Mining and Characterizing Hybrid Apps

Mohamed Ali Ali Mesbah
University of British Columbia
Vancouver, BC, Canada
{mohamedha, amesbah}@ece.ubc.ca

ABSTRACT

Mobile apps have grown tremendously over the past few years. To capitalize on this growth and to attract more users, implementing the same mobile app for different platforms has become a common industry practice. Building the same app natively for each platform is resource intensive and time consuming since every platform has different environments, languages and APIs. Cross Platform Tools (CPTs) address this challenge by allowing developers to use a common codebase to simultaneously create apps for multiple platforms. Apps created using these CPTs are called hybrid apps. We mine 15,512 hybrid apps and present the first study of its kind on such apps. We identify which CPTs these apps use and how users perceive them. Further, we compare the user-perceived ratings of hybrid apps to native apps of the same category. Finally, we compare the user-perceived ratings of the same hybrid app on the Android and iOS platforms.

CCS Concepts

•Information systems → Data mining; •Software and its engineering → *Software libraries and repositories;*

Keywords

Mobile Apps, Mining App Stores, Android, iOS, Hybrid

1. INTRODUCTION

Online app stores are the primary medium for the distribution of mobile apps. Through app stores, users can download and install apps on their mobile devices. App stores also provide an important channel for app developers to collect user feedback, such as the overall rating of their app, and issues or feature requests through user reviews.

Currently there are three popular ways to build mobile apps. The first method, "*Native*", developers use the software development kit (SDK) and frameworks for the targeted platform to build the app [16]. *"Native"*, allows developers to use all of the capabilities of a device, provides the best performance and is distributed through the platform's dedicated app store. However, implementing a native app for multiple platforms requires familiarity with the languages, APIs and SDKs of each specific platform, which is resource intensive and time consuming.

The second method, *"Mobile Web App"*, uses web technologies such as *HTML*, *CSS* and *Javascript* to build the application as one website which is optimized for mobile devices. This approach allows the app to be used across multiple platforms which reduces development cost and time. However, *"Mobile Web App"* cannot use device specific hardware features such as the camera or accelerometer [16].

The third method to build an app is "*Hybrid*", which bridges the gap between the first two methods. *"Hybrid"* uses a common code base to simultaneously deliver native-like apps to multiple platforms. These apps can access the hardware features of the device and are also distributed using the platform's app store. Hybrid apps are created using Cross Platform Tools (CPTs). These CPTs provide two approaches to create apps; the first approach allows the developer to use web technologies such as *HTML*, *CSS* and *Javascript* to create a code base which runs in an internal browser (*WebView*) that is wrapped in a native app. Examples of CPTs using this approach include PhoneGap [3] and Trigger.io [40]. The other approach CPTs use allows the developer to write their code in a language such as *C#* or *Javascript* which then gets compiled to native code for each platform. Examples of CPTs that use this approach include Xamarin [43], Appcelerator Titanium [2] and Adobe Air [1]. Mobile Apps created using a CPT are referred to as hybrid apps and we use that term to refer to these apps throughout this paper.

A recent study conducted by Viennot et al. [41] found that out of 1.1 Million Android apps, 129,800 (11.8 %) were hybrid apps.

We present a large-scale study on *hybrid* apps in order to understand their behavior, characteristics and analyze their various app-store attributes. By presenting real market data, our study can help developers decide if using a CPT to create their apps is the best solution.

The mining and study of apps has been studied extensively by other researchers [8, 18, 34, 4, 35, 36]. However, the existing studies address only native apps. To the best of our knowledge, we are the first to study *hybrid* mobile apps.

Overall, our work makes the following main contributions:

- We present a technique to identify hybrid apps.
- We present the first dataset of 15,512 hybrid apps. Our dataset is publicly available [17].

WAMA'16, November 14, 2016, Seattle, WA, USA
ACM. 978-1-4503-4398-5/16/11...$15.00
http://dx.doi.org/10.1145/2993259.2993263

Table 1: App-pair attributes

#	iTunes (iOS); Google (Android)	Description
1	name; title	Name of the app.
2	developerName; developer_name	Name of the developer/company of the app.
3	description; description	Indicates the description of the app.
4	category; category	Indicates the category an app belongs to.
5	isFree; free	True if the app is free.
6	price; price	Price ($) of the app.
7	numberOfReviewers; numberOfReviewers	Number of users rating the app.
8	starsVersionAllVersions; star_rating	Average of all stars (1 to 5) given to the app.
9	version; version_string	User-visible version string/number.
10	updated; updated	Date the app was last updated.

