Prioritizing The Devices To Test Your App On: A Case Study Of Android Game Apps

Hammad Khalid, Meiyappan Nagappan
Software Analysis and Intelligence Lab (SAIL)
Queen's University
Kingston, Canada
{hammad, mei}@cs.queensu.ca

Emad Shihab
Department of Computer Science and Software Engineering
Concordia University
Montreal, QC, Canada
eshihab@cse.concordia.ca

Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University
Kingston, Canada
ahmed@cs.queensu.ca

ABSTRACT

Star ratings that are given by the users of mobile apps directly impact the revenue of its developers. At the same time, for popular platforms like Android, these apps must run on hundreds on devices increasing the chance for device-specific problems. Device-specific problems could impact the rating assigned to an app, given the varying capabilities of devices (e.g., hardware and software). To fix device-specific problems developers must test their apps on a large number of Android devices, which is costly and inefficient.

Therefore, to help developers pick which devices to test their apps on, we propose using the devices that are mentioned in user reviews. We mine the user reviews of 99 free game apps and find that, apps receive user reviews from a large number of devices: between 38 to 132 unique devices. However, most of the reviews (80%) originate from a small subset of devices (on average, 33%). Furthermore, we find that developers of new game apps with no reviews can use the review data of similar game apps to select the devices that they should focus on first. Finally, among the set of devices that generate the most reviews for an app, we find that some devices tend to generate worse ratings than others. Our findings indicate that focusing on the devices with the most reviews (in particular the ones with negative ratings), developers can effectively prioritize their limited Quality Assurance (QA) efforts, since these devices have the greatest impact on ratings.

1. INTRODUCTION

Usage of Android devices has grown at a tremendous rate over the past few years [1]. To capitalize on this growth, both small and large companies are developing an enormous amount of applications (called mobile apps), designed to run on Android devices. However, the top-rated or the featured apps in the app markets, are the apps with the most downloads, and hence the most revenue [2, 3]. Also the app market is very competitive, especially for game app developers who have to compete with almost 120,000 game apps already in the Google Play store – more than any other category of apps. To compete in this environment, developers need to get (and maintain) good ratings for their apps [2]. This can be difficult since users are easily annoyed by buggy apps, and that annoyance could lead to bad ratings [4, 5]. Hence, app developers need to test their apps thoroughly on different devices to avoid a poor rating.

To make matters worse, there exists a large number of Android devices, each with its own nuances. In fact, dealing with device specific issues of (the many) Android devices is considered one of the biggest challenges developers face when creating an Android app [6]. A 2013 survey from Appcelerator, which has aggregated results from similar such surveys in the past three years, shows that developer interest in Android has fallen to 79% in 2013 from a high of 87% in 2011 [7]. A staggering 94% of the developers that avoid working on Android apps cited Android fragmentation as the main reason [8]. Android fragmentation refers to the concern, that since there are many devices with different screen sizes, different OS versions, and other hardware specifications, an app that functions correctly on one device might not work on a different one [9]. Joorabchi *et al.* [6] examined the challenges in mobile application development by interviewing 12 mobile developers. One of main the findings of their study is that dealing with device specific issues (e.g., testing these devices) remains a major challenge for mobile app developers. Even Google suggests that developers should test their apps on actual devices before they release the app [10]. There are now even business solutions based on providing devices remotely for testing [11]. However, with costs ranging in approximately a dollar for every 15 minutes of device time [12], the total expense incurred to developers can get very high. These concerns are especially worrisome for game app developers since they have to manually test their apps on-device instead of just relying on automated tests; due to the graphical and non-deterministic nature of video games [13].

Therefore, Android developers (and game developers in particular) need to carefully prioritize their testing and Quality Assurance (QA) efforts on the most important devices. While there has been some previous research in automated testing for Android apps [14, 15, 16, 17], to the best of our knowledge, there has not been any work on prioritizing testing and QA efforts on the devices that have the most impact on the rating of an app.

We coin the term 'review share', which measures the percentage of reviews for an app from a specific device. For example, a review share of 10% means that a specific device gave 10% of all ratings for an app. We use 'review share' to demonstrate the importance of focusing QA efforts on a smaller subset of devices. Through a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

study of 89,239 user reviews for 99 free game apps from various Android devices, we explore the following research questions:

RQ1. What percentage of devices account for the majority of user reviews?

We find that on average 33% of the devices account for 80% of all reviews given to a free game app. With this information, app developers can prioritize their testing and QA efforts by focusing on a small set of devices that have the highest potential review share.

RQ2. How can new developers identify the devices that they should focus their testing efforts on?

We find that developers can use the list of devices with the most review share in all other gaming apps as a good indicator of which devices they should focus their testing and QA efforts on.

RQ3. Do the star ratings from different Android devices vary significantly?

By examining the reviews from different devices for the same app, we find that some devices give significantly lower ratings. Developers can take corrective actions for such devices or remove support for them if they see fit.

Takeaway: Developers can better prioritize their QA efforts by picking the devices that have the most impact on the ratings of apps. Some of these devices may give worse ratings than others – developers can either allocate additional QA resources for these devices, or remove support for them.

The remainder of this paper is organized as follows: Section 2 surveys the related work. Section 3 discusses in detail the data that we analyze for our study. Section 4 presents the results of our study. Section 5 performs further analysis on paid apps, and apps in other categories, to see if our findings generalize. Section 6 discusses the implications of our findings for developers. Section 7 discusses the potential threats to the validity for our study. Section 8 concludes the paper.

2. RELATED WORK

In this section, we present work related to Android fragmentation, work related to mobile app reviews and work related to the testing of Android apps.

2.1 Work related to Android Fragmentation

A recent study by Han *et al.* manually labeled bugs reported for specific vendors of Android devices [9]. They highlighted evidence for Android fragmentation by showing that HTC and Motorola devices have their own vendor specific bugs. They considered this as evidence for Android fragmentation. Ham *et al.* came up with their own compatibility test to prevent Android fragmentation problems [18]. They focus on doing code analysis and API pre-testing to identify possible issues. Our study differs from their work since we seek to help app developers determine which devices they need to test their apps on. We determine this set of Android devices from the user reviews of an app.

2.2 Work Related to App Reviews and App Quality

Recent studies have examined the importance of app reviews. Harman *et al.*'s pioneering work on mining the BlackBerry store, found that there is a strong correlation between app rating and number of downloads [2]. Kim *et al.* also found ratings to be one of the key determinants in a user's purchase decision of an app [19]. Linares-Vasquez *et al.* show that the quality (in terms of change and fault-proneness) of the APIs used by Android apps negatively impacts their success, in terms of user ratings [20].

More recently there have also been studies on using the reviews for generating feature requests from the user comments. Pagano and Maalej [21] carried out an exploratory study on reviews from iOS apps to determine their potential for requirements engineering processes. There have also been studies on automatically extracting feature requests from the user reviews [22, 23]. Our work complements the aforementioned work since our goal is to use mobile app reviews like them, but to assist developers in determining whether they can use a small subset of devices to test their game apps.

