Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation

Ivano Malavolta*, Stefano Ruberto*, Tommaso Soru†, Valerio Terragni‡
*Gran Sasso Science Institute, L'Aquila, Italy - {ivano.malavolta,stefano.ruberto}@gssi.infn.it
†University of Leipzig, Leipzig, Germany - tsoru@informatik.uni-leipzig.de
‡The Hong Kong University of Science and Technology, Hong Kong, China - vterragni@cse.ust.hk

***Abstract*—One of the most intriguing challenges in mobile apps development is its fragmentation with respect to mobile platforms (e.g., Android, Apple iOS, Windows Phone). Large companies like IBM and Adobe and a growing community of developers advocate *hybrid mobile apps* development as a possible solution to mobile platforms fragmentation. Hybrid mobile apps are consistent across platforms and built on web standards. How hybrid apps are performing in production settings is still highly debated, with limited objective evidence.**

In this paper, we present the first empirical investigation into mobile hybrid apps involving a large number of mobile apps. Our goal is exploratory and we aim at identifying, analysing, and understanding the traits and distinctions of publicly available hybrid mobile apps within their *real-life context*. The study has been conducted by mining 11,917 free apps and their related metadata from the Google Play Store, and analyzing them from both a technical and end users' perception perspective.

I. Introduction

The mobile apps market now counts more than two millions apps, downloaded billions of times per year from a number of dedicated app stores (with Google Play Store and Apple App Store as clear market dominators [2]). However, code written for one mobile platform (e.g., the Java code of an Android app) cannot be used on another (e.g., the Objective-C code of an Apple iOS app) [1], making the development and maintenance of native apps for multiple platforms one of the major technical challenges affecting the mobile development community [8]. As a possible solution, large companies and many developers are investing resources and effort on the so-called *hybrid mobile apps*. Hybrid mobile apps are developed by using standard web technologies (i.e., HTML, CSS, and JavaScript) and all service requests to the Platform API are mirrored by a cross-platform JavaScript API. In this context, an hybrid development framework (e.g., Apache Cordova) can be defined as a software component that allows developers to create a cross-platform web-based mobile app by providing (i) a native wrapper for containing the web-based code, and (ii) a generic JavaScript API that bridges all the service requests from the web-based code to the corresponding platform API. Despite the obvious advantages of hybrid mobile apps, they also suffer from a number of shortcomings such as restricted access to hardware features, decrease in performance, etc. Although limited in-the-lab studies [9], [6], [10], [3] give some arguments to the strong debate pro and con, as of today existing hybrid mobile apps have not been empirically investigated yet.

As a step forward we present an empirical study about the traits and distinctions of hybrid mobile apps from both a developer's and user's perspectives. The purpose of this work is exploratory: we aim at studying hybrid mobile apps in their natural setting and letting the findings emerge from the observations [12]. The study has been conducted by mining and analysing the binaries of 11,917 free apps from the Google Play Store. By mining apps directly from the Google Play Store, this paper presents the first realistic investigation into hybrid mobile apps through an empirical strategy.

The *main findings* of our study are: (i) further efforts are needed to enhance hybrid mobile apps when dealing with low-level, platform-specific features, (ii) developers of hybrid mobile apps take heavily advantage of code reuse via both desktop and mobile-specific web libraries, (iii) end users value hybrid and native mobile apps similarly.

The rest of the paper is organized as follows. Section II presents the experimental design of our study. Section III discusses the results, whereas the threats to validity of our study are discussed in Section IV. Section V discusses related works and Section VI concludes the paper.

II. Design of the Study

The **research questions** of this study are:

- **RQ1 -** Are hybrid mobile apps published in the Google Play Store?
- **RQ2 -** What are the most used *hybrid development frameworks* for developing hybrid mobile apps?
- **RQ3 -** What are the most used *3rd-party web libraries* for developing hybrid mobile apps?
- **RQ4 -** How are hybrid mobile apps *integrated to the Android platform and other installed apps*?
- **RQ5 -** What is the difference in the *user perceived value* between hybrid and native mobile apps?

