Mining Android Apps to Predict Market Ratings

Eric Shaw, Alex Shaw, David Umphress
Department of Computer Science and Software Engineering
Auburn University
Auburn, AL USA

***Abstract*— Market rating systems give Android users the opportunity to provide feedback on an application (app). Developers aspire for the highest ratings possible, as they reflect upon user perceptions of their apps. However, no mechanism exists to predict in any way the market rating of an app before publication. We downloaded and reverse-engineered 10,740 apps from the Slide Me market, and analyzed them using quality related metrics. We compared the results of the 1,000 highest rated apps against the lowest rated 1,000. Our results show that traditional white box quality metrics do little to distinguish the groups, while certain Android specific user-perspective metrics are useful in prediction.**

Keywords- Android; market rating; data mining; quality

I. INTRODUCTION

Smart Phones have been wildly successful in recent years. Driving this success is the ability to install custom applications (apps) from a marketplace. Google's official Play Store reached one million applications, with over fifty billion downloads, in 2013 [1]. Google allows the distribution of apps through third party markets as well.

With such a large assortment of apps to choose from in today's marketplaces, users are faced with a difficult task in determining which to install. From this problem stems the community rating system, which allows app users to assign a score based on their respective experiences.

The developer views the rating as an assessment of his application [2]. It follows that he wants to maximize his score. Development and feedback is a time-consuming cycle. In addition, it is often difficult to extract useful information from reviews. As such, a mechanism to predict application ratings would be of value.

II. PREVIOUS WORK

In [2], the authors present a tool, WisCom, used to download and analyze user comments from 171,749 Google Play store apps, summarizing the data into an easily digestible format. The program also determines inconsistencies between user comments and ratings. AR-Miner [3] builds upon the work presented in WisCom. AR-Miner analyzes app store reviews and presents a ranked list of those that are most relevant.

Though some studies have been performed on the mining of market rating information, none of these have involved code analysis. We found no work that attempted to predict market rating based on quality metrics. Therefore we concluded that such a study would be of value.

III. METHODOLOGY

Our goal was to obtain a sample of free apps from an Android marketplace, reverse engineer them, and examine code features that might correlate with app quality. Our study involves the collection, unpacking, and mining of APK (Android Package) files.

The logical first step was to determine which market best fit our needs. We originally wanted to use Google Play for our study, as it is the most popular and has a widely used rating system. However, we found a major roadblock in doing so. A user may not download an APK file directly from Play to a PC. We turned our attention to third party markets, landing on Slide Me, which has a widely used rating system.

We then downloaded 10,740 APK files from the Slide Me market, and reverse-engineered them using apktool [4]. This produced Dalvik byte-code, along with resources and the manifest file. Next we developed a set of metrics with to measure app quality from an APK file.

A. Size Metrics

These metrics characterize the size of a software system in terms of amount of code. Collecting data such as this can be useful in controlling quality attributes as well as estimation.

- Number of Instructions. This is the byte-code equivalent of the lines of code (LOC) metric.
- Number of Methods. The app's total number of methods.
- Number of Classes. The app's total number of classes.
- Number of Methods per Class (MPC). Total number of classes divided by the total number of methods.
- Number of Instructions per Method (IPM). Total number of byte-code instructions divided by total number of methods.
- Cyclomatic Complexity. This metric measures the complexity of a piece of code.

B. Object Oriented Metrics

These metrics, introduced in [5], are intended to measure adherence to object oriented design principles.

MobiCASE 2014, November 06-07, Austin, United States
Copyright © 2014 ICST
DOI 10.4108/icst.mobicase.2014.257773

- Number of Children (NOC). The number of classes extending a particular class.
- Depth of Inheritance Tree (DIT). A measure of the number of levels in a given inheritance tree.
- Lack of Cohesion of Methods (LCOM). This metric measures the how related methods within a particular class are.
- Coupling Between Objects (CBO). The average number of objects that reference other objects.
- Percent Public Instance Variables (PPIV). The percent public instance variables across all classes in each app.
- Access to Public Data (APD). The number of accesses to public and protected fields within a class's methods.