In a previous study, we manually analyzed and tagged reviews of iOS apps to identify the different issues that users of iOS apps complain about [4, 5]. We hope to help developers prioritize the issues that they should be testing for. This study differs from our previous work as we are focusing on identifying the different devices that Android app developers should be focusing their QA efforts on.

2.3 Work Related to Testing Android Apps

Several recent studies have attempted to reduce the testing burden of Android developers. One of these studies is by Kim *et al.* [24] who look at testing strategies to improve the quality of mobile apps. Agarwal *et al.* study how to better diagnose unexpected app behavior [25]. Hu *et al.* suggested methods for automated test generation and analysis for Android apps [14]. Machiry *et al.* presented a dynamic input generation system which can be used by developers for black-box testing [17]. There has also been previous work which has aimed to automate testing in a virtual environment. Amalfitano *et al.* presented a tool which automatically generates tests for Android Apps based on their graphical user interface [15]. DroidMate is another such tool which uses genetic algorithms to generate input sequences (i.e., user interaction, or simulated events) [16].

While testing in a virtual environment is useful for identifying general issues, developers also test their apps on actual Android devices to identify device specific issues. Our work focuses on helping developers identify the devices that have the most impact on their rating and hence developer should focus their testing efforts on these devices.

Previous research has confirmed the effect of Android hardware and software fragmentation. In addition, recent studies have identified the importance of mobile app user reviews. In this paper, we seek to help developers understand how they can prioritize their QA efforts towards the devices that have the most impact on their ratings.

3. STUDY DESIGN

In this section we discuss the data used in our study, the main sources of this data, and the collection process for this data. The main data used in our study are the user reviews of the top 99 Android game apps. We collect these reviews from Google Play, which is the main market for Android apps. After collecting these reviews, we identify the devices that these reviews were produced from, as well as, the ratings associated with each review. Figure 1 provides an overview of the process used in our study. The following sections describe the data used in our study and our data collection method in further detail.

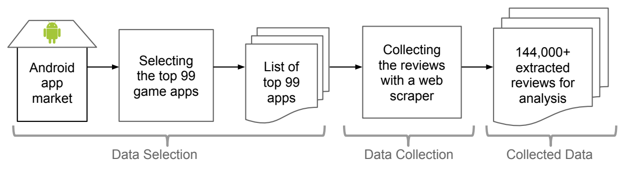

Android app market
Selecting the top 99 game apps
List of top 99 apps
Collecting the reviews with a web scraper
144,000+ extracted reviews for analysis
Data Selection
Data Collection
Collected Data

Figure 1: Overview of our process

3.1 Data Selection

For our study, we selected the top 99 free game apps as ranked by Google Play. Our proposed method is general (i.e., other apps can be examined – See Section 5 for a discussion about our analysis across the different categories of the market). Nevertheless, we picked these top game apps for two reasons (a) since game apps are the most popular apps in the Android market, and thus our results could have the greatest impact, and (b) top game app developers like Red Robot Labs, Pocket Gems, Storm8, and Animoca (More than 400 millions app download across their app portfolio), in an article on TechCrunch discussed the issues they are having with testing their apps on many devices [26]. Currently they rely on their experience (and app analytics data when available) for choosing 30-50 devices to test on.

3.2 Data Collection

To collect the user reviews, we build a web crawler using a web automation and testing tool called Selenium [27]. Our crawler extracts data such as the app name, the review title, the review description, the device used to review the app, and the numerical star rating of the review. This crawler opens a new instance of the Firefox browser, visits the URL of one of the selected apps, and then clicks through the review pages of this app. The required information is located on each page using Xpath then stored into a database [28]. We used a similar crawler to collect the URLs of the top apps. This crawler is much more limited and slower than typical web-crawling since Google Play puts heavy restrictions on crawlers. We found this crawling approach to be the only reliable method for extracting the reviews. All of the crawling is done using an anonymous user.

Also, Google Play limits the total number of reviews that a user can view to a maximum of 480 for each of the star ratings (i.e., a user is restricted to viewing a maximum of 2,400 reviews for each app as there are 5 levels of ratings (5*480)). Recent work by Pagano and Maalej shows that there exists a large amount of noise in reviews, for example one word reviews [21]. Hence even Apps Markets have moved to using more complex methods to calculate a global rating or ranking of an app instead of simply summing up all the reviews [29]. We use 'the most helpful reviews' as they are considered by App stores and users, as one of the most reliable sources of user feedback. The helpfulness is determined by other users voting for reviews. Such crowd-based filtering helps weed out spam reviews.

Summary of Collected Data In total we collect 144,689 reviews (across the 5 levels of star rating) from the studied apps. From this set of reviews, we only consider the ones that have a device associated with them. This reduces the set of reviews to 89,239, each submitted by a unique user. The implications of our review selection is discussed in more detail in Section 7.2. We limit our reviews to those given only after October 2012 to January 2013, a 3-month window since the rate of churn in devices is very high.

3.3 Preliminary Analysis

Prior to delving into our research questions, we perform some preliminary analysis on our dataset as a whole, i.e. using data from all 99 game apps taken together. We perform this analysis to determine whether our review dataset contains reviews from many different devices or a small set of devices. In other words, we would like to determine if Android fragmentation does exist in our data or not. For this, we look at how the reviews are distributed across the devices, for the apps taken as a whole. Using all of the reviews, we identify the number of reviews that each device gave.

In total, our dataset contains 89,239 reviews from 187 unique Android devices. Out of these 187 devices, 114 of these devices have provided more than 100 user reviews. These facts highlight the magnitude of the Android fragmentation problem and the importance of identifying the devices that give the most reviews to apps, for prioritizing QA efforts.

4. RESULTS

Now that we have determined that our dataset contains reviews from many different Android devices, in this section we answer our research questions.

RQ1) What percentage of devices account for the majority of user reviews?

Motivation: As mentioned earlier, Android fragmentation is a major concern of developers who are seeking to develop high quality Android apps [6, 8, 18]. There may be hundreds of devices that developers may need to test their apps on. Testing on such a large number of devices is not feasible for developers with limited time and budget. Therefore, our goal is to determine what percentage of devices account for the majority of reviews.

Approach: To answer this question, we use the reviews that we collected for the 99 free game apps. For each review, we determine the device that the review was posted from. Then we calculate the 'review share' for each device. We define 'review share' as the percentage of reviews from one device compared to the total number of reviews from all devices. For example, a review share of 10% means that a device gave 10% of all reviews. Initially we consider the reviews from all apps taken together. We then determine what percentage of devices is required to cover X% of the user reviews. In our case, X varies from 0 to 80%.

We also examine the results by breaking down the reviews by star ratings. Instead of looking at individual star ratings, we combine 1 and 2 star reviews into a group which we call 'bad reviews', and combine 4 and 5 star reviews into another group which we call 'good reviews'. We label 3 star reviews as 'medium reviews'. We then calculate the review share for each device, when we exclusively consider only bad, medium or good reviews.

