An Examination of the Current Rating System used in Mobile App Stores

Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst and Ahmed E. Hassan

Abstract Unlike products on Amazon, mobile apps are continuously evolving with new versions of apps in the app store replacing the old versions at a rapid pace. Nevertheless, many app stores still use the Amazon-style rating system for their hosted apps, where every rating assigned to an app over its entire life time is aggregated into one rating that is displayed in the app-store (which we call store-rating). In order to examine if the store-rating of an app is able to capture the changing user satisfaction levels with respect to new versions of the app, we mined the store-ratings of over 10,000 unique mobile apps in the Google Play market, every single day for an entire year. We find that many apps do increase their version-to-version rating, while the store-rating of an app is resilient to fluctuations once an app has gathered a substantial number of raters. Therefore, we conclude that the current store-rating of apps is not dynamic enough to capture the changing user satisfaction levels associated with the evolving nature of apps. This resilience is a major problem that can discourage developers from improving the quality of their app.

Keywords Mobile apps · Android · Google Play · review systems · rating

1 Introduction

Mobile-apps (apps) are applications running on mobile devices such as smartphones or tablets. Downloads of mobile apps across all platforms (e.g., iOS, Blackberry, and Android) are expected to cross 300 Billion by 2016, and the revenues are expected to grow to $74 billion by 2017 (Louis Columbus, 2013). The mobile app industry is centered around the concept of app-stores, such as the Google Play app-store[1] and Apple's App Store[2], which act as distribution platforms similar to Amazon.com for books[3] by centralizing sales of thousands of mobile apps.

Similar to buying books from Amazon.com, users can download apps from the app-stores, and also rate their experience in using the app with a scale of 1 up to 5 stars. Studies in other types of online markets such as online bookstores (e.g. Amazon.com), have analyzed the influence of such ratings on a customer's decision to acquire a product (Chevalier and Mayzlin, 2006; Zacharia et al, 1999). Similar to online bookstores, in app-stores, users of apps are influenced (among other factors) when deciding to acquire an app, by the rating of an app. Recent research shows

[1] Google Play app-store: play.google.com/store/apps
[2] Apple's App Store: www.apple.com/iphone/apps-for-iphone
[3] Amazon: www.amazon.com

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

that such ratings have a high correlation with download counts, a key measure of success for a mobile app (Harman et al, 2012).

However, there is a key difference between mobile apps and books (or other products that have a similar rating system): an app can be updated to a new version in a very short time period (e.g., apps updated on a weekly basis are not uncommon in the Google Play market), whereas books or other products take more time and are often released as a new product, and not just as a new version of an old product. Despite this difference, app-stores still use the same static rating system to help new users differentiate the apps that have high or low satisfaction levels among current users.

The ability to update apps, therefore, makes the current rating system insufficient. Four of the five most popular app-stores (Google Play, Amazon Appstore, BlackBerry Appworld, and Microsoft Marketplace), report only a cumulative average, which is calculated by aggregating the user ratings from all versions of the app taken together. Henceforth, in this paper we will call the cumulative average as 'Store-rating of an app', since this is the rating that is displayed in the app stores for each app. Due to the aggregation present in the store-rating of an app, the rating assigned to a particular version of the app is unavailable to the end user. Therefore, an end user could be downloading a version of the app that can be worse or better than what the store- rating might imply.

Therefore, in this paper we study the rating system used by one such app-store (Google Play) by mining the store-ratings of free-to-download mobile apps in the store during 2011 (10K+ unique apps in total), to highlight the issues with the store-rating of an app. We find that, store-ratings are very resilient - Once a substantial number of users have rated an app, the app's store-rating is resilient to fluctuations since it is based on the average over all the ratings in its entire lifetime. Hence, due to this fact, for apps with a large number of raters, we find that it is very difficult to move their store-rating (i.e., the displayed rating) up or down. Therefore, even after releasing an improved version of an app, it might be very difficult for an app to have a good store-rating if it has been given a poor rating by a substantial number of users before, and vice-versa.

The resilience of store-ratings is a major problem that can discourage developers from improving the app since the rating system cannot factor in recent improvements into the store-ratings. This resilience might also make developers less careful since botched releases will not have a major impact on their store-rating. On the other hand, this resilience might also encourage developers of well-established apps to experiment since any user discontent can be absorbed by their already high store-rating.

Overall, our study highlights the need for a careful re-thinking of the rating system used today by app-stores.