- A characterization of app attributes such as number of reviewers, stars and downloads.
- A measurement of how hybrid apps are perceived by users using a metric which combines the number of reviewers and stars.

2. METHODOLOGY

We address the following research questions in our study:

RQ1. How prevalent are hybrid apps and which cross-platform tool is widely used?

RQ2. Does the choice of a cross-platform tool influence how it is perceived by users?

RQ3. How do hybrid apps compare to native apps of the same category in terms of user-perceived ratings?

RQ4. Does using a cross-platform tool ensure the app is perceived similarly on multiple platforms?

We first describe how we identify hybrid apps and then explain the analysis steps we performed on those apps.

2.1 Data Collection

The first step in our work is to mine apps from the app stores. To this end, we use an open source dataset of Android and iOS app-pairs available on github [28]. The dataset contains the attributes of 80,000 app-pairs; the attributes are outlined in Table 1. This dataset only contains the attributes of the apps and does not include the source code. We used a dataset of app-pairs since one of the main usages of CPTs is to generate the app for more than platform; by looking at app-pairs the chances of finding hybrid apps is much higher. Additionally, having the app-pair allows us to answer RQ4 which compares how a hybrid app is perceived on the iOS and Android platforms.

2.2 Finding Hybrid Apps

In order to determine if an app is hybrid, a manual approach can be used. Such an approach would involve installing the app on a device and exercising its functionality and try to infer from the user experience if the app is hybrid. The manual approach is time consuming and subjective to the user's opinion, which can lead to many false positives. Furthermore, previous work [41] has quantified the number of hybrid apps and discovered that out of 1.1 Million Android apps 129,800 were hybrid. However, that dataset of hybrid apps is not publicly available and hence we had to build our own. For this work, we provide a fully automated technique

Algorithm 1: Identifying hybrid apps

```
   input  : Collection of Apps
   output : Collection of Hybrid Apps
 1 begin
 2     phoneGapApps ← []
 3     titaniumApps ← []
 4     adobeAirApps ← []
 5     foreach i = 0, i <COUNT(APPS), i++ do
 6         app ← APPS[i]
 7         appId ← app.id
 8         apk ← lookForApk(appId)
 9         classes ← classyShark(apk)
10         if classes.contains("org.apache.cordova") then
11             phoneGapApps.append(app)
12         end
13         if classes.contains("org.appcelerator.titanium")
            then
14             titaniumApps.append(app)
15         end
16         if classes.contains("com.adobe.air") then
17             adobeAirApps.append(app)
18         end
19     end
20 end
```

to detect hybrid apps with 100% accuracy. Our technique supports the detection of apps made using 3 CPTs which are PhoneGap [3], Appcelerator Titanium [2] and Adobe Air [1]. We target these CPTs since previous work [41] has shown that they are the most popular CPTs to develop hybrid apps.

To identify a hyrbid app, we download its Android application package file (APK) which is the file format used by the Android operating system to distribute and install application software and middleware. Since the dataset used in 2.1 does not include the APKs, our technique first attempts to find it by using the app's id to search through a dataset of 1.1 Million Android apps [30] and downloads the APK if it is available. Every Android APK includes a file called *"classes.dex"* which includes the classes of an Android app compiled in the dex file format. We use an open source tool called *android-classyshark* [6] to decompile the *"classes.dex"* into a readable format and then inspect it to check if an app uses a CPT. To determine if an app is hybrid we check its class contents for the following references; PhoneGap - "org.apache.cordova", Appcelerator Titanium -"org.appcelerator.titanium", Adobe Air -"com.adobe.air". This technique of inspecting *"classes.dex"* and looking for references of usage of CPTs allows us to identity hybrid apps with 100% accuracy. We chose to download the Android APK instead of the iOS application package file because such information is not publicly available for iOS apps. To validate that our technique correctly identifies hybrid apps, we manually compared app icons and screen shots between the iOS and Android version of the apps and looked for clues such as having the exact UI layout across platforms or the use of un-native UI elements to conclude that an app is hybrid. We sampled 100 random apps and all of them were indeed hybrid, indicating that there are no false positives.