C. Android Specific Metrics

Unlike the previous metrics, these come from attributes that directly influence the user's experience, and are specific to the Android platform.

- Unchecked Bundles. The getExtras() method of the `Activity` class may return null. The return value should be checked to avoid a `NullPointerException`.
- APK File Size. In addition to using more memory on device, large APK files take longer to download.
- Number of String Resources. The developer may declare strings by name in a strings.xml resource file.
- Bad Smell Method Calls. In [6], the authors identify a set of methods that commonly lead to crashes. We identified unchecked calls to these methods. Those we include are show(), setContentView(), onKeyDown(), and onBackPressed().

In order to meet our research goal of determining if any metrics could be used to predict app quality, we divided our apps into two groups: the 1,000 highest rated and the 1,000 lowest rated. The lowest had ratings from 0.5 to 2.0, whereas the highest were all rated at 5.0 (there were more than 1,000 apps rated at 5.0 – we selected 1,000 randomly). We compared the average values for each of our metrics between the groups.

IV. Results

Our goal is to quantify whether a given metric can be used to predict quality. We define a Difference Index (DI), a decimal number that is defined for each metric *m* as follows:

$$DI_m = \left| \frac{AVG_{Top}}{AVG_{Bottom}} - 1 \right| * 100. \quad (1)$$

A high DI indicates that there is a large disparity between the average for the top and bottom apps. A high DI therefore indicates that a specific metric is related to user rating. Table 1 shows the metrics values and DI for each metric.

TABLE I. Results

Metric	Top 1,000	Bottom 1,000	DI
Nº instructions	9093.44	8969.58	1.38
Nº methods	340.55	352.88	3.5
Nº classes	64.45	65.00	0.84
MPC	4.84	5.25	7.89
IPM	28.67	28.75	0.25
Cyclomatic complexity	6.69	6.36	5.17
NOC	1.05	0.79	32.23
DIT	1.40	1.37	2.15
LCOM	46.79	47.37	1.22
CBO	3.25	3.36	3.10
PPIV	17.40	19.07	8.75
APD	6.02	6.24	3.54
APK File Size	8.85	15.94	47.62
Nº strings	66.03	46.48	42.08
show()	9.14	10.11	9.57
setContentView()	0.62	2.97	79.24
onKeyDown()	0.001	0.03	96.25
onBackPressed()	0.001	0.01	81.13

V. Conclusion

Our study clearly demonstrates that traditional quality metrics are not strongly correlated with app ratings. However, we have introduced a number of Android-specific quality metrics that are predictors of app ratings. Thus we conclude that these metrics effective in assessing Android application quality.

References

[1] C. Welch, "Google: Android app downloads have crossed 50 billion, over 1M apps in Play", 2013, [Online] Available: http://www.theverge.com/2013/7/24/4553010/google-50-billion-android-app-downloads-1m-apps-available

[2] B. Fu, J. Lin, L. Li, C, Faloutsos, J. Hong, N. Sadeh: "Why people hate your app: Making sense of user feedback in a mobile app store," in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 1276–1284.

[3] 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. ACM, New York, NY, 2014, pp. 767-778.

[4] Winsniewski, R.: Android, "Apktool: a tool for reverse engineering Android apk files," 2012, [Online] Available: http://code.google.com/p/android-apktool/

[5] S. R. Chidamber, C. F. Kemerer: "A metrics suite for object oriented design," IEEE Trans. Softw. Eng., vol. 20, no. 6, 1994, pp. 476-493.

[6] M. Kechagia, D. Spinellis, "Undocumented and unchecked: exceptions that spell trouble," in Proceedings of the 11th ACM Working Conference on Mining Software Repositories, New York, NY, 2014, pp. 312-315.