Tug of Perspectives: Mobile App Users vs Developers

Sandeep Kaur Kuttal [#1], Yiting Bai [#2], Ezequiel Scott [*3], Rajesh Sharma [*4]

[#] *Tandy School of Computer Science, University of Tulsa*
USA
[1] sandeep-kuttal@utulsa.edu
[2] yitingbai95@gmail.com

[*] *University of Tartu*
Estonia
[3] ezequiel.scott@ut.ee
[4] rajesh.sharma@ut.ee

***Abstract*—Billions of apps are published each year in the mobile application distribution market. However, a large number of these apps are unsuccessful due to poor user attention and satisfaction as reflected by low ratings on the distribution markets. Recent studies have investigated the app popularity from users' perspectives, but none of the studies have compared it from a developer's perspective. To fill this gap, we analyzed the user ratings and reviews with the software characteristics: ratings, issue report contents, number of bugs and enhancements, and developers' team structure. In order to do this, we examined 274 apps on the Apple App Store that also had their source code available on GitHub. We collected 19,655 app reviews and 13,193 issue reports from both platforms. Generally, app users' satisfaction and reviews on App Store did not reflect the developers' preferences and issue report contents on GitHub. Furthermore, results suggested larger team sizes and the presence of subteams of developers had a significant impact on ratings. Our findings may have implications for practitioners to improve their mobile app development practices.**

I. Introduction

Application distribution markets ("App Stores") allow for a faster dissemination of mobile app software and are growing at a fast pace. Mobile app software development is a highly profitable and competitive business [1]. Each year, billions of apps are downloaded from app stores [2]. However, a large number of these apps fail due to poor user attention (few or no downloads) [3], [4]. For example, 80% of paid Android apps received less than 100 downloads and more than 60% of Apple apps were never downloaded [3], [5]. Hence, making profits on app stores is challenging for developers.

The revenue and profit for an app is directly dependent on the user base. To sustain an app it is crucial to improve users' experiences and satisfaction [6]. One way to measure this is to utilize the ratings systems on app stores that allows users to provide their opinions regarding an app [5].

Recent research has found relationships between app ratings and factors such as change and fault proneness to adopt a project, complexity of the user interfaces, and application churn [7], [8], [9], [10], [11]. However, it is still not apparent how factors related to mobile app development like bugs and enhancements reported in issue reports, user opinions expressed in app reviews, and team structures can impact app popularity. To fill this gap, we investigated the influence of these factors on the user experience to assist struggling developers.

To analyze and understand software engineering paradigm of app development, we collected two types of information - users ratings and reviews from the Apple App Store and project related information (e.g. issue reports, developers involved) from GitHub. These provided us with a rich and interrelated set of data.

We addressed the following research questions:

- **RQ1:** ***Is there any relationship between the ratings on the App Store and GitHub?*** We studied the distributions of the ratings and the Spearman correlation coefficient to determine whether there is a correlation between the App Store and GitHub ratings. The distributions of the ratings from the App Store and GitHub does not have any correlation, suggesting that app users and developers have different perspectives regarding their satisfaction with the app.
- **RQ2:** ***Do issue reports reflect the opinion of users of the App Store?*** We used Natural Language Processing techniques, in particular, Bag of Words representation and sentiment analysis to compare the contents of the user reviews from the App Store and the issue reports on GitHub. We found that the contents of the issue reports were seldom included in the contents of the app reviews and vice versa. Similarly, sentiment values of both users in user reviews and developers in issue reports did not correlate. These results signaled that users and developers have different views and feedback regarding the apps.
- **RQ3:** ***What is the relationship between the software quality and the popularity of the apps?*** We processed the labels of the issue reports and identified common labels for all the projects (i.e. bugs and enhancements). We analyzed their distributions and the Spearman correlation coefficient values. The projects that had high GitHub ratings had a high number of bugs and enhancements. Most of the high rated apps on the App Store had a

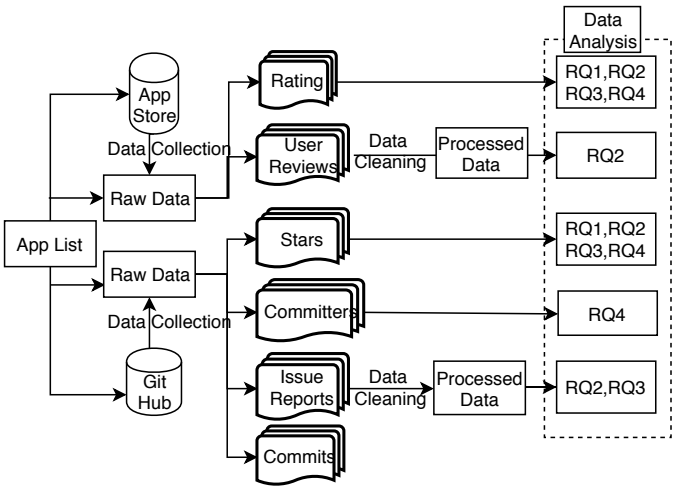

Fig. 1: Flow chart of the study evaluation process.

small number of bugs and enhancements. Thus, indicating different developer and user perspectives on the quality of the apps.

- **RQ4:** ***What is the relationship between the team structure and the popularity of the app?*** We built a social network of the developers for each app based on GitHub data and analysed the structure of the network using the app ratings on GitHub and the App Store as metrics. We found that bigger team sizes typically get better App Store ratings than medium or smaller-sized teams. Also, the more structured a team is (inferred by detecting communities) in terms of subteams, the better ratings it will attract from the GitHub developers, but not from App Store users. Therefore, team size is a significant factor in app popularity among developers and users.

The contributions of this paper are:

- We empirically verified that there are differences in the user satisfaction and developers' perception of the software quality of a mobile app. The users differed from developers in their ratings, sentiments, review contents, reaction to bugs and enhancements, and developers' team structure.
- We found that large sized teams developing mobile app software when grouped in sub-teams received better ratings by their peer developers in contrast to smaller sized teams. The well coordinated sub-teams received higher ratings on GitHub as well as App Store.

II. Data Preparation

In this section, we describe our process of data collection, preprocessing for extracting features, and the identification of variables for analyzing the data for our research questions. Figure 1 describes the steps of our methodology from data collection to data analysis.

A. Data collection

App List: We used a publicly available list of mobile iOS applications maintained by the GitHub community [12]. When we collected the data, the list contained 847 apps with information such as the project name, the URL to the Apple App Store, and the URL to the GitHub project for each app. The apps were categorized by functionality, we manually reviewed the categories in the original list and unified some of them (e.g. the category "fitness" was merged with "health"). The reviewed categories and their frequencies are reported in Table I.

TABLE I: Frequency of Mobile App Categories

Category	Frequency	Category	Frequency	Category	Frequency
Apple TV	14	Location	38	Extension	14
Apple Watch	44	Media	111	Finance	24
Browser	8	miscellaneous	160	Game	79
Calculator	6	News	26	Health	25
Calendar	2	Official	17	Keyboard	8
Clock	3	Reactive Prog.	20	Text	16
Color	3	Sample	33	Timer	6
Communication	29	Scan	2	Travel	6
Developer	56	Security	19	Weather	12
Education	11	Shopping	4	Event	15
Emulator	9	Social	13	Tasks	14

Data from the Apple App Store: We selected the Apple App Store because: (1) it is one of the most popular application distribution markets [13], (2) it was the first application distribution market (2008) that exploded the development of mobile apps [14], (3) it has strict development policies and publishing guidelines for every product/app to be approved and showcased in the market [15], and (4) it provides a Application Programming Guide with guidelines for various aspects of iPhone app development [16].