The questions aim at identifying and characterizing hybrid development frameworks from a developer's point of view. RQ1 aims at assessing the use and spread of hybrid development frameworks in the Google Play Store. RQ2 to RQ4 have the form of classification questions and are related to what are the most used hybrid development frameworks among top-level mobile apps, how they build on 3rd-party software components, and how they integrate with the Android platform. Based on the reflection that *consumers expect things to just work, and rightfully so* [5], RQ5 considers hybrid

development frameworks from the end users' value perception viewpoint.

The objects of our study are the 11,917 free Android apps from one of the major markets, the Google Play Store. We decided to analyse mobile apps in the Google Play Store because of its large number of available apps having binaries easy to reverse engineer. To identify a reference set of hybrid development frameworks for mobile apps, we considered the publicly available reference list[1].

The **dependent variables** of our study are:

➢ **type** (RQ1, nominal-binary): goal of this variable is to identify whether the mobile app is hybrid or native.

➢ **hybridFramework** (RQ2, nominal): goal of this variable is to identify which hybrid development framework has been used for developing the mobile app.

➢ **libraries** (RQ3, nominal set): set of nominal variables, each of them having a value within the set $\{YES, NO, NOT\_APPLICABLE\}$. The aim of each of these variables is to identify if a specific 3rd-party web library has been used during the development of the hybrid mobile app. Since we cannot know a priori all existing 3rd-party web libraries, the $libraries$ set will be defined in parallel with the data extraction process, similarly to as data extraction is performed in grounded theory approaches [11].

➢ **permissions** (RQ4, nominal-binary set): a set of nominal binary variables, where each of them can have a value in the $\{YES, NO\}$ set. Similarly to 3rd-party web libraries, the variables within the $permissions$ set will be defined in parallel with the data extraction process.

➢ **rating** (RQ5, ratio): this variable is estimated as the average rating provided by the users of the mobile app as coming from the 5-stars ratings in the Google Play Store.

➢ **reviewsCount** (RQ5, ratio): based on the fact that in principles high-quality mobile apps tend to get more reviews in its app lifecycle [4], this variable represents the number of reviews of the mobile app provided by end users.

The **data extraction process** is composed of three main steps (Figure 1).

1. Top-500 identification. We considered the top 500 most popular free apps for each category of the Google Play Store, as of November 23, 2014. By following the guidelines in [12, §10.2], from the 27 categories we exclude the *Widget* and *Live Wallpaper* categories because they are redundant as they are aggregations of apps belonging to other categories. The result of this step is a list of 12,500 app IDs.

2. APKs download. For each app identified in the previous step we downloaded its corresponding APK file. In this step we developed a Java tool for automatically downloading the APK files. The tool is based on an open-source third-party library[2] and on other publicly available third-party repositories of APK files. When obtaining these data, we also downloaded from the Google Play Store website the ratings and reviews counts values of each app. At the time of writing, some of the 12,500 apps identified in the previous step are not available to download and we decided to exclude these apps from the study, resulting in a reduced dataset of 12,468 apps.

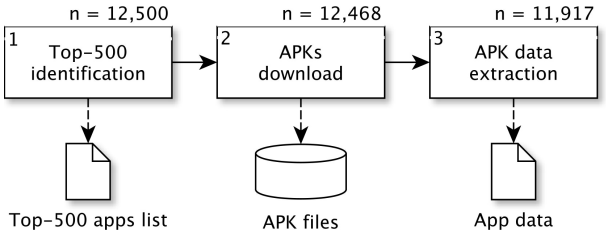

Fig. 1. Data extraction process.