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

1.1 Paper Organization

The rest of the paper is organized as follows. Section 2 presents the data used in our study and some background information on the store-rating of apps. Section 3 presents the results of our study. Then, Section 4 outlines the threats to validity, and Section 5 discusses related work. Finally, Section 6 concludes the paper.

2 Background and Data Sources

To set the stage of our case study, we describe the rating mechanism used in in our case study, and our sources of data.

2.1 Mobile Apps and Ratings

App-stores commonly employ simple rating mechanisms. In at least five of the most popular and largest app-stores, including Apple's App Store, Google Play, Windows Phone Store, Amazon Appstore, and Blackberry Appworld, the store-rating of an app is represented as a number of "stars" from 1 up to 5, aggregated from individual user ratings.

In our case study subject Google Play, the store-rating of an app is the cumulative average of all individual user ratings given to the app over all the versions (i.e., the entire life time of the app in the app store). For example, if two users gave 3 stars to version 1 of an app, and five users give 4 stars to version 2, then the app's store-rating is $2*3+5*4 = 3.7$. Four of the five largest app stores (Apple's App Store being the 2+5 exception) only display the aggregated average as the store-rating for each app.

2.2 Data Source

Our study analyzes ratings of the free apps[4] that were available in Google Play throughout 2011. We chose Google Play as it is—next to Apple's App Store—one of the two largest app-stores in existence today, and we were able to extract the store-rating and the number of people who have rated an app for every version of an app. We believe that our selection is representative, and that our results also apply to other app-stores since they follow a similar rating system.

The dataset was extracted previously by two of the authors of this paper by crawling Google Play throughout 2011. This resulted in 242,089 app versions of 131,649 mobile apps. See a technical note (Dienst and Berger, 2012) for further details.

[4] More precisely, the free-to-download apps, since apps can require in-app purchases (freemium model).

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Since Google can adjust ratings in case of fraud[5], or in order to assign a rating to unrated apps[6], we had to sanitize our data to eliminate anomalies in ratings, such as the cases where the total number of raters in a later version was less than the total number of raters in an earlier version. 3,891 versions of 3,454 apps had this issue. After filtering out apps with anomalous ratings, we ended up with 238,198 versions of 128,195 apps.

The store-rating of apps with few raters are not reliable, since the developers and their friends could have rated the app highly. Such apps could be very good apps, but we cannot be sure of it. (Whitby et al, 2004) present a statistical filtering technique to exclude unfair ratings of each individual user. We also observed such a rater bias phenomenon, and we performed filtering on our dataset in order to obtain a more representative dataset for our study. However, our filtering is different from (Whitby et al, 2004), since we do not filter any particular user's ratings. Instead, we filter apps that have less than 10 raters.

After filtering apps with fewer than 10 raters, we also filter apps with just one app version in our dataset, since in order to calculate the rating of a specific version, we need to have the store-rating and the number of raters for the previous version as well (the exact formula for calculating the rating of a specific version is presented in Section 3.2). Hence, the set of apps that we use (32,596 app versions across 10,150 apps) have at least 10 raters and at least two app versions each.

On every day in 2011, for each of the 10,150 apps in our dataset, we mined the store-rating displayed in the apps store (which is on a scale of 1 star to 5 stars). We also mined the number of people that had rated the app till that day. We did not mine the reviews for the apps, which are a subset of the people who have rated the app (a review is not required to rate an app, but a rating is required to review an app). Only when a review is submitted can we get other meta data like author name, device name etc. However, we downloaded the app binary everyday. When a new version of the app binary was available on a particular day, we knew that a new version of the app was released on that day. Using this information, we were able to reconstruct the version history of an app and how the ratings evolved from version to version. In the Google Play store, only store-ratings are available and no version-ratings are available. Hence, if we want to know the version-rating, we have to manually reconstruct it from the version history. More details on how the version-ratings are calculated is in subsection 3.2. We have made the data available in order to make our study replicable[7].

[5] http://support.google.com/googleplay/bin/answer.py?answer=113417
[6] Google Play Developer Distribution Agreement: http://play.google.com/about/developer-distribution- agreement.html
[7] http://sailhome.cs.queensu.ca/replication/2014/Ratings_IEEE_SW/ Ratings_Data.csv

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

3 How responsive is the store-rating to new versions of an app?

3.1 Motivation

The displayed store-rating of an app is an important indicator of its user-perceived satisfaction. It is essential for such a rating to be representative of the evolving nature of apps, in contrast to books, which do not undergo much change (often no change) after they are released. In this research question, we examine whether we would no- tice a change in the store-rating of an app, given a rise or a drop in the rating of a specific version of that app.