To ensure that our grouping makes sense, we run a sentiment analysis tool over the text of the reviews in these groups to validate that these groupings are appropriate (i.e., 'good reviews' actually have the most positive reviews and vice versa) [30]. This tool assigns an integer to the sentiment expressed in the reviews (where a negative integer represents a negative review). In our case, the 'negative reviews' group was assigned a score of -0.32, the 'medium reviews' group was assigned a score of 0.44, while the 'good reviews' group was assigned a score of 1.25. These scores support our grouping scheme.

Findings: Figure 2 shows the percentage of devices (x-axis) vs. the cumulative percentage of reviews (y-axis) in the different rating groups (different coloured curves). From this figure, we find that *20% of the devices account for approximately 80% of the posted*

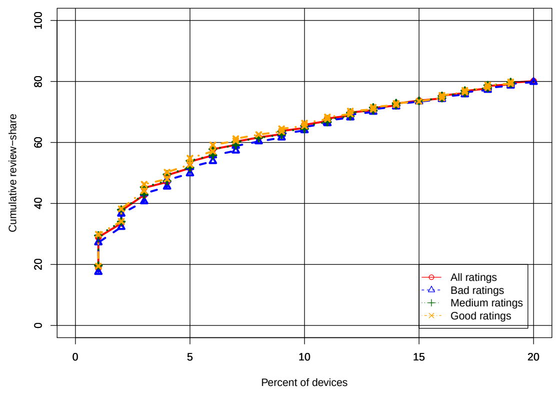


Figure 2: Percent of Android devices used to give X% (X ranges from 0 - 80) of the reviews for free game apps

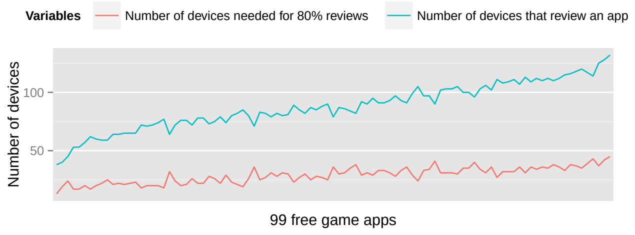


Figure 3: Number of Android devices that account for 80% of the reviews for each game app, compared with the total unique devices that review the app

user reviews. This finding suggests that developers working on free game apps, only need to focus their testing efforts on 20% of the devices to cover the majority (i.e., approximately 80%) of the reviews.

Observing Figure 2, we note that the bad, medium and good ratings curves are very similar. After comparing how different devices gave bad, medium and good reviews, we find that while the devices that gave the bad ratings were fairly identical to the devices that gave good ratings, there were a few discrepancies. The set of devices that had a cumulative sum of 80% of the bad ratings, had four devices that were not in the set of devices that gave most of the good reviews. This finding suggests that there is additional variation in how specific devices rate these game apps.

Discussion: From RQ1, we know that a few devices account for most of the reviews in an app. We wanted to dig deeper and find if a similar trend exists at the app level as well, i.e., for each app, if most of the reviews came from a small subset of devices. The line chart in Figure 3 compares the number of total unique devices that rate an app, with the number of devices that account for 80% of the reviews for that app. The top line in this chart shows the number of unique devices. The minimum number of devices is 38, the median is 87 devices, and while 132 is the maximum number of unique devices that rate an app. This finding clearly indicates that fragmentation exists even at the app level.

The bottom line in Figure 3 shows the number of devices that account for 80% of the reviews. As this figure shows, this number is much lower than the total number of unique devices. We find a minimum of 13, a median of 30, and a maximum of 45 devices accounted for 80% of the reviews, per app. While these numbers may seem large, one would have to keep in context that Android developers often have to think about testing their apps on hundreds of devices, otherwise.

To get a better idea of the comparison above, we calculate the percentage of devices that account for 80% of the reviews in each of the 99 free game apps. We do this on a per app basis. We observe that, 80% of the reviews can be addressed by considering a minimum of 22% of the devices, and at most 53% of the unique devices. The average percentage of devices required to cover 80% of the review share is 33% (and the median is 32%). This finding indicates that app developers can cover the majority (i.e., 80%) of the reviews by focusing their QA efforts on 33% of the devices on average.

A small set of devices are needed to cover 80% of the reviews. On average, 33% of all devices account for 80% of reviews given to free game apps.

RQ2) How can new developers identify the devices that they should focus their testing efforts on?

Motivation: Thus far, we have shown that a small percentage of devices make up the majority of user reviews. An implication of this finding is that, if developers carefully pick their set of focus devices, then they can maximize the effectiveness of their QA efforts. To illustrate, consider a scenario, where a team of developers is working on a new free game app but they can only afford to buy 10 devices to test their app on. Identifying the optimal set of devices is even more important for such developers with limited resources who can only afford a few devices.

One method of picking these devices is to aggregate the review data of every top rated app in a category, and select the devices which would lead to the most review share. This method can provide developers a working set of devices to start from, which they can later augment with other devices. In this RQ we examine the effectiveness of this method.

Using our app review data, we generate Figure 4 which contains a stacked area chart that shows the percentage of reviews that 10 devices with highest review share would cover. This figure shows that even a small number of devices, if carefully chosen (i.e., by looking at which devices frequently post reviews for apps), can account for a considerable number of reviews – more than half of the reviews in most cases.

This leads to the question - how can developers without any reviews for their app pick the best set of devices that they should focus their QA efforts on? For example, can a developer use the review share data from all the top rated apps that are currently present in the game app category? Should they just pick the most popular devices? We explore this issue in RQ2.

Approach: To answer this RQ (working with our dataset of 99 free game apps), we identify a set of 10 devices with the most review share for each app, and compare this set with the set of 10 devices with the most review share in the remaining 98 apps combined. Doing so allows us to simulate an app developer who picks the 10 devices that provide the most reviews for other apps, and using these devices to test his or her own app. Since we have the reviews for that appas well in our dataset, we use these reviews to measure the review share that would be covered for an app based on the selected 10 devices. In other words, this analysis allows us to identify the devices that were in the set of devices that gave the most reviews for an app, but not in the set of devices that gave the most reviews for the rest of the group (essentially a set difference). If such devices are identified, we sum the review share of these devices to highlight the review share that the developer would have