3. APK data extraction. APK file of an Android app contains its binary code, its static resources and its manifest file providing essential information of the app to the Android system. We developed a Java-based tool for automatically extracting relevant information from APK files. The tool is publicly available on GitHub[3] and we are actively maintaining it. The tool is able to automatically get all the needed information for extracting the values of the *type*, *hybridFramework*, $libraries$, and $permissions$, dependent variables More specifically, the *type* and *hybridFramework* are extracted by checking a series of assertions predicating on the resources within the APK file. We defined those assertions by manually inspecting examples of apps we created with each hybrid framework. The $libraries$ variable is extracted by collecting the names of all the JavaScript source files within the APK file, and stemming them with respect to their extension and potential version identifiers. The $permissions$ variable is extracted from the value of the *android:name* attribute of the $< permission >$ tag in the Android manifest of the app. During the analysis, we noticed that 551 APK files have been encoded in a way that reverse engineering them is not possible. Since these cases represent only a small fraction of our dataset and they are scattered through many categories, we could safely exclude them from our study with low impact to its validity. To allow easy replication and verification of our study, we provide to interested researchers a complete replication package[4].

III. Results

Presence in the Google Play Store (RQ1). Overall, we have identified 445 hybrid mobile apps in our dataset, counting for a 3.73%. On one side, this result clearly shows that hybrid mobile apps are significantly uncommon among the top-500 apps within 25 Google Play categories. On the other side, hybrid mobile apps are not completely neglected by top Android developers, this result may be encouraging for the future growth of hybrid development practice. We observe that the categories containing the lower number of hybrid mobile apps (*Photography*, *Music & Audio*, *Tools*, *Game* and *Personalization*) are those that require a closer interaction with the Android platform and hardware. For example, photo-based apps for manipulating photos, music apps for playing

[1] http://en.wikipedia.org/wiki/Multiple_phone_web-based_application_framework

[2] http://code.google.com/p/android-market-api

[3] http://github.com/GabMar/ApkCategoryChecker

[4] Replication package - http://cs.gssi.infn.it/hybrid_googleplay_analysis

songs from the device's music library, tools for launching background and system tasks, games for their performance requirements, personalization apps for customizing the standard menus and features of the device. Because of their cross-platform portability, hybrid development frameworks suffer from the lack of these capabilities, often falling back to platform-specific plugins and add-ons. This results is a clear indicator of a future area of work for developers and vendors of hybrid development frameworks.

Used hybrid development frameworks (RQ2). The most used hybrid development frameworks are Apache Cordova and Appcelerator Titanium counting to 258 and 116 apps, respectively , whereas all the other frameworks are very less used across all categories. This result is in line with informal claims in other research papers [9], [10], [3], and it confirms them empirically. The use of Apache Cordova is regularly distributed across categories, whereas the use of Appcelerator Titanium has a spike in the *Finance* category. By carefully analysing the official documentation and support material of these two frameworks, we conjecture that this result resides in the difference about how they consider security. Indeed, Appcelerator Titanium, differently from Apache Cordova, provides a set of security features that are required by mobile app developers in the financial domain, like the support for enterprise-level authentication and authorization.

Third-party web libraries for hybrid mobile apps (RQ3). Table I shows the top twenty most used third-party libraries in our dataset. In this context, the clear winners are: (i) jQuery, a popular library for manipulating, querying, and interacting with the Document Object Model (DOM), events handling, animations, etc., (ii) jQuery Mobile, a front end library providing a set of UI components specific to mobile apps (e.g., lists of elements, buttons, tab bars, title bars, search fields, icons) and mobile-specific events such as page changes, swipes, etc., and (iii) Json2, a browser-independent library for encoding and decoding JSON objects (mainly used for supporting JSON objects in older browsers).

Interestingly, 16 out of those 20 top used libraries are not specific to mobile browsers, and actually have been initially devised for targeting desktop browsers only. This is a clear indicator of one of the most powerful advantages of developing hybrid mobile apps from a developers' point of view: the possibility to reuse and build on any of the thousands of JavaScript frameworks already existing in the desktop web area today. We also notice that among the top-20 libraries there are two MVC JavaScript frameworks, namely AngularJS and Backbone. The purpose of an MVC framework is to give a well-defined structure to a web application by building it according to the Model-View-Controller design pattern. An MVC framework is generally used when the logic of a JavaScript web application gets larger and difficult to maintain. This is exactly the case of hybrid mobile apps, in which the logic running on the device is defined entirely in JavaScript.