3.2 Approach for Calculating Store and Version-Rating

As explained in section 1, each app has a rating that is displayed in its page on the app store, which we call the store-rating. The store-rating of an app is calculated by the app stores (and not by us) as follows:

$$store_i = \frac{sum\ of\ all\ the\ individual\ ratings\ assigned\ to\ the\ app\ until\ version\ i}{nr\_ratings\_until\_i} \quad (1)$$

where $store_i$ is the store-rating of the app recorded at version i, and nr_ratings_until_i is the total number of ratings across all versions up until version i. Essentially, $store_i$ is the average rating of an app overall, from the first version until the current version.

A key metric in addition to the store-rating and the number of raters (which are directly gathered from Google Play (Google, 2013)), is the rating assigned to a specific version of an app, which we call version-rating. Since Google Play does not offer this information—if it is recorded at all—we reconstruct the version-rating of the i^{th} version of an app as follows:

$$version{-}rating_i = \frac{store_i * nr\_ratings\_until\_i - store_{i-1} * nr\_ratings\_until\_i - 1}{nr\_ratings\_until\_i - nr\_ratings\_until\_i - 1} \quad (2)$$

The numerator corresponds to the total number of stars awarded to version i by all of version i's raters, while the denominator corresponds to the number of ratings of version i. Our analysis focuses on apps with at least two (when studying version-rating) or three versions (when studying increases in version-rating), since we cannot calculate the version-rating metric for the first fetched version. We calculate the version-rating for each app version using Equation 2 above.

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

3.3 Case Study Results

We use hexbin plots to examine our research question. A hexbin plot adds an additional dimension to the regular scatterplot. Besides the relationship between the x and y-axis measures, the hexbin plot captures the frequency of each point in the scatterplot, by dividing the plot in different hexagons, each of which gets a color rep- resenting the number of observations falling within it. In our case, we plot the change (increase/decrease) in the store-rating of an app from one version to another on the x- axis and the change in the version-rating of the corresponding app from the same two versions on the y-axis. The darkness of a hexagon in the hexbin plot indicates the frequency of apps that have a particular combination of version-rating and store-rating. Figure 1 plots for each successive pair of app versions the increase in version-rating versus the increase in store-rating.

We can observe three phenomena in the hexbin plot of Figure 1. First, the majority of rating changes float around 0 for both version- and store-rating (dark black cells), i.e., this is the case where no rating change happens. Second, we see a long vertical stretch of version delta ratings (between -3 and 3), with only a short horizontal stretch of store-rating deltas (between -1 and 1). Thus, changes in version rating did not have any corresponding change in the corresponding store-rating of the app, confirming that the store-rating indeed dampens the effect of fluctuations in quality. Third, we observe a slight diagonal pattern from the bottom left to the upper right showing that changes in version-rating can have a corresponding, but dampened (across a smaller range of changes) change in store-rating of the app.

To better understand the dampening effect, we also calculate the percentage of app versions in which a change in version-rating results in a visible change in store-rating. “Visible” here means that the store-rating changed more than half a star (since app stores typically visualize ratings in terms of stars). In 78.8% to 97.4% of all app versions, version-rating star changes do not result in a change in store-rating stars. 11.2%, 21.0% and 14.3% of app versions that lose 1, 2 or 3 version-rating stars respectively, only lose 1 store-rating star, while 9.3%, 20.0% and 14.3% of apps that won 1, 2 or 3 version-rating stars respectively, only won 1 store-rating star.

The store-rating is very resilient to changes in the version-rating.