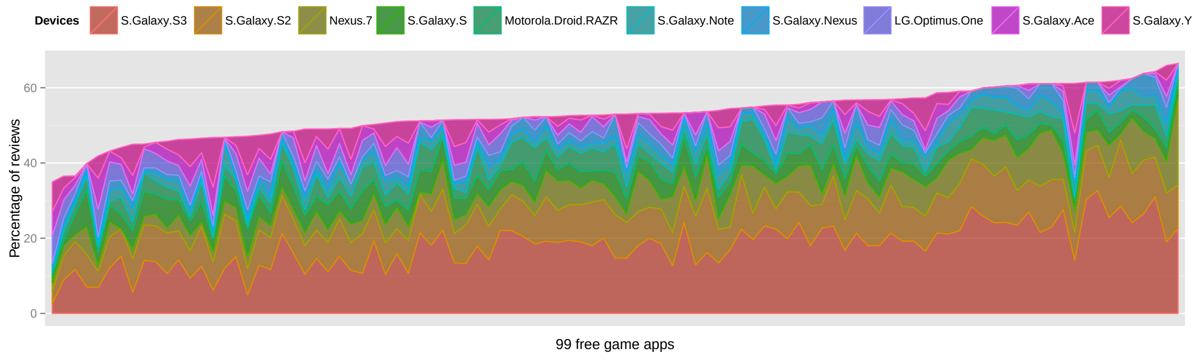
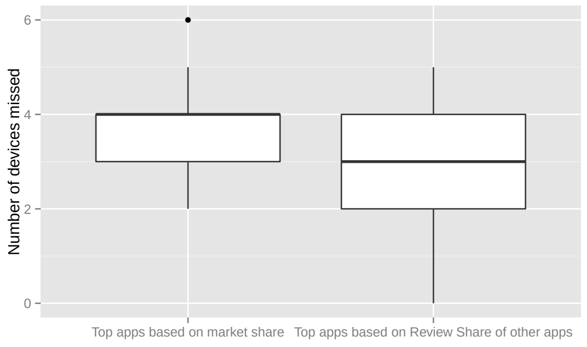
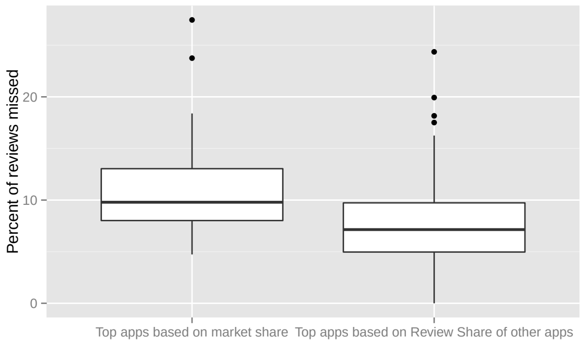


Figure 4: Percent of reviews of each app accounted for, by selecting the 10 devices with the most review share for all 99 free game apps taken together



(a) Number of devices missed



(b) Percent of review share missed

Figure 5: Number of devices and percent of review share missed if 10 devices with the most reviews for all the remaining apps are chosen

lost/missed if he or she had picked the 10 devices with highest review share among all the other 98 apps in that category. These results are shown in Figure 5.

Findings: Using our proposed method to prioritize the devices works well overall. In the majority of the cases the developer will identify the 7 most review-producing devices. Figure 5a (right box plot) shows the distribution of these missed devices for the 99 free game apps. We observe that the median number of missed devices is 3 devices, with a max of 5 devices and a min of 0.

Now, we want to know what percentage of the review share would be impacted due to these missed devices. Figure 5b (right box plot) contains a box plot that shows the percentage of review share impacted due to the missed devices. The median of missed review share is 7.1%. This is not a large loss in review share, especially given that, using this method (i.e., using devices that have a large review share in the app's category), developers will have the benefit of picking an adequate set of devices even before releasing their app to the market.

Discussion: While identifying the devices with the most review share within an app's category is useful, some developers may opt to pick the devices with the most overall *market share* (which is posted on many mobile analytics websites, e.g., App Brain [31]) to prioritize their QA effort. Market share is generally determined by the number of active Android devices.

To compare this method of simply using market share to our proposed method, which uses the reviews, we compare the devices with the most market share to the devices that have the most review share for each of the game apps. We obtain the list of devices with the most market share from App Brain, which gives us a list of the 10 devices (shown in Table 1) for our studied time period (October 2012 to January 2013).

Figure 5a (left box plot) shows the distribution of missed devices for each the 99 free game apps, when market share devices are compared with the 10 devices that have the highest review share for the corresponding app. We find that the median number of devices missed if we consider the market share devices is 4, which is higher than if we use review share in the app's category. Moreover, Figure 5b (left box plot) shows the distribution of the percentage of review share impacted due to the missed devices. Once again, we see that the median value is 9.8%. This rate is higher than the median if our method was used (which has a median of 7.1%). The difference in the number of missed devices and missed review share between choosing our review-share method and the market-share method is statistically significant (p-value « 0.01 for a paired Mann Whitney U test). Our findings suggest that simply using the market share is not sufficient, and using our method, which uses reviews from apps in the same category can identify devices that have a greater chance to review the app. Thus by using our method, a developer can improve the effectiveness of their device prioritization efforts, since they will be able to identify devices that have a greater impact on the ratings of an app.

While examining the 10 devices with the most review share in the

Table 1: The market share of the most popular Android devices for our studied time period (October 2012 to January 2013)

Top market share device	Market share (%)
Samsung Galaxy S3	9.4
Samsung Galaxy S2	7.8
Samsung Galaxy S	2.6
Samsung Galaxy Ace	2
Samsung Galaxy Note	1.9
Samsung Galaxy Y	1.9
HTC Desire HD	1.5
Asus Nexus 7	1.1
Samsung Galaxy Tab 10.1	1.1
Motorola Droid RAZR	1.1

game category, we notice not all of the devices rate apps the same way. This makes us wonder if some specific devices give worse ratings than others, and thus need special attention from developers. We explore this question next in RQ3.

Devices with the most review share across all existing top rated game apps are a good indicator of which devices are likely to have a large review share for each game app. Using this list of devices, a developer can focus their QA efforts even before they release the first version of their app.

RQ3) Do the star ratings from different Android devices vary significantly?

Motivation: Since different devices have varying specifications, it could be the case that users of the different Android devices perceive and rate the same app differently. Understanding how different devices rate apps will allow developers to understand why their apps were given the ratings that they receive (i.e., is a device issue or an app issue). If different devices give varying ratings, this would imply that not all reviews of apps should be treated the same way. With this kind of information, the app developer can do one of two things: 1) they can either allocate even more QA efforts (e.g., testing or focus group) to devices that give a poor rating, when the revenue from that device is critical or 2) if there are not enough users of the app on the particular device, then developers can manually exclude these devices [32], from the list of supported devices on Google Play. In either case knowing if certain devices give worse ratings than others will help developers prioritize their QA efforts even further.

Approach: We separately compare the bad (1 and 2 star reviews), medium (3 star reviews) and good (4 and 5 star reviews) reviews from the 20 devices with the most review share of the 99 free game apps. For example, to compare the ratings of the devices that rated game apps poorly, we create a table where the columns are the 99 game apps and the rows are a percentage of ratings from the top 20 devices. Each cell in the table has a bad-ratings:all-ratings percentage (which is the number of 1 and 2 star reviews over the total number of reviews) given to an app, by a specific device. Table 2 is an example of such a table. For instance, the ratio 21:100 in the cell that corresponds to the row 'Samsung Galaxy S3' and the column 'Angry Birds', means that 21 out of every 100 ratings that S3 devices gave to Angry Birds game were bad (i.e., 1 and 2 star reviews). Similarly, 33 out of every 100 ratings from the 'Samsung Galaxy S2' device for the 'Temple Run' game were bad.