Algorithm 1 summarizes our approach and is used on the dataset in 2.1 to create a dataset of hybrid app-pairs. We use this dataset to answer RQ1 and the results are presented in section 3.

2.3 User-Perceived Rating

There are many ways to measure how an app is perceived by users. Machine learning and NLP techniques can be used to classify user reviews and understand what users think of an app. Such techniques, however, are not accurate enough to measure how an app is perceived by users[14], [38]. Further, the number of app downloads can be used, however Tian et al. [39] has shown that many users can download an app without actually using it.

The star rating of an app, which is the average rating of an app between 1 and 5 has been used by many studies to measure how well an app is perceived by users [39], [13], [15] [5]. However, using just the average rating of an app might not be an accurate measure since it does not consider the number of reviewers of the app.

For instance the *Buckhead app* on the Google Play store, has an average star rating of 5 and only 1 reviewer. Another app, *Instagram* has an average star rating of 4.5 with 37 million reviewers. Trivially the *Instagram* app is more successful since it has much more reviewers. To accurately measure how users perceive an app, we combine the number of reviewers and average star rating as follows:

$$Aggregated\ User-perceived\ Rating(AUR) = \frac{v \times r + m \times c}{v + m} \quad (1)$$

where

1. v = number of reviewers for the app
2. r = average star rating for the app
3. m = average number of reviewers for all apps in the dataset
4. c = average number of star ratings for all apps in the dataset

This formula provides a true Bayesian estimate [10], a variation of which is currently being used by the Internet Movie Database (*IMDb*) [19]; the popular online database of information related to films, television, to generate the list of top 250 films [20]. Essentially, if an app has a few number of reviewers (less than m) we can not place much trust on it to accurately measure how it is perceived by users; so the formula compensates and relies on a conservative estimate using the average number of reviewers and stars in the dataset to calculate AUR.

The output of this formula is a number which ranges between [1–5], which we convert into a percentage to better represent the results. It is possible that an app has no reviewers and hence no star ratings. In our work, we only consider apps which have at least 1 reviewer.

We conducted a small experiment to illustrate the need to combine the average stars and number of reviewers to accurately measure how an app is perceived by users. We randomly selected 100 apps from our dataset where the average number of reviewers (m) and stars (c) across the 100 apps was 163 and 4.2, respectively. We first ranked the apps using the average stars only and then ranked them using AUR. Table 2 shows the rankings using both criteria for 3 apps. Using only the stars ranks app A first despite having only 1 reviewer and app C ranks last despite having 2172 more reviewers and only 0.2 less stars. Using AUR and taking the number of reviewers into account ranks app C first and app A last.

CPT vs App AUR. Since every CPT uses a different set of programming languages and techniques to generate hybrid apps, the goal of RQ2 is to analyze the relationship between the CPT used to develop an app and how it is perceived by users. We use the metric discussed earlier to measure the AUR of apps generated by each CPT and compare the results in Section 3.

Table 2: Ranking of apps using different metrics.

App	Ratings (R)	Stars (S)	Rank (S)	Rank (AUR)
A	1	5.0	1	3
B	5	4.9	2	2
C	2173	4.8	3	1

Hybrid vs Native Apps. While Hybrid apps have been increasing in popularity, native apps still dominate the market place due to their competitive advantage in terms of performance and supported features. The goal of RQ3, is to compare native apps and hybrid apps in terms of AUR. Since there is no way of directly comparing the apps, we compare the categories of apps with one another.

AUR Across Platforms. One of the main reasons behind using a CPT is the convenience of generating an app for multiple platforms using a single code base. Furthermore, this ensures that the user experience is uniform across platforms. The final RQ examines whether these identically created apps are also identical in terms of their AUR. The dataset we use from Section 2.1 contains information about Google's Android and Apple's iOS app-pairs (the same app implemented for different platforms). We again use the formula discussed earlier to measure the AUR of hybrid apps across these platforms and present the comparisons in Section 3. Since we are performing cross-platform analysis, we require that the app has at least 1 rating on both platforms.