From the app list, we were able to collect 274 apps from the Apple App Store that had active links; and, out of those 274 apps, 161 contained app reviews. The app reviews were collected from the App Store using a crawler that collected the RSS feeds of each app. These RSS feeds were processed to extract the following features:

1) **App Store Ratings:** The Apple App Store allows its users to give their aggregated opinions about an app they downloaded in the form of 5-star ratings.
2) **App Reviews:** Apple App Store also allows its app users to leave detailed review comments of the app. This is the App Store's effort to allow the community of users to provide feedback on whether the apps are reliable, perform as expected, and the experiences users have with the apps.

Data from GitHub: GitHub is arguably the largest collaborative software development platform, hosting 100 million repositories[17]. Some activities on GitHub include forking projects (creating a copy), reporting issues, contributing code, and global collaboration [18]. From the app list, we were able to retrieve 808 apps with active links on GitHub; the remaining projects no longer existed, because they were deleted by the authors. To collect data from GitHub, we used a web crawler to scrape the following features of each project:

1) **GitHub Ratings:** GitHub uses a unary (like-based) rating system. Each project on GitHub may be given a star by developers as a medium of expression to show their liking for the project. The number of stars represents the

number of developers that liked the project, and a high number of stars signals popularity of the project with developers.

2) **Commits:** A commit is an individual change to a file or a set of files and allows committers to leave comments. Every change on a Git repository creates an unique ID (SHA) that allows to keep a log of the changes with information of the files changed, the changes made, the comments, the time, and the user who made that change (committer).
3) **Committers:** GitHub users that have commit access to the project repository are known as contributors. Contributors collaborate in the project and are allowed to make changes to the project but they can be removed from it by the project owners. To have more realistic measurements about the development teams, we extracted the name of the users who made commits to the project (a.k.a. committers). We consider the use of committers more appropriate since the commit log of a project is permanently saved with the project, along with the commit comments and other GitHub activities. We consider number of committers/developers/contributors as equivalent to the team size of a project.
4) **Issue Reports:** Issue reports contain information on bugs, enhancements, or other requests from developers. In our data, we collected the title and body of the issue reports. In addition, we collected the labels associated to the issue reports such as *bug* and *enhancement*.

We wrote the web crawlers for both Apple App Store and GitHub using Python's `beautifulsoup` library [19].

B. Preprocessing Data

The raw data related to user reviews collected from the App Store and the issue reports from GitHub was preprocessed using the following steps:

Extracting content: We extracted the title and the body from app reviews and issue reports.

Changing to lowercase: For consistency, we converted all upper cases to lower cases.

Removing non-words: For some reviews, we removed numbers and punctuation symbols, as they do not add emotional value to the content and make the interpretation of frequencies inaccurate.

Removing stop words: We removed popular stop words such as *the, a, in*, as these do not provide context.

Removing URLs: We removed URLs from the content since they cannot be interpreted.

From the 161 Apple (iOS) apps that had reviews, we collected 22,671 users' app reviews. We extracted 133324 words from app reviews and after preprocessing, 126394 words remained for the final analysis.

Out of 808 projects, only 493 projects contained issue reports on GitHub. From those projects, we were able to collect a total of 19,605 issue reports, that had a total of 2,73,371 words. After preprocessing, 214196 words remained for the final analysis. We collected the data over a time period of around 3 months and we made the dataset and the replication package publicly avaiblable[1].

We used the library `tm` in R for preprocessing the raw data.

TABLE II: Descriptive statistics about the variables for each research question.

RQs	Term	Std	Med	Avg	Max	Min
RQ1,	GitHubRatings	1480.49	76.50	523.80	20169.00	0.00
RQ3,	AppStoreRatings	0.79	3.93	3.85	5.00	1.00
RQ4:	NumOfContributors	27.08	2.00	7.70	288.00	0.00
	NumOfCommitters	62.69	2.00	11.96	953.00	0.00
RQ2:	IssueReports	1243.47	73.00	434.47	13300.00	1.00
	AppReviews	862.43	369.00	785.06	3170.00	9.00
RQ3:	IssueLabel(bug)	16.75	0.00	2.54	280.00	0.00
	IssueLabel(enhancement)	11.04	0.00	2.86	133.00	0.00
RQ4:	Disconnected_components.	0.10	1.00	1.01	2.00	1.00
	Modularity.	0.06	0.00	0.03	0.48	0.00
	NumCommunities	1.15	1.00	1.61	11.00	1.00
	AvgOfCommonRepositories	17.63	5.00	11.04	181.00	1.00
	AvgOfDensity	0.18	0.92	0.84	1.00	0.24
	AvgOfNumOfConnections	301.05	3.00	48.73	3412.20	1.00
	AvgOfSize	13.83	3.00	6.20	157.67	2.00
	MedianOfCommonRepositories	17.30	4.33	10.19	181.00	1.00
	MedianOfDensity	0.19	1.00	0.84	1.00	0.20
	MedianOfNumOfConnections	226.93	3.00	35.49	3073.50	1.00
	MedianOfSize	12.45	3.00	5.87	157.00	2.00

C. Measuring Variables

After preprocessing the data, we identified variables from the Apple App Store and GitHub that were necessary to answer each research question. Table II shows the descriptive statistics for the variables used in our study.

III. Approach and Results

In this section, we present the motivation, approach, and findings for each research question.

A. RQ1: App Store vs. GitHub Ratings

Motivation. The apps on the app stores and projects on GitHub use a rating system to show the popularity of an app or a project. Apps on app stores use a 5-star ratings while projects on GitHub use a unary rating system. We conjectured that apps which have high ratings on the app store and are popular with users may have high ratings on GitHub from developers as well. Hence, we formulated ***RQ1: Is there any relationship between the ratings on the App Store and GitHub?***

Approach. We studied the popularity metrics of both communities: the 5-star ratings provided by the App Store and the number of stars collected from GitHub. We chose the ratings rather than reviews, as they represents an average for the whole app and combines both positive and negative evaluations of a feature [20]. The Spearman correlation [21] coefficient was used to determine whether there is a correlation between the App Store and GitHub ratings.

Findings. We selected 149 apps with ratings that appeared on both the App Store and GitHub. We investigated RQ1 using the Spearman correlation coefficient r between the ratings on App Store and GitHub but found no significant relationship between both variables ($r = -.111; p = .176$). To further understand these results, we did an in-depth analysis and found:

[1]https://drive.google.com/open?id=1ZvXjZihdVxXf-zuDkpGwdUN-IdnrRyJE

Observation.1a: ***More apps were popular with App Store users in comparison to GitHub developers.*** Figure 2 (scatter plot), shows that most of the apps that had a high rating on the App Store had relatively low ratings on GitHub. Figure 3 illustrates the distributions of the ratings found on the App Store (left) and GitHub (right). 83% of the apps on the App Store had a rating $>$ 3 stars. In contrast, 13% of the total number of projects on GitHub had ratings $>$ 2000 stars (representing the app being liked by 2,000 developers). These result indicate a gap in users' and developers' perception of the quality of an app.

