sustainability. Overall, mobile software development and maintenance require continuous monitoring of app store metrics, proactive response to user feedback, and strategic management of app quality to ensure long-term sustainability.

3. Methodology

In this section, we describe our methodology to collect and filter our dataset and conduct our analyses.

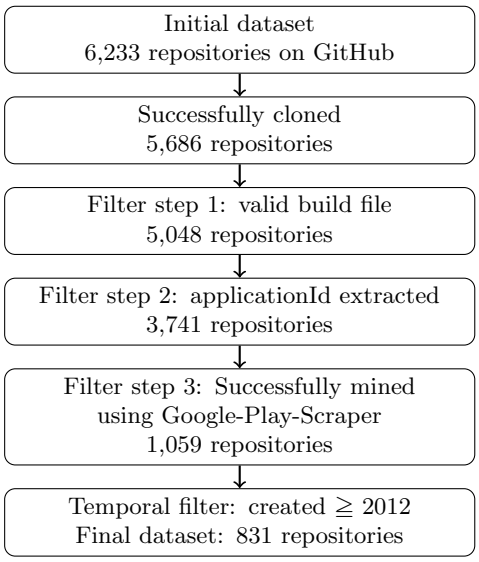

Fig. 1 Overview of dataset filtering process and remaining repositories at each step.

3.1 Dataset Creation

We first obtained a list of 6,233 open-source Android application repositories [16] hosted on GitHub, of which 5,686 were successfully cloned. To ensure that each repository corresponds to a valid Android application with accessible marketplace metadata, we searched the directory structure of all projects to find valid Gradle build files.

From the *build.gradle* files, we extracted each project's applicationId, then queried the Google Play API using the *google-play-scraper*[2] python package to collect app marketplace metadata of all projects. This filtering step resulted in 1,059 valid repositories.

Using the Google Play API, we were able to collect extensive metadata for over 1,059 Android applications. This data included general information such as the application title, user ratings, install counts, number of reviews, release date, and last update timestamp.

Next, we again used the *google-play-scraper* package to collect user reviews for our analyses. From this mining process, we were able to collect all user reviews for our studied projects along with additional information such as the review identifier, author name, submission date, star rating, review text, thumbs-up count, app version, and any developer replies. This provided a comprehensive corpus of user feedback reflecting real-world user experiences.

(1): https://github.com/aserg-ufmg/truck-factor

(2): https://pypi.org/project/google-play-scraper/

Finally, we wanted to ensure that our analyses accurately reflect the modern state of mobile software development and app marketplace usage. Because the Android app ecosystem was effectively established only after the smartphone boom and the introduction of the Google Play Store in 2012, we excluded all projects created before 2012. Our final dataset therefore consists of 831 Android applications.

3.2 Truck Factor and Truck Factor Developer Detachment Calculations.

To identify applications that were abandoned by their developers (i.e., experienced a TFDD), we first needed to calculate the yearly truck factor of every application in our dataset. For each repository, we retrieved it initial creation date and then traversed the project's development history in one-year increments using the *git checkout* command. At each yearly checkpoint, we calculated the truck factor using Avelino *et al.*'s tool, which automatically collects commit and file-level information of a project, determines the main programming language, and computes the number of truck factor developers.

Since the tool relies on cumulative contribution history, developers who had made substantial contributions in the past but were inactive in the target year could still appear as core developers. To address this issue, we used the *git log* command to check the commit history of each developer. Specifically, we verified whether each candidate developer had actually committed during the corresponding year, and excluded those who did not. This filtering ensured that only active contributors were considered as core developers in each yearly snapshot. For all instances where a given year had no active core developers, we marked the project has having experienced a TFDD.

3.3 LLM-Based Analysis of User Reviews

In RQ3, we investigate user reviews to find patterns that could indicate that if an app is likely to remain stable or if it is likely to face a TFDD. Specifically, we first conduct a qualitative analysis of user reviews to classify the purpose of each user review based on the content of the review. For instance, a review may give feedback to the developer, or it could let the developer know that there is a bug that needs fixing.