To compare how the top review share devices give bad, medium and good ratings, we use the Scott-Knott test [33]. The Scott-Knott test is a statistical multi-comparison procedure based on cluster analysis. The Scott-Knott test sorts the percentage of bad reviews for the different devices. Then, it groups the devices into two different groups that are separated based on their mean values (i.e., the mean value of the percentage of bad reviews to all reviews for each device). If the two groups are statistically significantly different, then the Scott-Knott test runs recursively to further find new groups; otherwise, the devices are put in the same group. In the end of this procedure, the Scott-Knott test comes up with groups of devices that are statistically significantly different in terms of their percentage of bad reviews to all reviews.

Table 2: An example of the table used to compare bad ratings given by devices

Device	Angry Birds	Temple Run
Samsung Galaxy S3	21:100	19:100
Samsung Galaxy S2	29:100	33:100

Table 3: Scott-Knott test results when comparing the mean percentage of bad ratings given from each device to free game apps, divided into distinct groups that have a statistically significant difference in the mean

Group	Device	Mean % of bad ratings for the device per app
G1	Motorola Droid X2	45.79
G2	Droid Bionic	39.25
	Motorola Droid X	39.20
	HTC Sensation 4G	39.10
	HTC Evo 4G	39.03
	HTC Desire HD	36.81
	Samsung Galaxy Nexus	35.72
	HTC EVO 3D	35.53
	HTC One S	35.31
G3	Motorola Droid RAZR	33.51
	Samsung Galaxy S	33.26
	LG Optimus One	31.11
	HTC One X	32.76
G4	Samsung Galaxy Ace	30.02
	Samsung Galaxy Note	29.68
	Samsung Galaxy S3	28.19
	LG Cayman	28.17
	Samsung Galaxy S2	27.83
	Asus Nexus 7	26.90
	Samsung Galaxy Y	26.78

Findings: The 20 devices (which we examine in this RQ) are divided into 4 statistically significantly different groups. Table 3 shows the significantly different groups of devices as indicated by the Scott-Knott test. Table 3 also lists the devices that are in the group and the mean percentage of bad reviews for each of the devices.

Our findings show that indeed, the users of some devices such as the 'Motorola Droid X2' give more bad ratings to apps than others. We find that this device has a significantly higher ratio of bad ratings than the devices that give the least ratio of bad ratings to all ratings (i.e., Samsung Galaxy Y). The Scott-Knott test also shows that this device has the lowest ratio for good ratings. A reason behind these poor ratings could be manufacturer specific problems. A recent study by Han *et al.* provided evidence of vendor specific problems when they compared bug reports of HTC devices with Motorola devices [9]. A report of this device from Android Police (an Android dedicated web blog) describes its sluggish performance and poor screen resolution [34]. The poor screen resolution may be the main issue with this device since most game apps require a good screen resolution and performance.

On the other hand, we notice that the users of some devices such as the 'Samsung Galaxy Y', 'Asus Nexus 7' and 'Samsung Galaxy S2' give less bad ratings than other devices. To better understand

the results, we further investigated the data to see whether the bad reviews were given to the same app or whether the bad reviews were given from different apps. We discover that the bad reviews are different for all devices (i.e., it is not the same apps which are receiving the bad ratings across the different devices). For example, 92% of all ratings given from the 'HTC Sensation 4G' device to the 'Tap Tap Revenge 4' game app are bad ratings while the median percentage of bad ratings given by the same device is 37.5%. We also find that none of the other top devices gave this app such poor ratings. Thus we see that this particular app just does not work well on the 'HTC Sensation 4G' device. On further examination, we find many complaints by users of this device on the forum of the developer of 'Tap Tap Revenge 4'. Examining the reviews from this device, we see that this app crashes after most major events (i.e., a song ending in the app) [35].
Discussion: Our findings imply that developers should be aware that a few devices may give significantly worse reviews than others. These bad ratings may be given because the device itself provides a poor user experience (i.e., 'Motorola Droid X2'), or that the app itself does not work well on a device (as in the case of 'HTC Sensation 4G' device and the 'Tap Tap Revenge 4' app). In either case, developers aware of this finding can specifically address the concerns expressed in the reviews from such devices (e.g., do detailed testing) or remove the support for such devices. We are not suggesting that developers only need to test on devices that give statistically different star ratings than others; developers just need to devote specific attention towards problematic devices. Monitoring how different devices rate apps can reveal devices that are bringing down the overall rating of an app. Additionally, our findings suggest that the user's perception of the quality of an app depends on the device on which it runs. Hence research on testing Android apps, should factor in the effect of the device.

Another possible reason why some devices give worse ratings than others could be that those devices are simply older. Older devices also tend to have worse hardware specifications than new devices so the performance difference may be the main reason for these bad ratings. To test this theory, we identify the release dates of the 20 devices with the most review share. Then we do a correlation test of their release dates and the median of the percentage of bad ratings to all ratings for the 20 devices. Using the 'Spearman' correlation test we find a correlation of -0.61. Since the correlation is negative, it means that newer devices have a lower percentage of bad ratings. Therefore, this result implies that when it comes to the top 20 devices, the age of the device may be a factor for bad ratings. It is important to note here that what we are observing is a correlation, not causation.

For the devices that give the most reviews for game apps, we find statistical evidence suggesting that some of these devices give worse ratings than others. Developers can take corrective actions for such devices by allocating more QA effort, or removing support for these devices if they see fit.

5. GENERALIZING THE RESULTS

We focus on free game apps since this lets us examine a very focused context, thus avoiding other confounding factors such as cost, functionality, and end user profile. We now wish to examine whether our findings generalize. First we compare our results of free game apps with paid ones. Then, we compare our results for different categories.

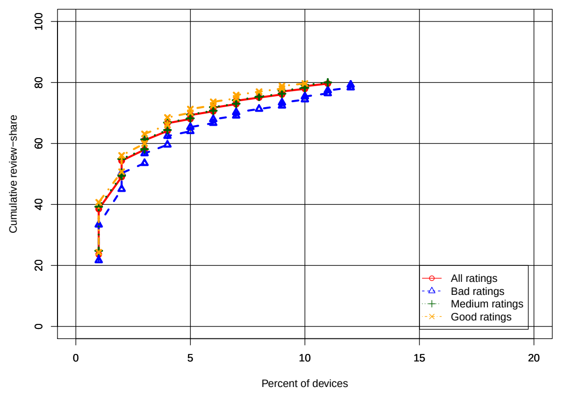


Figure 6: Percent of Android devices which contribute X% (X ranges from 0 - 80) of the reviews for all paid game apps

5.1 Comparison with Paid Game Apps

Although it is likely that many devices review the paid game apps as well, we believe that there will be a more even distribution of reviews among the devices. The move even distribution is because users of paid apps may be more inclined to give reviews, since they paid for the apps [21]. To conduct our comparison, we collected all the reviews for the paid game apps. In total, we collected 61,996 reviews, of which 42,110 were associated with one of 159 devices.