Observation.1b: ***Apps for developers or projects that had large teams were popular on both the App Store and GitHub*** For example, three apps - CodeHub, Dash, and Kodi Remote- had ratings $>$ 4 on App Store and $>$ 5,000 on GitHub (refer Figure 2). CodeHub [22] is the best way to browse and maintain GitHub repositories, and Dash [23] allows instant offline access to 200+ API documentations. Both these developer related apps are designed for hand-held devices like iPhone, iPod Touch, and iPad. In particular, Kodi is an home theater/media center software and entertainment hub for digital media [24] which consisted of a large team of 623 contributors, 211 issue reports, and 52,228 commits. These observations align with past research that suggest that developers create better software systems when they understand the customers' requirements and work in large teams [25], [26].

Observation.1c: ***General purpose apps were not popular with developers, and user preferences for them depended on the apps' category*** For example, seven apps with ratings $>$ 4 on App Store and $<$ 10 on GitHub were found: Spin Zone (game), Monotone Delay (audio), Northern California Cherry Blossom Festival (official), Conjugar (education), Phone Battery (Apple Watch), Be my eyes (communication) and SnowHaze (game). On analyzing these seven apps, one project (SnowHaze) had 2 open_issues; only one project (Be my eyes) had 7 contributors and the rest had just one. Analyzing user reviews by category, users who rated high for apps in the entertainment category (games, audio) mentioned they appreciated if the game was fun and addicting; where as in education and official categories, users mentioned they wanted apps that were well organized, had clean User Interfaces, and were more efficient. Be my eyes, was a special app, that was highly popular with users for supporting the visually impaired community. Users left positive feedback for its contribution to help both blind people and society at large. Hence, these results verify the differences in users' and developers' perception of a good app.

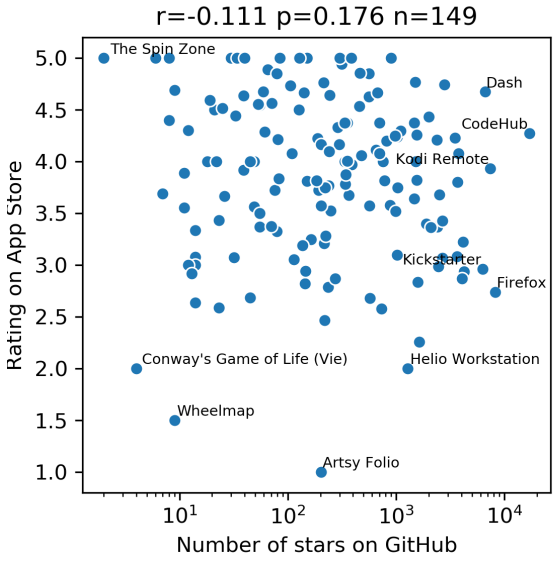

Fig. 2: Relationship between the ratings of the App Store and the number of stars on GitHub.

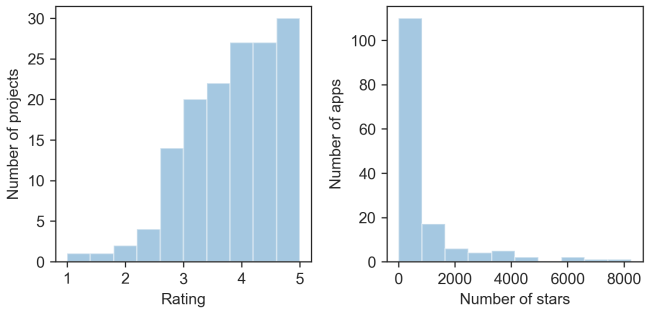

Fig. 3: Histograms of the ratings on App Store (left side) and stars on GitHub (right side).

B. App Reviews vs. Issue Reports

Motivation. Based on the results of RQ1, we wanted to understand whether the users' reviews/feedback on the App Store and developers' reported issues on GitHub are similar. We utilized user reviews/feedback on the App Store and Issue reports from GitHub. Recent empirical studies suggest that App Store reviews can be effectively utilized by analysts and designers for collecting information related to user requirements, bug reports, feature requests, and documentation of user experiences with specific app features [27], [28]. On the other hand, issue reports on GitHub contains information regarding bugs, enhancements, or other requests by developers. Based on findings from a survey by Al-Subaihin et al. [29], 51% of app developers performed perfective maintenance based on feedback from the app reviews. Hence, we expected app reviews to inform about possible bugs and enhancement requests. We formulated ***RQ2: Do issue reports reflect the opinion of users of the App Store?***

Approach. In order to address this question, we analyzed the contents of the app reviews from the App Store and the issue reports from GitHub for each app/project. The following techniques were used:

Bag of Words: The analysis was done using a Bag of Words (BoW) representation of the contents of feedback from the users of both platforms (i.e. users on the App Store and developers on GitHub). We used the preprocessed data as de-

tailed in Section II to create two corpora for each app/project: the first corpus consisted of all the app reviews for a given app whereas the second corpus consisted of all the issue reports of the corresponding project. Then, we used a BoW representation for each corpus (sets A and B, respectively) to determine whether A and B are similar or not. We applied two similarity functions, the Jaccard coefficient (Eq. 1) and the cosine similarity (Eq. 2). The former measures the fraction of words that the two sets have in common, whereas the later measures the similarity between the two sets but taking the word count into account. It should be noted that the cosine similarity requires to convert the sets of words A and B into two document vectors $\vec{a}$ and $\vec{b}$, respectively.

$$J_{A,B} = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

$$\alpha_{A,B} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2}\sqrt{\sum b_i^2}} \tag{2}$$

In addition, we calculated the percentage of inclusion ($IR \subseteq AR$) (eq. 3) to measure how many similar words were in both the issue reports (IR) and the app reviews (AR).

$$I_{AR,IR} = \frac{|AR \cap IR|}{|IR|} \tag{3}$$

Sentiment Analysis: The sentiment analysis was done by classifying each corpus as positive or negative depending on the number of positive and negative words. To do this, we used the `Syuzhet` package[2] available in R to calculate the number of positive (Pos) and negative (Neg) coded sentiment words and the number of all other words (O) on each corpus, separately. Then, we determined the overall sentiment (S) of the corpus by its relative proportional difference (Equation 4) [30]. As a result, the value of S is in the range $[-1, 1]$, where negative values refer to an overall negative sentiment, and positive values to an overall positive sentiment. Using this equation, we compared sentiments of the contents in issue reports S_{IR} and app reviews S_{AR}.

$$S = \frac{Pos - Neg}{Pos + Neg} \tag{4}$$

Findings. From a list of 808 apps, we were able to collect 22,671 app reviews from 161 apps and 19,605 issue reports from 493 projects. We found 125 apps/projects with both app reviews and issue reports. On analyzing these 125 apps/projects, we found:

Observation.2a: ***Few app reviews contained the same words in the issue reports.*** The percentage of inclusion was lower than 20% for most of the apps. Only 15 out of 125 apps (12%) had more than the 20% of the words of the issue reports included in the app reviews (refer Figure 4). This was also supported by the low cosine similarity and Jaccard index values: 21 out of 125 (16.80%) apps had more than 0.5 of cosine similarity and Jaccard index values.