To get a list of functional labels for this analysis, we first randomly sampled 500 hundred user reviews. We then used ChatGPT-4 to assign describe the purpose of each review. From this initial classification, two authors manually checked the labeling, corrected wrong labels and derived a list of labels describing the purpose of user reviews. From this analysis, we ended up with only three labels:

- **Feedback**: A user provides feedback to the developer on their application: *"Great browser"*
- **Feature**: A user requests a new feature to the developer: *"I wish it had a translate feature like Google does..."*
- **Bug**: A user reports a bug to the developer: *"I can't import pdf file. "FormatException: Unexpected extension byte (at offset 11)" . But no problem in latest nightly version"*

After deriving a list of labels, we used the GPT-4o mini model via OpenAI's API to classify all 127,131 reviews of TFDD apps, then repeated the process for Stable apps after randomly sampling the same number of reviews.

After performing the functional analysis of user reviews (i.e., the classification Feedback/Feature/Bug classification), we reused the same dataset to conduct a second qualitative analysis with the purpose of classifying the overall sentiment the sampled user reviews. For this analysis, the LLM was tasked to analyze the content of each review and assign one of three labels: Positive/Neutral/Negative.

4. Results

4.1 RQ1: To what extent do mobile apps experience TFDD, and how frequently are they abandoned?

Motivation. The sustainability of software projects heavily depends on the continued participation of core developers. For mobile applications, which require frequent updates for compatibility and security, the departure of core developers can have particularly severe consequences.

Approach. To better understand how vulnerable are mobile applications to abandonment, we first analyzed the annual trend in the number of core developers across all applications between the years 2013 and 2024. We then calculated the ratio of mobile application projects that experienced a TFDD and classified each app into one of two groups:

- **Stable:** Projects that have never experienced a TFDD.
- **TFDD:** Projects that experienced at least one TFDD event.

For the TFDD group, we further distinguished between two subcategories:

- **Survival:** Projects where core developers resumed their development activity or new core developers joined the project following a TFDD event.
- **Abandoned:** Projects where no core developer activity ever took place following a TFDD event.

After categorizing all projects, we calculated the ratio of applications that were able to survive a TFDD. We also further investigated the timing when TFDD events took place relative to the project's life, its release date and the date of its last update on the app marketplace.

Results. Table 1 shows the proportion of projects that experienced a TFDD. Out of 831 applications, we find that 422 (50.7%) experienced at least one TFDD, while 409 (49.2%) never experienced any TFDD event.

Table 1 Frequency of TFDD in mobile applications

Category	# of Apps	Percentage
TFDD	422	50.7%
Stable	409	49.2%
Total	831	100%

Ratio of Survival and Abandoned projects. As shown in Table 2, among projects that experienced a TFDD, 144 (34.1%) were able to survive and have a core developer continue maintenance and development activities, while 278 (67.9%) were abandoned.

As shown in Figure 2, the median number of TF developers throughout the years remains stable across our studied projects for both Stable apps and TFDD apps. The median

Table 2 Survival vs. abandonment after TFDD

Category	# of Apps	Percentage
Survival	144	34.1%
Abandoned	278	67.9%
Total (TFDD)	422	100%

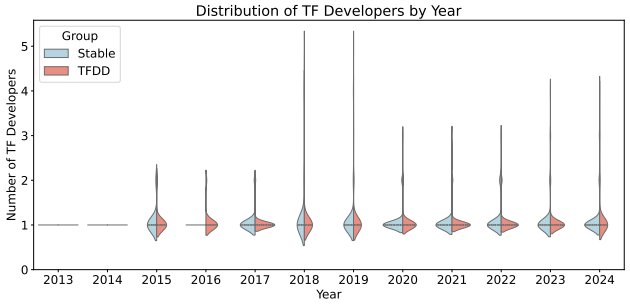

Fig. 2 Distribution of the number of TF Developers by year.

for both groups was 1, while the mean was slightly higher for Stable projects (1.083) compared to TFDD projects (1.063).

For most apps, we find that TFDD events take place during the early part of the project's lifecycle. For over 80% of projects that will experience a TFDD during their life, the TFDD event will take place within the first 3 years of development as shown in Figure 3. Figure 4 shows the time elapsed between a TFDD event and the time the app was released on the official marketplace (for apps that experienced a TFDD). Our results show in most case the TFDD will take place early after being published and that close to 20% of mobile application will even experience a TFDD event prior to their release on the official app store (as shown by negative numbers on the plot).