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

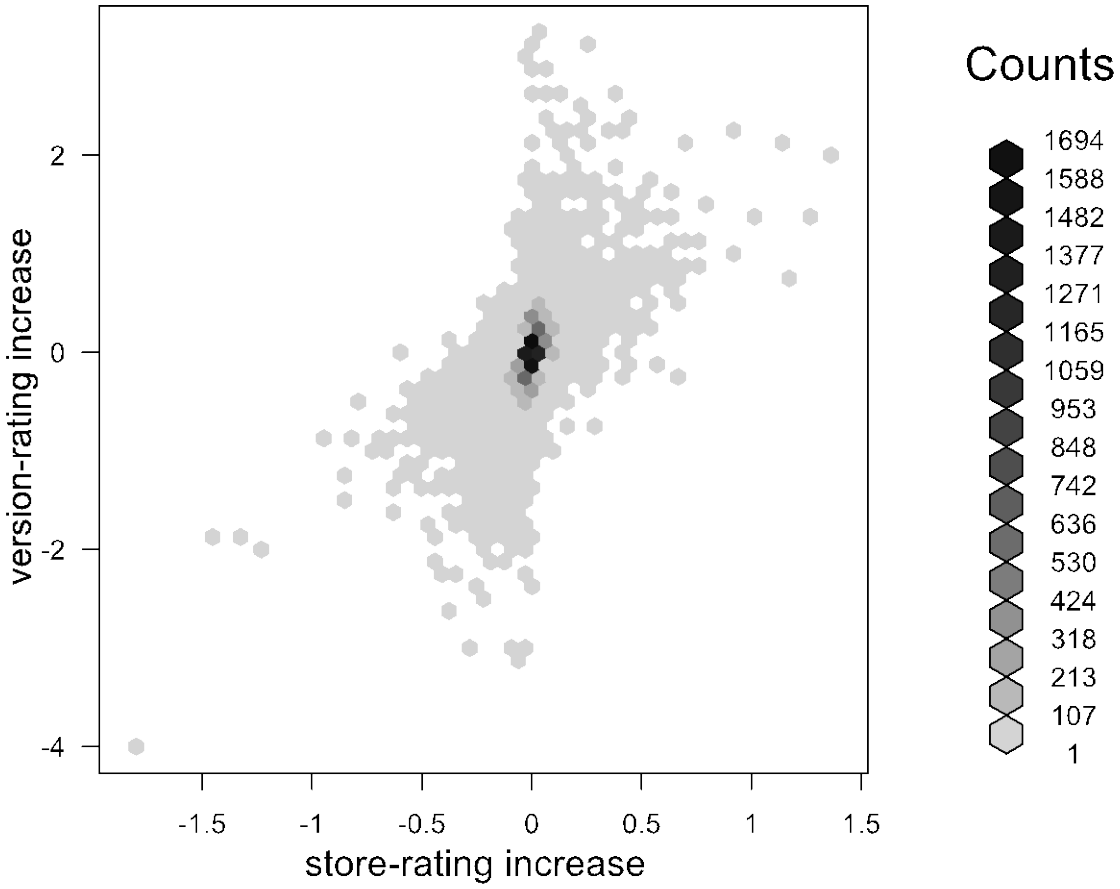


Fig. 1 Scatterplot of increase in version-rating for each successive pair of all versions of the apps with at least two versions in 2011 and 10 raters per version versus the corresponding increase in store-rating of the app.

3.4 Discussion: Why are store-ratings resilient to change?

For each app in our filtered dataset, we determine the change in store-rating between their first and last version in 2011. We also determine the total number of people who had rated each app until the first version in 2011. Then we plot a scatterplot in Figure 2 with the change in rating on the y-axis and the initial number of raters on the x-axis.

Given that the X axis is in logarithmic scale, the plot clearly shows that the more raters an app has at the beginning of 2011, the more resilient the app will be to changes in its store-rating, even after an entire year of user ratings are accumulated. For example, the variance and standard deviation for the change in store-rating among apps that have fewer than 5000 raters to begin with, is 28 times and 5 times higher than apps with more than 5000 raters respectively. Therefore, this implies that developers of an app with a high store-rating and high number of raters have