[2]https://cran.r-project.org/package=syuzhet

Out of those 21 apps, 5 apps had Jaccard index values >0.7 and app store ratings >4. We found only one app with Jaccard coefficient >0.7 and ratings of 2.69 on the App Store. This app was "MyHeartCounts" (Health), a research kit. Inspecting the app reviews, we found that recent users reported problems mostly related to inaccurate results and login issues. This might indicate that developers may not be aware of the contents of the user reviews, which may impact the quality of the apps and lead to lower ratings [27], [28].

These results aligns with RQ1 that user and developers have different perspectives and priorities.

Observation.2b:***The sentiment values of users and developers for an app/project did not correlate.*** Figure 5 (right) shows the relationship between the relative proportional difference of sentiment of app reviews (S_{AR}) and issue reports (S_{IS}). There is no significant correlation between S_{AR} and S_{IS} ($r = 0.139, p = 0.123$). Figure 5 shows the results from the sentiment analysis. These results suggests that developers' and users' sentiments values were different.

Observation.2c: ***Sentiments between app reviews and issue reports were in contrast.*** On further analysis of the sentiments of the app reviews and issue reports, we found 6 apps that had negative sentiment $S_{AR} < 0$ or $S_{IS} < 0$ but had opposite values of S_{AR} and S_{IS}. There were three apps with $S_{AR} < -0.3$ and $S_{IS} > 0.4$, and other three apps have $S_{AR} > 0.4$ but $S_{IS} < 0$. For instance, the app "Battle for Wesnoth" (Game) showed opposite sentiment values in the content of the issue reports and app reviews ($S_{IS} = -0.333; S_{AR} = 0.524$). The word cloud of app reviews depicted in Figure 6 shows that users on app store liked the app, since the most frequent words were "great", "love", and "good". On the other hand, Figure 7 there were frequent negative words like "missing" and "wrong" in the issue reports.

Observation.2d: ***Negative user reviews may contain rich bug and enhancement information.*** On further analysis, we found 3 apps - Artsy Folio (Media), Wheelmap (Location), and Helio Workstation (Media) - that scored less than 2 stars. The Wheelmap issue report was not written in English. For Helio workstation, the users mentioned in the reviews that the app was difficult to use; this matched with issue reports, which asked for more features and tips. All users rated Artsy Folio as 1 and mentioned that the app was useless to them, as it had a bad explanation for the targeted users. These results may suggest that negative ratings may help in finding bug and enhancement labels for projects.

C. Software Quality vs. App Success

Motivation. As noted before, issue reports are often labelled by different categories such as bug, enhancement, and question, among others. These labels are maintained and verified by the community and can be used as indicators of the quality of the software. Thus, we quantified the number of bugs and enhancements reported by the developers. We wanted to understand if there was a relationship between the quality of the projects and their popularity. Hence, we formulated

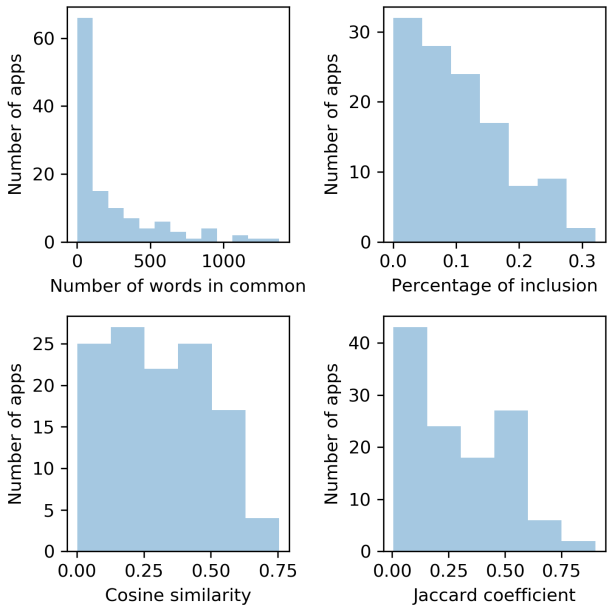

Fig. 4: Distribution of the similarity measurements obtained from comparing the contents of the app reviews and the issue reports.

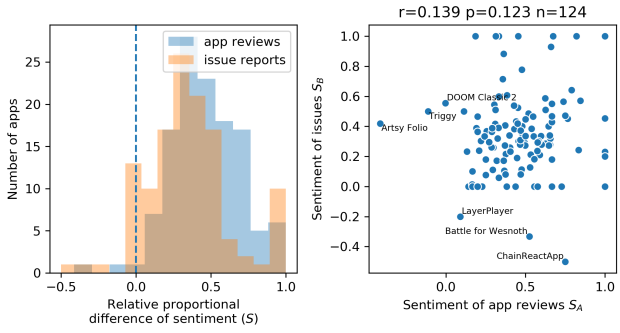

Fig. 5: Distribution of the relative proportional difference of sentiment (S) for both app reviews and issue reports.

Fig. 6: Word cloud representing the most frequent words in the contents of the app reviews of ”Battle for Wesnoth”

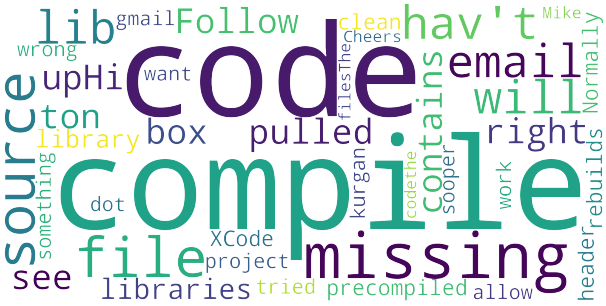

Fig. 7: Word cloud showing the most frequent words in the contents of the issue reviews of the app ”Battle for Wesnoth”

RQ3: What is the relationship between the software quality and the popularity of the apps? Previous studies have shown that there is no correlation between the quality of the source code of mobile applications and its success [31]. Other studies have pointed out that the number of bugs impacts users' decisions about uninstalling the app. [32], [33]. The purpose of this question is to determine whether indicators of the software quality of the app such as the number of bugs and enhancements reported on GitHub impact on the popularity of the app.

Approach. We collected issue reports from GitHub and identified common labels for all the projects (i.e. bugs and enhancements). These labels were used to count the number of reported bugs and enhancements on each app/project. In order to determine their relationship with the ratings on the App Store and GitHub, we analyzed their distributions and the Spearman correlation coefficient values. We also reported outliers to complement the analysis. In these cases, we considered high-rated apps as those with more than 4 stars on the App Store; acceptable apps as those between 4 and 2 stars, and low-rated apps with a rating lower than 2 stars.