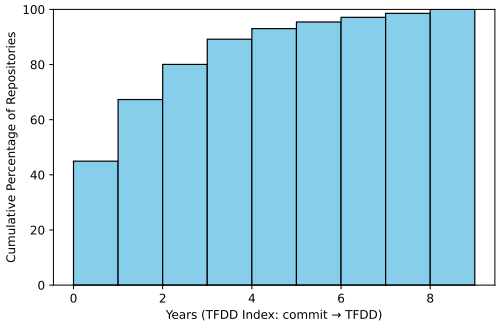

Fig. 3 Cumulative proportion of projects that experience a TFDD within N years of their creation.

For Abandoned projects, we investigated the time elapse between when a TFDD occured and the last time an update was pushed to the app on the app marketplace. As shown in Figure 5, over 80% of TFDDs happen within the first year when a developer does not update its application on the app store. In a few cases, we also found that a mobile software project had already faced a TFDD, survived, then released the app on the marketplace as shown by the negative numbers in Figure 5.

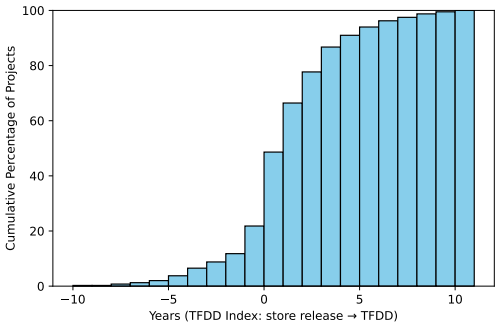

Fig. 4 Cumulative proportion of projects that experience a TFDD within N years of their release on the app marketplace.

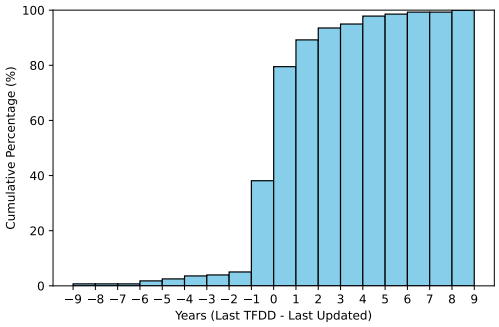

Fig. 5 Time elapsed between the date when projects experience a TFDD and the last update of the app on the app marketplace (cumulative proportion of Abandoned projects)

We find that half of our studied mobile applications experienced a TFDD with most TFDD events taking place within the first three years of development. Our results indicate that only 34.1% of mobile applications are able to recover from a TFDD.

4.2 RQ2: How do abandoned apps differ from surviving apps?

Motivation. This analysis aims to clarify the differences in app characteristics between TFDD and Stable applications based on app store metadata. Specifically, we aim to find out if applications are usually abandoned by the developers due to low adoption by users or low users ratings.

Approach. For each group (Stable and TFDD), we analyze the distributions for the app ratings, and the number of installations and user reviews.

Results. As shown in Figure 6, both Stable and TFDD groups had median ratings of 4 out of 5 stars. However, the Stable application distribution exhibits more cases of poorly rated apps. In contrast, TFDD apps showed a more concentrated distribution, indicating more consistent user ratings.

Figure 7 shows the distributions for the number of installations between Stable and TFDD groups. Interestingly, the TFDD group had a higher median number of installations than stable applications, indicating that applications that were abandoned by their developers tend to have a larger user base.

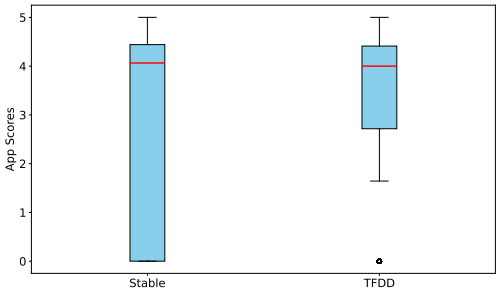

Fig. 6 App ratings for Stable and TFDD groups

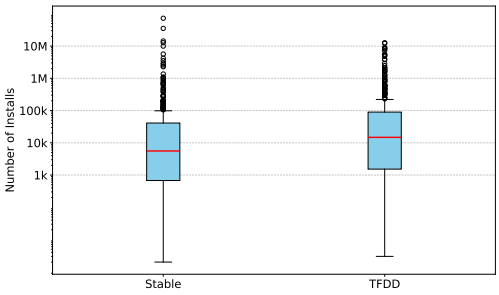

Fig. 7 Number of installs for Stable and TFDD groups

We find that mobile applications that were abandoned by their development team typically had more consistent high reviews and also a higher median number of installations than apps that never faced a TFDD event.