Similar to RQ1, we identify the percentage of devices that are needed to account for the majority of the reviews given to the paid game apps. We use the same methods that we used in RQ2, to determine an ideal method to choose the top 10 devices for focussed QA efforts - based on the reviews from other apps, or based on market share. We use the same method that we used in RQ3 to identify if ratings from different devices vary, for paid game apps.
What percentage of devices account for the majority of user reviews? For paid game apps, we find that the median number of unique devices that review each paid game app is 50. Compared to free game apps, which have a median of 87 devices, we find that paid game apps have much fewer unique devices that rate an app. In terms of percentage of devices needed to account for 80% of the reviews on a per app basis, we find that while the median is the same as free game apps, the range of these percentages is more for paid game apps in comparison to free game apps. The minimum percentage is 16.7% whereas the maximum percentage is 69.2%.

Figure 6 shows the percentage of devices vs. the cumulative percentage of reviews in the different rating groups for all the paid game apps taken together. Again, compared to free game apps, we find that much less devices are needed to account for 80% of the reviews. We find that only 12.6% of the devices are needed to cover 80% of the reviews (compared to the 20% for the free game apps). Our finding suggests that developers working on paid apps should be even more attentive of their analytics since in some cases a select few devices have a huge impact on their ratings, and hence their future revenue.
How can new developers identify the devices that they should focus their testing efforts on? From Figure 7a and Figure 7b, we can see that it is indeed more beneficial to target devices based on the review share of the other paid game apps instead of using the market share, since we will be targeting devices that are used to review the apps more. The difference in both the cases is statistically significant (p-value « 0.01 for Mann Whitney U test). This result

Table 4: Scott-Knott test results when comparing the mean percentage of bad ratings given from each device to free game apps, divided into distinct groups that have a statistically significant difference in the mean

Group	Device	Mean % of bad ratings for the device per app
G1	EeePad Transformer TF101	42.83
	Motorola XOOM	41.20
	HTC Evo 4G	40.32
	Samsung Nexus S	39.16
	Galaxy Tab 10.1	38.79
	HTC Desire HD	36.55
	Droid Bionic	36.26
	EeePad Transformer TF300	35.69
G2	Samsung Galaxy Note	33.55
	HTC Sensation 4G	33.25
	HTC EVO 3D	32.33
	Samsung Galaxy S	32.84
	SEMC Xperia Play	31.53
	HTC One S	31.29
	Motorola Droid RAZR	30.29
	HTC One X	29.29
	Samsung Galaxy S2	29.11
	Samsung Galaxy Nexus	29.04
G3	Samsung Galaxy S3	26.72
	Asus Nexus 7	22.63

is similar to the result for the free game apps. However one noticeable difference is that the number of devices missed (median of 5 devices) and the review share missed (median of 15.9%) for paid apps when using the market share data is slightly higher than in the case of free game apps (where the median values are 4 devices and 9.8%). Thus we can see that in the case of paid game apps, the market share data is much less accurate in helping the developer identify the devices to test their app on first.

Do the star ratings from different Android devices vary significantly? For the paid game apps, we illustrate the differences in the percentage of bad ratings from each device in Table 4. Our findings show that indeed, even for paid game apps, different devices provide different levels of bad ratings to apps. The Scott-Knott test groups the devices into 3 statistically significantly different groups. For example, the 'Asus Nexus 7' and the 'Samsung Galaxy S3' devices give significantly better ratings to paid game apps than many of the other devices (i.e. Motorola Xoom, EeePad TF101, HTC Evo 4G). We also find that the set of devices that give a higher percentage of bad ratings in paid apps is different from the set of devices that give a higher percentage of bad ratings in free game apps. For example, the Motorola Xoom and EeePad TF101 are not even in the top 20 devices that review free game apps. *Thus developers need to be careful about using free games apps to prioritize their QA efforts for paid game apps, as the devices that the users use for paid game apps do vary.* Next, we examine the devices that review apps in other categories (not just games).

Trends and results in paid game apps are similar to free game apps as well. However, the argument for prioritization is more pronounced in the case of paid game apps. Therefore paid game app developers can make optimal use of their QA budget by prioritizing their efforts based on the share of reviews from a device.

5.2 Analysis of Apps in Other Categories

While the findings of the study so far are most relevant for developers of game apps, we want to see if our findings hold for apps in other categories as well. More specifically, we want to examine if the reviews of apps in the Business, Education, Sports and Entertainment categories have similar patterns to those we found in our study – this can help other developers deal with Android fragmentation as well. Statistics about the data used to perform this analysis is summarized in Table 5.

Table 5: Reviews collected and number of devices for the other 4 categories of free apps

Category	# of Reviews	# Reviews Linked to a Device	# of devices
Business	21,365	13,901	153
Education	14,097	9,000	168
Sports	16,790	12,102	157
Entertainment	64,690	40,399	180

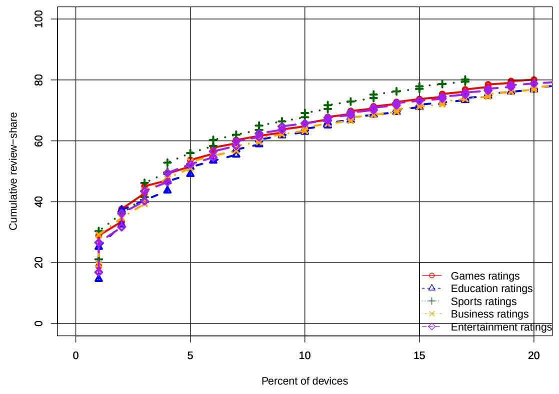
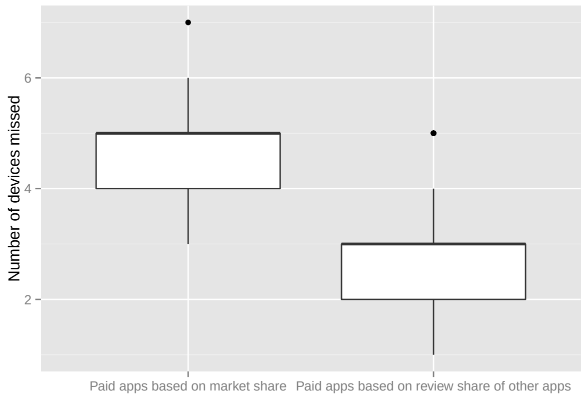


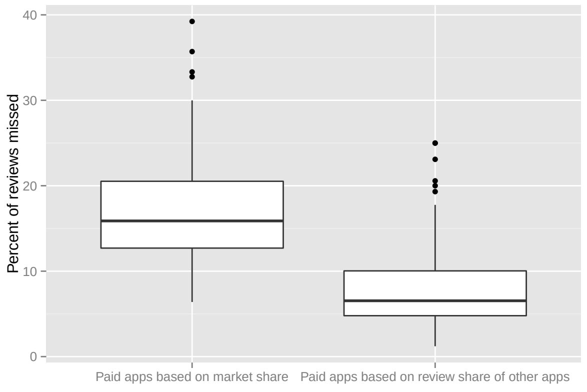
Figure 8: Percent of Android devices used to give X% (X ranges from 0 - 80) of the reviews of free apps in 5 categories

What percentage of devices account for the majority of user reviews? Figure 8 shows the percentage of devices vs. the cumulative percentage of reviews in the different rating groups for each of the 5 categories. We find that when considered together, 21.1%, 22.2% and 22.6% of the devices account for 80% of the reviews given to apps in the Entertainment, Business and Education categories respectively. For apps in the Sports category, only 17.1% of the devices account for 80% of the reviews. These numbers are similar to the 20% in the free game apps category. Similar to game apps, these findings imply that developers working on these categories only need to focus on a small subset of devices to cover the majority of the reviews given to their apps. Our finding suggests that developers working on apps in categories other than games can also greatly improve their efficiency by focusing on the few important devices since these devices make up the majority of the reviews given to an app.