Findings. We found that:

Observation.3a: ***Popular apps/projects on GitHub may have more bugs and enhancements, giving developers opportunities to improve their skills.*** We found marginally significant correlation between the stars on GitHub vs number of bugs ($r = .254; p = .008; n = 107$) vs enhancements ($r = .234; p = .007; n = 134$). (Figure 8 upper-left): there were three apps with a high number of bugs (more than 50 bugs) that have also a high number of ratings (more than 100) on GitHub. On GitHub (Figure 8 upper-right), the ratings were diverse ranging from (1 to 10^4 stars) in apps that have a low number of enhancements (less than 20 enhancements).

Figure 9 shows apps with high ratings on GitHub (more than 10^3), often have several bugs and enhancements. For the rest of the apps, the number of GitHub ratings increased as more bugs and enhancements were reported. We conjecture that more popular apps on GitHub may have more developers involved; mobile software development and testing is challenging because of the need to test and develop across heterogeneous devices, operating systems, and networks [34]. Hence, more bugs/enhancements gives developers opportunities as reflected

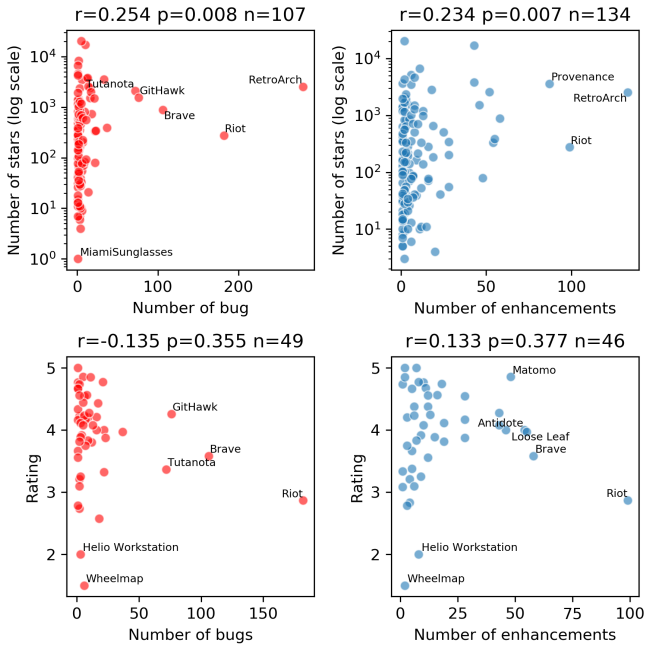

Fig. 8: Distribution of the number of bugs (right) and enhancements (left) with regard to GitHub ratings (upper) and App Store ratings (lower). Spearman correlation coefficient value (r), the statistical significance (p), and the number of apps considered (n). In all the cases, zero values were discarded from the analysis (e.g. apps with zero bugs reported or zero number of ratings).

in ratings.

Observation.3b: ***Popular apps on App Store have less number of bugs and enhancements.*** We found no significant relationship between the ratings on the App Store with number of bugs ($r = .135; p = .355; n = 49$) and number of enhancements ($r = .133; p = .377; n = 46$). When comparing the number of bugs and the rating found in the App Store (Figure 8 lower-left), only only three apps were found with a high number of bugs (more than 50 bugs) and an acceptable rating on the App Store (higher than 3). A few high-rated apps (5 apps) had several enhancements reported (more than 50 enhancements). All these consisted of small groups of team members, except one (Evolution (developer)) that had 16 team members.

Figure 9 shows distribution of both popularity indicators (i.e., ratings on GitHub and App Store) and the number of bugs. Figure 9, it is clear that the high rated apps on App Store (more than 4.5) did not contain bugs or enhancements. Out of 13 five stars projects, only one project, ChainReactApp (Event), reported 1 bug and 0 enhancements; two projects, Evolution (Developer) and Blahker (Extension), both reported 0 bugs and 2 and 7 enhancements, respectively; the remaining 10 projects had 0 enhancements and 0 bugs.

D. Team structure vs. App Success

Motivation. Studies have shown that the characteristics of the team and the people involved in the development play

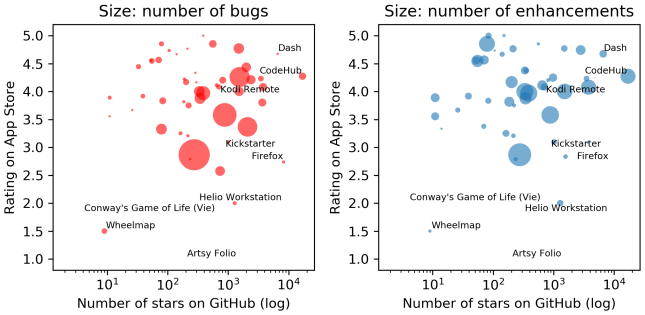

Fig. 9: Distribution of the number of bugs (left side) and enhancements (right side) with regard to the number of stars and rating of the apps.

a critical role in the success of the project [26], [35], [36]. It has been reported that apps are developed by small teams [37], [38]. We wanted to investigate if team size and the interactions among team members have an effect on the popularity of the app. We formulated ***RQ4: What is the relationship between the team structure and the popularity of the app?***

In mobile software engineering, "scaling up" of a project needs to be addressed by finding appropriate techniques to manage complexity[34]. As teams of developers need mechanisms for coordination and reporting, we believe that thorough understanding of developers' behaviors on GitHub will yield new ways to increase collaboration among mobile app developers. Hence, this will help researchers and practitioners to gain more insights needed to design better tools for supporting struggling developers trying to create successful apps.

Approach. To understand the team structure, we first created a team members' network that was modeled as a graph $G(N, E)$. N represents the set of all the developers under consideration in our dataset. E denotes the set of all the connections that are present among various developers. Two developers are connected if they have worked on a project together. We calculated the following three metrics to understand the team structure of the developers:

- Number of disconnected components: It is to be noted that our definition of edge creation assumes that if there is a commit by any two team members for the same file, then they have communicated with each other and that is represented by the edges in our networks. However, the network of a particular project might not be fully connected as some team members may not have communicated with each other. For example, a backend developer may not commit in the same file as the technical writer of a project. We captured this disconnectivity by calculating the number of disconnected components.
- Number of communities: Within a connected component, it is possible that some set of team members communicate with each other more than other sets of team members. In a network, a community C can be defined as a set of nodes where $C \subseteq G$, such that the nodes $\in C$ are densely

connected with each other compared to rest of the nodes. To detect the communities, we used implementation of Louvain algorithm [39] using networkX library in Python.

- Modularity: First proposed by Newman and Girvan [40], the "modularity" metric helps in understanding how tightly knit the nodes are with each other in a network. A developer network with high modularity has dense connections between the nodes, indicating a closely connected social community (i.e. a highly connected team structure). In such a tightly knit community, the rate of transmission of information is reliable and faster when compared to a community with low modularity. Good communication is a key factor that influences the success of a team. [41].

The number of stars on GitHub is used as an indicator of the success of the project from the developers' perspective, whereas ratings on the App Store indicate success from the user's perspective.

Findings.