TABLE I
MOST USED THRID-PARTY WEB LIBRARIES (TOP 20)

#	Web library	# apps	#	Web library	# apps
1	jQuery	267	11	Underscore	29
2	jQuery Mobile	106	12	Backbone	27
3	Json2	99	13	Jasmine	27
4	Ionic	58	14	Lo-Dash	21
5	AngularJS	55	15	RequireJS	21
6	Google Analytics	38	16	Bootstrap	20
7	Fastclick	35	17	Mobiscroll	20
8	jQuery UI	32	18	Crypto-js	16
9	Moment.js	32	19	Datejs	15
10	Facebook SDK	30	20	TweenJS	14

Integration to the Android platform and other apps (RQ4). It does not come as a surprise that permissions to access the Internet and the network connection state are dominant here, followed by the permissions to write to an external storage drive (e.g., an SD card), or the ones to access the geographic position of the user. Also, it does not come as a surprise that 17 out of the top 20 permissions are requested to the Android platform, whereas the remaining 3 permissions are requested to Google services. The three permissions requested to Google services are to: (i) register and receive push messages via the Google Cloud Messaging service, (ii) access web-based services like Google Maps, and (iii) access the Google in-app purchases services. By looking at the data, we notice that permissions requested by hybrid mobile apps are generally in line with those requested by native mobile apps.

Perceived value (RQ5). The average of end user *ratings* for both hybrid and native apps is 3.75 and 3.35, respectively. This result does not come as a surprise because, as suggested by Hu et al. [7], end users suffer from purchasing bias, i.e., they are more likely to view their acquired product more positively since they committed the time (and money) to purchase it. More interestingly, hybrid and native mobile apps are performing equally with respect to end users' star-rating across all categories, with neglectable differences. For what concerns *app review counts*, we notice that there is a relevant difference between the number of reviews of native apps with respect to hybrid mobile apps. Indeed, in our dataset, native apps have been reviewed in average 24880 times more than hybrid mobile apps. By following the theory proposed by [7], saying that end users who are reviewing a product are only doing so when they are either incredibly satisfied or dissatisfied, we can interpret this result as an indication of the fact that hybrid mobile apps are neither perceived as too satisfying nor dissatisfying. Together, the two variables we discussed above let us conclude that, using a hybrid development framework or developing an app natively is not a key discriminator with respect to end users' perception of the app.

IV. THREATS TO VALIDITY

Threats to external validity. We reduced this threat by considering a large data set. Furthermore, a random selection of all apps in the Google Play Store is likely to select poor quality apps with a small number of downloads and user reviews. By considering the most popular free apps per category we increased the chance to include the apps with the

best (both current or expected) user base. Indeed, free apps represent 75% of all Google Play Store apps.

Reliability validity threats concern the possibility of replicating this study. We mitigated this possible threat by releasing the replication package, which contains all the data extracted from both the Google Play Store and the APK files.

Threats to conclusion validity concern the relation between the treatment and the outcome. Google Play Store categories have been considered as homogeneous entities during the analysis, however this is not the case. For example, categories like *Libraries & Demo* contain misc apps with many different functionalities. Also, the other categories have an important amount of heterogeneity. In this condition, generalizing considerations to a whole category presents some risk and is not completely appropriate. Further research is needed to find out if the results apply to other app markets and mobile platforms.

V. Related Work

Research studies analysing hybrid mobile apps are emerging only recently. An observational study to provide a guide to choose the right technology for implementing a mobile app is presented in [9]. The resulting decision framework takes into consideration five dimensions: user needs, device features, development technologies, supported platforms, and development approaches.

An experiment on evaluating the execution time and performance overhead between a PhoneGap-based hybrid mobile app with respect to an identical native application is reported in [3]. The results of the benchmark show that in 7 out of 8 cases, the hybrid app implementation was slower than the native one; however, the authors also noted that for general-purpose business applications, this performance issue can be considered as a slight one.