How can new developers identify the devices that they should focus their testing efforts on? Similar to our results for free game apps, we find that developers get more coverage of the reviews, if they focus on the devices with the most review share instead of the devices with the most market share. We find that by focusing on the devices with the most review share, instead of the devices with the most market share, developers can gain an extra 7.69%, 8.51%, 6.48%, 7.91% review coverage on apps in the Business, Education, Entertainment and Sports categories respectively.



(a) Number of devices missed



(b) Percent of review share missed

Figure 7: Number of devices and percent of review share missed if 10 devices with the most reviews for all the remaining paid game apps are chosen

Do the star ratings from different Android devices vary significantly? We now compare how different devices review free apps in Entertainment, Business, Sports, Game and Education categories. Using the Scott-Knott test, we observe that the variation of ratings from different devices is also present for the apps in these categories of apps. In each category, the Scott-Knott test divided the set of devices into four statistically significantly different groups based on the percentage of bad reviews to all reviews in an app. We note that many of the devices that give the most ratings to apps in these categories vary in terms of their bad and good rating. We also note that the Asus Nexus 7 gave more bad ratings, and less good ratings for the apps in the Entertainment category than it did in the game category. The fact that the Asus Nexus 7 gave more bad ratings in the Entertainment category may be because it is a tablet and not all apps scale well to larger screens, indicating that some devices may rate apps in different categories with varying criteria. Our findings further suggest that developers working on Android must be attentive of their analytics as it could help them identify problematic devices.

After comparing the results of our RQs for the free game apps, with paid game apps and apps in four other categories, we find evidence that our results do generalize. Our finding suggests that developers working on apps in other categories can also use our methods to better prioritize their QA efforts.

When working on apps in the Entertainment, Business, Sports, and Education categories, developers only need to focus their testing efforts on a small set of devices to cover the majority of the reviews for their apps. Moreover, we find that many of the devices give different ratings to apps in these categories.

6. IMPLICATIONS FOR DEVELOPERS

While we think that the results from this study will be useful for developers, device usage may have changed by the time most developers view this study. The change is because of the rapid growth and evolution of the mobile industry where newer devices are constantly being released [1]. Thus, developers should focus on our general findings and method rather than the device specific results.

For example, developers can take away the idea of prioritizing their QA efforts on a subset of impactful devices rather than all devices. Moreover, they can use our method of analyzing reviews per device to potentially identify devices that consistently give poor ratings. Once developers identify these problematic devices, they can remove support for them to avoid additional QA and their bad reviews all-together. If developers feel that the additional downloads (which can generate high ad revenue) from these devices are worth potentially lower average ratings, they can allocate additional QA resources for these devices.

While it is ultimately up to the developers to decide how they are going to prioritize their QA efforts, we think that focusing on the devices that have the most impact on the app's ratings is an effective method (especially because ratings are directly correlated to the number of downloads [2], and thus the revenue generated by apps).

7. THREATS TO VALIDITY

In this section we discuss the perceived threats to our work and how we address them.

7.1 Construct Validity

We compared the bad, medium and good reviews given from different devices to identify if certain devices give different (and worse) ratings than other devices. Note that we are not raising a causal link here in the paper. We are not claiming that an app gets a poor rating because of a device. We are just saying that apps get rated frequently and sometimes poorly from a small set of devices. This is similar to the vast literature on using software metrics for QA prioritization. Such literature do not claim a causal link between software metrics and software defects, but just suggest that software metrics like churn can be used to prioritize QA efforts. The underlying reason for poorer or more frequent reviews from a particular device could be the hardware specification of the devices, the OS running on the devices, or just that the people using a particular device may have a specific profile. More research has to be conducted to identify the underlying causes. Note that data on user profile or which OS version is running on a device is currently not available openly to be mined by researchers. Hence, a major data collection effort has to be launched to examine these underlying factors.

7.2 Internal Validity

Since we limited our reviews to only the reviews from a few months (i.e., October, 1st, 2012 till January, 15, 2013), our data may not accurately represent ratings for the entire year or the entire life of devices. However, to mitigate this threat, we made sure to apply statistical tests, where applicable, to ensure that our findings are statistically significant. Also note that we extract the device information from user reviews. This information, as far as we can tell is very accurate, and cannot be faked, since a user cannot manually change this information when posting a review. The device information is automatically taken from the device from which the review is posted.

Since we require reviews to contain the device information, we had to ignore reviews that were not linked to a device. To determine the impact of this issue on our findings, we measured the average rating from reviews that were linked to a device and reviews that were not linked to a device. We found that for free game apps, the average rating for reviews that were linked to a device is 3.22, while it is 3.43 for reviews that were not linked to a device. For paid game apps, the average rating for reviews linked to a device is 3.53, while it is 3.52 for reviews that are not linked to a device. If we consider all of the apps, not just game apps, we found that the average rating for reviews that are linked to a device is 3.37 and for reviews that are not linked to a device is 3.51 (and we found similar results when we took each category of apps separately). We performed the Wilcoxon statistical significance test and found that there is a statistically significant difference for free game apps and all apps, however, there is no statistically significant difference for paid game apps. In all statistically significant cases, we find the rating tend to be lower for reviews that are linked to a device. This finding indicates that reviews that are linked to a device are more critical, and hence, are more important to developers who are trying to avoid negative reviews.

User reviews can contain some spam reviews that serve as noise in our dataset [21]. To mitigate this issue and ensure the quality of the reviews used in our study, we selected the ‘most helpful reviews’, since they provide us with the most reliable information.

All of our findings are derived from user reviews. In certain cases, user reviews may not directly correlate with other measures of quality such as defects, for example. However, prior research showed that user reviews are directly correlated with app revenues (even for free apps, which make their money through ads). Therefore, we believe that using user reviews is a good proxy of success of an app.

7.3 External Validity

Since our study was performed on 99 mobile apps, our results may not generalize to all game apps. To address this threat, we pick apps which are labeled as ‘Top apps’ by Google Play. We feel that these apps are an appropriate representation of the apps in the game category, and a better choice than hand picking apps. In addition we extended our study to paid game apps, and free apps from four other app categories in Google Play. We found that, our results were often consistent, and sometimes even more pronounced in these apps, when compared to our results for free game apps.