2.4 Dataset and Results

Our extracted dataset and results for the identified hybrid apps, as well as all our scripts are available for download at [17].

3. FINDINGS

In this section, we present the results of our study for each research question.

3.1 Prevalence and Popularity of CPTs (RQ1)

Out of the 80,000 apps that were inspected, our technique (described in 2.2) was able to find a total of 15,512 hybrid apps. As shown in Table 3, 10,562 hybrid apps used the PhoneGap CPT, 2,881 used Appcelerator Titanium, and 2,069 used Adobe Air. Furthermore, Figures 4–6 show the distribution of hybrid apps across the various categories in the Google Play store for each of the CPTs. The most popular categories for PhoneGap are *business, lifestyle, travel & local, sports and education.* For Titanium, the most popular categories are *travel & local, lifestyle, finance, business and education.* Finally for Adobe Air the most popular categories are *games, education, business, lifestyle, and entertainment.* Looking at the number of paid vs free apps, we found that *all* the hybrid apps in our dataset were free, regardless of which CPT they used.

Number of Reviewers, Star Ratings & Downloads. We found that 79% of all the hybrid apps in our dataset, 76% of the PhoneGap apps, 81% of the Titanium apps and 90% of the AdobeAir apps have at least one reviewer. As depicted in Table 4, the median for the number of reviewers is 7 overall, 6 for PhoneGap, 7 for Titanium and 19 for Adobe Air.

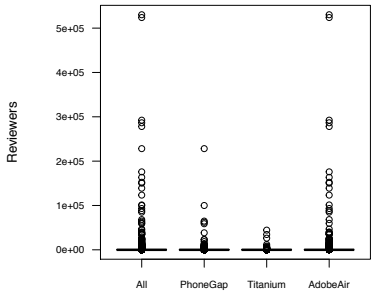

Figure 1: Reviewers

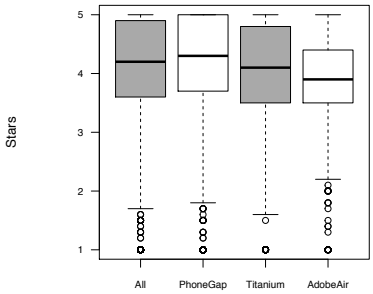

Figure 2: Stars

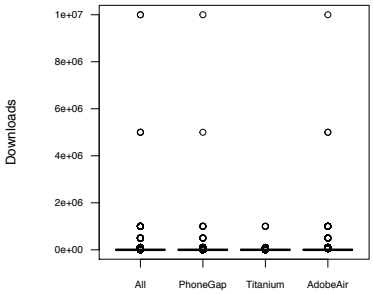

Figure 3: Downloads

Table 3: Number of Hybrid apps using different CPTs

CPT	# of Apps	% of Hybrid Apps
PhoneGap	10,562	68.0%
Titanium	2,881	18.5%
Adobe Air	2,069	13.5%
Total	15,512	100.0%

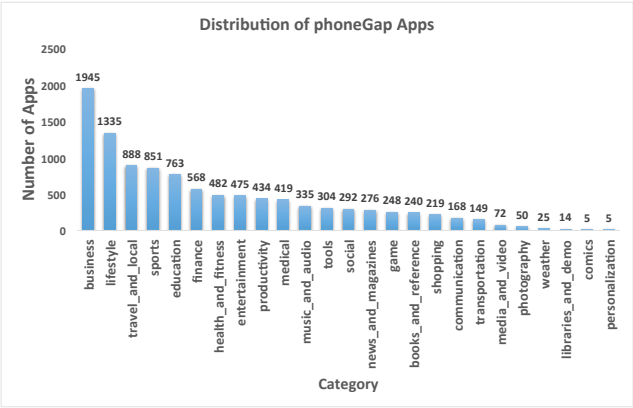

Figure 4: Number of apps in each category created using the PhoneGap CPT.

As for the star ratings, the median is 4.20 overall, 4.3 for PhoneGap, 4.1 for Titanium and 3.9 for Adobe Air. 99% of the apps have been downloaded at least once. The median was 100 downloads overall, 100 for PhoneGap and Titanium and 500 for AdobeAir.