4.3 RQ3: How do user reviews relate to TFDD events in mobile applications?

Motivation. The sustainability of mobile applications is influenced not only by development activity but also by user feedback. User reviews in app stores directly reflect user satisfaction and dissatisfaction, and can thus serve as important indicators of an app ' s long-term viability. While prior research has primarily focused on developer-side factors (e.g., core developer turnover or contribution volume), the relationship between user reviews and app abandonment (TFDD) has not been sufficiently explored. We therefore conduct an in-depth analysis into user reviews to understand how their content relate to TFDD events.

analyze the number of reviews, trends in user sentiment toward th, and topics to examine whether user feedback provides early warning signals of app abandonment.

Approach. We first investigate how the number of reviews differ between Stable and TFDD applications. Following the methodology described in Section 3, we use OpenAI's GPT-4o mini model to analyze the content of user reviews and determine 1) the purpose of the review, and 2) the overall sentiment of the user review. Using the LLM, we classified every review from TFDD applications an collected an equal-size random subset of user reviews from Stable applications.

Results. As shown in Figure 8, TFDD apps generally had a higher median number of reviews, indicating greater user engagement. In contrast, Stable apps tended to have fewer reviews overall, with a larger proportion of applications receiving only a small number of user reviews.

Table 3 summarizes the result of the LLM-based analysis of user reviews for both Stable and TFDD apps. Overall,

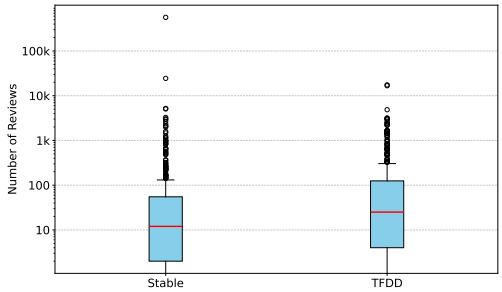

Fig. 8 Number of reviews for Stable and TFDD groups

Stable apps had a higher proportion of feedback-oriented reviews (85.0%) with user reviews requesting features or mentioning bugs/crashes which most likely correlates with the high percentage of positive comment (78.6%) we observed. Alternatively, TFDD apps showed lower user satisfaction, with only 64.3% positive reviews and over double the amount negative of negative (18.8%) when compared when Stable applications. TFDD apps also noticeably had a higher proportion of feature requests (18.5%) and bug reports (17.2%), suggesting that users of these apps frequently expressed unmet needs or technical issues.

Table 3 Distribution (in %) of review topics and sentiments over all analyzed user reviews.

Class	Feedback	Feature	Bug	Positive	Neutral	Negative
Stable	85.03	8.68	6.29	78.58	13.23	8.20
TFDD	64.36	18.46	17.18	64.32	16.86	18.82

We find that applications that get abandoned often have a higher number of user reviews. Our results also show that TFDD apps tend to have a larger proportion of negative reviews, bug reports, and feature requests.

5. Discussion

Our findings provide several new insights about developer turnover and the impact of external factors such as user feedback in the context of mobile application development. Specifically, our analyses reveal that it is not uncommon for mobile applications published on an app marketplace such as the Google Play Store to get abandoned by its developers.

Our results in RQ1 show that mobile applications have always relied on very small core development teams since the release of the Google Play Store. Our results show that, in most cases, a single core developer is responsible for continuing development and maintenance. This phenomenon is likely one of the main factor leading to the 50% app abandonment rate we observed. However, our results do seem to indicate that development tends however that although

While most people would assume that mobile applications get abandoned due to low ratings and a lack of user engagement, our results RQ2 and RQ3 results show that apps that get abandoned often have consistently higher ratings, higher number of installations, and more user engagement in the form of user reviews. These results show that even apps that are successful (relatively to other apps on the marketplace)

are at risk of getting abandoned by their developers.

The result of our qualitative analysis in RQ3 highlights the main differences which separate apps that faced abandonment and those that did not. Based on the analysis of user reviews, it is clear that a higher rate of negative reviews (which includes user reviews describing bugs) correlates to a higher chance of developers disengaging from the project. Additionally, while feature requests are not innately negative, having a large percentage of user reviews requesting additional features could be an indication that the users are not satisfied with the current state of the application.