Given that the Android OS and app ecosystems are quickly evolving, the device specific analysis in this paper may not be applicable in a few years (or even months). However, we would like to emphasize that the main takeaways from this study are not about specific devices, but are about our generalizable method for prioritizing QA efforts.

7.4 Conclusion Validity

We assume that testing apps or conducting focus groups for certain devices will find problems (e.g., bugs) which can be fixed by the developer of an app, thereby improving the quality and hence the revenues of the app. Even though it may seem logical, it still is an assumption. For example, we did not verify whether the test effort prioritization does actually improve quality or increase revenues. However, this assumption (testing finds bugs that can be fixed to improve quality) is the basis for most testing efforts. Nevertheless more indepth studies are needed to study such assumptions.

8. CONCLUSION

This study seeks to help game app developers deal with Android fragmentation by picking the devices that have the most impact on their app ratings, thus aiding developers in prioritizing their QA efforts. By studying the reviews of game apps, we find that a small percentage of devices account for most of the reviews given to apps. Thus, developers can focus their QA efforts on a small set of devices. New developers can use data from other apps in the same app category to prioritize their testing and other QA efforts if they do not already have reviews for their app. We also find that some devices give statistically significantly worse ratings than others. Therefore, developers should identify particularly problematic devices, and prioritize their QA efforts even further towards such devices. Finally we find that the results from the free game category to generalized to paid game apps, and free apps in the four other categories that were examined.

In conclusion, developers can adopt our method of analyzing Android app reviews in order to effectively alleviate the QA challenges brought forth by Android fragmentation. In future work, we would like to investigate the problems reported from each device, and why certain Android devices have certain kinds of problems.

References

[1] L. Goasduff and C. Pettey, “Gartner says worldwide smartphone sales soared in fourth quarter of 2011 with 47 per cent growth,” *Gartner, Inc*, vol. 15, 2012.

[2] M. Harman, Y. Jia, and Y. Z. Test, “App store mining and analysis: Msr for app stores,” in *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR ’12)*, Zurich, Switzerland, 2-3 June 2012, pp. 108–111.

[3] “Distimo.” [Online]. Available: http://www.distimo.com/

[4] H. Khalid, “On identifying user complaints of ios apps,” in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 1474–1476.

[5] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, “What do mobile app users complain about? a study on free ios apps,” in *Accepted to be published in IEEE Software*. IEEE Press, 2014.

[6] M. E. Joorabchi, A. Mesbah, and P. Kruchten, “Real challenges in mobile app development,” in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013, pp. 15–24.

[7] “Appcelerator / IDC - Q4 2013 mobile trends report,” November 2013. [Online]. Available: http://www.appcelerator.com.s3.amazonaws.com/pdf/q4-2013-devsurvey.pdf

[8] H. Bodden, “Android’s fragmentation problem,” November 2012. [Online]. Available: http://greyheller.com/Blog/androids-fragmentation-problem

[9] D. Han, C. Zhang, X. Fan, A. Hindle, K. Wong, and E. Stroulia, “Understanding android fragmentation with topic analysis of vendor-specific bugs,” in *Reverse Engineering (WCRE), 2012 19th Working Conference on*. IEEE, 2012, pp. 83–92.

[10] “Using hardware devices.” [Online]. Available: http://developer.android.com/tools/device.html

[11] “Appthwack.” [Online]. Available: https://appthwack.com/

[12] “Appthwack - pricing.” [Online]. Available: https://appthwack.com/pricing

[13] D. Irish, *The game producer’s handbook*. CT-Press, 2005.

[14] C. Hu and I. Neamtiu, “Automating gui testing for android applications,” in *Proceedings of the 6th International Workshop on Automation of Software Test*. ACM, 2011, pp. 77–83.

[15] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon, “Using gui ripping for automated testing of android applications,” in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2012, pp. 258–261.

[16] Z. Jamrozik, Gross, “Droidmate: Fully automatic testing of android apps,” September 2013. [Online]. Available: http://www.droidmate.org/

[17] A. Machiry, R. Tahiliani, and M. Naik, “Dynodroid: An input generation system for android apps,” in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013, 2013, pp. 224–234.

[18] H. Ham and Y. Park, “Mobile application compatibility test system design for android fragmentation,” *Software Engineering, Business Continuity, and Education*, pp. 314–320, 2011.

[19] H.-W. Kim, H. L. Lee, and J. E. Son, “An exploratory study on the determinants of smartphone app purchase,” in *The 11th International DSI and the 16th APDSI Joint Meeting*, Taipei, Taiwan, July 2011.

[20] 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, 2013, pp. 477–487.

[21] D. Pagano and W. Maalej, “User feedback in the appstore: An empirical study,” in *Requirements Engineering Conference (RE), 2013 21st IEEE International*, July 2013, pp. 125–134.

[22] L. V. Galvis Carreño and K. Winbladh, “Analysis of user comments: An approach for software requirements evolution,” in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE ’13, 2013, pp. 582–591.

[23] C. Iacob and R. Harrison, “Retrieving and analyzing mobile apps feature requests from online reviews,” in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, May 2013, pp. 41–44.

[24] H. Kim, B. Choi, and W. E. Wong, “Performance testing of mobile applications at the unit test level,” in *IEEE Int’l Conf. on Secure Software Integration and Rel. Improvement*, ser. SSIRI ’09, 2009, pp. 171–180.

[25] S. Agarwal, R. Mahajan, A. Zheng, and V. Bahl, *Diagnosing mobile applications in the wild*. ACM Press, 2010, pp. 1–6.

[26] K.-M. Cutler, “How do top android developers qa test their apps?” June 2012. [Online]. Available: http://techcrunch.com/2012/06/02/android-qa-testing-quality-assurance/

[27] “Selenium - web browser automation,” June 2012. [Online]. Available: http://seleniumhq.org/

[28] J. Clark, S. DeRose *et al.*, “Xml path language v-1.0,” 1999.

[29] S. L. Lim and P. Bentley, “Investigating app store ranking algorithms using a simulation of mobile app ecosystems,” in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 2672–2679.

[30] “Sentistrength - sentiment strength detection,” July 2013. [Online]. Available: http://sentistrength.wlv.ac.uk/

[31] “Android phone market share - appbrain,” March 2014. [Online]. Available: http://www.appbrain.com/stats/top-android-phones

[32] “Using the device availability dialog,” February 2013. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/1286017

[33] A. J. Scott and M. Knott, “A Cluster Analysis Method for Grouping Means in the Analysis of Variance,” *Biometrics*, vol. 30, no. 3, pp. 507–512, 1974.

[34] “Motorola droid x2: Even raw horsepower can’t save this phone,” September 2013. [Online]. Available: http://www.androidpolice.com/2011/05/28/review-motorola-droid-x2-even-raw-horsepower-cant-save-this-phone-from-mediocrity/

[35] “Tap tap revenge 4 keeps closing,” September 2013. [Online]. Available: https://getsatisfaction.com/tapulous/topics/tap_tap_revenge_4_keeps_closing_after_i_complete_a_song