To determine the effect size of the relationships found, we interpret the Spearman r coefficient values by following Cohen's guidelines [42], [43]: small, medium, and large effect sizes are respectively 0.1, 0.3, and 0.5. Table III shows the Spearman correlations coefficients r for the network structure metrics and the popularity indicators (i.e. number of stars on GitHub and ratings on the App Store). The results are grouped by team size; n represents the number of apps analyzed, and p the p-value was to test for non-correlation.

We analyzed 534 out of 847 (63.05%) projects that had teams of more than one committer. Out of those 534 projects, we kept 482 projects with teams of less than 23 committers, since they accounted for 90% of the data. To simplify the analysis, we classified each app as small (S), medium (M), and large (L) team sizes. We found 248 small-sized projects (between 1 and 3 developers), 155 medium-size projects (between 4 and 8 developers), and 79 large-size projects (more than 9 developers). The number of developers for each project size were set by analyzing the distribution of the number of committers.

Observation.4a: ***The number of developers in a team project influences the GitHub popularity, but not App Store popularity*** The number of stars on GitHub showed a positive correlation with the number of committers who participated on the development of the app ($r = 0.514; p = 0; n = 778$). The effect size of this relationship is large, and it is consistent with previous studies [41]. This indicates that larger team sized projects attract more visibility in the developer community and, subsequently, attract high ratings. On the other hand, we found no significant results ($r = -0.071; p = 0.386$) indicating correlation between popularity on the App Store and the number of committers. We can infer that the developers have an inclination towards liking projects with large team sizes, as, from a developer's perspective, these teams have more focused responsibilities and better coordination [41].

Observation.4b: ***An increasing number of common projects between developers led to decreasing ratings on the App Store and GitHub.*** Regardless of the team size, the popularity on the App Store decreases as the average number of common repositories increases ($r = -0.24; p = 0.007; n = 124$), but when taking the team sizes into account, the effect size seems to be significant for small-size ($r = -0.459; p = 0.010; n = 31$) and large-size teams ($r = -0.441; p = 0.028; n = 25$). It is worth mentioning that the small sample of apps makes these results statistically insignificant, since, according to Cohen [42], a sample size of 617 is required to support a small size effect at $\alpha = .10$. The popularity on GitHub was significant for the large-size ($r = -0.326; p = 0.006; n = 69$), medium-size ($r = -0.321; p = 0; n = 148$) and small-size ($r = -0.146; p = 0.030; n = 221$) teams. This inverse correlation between common repositories can signal high homophily among the team members on GitHub in terms of the topics of projects they are working on. This also means the team members have limited visibility, which transitively attracts low ratings from the GitHub community.

TABLE III: List of Spearman correlation coefficient values r for the network structure metrics and the popularity indicators. Only significant results are shown. The results are grouped by team size (S: small, M: medium, L: large, A: all sizes), n represents the number of apps analyzed, and p the p-value to test for non-correlation.

Size	Metric	Popularity	n	r	p
A	NumOfCommiters	stars	778	0.514	0.000
A	Modularity	stars	168	0.317	0.000
A	AvgOfCommonRepositories	rating	124	-0.240	0.007
L	NumOfCommunities	stars	69	0.331	0.006
L	NumOfCommiters	stars	74	0.312	0.007
L	Modularity	stars	61	0.227	0.079
L	AvgOfCommonRepositories	stars	69	-0.326	0.006
L	AvgOfCommonRepositories	rating	25	-0.441	0.028
M	Modularity	rating	14	0.666	0.009
M	Modularity	stars	56	0.314	0.019
M	NumOfCommunities	stars	149	0.240	0.003
M	NumOfCommiters	stars	152	0.212	0.009
M	AvgOfCommonRepositories	stars	148	-0.321	0.000
S	NumOfCommiters	stars	239	0.160	0.013
S	AvgOfCommonRepositories	stars	221	-0.146	0.030
S	AvgOfCommonRepositories	rating	31	-0.459	0.010

Observation.4c: ***The popularity on GitHub of apps/projects developed by medium and large-size teams increases as the number of subteams within these teams increases.*** We found a significant relationship between the popularity on GitHub and the number of communities in projects that were developed by teams of medium-size ($r = 0.24; p = 0.003; n = 149$) and large-size ($r = 0.331; p = 0.005; n = 69$). When this relationship was studied without taking team size into account, a large effect size ($r = 0.5; p = 0; n = 752$) was found. However, a small effect size was found when analyzing the relationship between the number of communities and the popularity on the App Store ($r = -.127; p = .124; n = 147$). As mentioned before, we cannot consider these values highly significant, as the number of apps analyzed is not big enough

[42]. It can be inferred that higher number of communities within medium or large-size teams was an indication of well-structured subteams or subgroups that were responsible for the development of particular components. Further, this indicated that those subteams were more focused and could have lead to more successful projects [41]. In addition, having well defined components in a software project is considered a good practice [44], and this fact might explain the popularity among the developers.

Observation.4d: ***Coordinated subteams achieved higher ratings on the App Store and GitHub.*** Regardless of the team size, the modularity values calculated from the communities in the developers' network showed a significant correlation of medium effect size with the number of stars ($r = 0.317; p = 0; n = 168$). In addition, a significant relationship between the modularity values and the ratings on the App Store was also present for medium-size teams ($r = 0.666; p = 0.009; n = 14$). Tightly knit project teams often attract better ratings from the app users. Although, according to Cohen [42], a sample size of 22 is required to support a large-sized significant test at $\alpha = .10$. Higher modularity means there is good coordination among the team members, which can result in a high quality app that gets highly rated by users and developers.

IV. Implications for Tools

Bridging the gap between the mobile app developers and users can help in creating a more transparent mobile market place. Our findings have implications for future software tools for developers.

Context based categorising using app reviews and issue reports: We can utilize both issue reports and user review to generate a context based words/features (*Observation.1b*, *Observation.1c*) using topic modelling techniques to generate features [45]. Thus, both sources of information may be utilized in a complementary way while analyzing the feature (*Observation.2*). Such tools can be utilized by developers to elicit requirements from user reviews as they contain "requirements for the masses; requirements from the masses" allowing a user participatory cyclic development model [27] (*Observation.1b*). However, Chen et al.[46] found that more than 60% of user reviews did not contain any useful opinions[46]. To effectively detect emerging issues, previous research has established dictionaries for preprocessing reviews, filtering out non-informative reviews, or classifying reviews to predefined topics [47], [48], [6], [49]. Although, apps made for developers were rated higher on GitHub, as developers are able to elicit the requirements for those apps better, providing better customers' perspective (*Observation.1b*).

Category-based recommendations: Reviews and feature requests can be influenced by the categorization of the app (*Observation.1c*). An effective categorisation may facilitate better application discovery and more exposure to newly emerging apps [50]. Apps reviewed based on categories led to positive feedback. For example, for games, users expected apps to be fun and interactive, while education or office-related apps are expected to be well organized and efficient with a good user interface (*Observation.1c*). Clustering the apps based on the categories may be utilized by Al-Subaihin et al. [50].