***Finding 1**: The PhoneGap CPT dominates the hybrid app market with a 68% share and is mainly being used to develop business, lifestyle, travel & local apps. Despite having the smallest market share at 13.5%, apps created using the AdobeAir CPT are downloaded and reviewed much more by users. The AdobeAir CPT is mainly used to develop games, and we attribute its popularity to this reason.*

3.2 Effect of CPT on App's AUR (RQ2)

Table 5 below shows the AUR across all hybrid apps and for each of the CPTs. In our analysis, we only keep apps that contained at least 1 reviewer; this reduced the number of apps to 9948. Overall the median for AUR was 84%, 86%

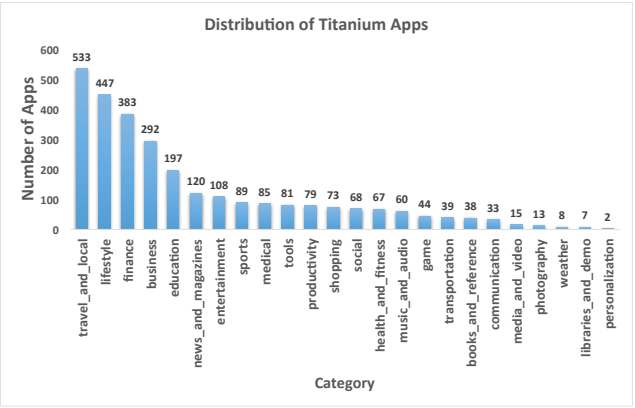

Figure 5: Number of apps in each category created using the Titanium CPT.

Table 4: Descriptive statistics for the hybrid apps: Reviewers (R), Stars (S), and Downloads (D).

ID	Type	Min	Mean	Med	SD	Max
R	All	1	609.40	7.00	10428.74	530200
	PhoneGap	1	182.00	6.00	3465.35	228200
	Titanium	1	146.30	7.00	1487.72	44400
	AdobeAir	1	3088.00	19.00	25792.13	530200
S	All	1	4.07	4.20	0.85	5
	PhoneGap	1	4.13	4.30	0.85	5
	Titanium	1	4.00	4.10	0.89	5
	AdobeAir	1	3.88	3.90	0.75	5
D	All	1	11290.00	100.00	177195.30	10M
	PhoneGap	1	6295.00	100.00	143812.30	10M
	Titanium	1	4835.00	100.00	46610.65	1M
	AdobeAir	1	41610.00	500.00	339959.60	10M

for PhoneGap, 82% for Titanium and 78% for AdobeAir. As can be seen in Figure 7, the PhoneGap CPT generates apps with a better AUR score, followed by Titanium and AdobeAir. The results are statistically significant, with the p-value (Mann-Whitney) being 0.00 for all comparisons between CPTs.

***Finding 2**: The PhoneGap CPT results in apps which are better perceived by users. Furthermore, our results indicate that a high number of downloads and reviewers does not necessarily mean that an app is well received by users. This is evident by the results of the AdobeAir CPT, which had the highest number of reviewers but resulted in apps with the lowest AUR.*

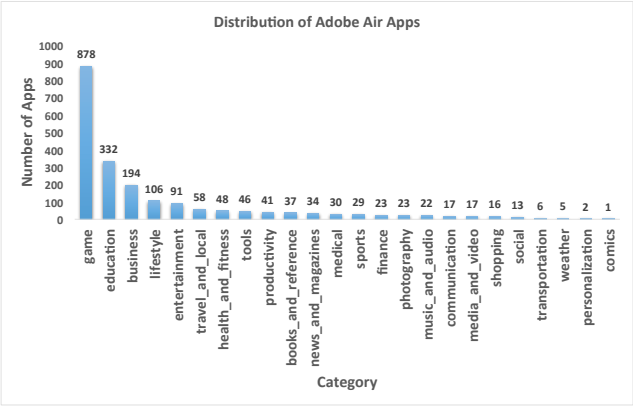

Figure 6: Number of apps in each category created using the Adobe Air CPT.