Our study provides novel insights about mobile application sustainability. Namely, our results prove that app abandonment is a significant issue in app marketplace and that all applications (not only unsuccessful ones) are at risk of getting abandoned by their core developers.

For researchers, our analyses provide valuable new insights about factors that impact the success and death of mobile applications in large marketplace. Our results can help researchers develop better predictive models of app success and sustainability.

For users, our results show that looking at the content of recent reviews could be a good indicator of the health of the mobile application. If most recent reviews are negative, mentioning bugs, or requesting new features, it is likely that the application has either already been abandoned or is on its way to being abandoned. Users can therefore make a more informed decision whether they wish to use a specific app or explore alternatives.

For app store operators, monitoring app metadata (last update, ratings, installs) alongside review patterns can help detect at-risk apps, guide users, and support developers to maintain a healthy ecosystem.

6. Conclusion

In this study, we investigated factors that lead to mobile application abandonment, focusing on core developers abandonment and user reviews. We found that half of our studied mobile apps have faced abandonment from their core developers. Our result also reveal that highly rated and popular apps are also at risk of facing abandonment from their developers. Lastly, our findings show that mobile applications that have faced abandonment tend to have a larger proportion of feature requests, bug reports, and overall negative user reviews when compared to apps that have not faced abandonment.

This study helps improve our understanding of what factors impact the overall health of the mobile application ecosystem. Using our findings, researchers and developers developing tools to model ecosystem health or to predict the adoption/success of mobile software projects can improve their models and users can make more informed decisions for their choice of mobile applications.

References

[1] Y. Ouyang, B. Guo, T. Guo, L. Cao, and Z. Yu, "Modeling and forecasting the popularity evolution of mobile apps: A multivariate hawkes process approach," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol.2, 2018.

[2] S. Shen, X. Lu, Z. Hu, and X. Liu, "Towards release strategy optimization for apps in google play," Proceedings of the 9th Asia-Pacific Symposium on Internetware, Association for Computing Machinery, 2017.

[3] F. Bemmann and S. Mayer, "The impact of data privacy on users' smartphone app adoption decisions," Proceedings of the ACM on Human-Computer Interaction, vol.8, 2024.

[4] G. Avelino, L. Passos, A. Hora, and M.T. Valente, "A novel approach for estimating truck factors," Proceedings of the 24th IEEE International Conference on Program Comprehension (ICPC), pp.1–10, 2016.

[5] O. Nourry, M. Kondo, S. Saito, Y. Iimura, N. Ubayashi, and Y. Kamei, "Myth: The loss of core developers is a critical issue for oss communities," arXiv preprint arXiv:2412.00313, 2024.

[6] L. Williams and R. Kessler, Pair Programming Illuminated, Addison-Wesley, 2003.

[7] G. Avelino, E. Constantinou, M.T. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," International Symposium on Empirical Software Engineering and Measurement, pp.1–12, IEEE, 2019.

[8] F. Ricca and A. Marchetto, "Are heroes common in floss projects?," Proc. 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp.1–4, Association for Computing Machinery, 2010.

[9] 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.

[10] F. Calefato, M.A. Gerosa, G. Iaffaldano, F. Lanubile, and I. Steinmacher, "Will you come back to contribute? investigating the inactivity of oss core developers in github," Empirical Softw. Engg., vol.27, 2022.

[11] M. Ferreira, M. 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.

[12] E. Jabrayilzade, M. Evtikhiev, E. Tüzün, and V. Kovalenko, "Bus factor in practice," Proc. 44th International Conference on Software Engineering: Software Engineering in Practice, pp.97–106, Association for Computing Machinery, 2022.

[13] S. Sigg, E. Lagerspetz, E. Peltonen, P. Nurmi, and S. Tarkoma, "Sovereignty of the apps: There ' s more to relevance than downloads," arXiv preprint arXiv:1611.10161, 2016.

[14] AppTweak, "Impact of app store ratings and reviews on app visibility." `https://www.apptweak.com/en/aso-blog/impact-of-app-store-ratings-reviews-on-app-visibility`, 2023.

[15] D. Martens and T. Johann, "On the emotion of users in app reviews," arXiv preprint arXiv:1703.02256, 2017.

[16] Anonymous, "F-droid tabler: Web dashboard interface for f-droid repository [online]." Available: `https://fdroid.tabler.dev/`. Accessed: Oct. 18, 2025.