Automated bug and enhancement labels generation from negative user reviews: Negative sentiments of user reviews can help in generating automated bug labels and enhancement labels for issue reports (*Observation.2d*). This will help in reducing the sample set of user reviews to just reviews with lower rating and negative reviews. For the rest of the apps, we can see that the number of GitHub ratings increased as more enhancements were reported (*Observation.3a*). The relationship between GitHub ratings and enhancements can be explained by the app popularity. We conjecture that more popular apps may have more developers involved, hence, more enhancements are reported, as these developers may get opportunities to improve the app. The weak linear (*Observation.3*) relationship might be due to popularity. More popular apps on GitHub have more reported issues (both enhancement and bugs). The results confirmed that users prefer apps with fewer bugs, while these same users do not trend to report enhancements. Although, automatic identification of sentiments from user reviews is still beyond the reach of state-of-the-art natural language processing tools [6].

Automated notifications generation: Any feature or bug that is resolved in issue reports can be used to generate automated notifications for the corresponding apps on the App Store (*Observation.2a*).

Socially informed collaborative system: Utilizing the social network analysis to notify the developers regarding the factors observed in (*Observation.4*) to automatically incite more collaborations and coordination among mobile app developers.

V. Threats to Validity

Like any other empirical study there are several threats to validity.

External validity We studied just one application distribution market, the App Store, so the results can not be generalized. Though, this choice was intentional as previous research has found mobile apps tend to depend highly on these platform-specific APIs [51]. In addition, the relatively small data set of apps/projects and their open source categorization do not represent all kinds of mobile applications, particularly for those developed in commercial settings. Thus, further research is needed to improve external validity.

Internal validity Our results could be influenced by the evaluation metrics. Metrics, such as number of downloads and user reviews [52], were not used, as the downloads do not necessarily reflect usage, and user reviews vary widely in quantity and quality [20]. Further, it is difficult to determine the ratio of positive and negative feedback for specific features via user reviews [20]. There are several ways to build the developers' network and measure metrics to understand team dynamics, but we focused only on one network structure with several structural metrics. More studies need to be conducted using different network structures and metrics.

Conclusion validity We used a data-driven approach (Section II) to collect the apps, which may have affected our findings. We limited the threat by augmenting our findings with the description of outliers and particular cases and referring to qualitative studies that support our findings.

VI. Related Work

In recent years, an increased amount of research on user-reviews has been reported, exhibiting how they can be useful for software engineering activities. For example, Al-Subaihin et al. [29] have investigated, from a developers' perspective, to what extent app stores affect software engineering tasks. In a survey, Martin et al. [53] have identified user-review analysis as one of the key fields of App Store Analysis.

User-review analysis is primarily intended to provide insights regarding the users' experience and needs. For instance, several studies have tried to classify user-reviews into groups such as bug reports and feature requests [54], [55], [56], [57] whereas others have tried to automatically summarize the app reviews [56]. The performance of different machine learning algorithms for text classification has been studied. For instance, Faiz et al. [54] compared the performance of classifiers based on the Bag-of-Words approach with more advanced models based on Convolutional Neural Networks (CNN).

Previous studies have also explored the relationships between user-reviews and the App Store's ecosystem. For example, several authors have studied how user-reviews are related to the source code of the application. Based on the issues described in the user-review, Ciurumelea et al.[58] recommends the source code files that should be modified to address the issue. Palomba et al. [59] propose a solution for tracing informative crowd reviews into source code changes, and McDonnell et al. [60] studied the API updates of different apps using source code from GitHub. Syer et al. [51] studied the different code practices between app stores.

Many studies are focused on explaining the reasons why an app might be rated higher or more frequently downloaded. Zhong and Michahelles [61] analyzed the number of app downloads and ratings from Google Play. They found that a small number of popular apps have the majority of app downloads. Petsas et al. [62] also found that popularity follows a power-law distribution against app price, for paid apps. Corral and Fronza [31] studied open source apps that are available on the Google Play store. They analyzed the relationship between source code quality metrics and the number of downloads, number of reviewers, and average rating. As a result, the authors did not find a correlation between source code quality and app ratings.

The effects of network structure on performance has been widely studied in general [63], [64], [26], [35]. In a well connected network, it has been observed that i) communication is much better among the team members [65], ii) there is high efficacy [66], iii) the goal is very clear [67], iv) performance of such teams is much better [68].

We added insights to the software engineering literature regarding the correlation of App Store and GitHub popularity (ratings), explored the relationship between user opinion (reviews) on app store with issue reports on GitHub, analyzed the relationship between the software quality and app popularity, and investigated the impact of team structure (of developers) and popularity on mobile apps.

VII. Conclusion

In this paper, we studied the effect of GitHub ratings, issue reports, and team structure on app ratings and reviews. We observed that there is no correlation between the ratings on the App Store and GitHub. Furthermore, most of the apps that were rated high on the App Store were rated low on GitHub. The contents of the user reviews and issue reports were not similar, implying that developers do not use user reviews as requirements to improve their apps/projects. The apps/projects with high number of bugs and enhancements were rated high on GitHub but low on the App Store. Therefore, the apps that get high attention from developers may have lower user satisfaction. The social network analysis concluded that teams of large size attract high ratings from developers but not from users. Furthermore, the GitHub ratings of the apps/projects developed by large size teams increased as the number of subteams within these teams increased, and coordinated subteams achieved higher ratings on the App Store, too. Finally, we discussed the implications for the researchers and practitioners to create a more transparent mobile market by developing tools that bridge the gap between the mobile app developers and users.

References

[1] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telemat. Inf.*, vol. 28, no. 1, pp. 22–31, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.tele.2010.05.006

[2] "Statistics," Retrieved Feb-2020, https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[3] S. L. Lim, P. J. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, "Investigating country differences in mobile app user behavior and challenges for software engineering," *IEEE Transactions on Software Engineering*, vol. 41, no. 1, pp. 40–64, Jan 2015.

[4] "A. p. d. survey," Retrieved Feb-2020, http://app-promo.com/wp-content/uploads/2013/06/SlowSteady-AppPromo-WhitePaper2013.pdf.

[5] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer rating reactions can be predicted purely using app features," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Aug 2018, pp. 76–87.

[6] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2015, pp. 749–759.

[7] M. D. Syer, M. Nagappan, B. Adams, and A. E. Hassan, "Studying the relationship between source code quality and mobile platform dependence," *Software Quality Journal*, vol. 23, no. 3, pp. 485–508, 2015.

[8] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, June 2012, pp. 108–111.

[9] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "Api change and fault proneness: A threat to the success of android apps," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 477–487. [Online]. Available: http://doi.acm.org/10.1145/2491411.2491428

[10] L. Guerrouj, S. Azad, and P. C. Rigby, "The influence of app churn on app success and stackoverflow discussions," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, March 2015, pp. 321–330.

[11] G. Bavota, M. Linares-Vásquez, C. E. Bernal-Cárdenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk, "The impact of api change- and fault-proneness on the user ratings of android apps," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, April 2015.

[12] "List of apps on apple store," Retrieved Aug-2018, https://github.com/dkhamsing/open-source-ios-apps.

[13] K. W. Tracy, "Mobile application development experiences on apple's ios and android os," *IEEE Potentials*, vol. 31, no. 4, pp. 30–34, July 2012.