Table 5: Descriptive statistics for the hybrid apps: AUR

ID	Type	Min	Mean	Median	SD	Max
AUR	All	20.00	81.48	84.00	17.11	100.00
	PhoneGap	20.00	82.75	86.00	17.17	100.00
	Titanium	20.00	80.14	82.00	17.86	100.00
	AdobeAir	20.00	77.67	78.00	15.03	100.00

3.3 Hybrid Vs Native (RQ3)

Figure 8 plots the AUR scores of various app categories for Native and Hybrid Apps. To fairly compare the categories, we used an equal number of apps, distributed equally across the categories. The total number of Hybrid apps was 9948 and an equal number of Native apps was used. When exploring this RQ, we expected the native apps to be better perceived than the hybrid ones due to their advantage in terms of performance and supported native features. To our surprise, the hybrid apps were very close to the native ones in terms of AUR scores and even scored slightly higher in some of the categories. Out of the 25 possible categories for apps, the Hybrid scored higher in 17 of them.

Finding 3: *The hybrid Apps analyzed had AUR scores which matched; and in some categories were higher than the native apps. This indicates that it is possible to create a successful hybrid app that is well perceived by users and that can compete with native variants.*

3.4 AUR Across Platforms (RQ4)

Figure 9 plots the AUR scores of 1400 hybrid app-pairs. The results show that using a CPT to create hybrid *identical* apps for the iOS and Android platform does not necessarily mean it will be equally perceived by users on both platforms. The far ends of the plot show apps that are better perceived on one platform but not the other.

4. THREATS TO VALIDITY

The Hybrid apps detected in our study are a subset of all the possible hybrid apps. Our study uses an existing dataset of 80,000 to identify 15,512 hybrid apps.

In terms of representativeness, our identified hybrid app dataset contains apps from all the categories available on the Google Play store. With respect to generalizability, iTunes

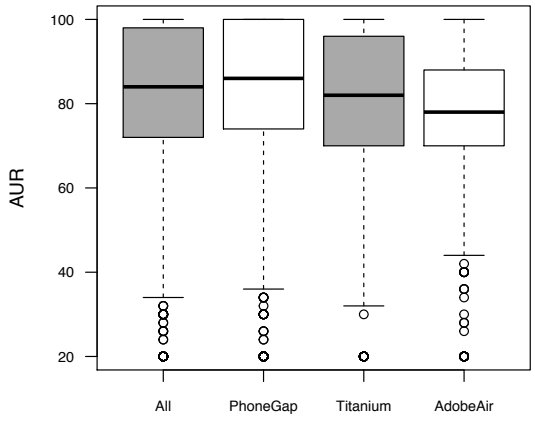

Figure 7: AUR rates for the apps overall and across each CPT.

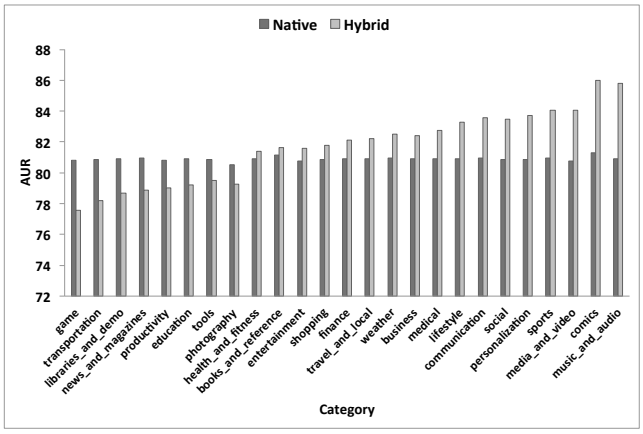

Figure 8: Average AUR for app categories for Native and Hyrbid Apps.

and Google Play are the most popular systems currently, although apps in other app stores could have other characteristics. Regarding replication, all our data is publicly available [17], making the findings of our study reproducible.

5. RELATED WORK

Mobile app stores provide app developers with a new and critical channel to extract user feedback. As a result, many studies have been conducted recently through mining and analysis of app store content such as user-reviews [7, 8, 34, 33, 39, 27, 42, 22, 23, 11, 21, 31, 26, 18, 15, 24], app descriptions [9, 12, 25, 37], and app bytecode [4, 35, 36, 45].