An in-the-lab study about hybrid mobile apps with respect to developers' needs (e.g., used programming language, debugger, extensibility with native code, etc.) and user expectations (mainly focussing on performance issues such as app package size, required RAM, etc.) is presented in [10]. They extracted data on a (non-exhaustive) set of hybrid development frameworks from vendor documentation, and performed an additional subjective analysis of the apps binaries.

Heitkotter et al. evaluated mobile web apps, hybrid apps, and native apps with respect of a set of common criteria [6], such as license and cost, supported platforms, application speed, scalability, etc. The whole study is based on the authors' own research and experiences and on opinions from experienced developers.

In summary, differently from past research, our paper presents the first study for characterizing hybrid apps that (i) has an empirical strategy, (ii) is based on a large dataset comprising 11,917 apps and 3,041,315 user ratings, and (iii) analyses hybrid apps in their actual context of use. Also, our study is one of the first investigations into hybrid mobile apps from both the developers' and end users' viewpoints.

VI. Conclusion and Future Work

Inspired by recent trends in which hybrid apps have been identified as one of the most effective mobile development strategies, this paper satisfies the need for a detailed analysis of existing hybrid apps in their actual context of use (i.e., the Google Play Store). The results of our study show that there is still room for working on hybrid development frameworks, especially for supporting platform-specific features.Hopefully, the above mentioned results will shed light on the current traits and distinctions of hybrid apps today, thus impacting future research, methods, and techniques for developing and managing hybrid mobile apps.

As future work, we are planning to (i) perform a more detailed study on the users' perceptions about hybrid apps. (ii) perform a cross-platform study that compare how a hybrid mobile app perform differently on different platforms (iii) design and conduct a survey targeting hybrid mobile app developers with a focus on practitioners' perceived strengths, limitations and needs associated to existing hybrid development frameworks. Based on the findings of the previous points, we will design new solutions and techniques for enhancing hybrid app development in close contact with vendors of hybrid development frameworks, app store moderators, and independent app developers.

Acknowledgement

The authors would like to thank Gabriele Martini for the development, maintenance, and support of the APK analyzer at the basis of our study. This work is partially supported by the Research Grants Council (GRF 611813) of Hong Kong.

References

[1] IBM Corporation. Native, Web or Hybrid Mobile-app Development. Document Number: WSW14182USEN, 2012.
[2] The U.S. Mobile App Report. *comsCore Whitepaper*, 2014.
[3] L. Corral, A. Sillitti, and G. Succi. Mobile Multiplatform Development: An Experiment for Performance Analysis. *Procedia Computer Science*, 2012.
[4] D. Datta and S. Kajanan. Do App Launch Times Impact their Subsequent Commercial Success? An Analytical Approach. In *CloudCom-Asia*, 2013.
[5] B. Fling. Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps. *OReilly Media Inc.*, 2009.
[6] H. Heitkötter, S. Hanschke, and T. A. Majchrzak. Evaluating Cross-Platform Development Approaches for Mobile Applications. In *WEBIST*, 2013.
[7] N. Hu, J. Zhang, and P. A. Pavlou. Overcoming the J-Shaped Distribution of Product Reviews. *Communications of the ACM*, 2009.
[8] M. E. Joorabchi, A. Mesbah, and P. Kruchten. Real Challenges in Mobile App Development. In *ESEM*, 2013.
[9] E. Masi, G. Cantone, M. Mastroni, G. Calavaro, and P. Subiaco. Mobile Apps Development: A Framework for Technology Decision Making. In *MobiCASE*, 2013.
[10] J. Ohrt and V. Turau. Cross-Platform Development Tools for Smartphone Applications. *Computer*, 2012.
[11] A. Strauss and J. Corbin. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. *SAGE Publications*, 1998.
[12] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in Software Engineering. *Springer Science & Business Media*, 2012.