[14] V. N. Inukollu, D. D. Keshamoni, T. Kang, and M. Inukollu, "Factors influencing quality of mobile apps: Role of mobile app development life cycle," *CoRR*, vol. abs/1410.4537, 2014.

[15] L. Corral, A. Sillitti, and G. Succi, "Defining relevant software quality characteristics from publishing policies of mobile app stores," 08 2014, pp. 205–217.

[16] "Programming guide from iphone apps," https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html, Retrieved Feb-2020.

[17] "Github repos," Retrieved Feb-2019, https://github.com/about.

[18] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang, "Network structure of social coding in github," in *Software maintenance and reengineering (csmr), 2013 17th european conference on*. IEEE, 2013, pp. 323–326.

[19] "beautifulsoup4 python library," Retrieved Feb-2019, https://pypi.org/project/beautifulsoup4/.

[20] 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 *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2015, pp. 301–310.

[21] G. L. Shevlyakov and H. Oja, *Robust Correlation: Theory and Applications*. John Wiley & Sons, 2016.

[22] "Code hub," Retrieved Feb-2020, https://www.codehub.gr/.

[23] "Dash," Retrieved Feb-2020, https://www.dash.org/.

[24] "Kodi," Retrieved Feb-2020, https://kodi.tv/.

[25] C. Courage and K. Baxter, *Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[26] J. S. Reel, "Critical success factors in software projects," *IEEE software*, no. 3, pp. 18–23, 1999.

[27] F. Sarro, A. A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang, "Feature lifecycles as they spread, migrate, remain, and die in app stores," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, Aug 2015, pp. 76–85.

[28] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requir. Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016. [Online]. Available: http://dx.doi.org/10.1007/s00766-016-0251-9

[29] A. Al-Subaihin, F. Sarro, S. Black, L. Capra, and M. Harman, "App store effects on software engineering practices," *IEEE Transactions on Software Engineering*, vol. 50, no. 8, p. 1, 2018.

[30] W. Lowe, K. Benoit, S. Mikhaylov, and M. Laver, "Scaling policy preferences from coded political texts," *Legislative studies quarterly*, vol. 36, no. 1, pp. 123–155, 2011.

[31] L. Corral and I. Fronza, "Better code for better apps: a study on source code quality and market success of android applications," in *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems*. IEEE Press, 2015, pp. 22–32.

[32] S. A. Scherr, F. Elberzhager, and S. Meyer, "Listen to your users–quality improvement of mobile apps through lightweight feedback analyses," in *International Conference on Software Quality*. Springer, 2019, pp. 45–56.

[33] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE Software*, vol. 32, no. 3, pp. 70–77, 2015.

[34] A. Wasserman, "Software engineering issues for mobile application development," 01 2010, pp. 397–400.

[35] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018.

[36] C. Zhou, S. Kuttal, and I. Ahmed, "What makes a good developer? an empirical study of developers' technical and social competencies,doi = 10.1109/VLHCC.2018.8506577," 10 2018, pp. 319–321.

[37] "Small teams," Retrieved Feb-2020, https://stormotion.io/blog/6-tips-on-how-to-structure-a-development-team/.

[38] M. D. Syer, M. Nagappan, A. E. Hassan, and B. Adams, "Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source android apps," in *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, ser. CASCON '13. Riverton, NJ, USA: IBM Corp., 2013, pp. 283–297. [Online]. Available: http://dl.acm.org/citation.cfm?id=2555523.2555553

[39] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008. [Online]. Available: http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

[40] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review*, vol. E 69, no. 026113, 2004.

[41] M. Klug and J. P. Bagrow, "Understanding the group dynamics and success of teams," *Royal Society open science*, vol. 3, 2016.

[42] J. Cohen, "A power primer." *Psychological bulletin*, vol. 112, no. 1, p. 155, 1992.

[43] ——, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[44] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[45] A. Ghosh and S. Kuttal, "Semantic clone detection: Can source code comments help?" 10 2018, pp. 315–317.

[46] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace." in *ICSE*, P. Jalote, L. C. Briand, and A. van der Hoek, Eds. ACM, 2014, pp. 767–778.

[47] ——, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767–778. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568263

[48] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 499–510. [Online]. Available: http://doi.acm.org/10.1145/2950290.2950299

[49] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, May 2018, pp. 48–58.

[50] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '16. New York, NY, USA: ACM, 2016, pp. 38:1–38:10. [Online]. Available: http://doi.acm.org/10.1145/2961111.2962600

[51] M. D. Syer, B. Adams, Y. Zou, and A. E. Hassan, "Exploring the development of micro-apps: A case study on the blackberry and android platforms," in *2011 11th IEEE International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2011, pp. 55–64.

[52] L. Corral and I. Fronza, "Better code for better apps: A study on source code quality and market success of android applications," in *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, May 2015, pp. 22–32.

[53] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE transactions on software engineering*, vol. 43, no. 9, pp. 817–847, 2017.

[54] F. Ali Shah, K. Sirts, and D. Pfahl, "Simple app review classification with only lexical features," in *Proceedings of the 13th International Conference on Software Technologies - Volume 1: ICSOFT,*, INSTICC. SciTePress, 2018, pp. 112–119.

[55] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 14–24. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884818

[56] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 499–510.

[57] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "Ardoc: App reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 1023–1027.

[58] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H. C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 2017, pp. 91–102.

[59] F. Palomba, M. Linares-Vasquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "User reviews matter! tracking crowdsourced reviews to support evolution of successful apps," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 291–300.

[60] T. McDonnell, B. Ray, and M. Kim, "An empirical study of api stability and adoption in the android ecosystem," in *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*. IEEE, 2013, pp. 70–79.

[61] N. Zhong and F. Michahelles, "Google play is not a long tail market: an empirical analysis of app adoption on the google play app market," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 499–504.

[62] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis, "Rise of the planet of the apps: A systematic study of the mobile app ecosystem," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 277–290.

[63] C. Tröster, A. Mehra, and D. van Knippenberg, "Structuring for team success: The interactive effects of network structure and cultural diversity on team potency and performance," *Organizational Behavior and Human Decision Processes*, vol. 124, no. 2, pp. 245–255, Jan. 2014.

[64] M. L. M. Gnter K Stahl, A. Voigt, and K. Jonsen, "Unraveling the effects of cultural diversity in teams: A meta-analysis of research on multicultural work groups," *Journal of International Business Studies*, vol. 41, pp. 690–709, 2010.

[65] D. L. Gladstein, "Groups in context: A model of task group effectiveness," *Administrative Science Quarterly*, vol. 29, no. 4, pp. 499–517, 1984.

[66] M. A. Campion, G. J. Medsker, and A. C. Higgs, "Relations between work group characteristics and effectiveness: Implications for designing effective work groups," *Personnel psychology*, vol. 46, no. 4, pp. 823–847, 1993.

[67] J. Hu and R. C. Liden, "Antecedents of team potency and team effectiveness: an examination of goal and process clarity and servant leadership." *The Journal of applied psychology*, vol. 96 4, pp. 851–62, 2011.

[68] M. Kilduff and D. Krackhardt, "Bringing the individual back in: A structural analysis of the internal market for reputation in organizations," *The Academy of Management Journal*, vol. 37, pp. 87–108, 1994.