However, all of these works have only considered native apps on one platform. Very little research has been done on cross platform apps [29] [44] and specifically hybrid apps.

Heitkötte *et al.* [16] proposed a set of criteria to assess cross platform development approaches and compare them to the native approach. The authors examine the PhoneGap

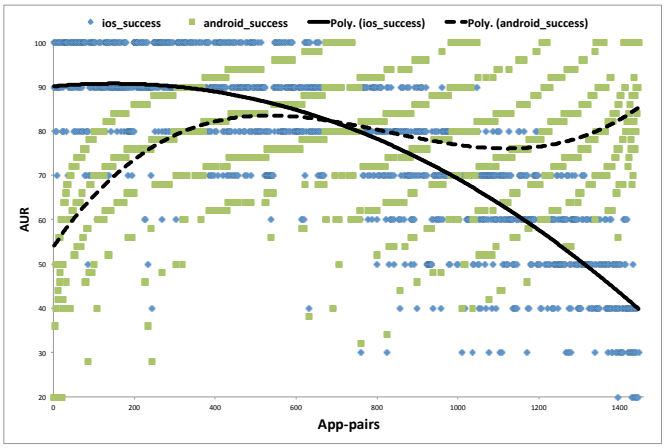

Figure 9: AUR scores for 1400 hybrid app-pairs. Each pair of diamond(iOS) and square(Android) dots represents an app. The solid and dashed lines show the trend of AUR across the apps.

and Titanium CPTs and their findings indicate that Hybrid apps are a viable alternative to native ones.

Palmieri *et al.* [32] compared four very popular CPTs, which are Rhodes, PhoneGap, DragonRad and MoSync. The comparison focused on providing an overview on the availability of application programming interfaces, programming languages, supported mobile operating systems, licenses, and integrated development environments. Their results help developers choose the right CPT for their needs.

Our work, on the other hand, mines hybrid apps and provides an analysis of how they are perceived by users and how they compare to native apps and across multiple platforms. To the best of our knowledge, this is the first work to report a large-scale study on hybrid apps.

6. CONCLUSION & FUTURE WORK

In this paper, we present the first study of hybrid mobile apps. We mined 15,512 hybrid apps, identified the CPTs they used and analyzed their attributes like the number of reviewers, star rating and number of downloads. We indirectly compared the success of the CPTs by measuring how the app is perceived by users. We compared the AUR of hybrid apps to native apps of the same category. Additionally, we compared the AUR of the same hybrid app on the Android and iOS platforms.

For future work, we plan to identify more hybrid apps and expand our dataset. Furthermore collecting the user reviews and using machine learning to classify them can reveal interesting and specific problems to hybrid apps. Finally, it would be valuable to survey the developers of hybrid apps and get their feedback and comments on the subject.

7. REFERENCES

[1] Adobe Inc. https://get.adobe.com/air/.

[2] Appcelerator Inc. http://www.appcelerator.com/.

[3] Adobe Inc. http://phonegap.com/.

[4] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, S. Arzt, S. Rasthofer, and E. Bodden. Mining apps for abnormal usage of sensitive data. In *Proceedings of the 37th International Conference on Software Engineering*, ICSE 2015. ACM, 2015.

[5] G. Bavota, M. Linares-Vasquez, C. Bernal-Cardenas, 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*, 99(PrePrints):1, 2015.

[6] Boris Farber. ClassyShark. https://github.com/google/android-classyshark.

[7] R. Chandy and H. Gu. Identifying spam in the iOS app store. In *Proceedings of the Joint WICOW/AIRWeb Workshop on Web Quality*, WebQuality '12, pages 56–59. ACM, 2012.

[8] 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 *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 767–778. ACM, 2014.

[9] Y. Chen, H. Xu, Y. Zhou, and S. Zhu. Is this app safe for children?: A comparison study of maturity ratings on Android and iOS applications. In *Proceedings of the International Conference on World Wide Web*, WWW '13, pages 201–212. ACM, 2013.

[10] M. A. Figueiredo. Lecture notes on bayesian estimation and classification. *Instituto de Telecomunicacoes-Instituto Superior Tecnico*, page 60, 2004.

[11] L. Galvis Carreno and K. Winbladh. Analysis of user comments: An approach for software requirements evolution. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 582–591. IEEE Computer Society, 2013.

[12] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1025–1035. ACM, 2014.

[13] 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)*, pages 321–330, March 2015.

[14] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162, 2014.

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

[16] H. Heitkötter, S. Hanschke, and T. A. Majchrzak. Evaluating cross-platform development approaches for mobile applications. In *International Conference on Web Information Systems and Technologies*, pages 120–138. Springer, 2012.

[17] Hybrid Apps Study: Toolset and dataset. https://github.com/saltlab/hybrid-apps-study.

[18] C. Iacob, V. Veerappa, and R. Harrison. What are you complaining about?: A study of online reviews of mobile applications. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*, BCS-HCI '13, pages 29:1–29:6, Swinton, UK, UK, 2013. British Computer Society.

[19] IMDB. http://www.imdb.com/.

[20] IMDB. http://www.imdb.com/help/show_leaf?votestopfaq.

[21] H. Khalid. On identifying user complaints of iOS apps. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1474–1476. IEEE Press, 2013.

[22] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan. Prioritizing the devices to test your app on: a case study of Android game apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pages 610–620. ACM, 2014.

[23] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan. What do mobile app users complain about? a study on free iOS apps. *IEEE Software*, 99, 2014.

[24] A. Kumar Maji, K. Hao, S. Sultana, and S. Bagchi. Characterizing failures in mobile oses: A case study with Android and symbian. In *Software Reliability Engineering (ISSRE), IEEE International Symposium on*, pages 249–258. IEEE, Nov 2010.

[25] D. Lavid Ben Lulu and T. Kuflik. Functionality-based clustering using short textual description: Helping users to find apps installed on their mobile device. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 297–306. ACM, 2013.

[26] 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 International Symposium on the Foundations of Software Engineering*, ESEC/FSE 2013, pages 477–487. ACM, 2013.

[27] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang. The app sampling problem for app store mining. In *12th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2015.

[28] Mining iOS and Android mobile app-pairs: Toolset and dataset. https://github.com/saltlab/Minning-App-Stores.

[29] M. Nagappan and E. Shihab. Future trends in software engineering research for mobile apps. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution, and Reengineering, FoSE*, 2016.

[30] Nicolas Viennot. Playdrone. https://github.com/nviennot/playdrone.

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

[32] M. Palmieri, I. Singh, and A. Cicchetti. Comparison of cross-platform mobile development tools. In *Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on*, pages 179–186, Oct 2012.

[33] F. Palomba, M. Linares-Vasquez, G. Bavota, R. Oliveto, M. D. Penta, D. Poshyvanyk, and A. D. Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015.

[34] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015.

[35] I. Ruiz, M. Nagappan, B. Adams, and A. Hassan. Understanding reuse in the Android market. In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, pages 113–122. IEEE, June 2012.

[36] I. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. Hassan. On the relationship between the number of ad libraries in an Android app and its rating. *IEEE Software*, 99, 2014.

[37] S. Seneviratne, A. Seneviratne, M. A. Kaafar, A. Mahanti, and P. Mohapatra. Early detection of spam mobile apps. In *Proceedings of the 24th International Conference on World Wide Web*, WWW, pages 949–959. ACM, 2015.

[38] M. Thelwall, K. Buckley, and G. Paltoglou. Sentiment strength detection for the social web. *J. Am. Soc. Inf. Sci. Technol.*, 63(1):163–173, Jan. 2012.

[39] 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 *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015.

[40] Triggercorp Inc. https://trigger.io/.

[41] N. Viennot, E. Garcia, and J. Nieh. A measurement study of google play. In *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '14, pages 221–233, New York, NY, USA, 2014. ACM.

[42] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach. *CoRR*, abs/1505.04657, 2015.

[43] Xamarin Inc. https://www.xamarin.com/.

[44] S. Xanthopoulos and S. Xinogalos. A comparative analysis of cross-platform development approaches for mobile applications. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 213–220, New York, NY, USA, 2013. ACM.

[45] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck. Appcontext: Differentiating malicious and benign mobile app behaviors using context. In *Proceedings of the 37th International Conference on Software Engineering*, ICSE 2015. ACM, 2015.