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

some leeway to experiment in a new version without their store-rating being

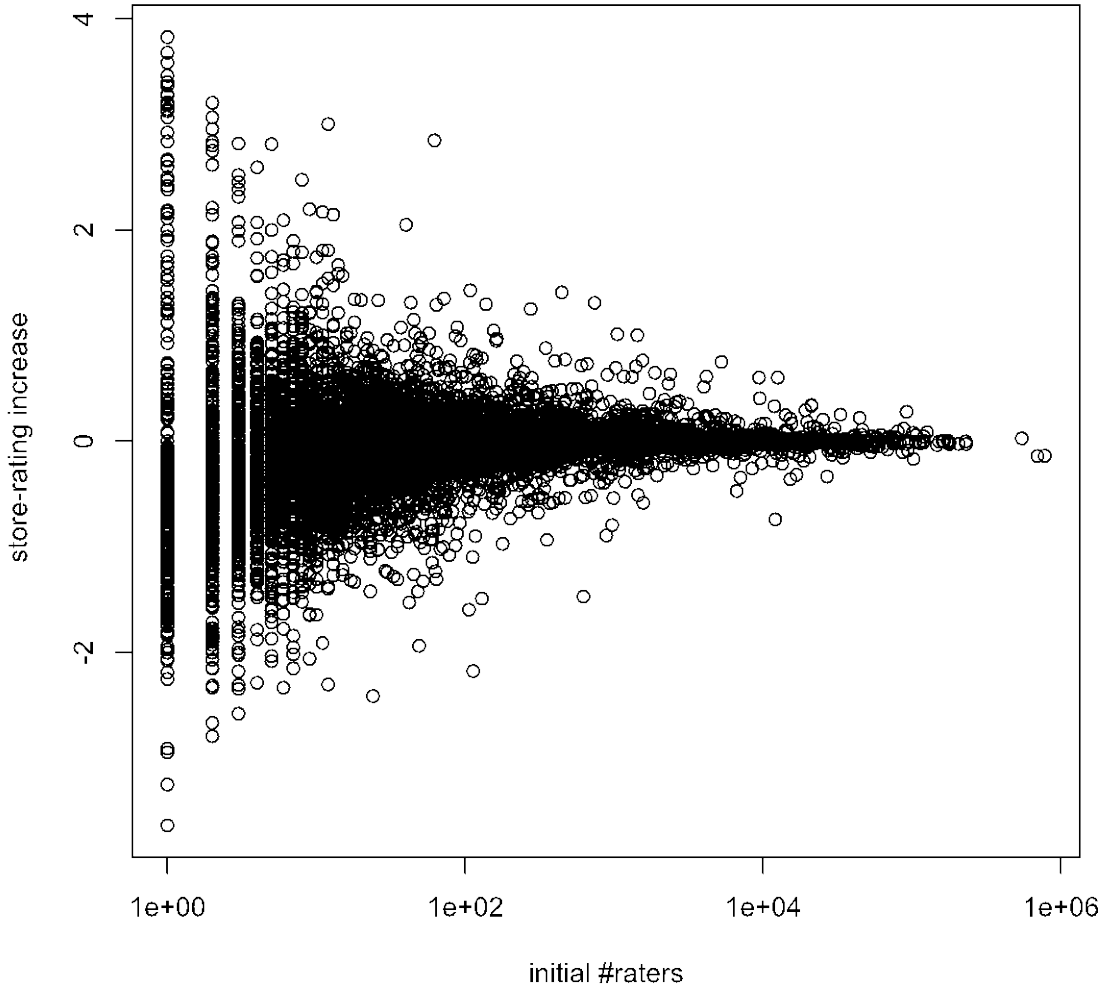


Fig. 2 Scatterplot of store-rating increase between the first and last version of apps with at least two versions in 2011 versus the total number of raters up until the first version in 2011.

affected negatively, while developers of apps with a poor store-rating yet high number of raters will find it nearly impossible to improve their store-rating.

We looked at six more months of rating data from KakaoTalk (i.e., a total period from January 2011 to June 2012), which was one of the apps with the largest number of ratings, to see how its store- and version-ratings have fluctuated. KakaoTalk had 229,869 raters by Jan 2011, and 936,497 raters by June 2012. It stayed at a store- rating of 4.5. However, the version-rating dropped once by 1.54 and increased once by 1.44 in those six months. KakaoTalk is an example of an app with highly fluctuating version-ratings that do not impact its very stable store-rating.

These observations suggest that the real quality of an app does not really matter once a massive number of users has rated the app favourably, since the displayed store-rating remains the same. While this displayed store-rating might be resilient,

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

the version-rating of a specific app version provides a more instantaneous and hence accurate view of the perceived quality of a particular app.

Store-ratings are resilient to change once a substantial number of people have rated the app, even though the version-rating might fluctuate heavily.

4 **Threats to Validity**

4.1 External Validity

We studied the Android platform with data spanning 2011 since we had readily avail- able APIs to mine apps from the Google Play store. We chose to study only the free apps because we required access to the source code or bytecode, which is not possible for paid apps without actually paying. Since the Android platform has the largest user base (Kellogg, 2011), it has the largest number of downloads (Pettey and van der Meulen, 2012), and free apps represent 75% of the total number of apps in the Google Play app-store (AppBrain, 2013). Additionally, we only look at data from 2011. However, the rating system in the Google Play and various other app stores are still the same. Hence, we believe that our case study space is broad and practical enough to draw valid conclusions.

4.2 Internal Validity

We used publicly available APIs to mine the Google Play app-store. Our crawling tools collected the data from the store automatically. Although we have taken every possible step to ensure correct working of our tools, we could not collect every app in the market every day, since we are restricted by the number of queries that can be run against the app-store. This means that we may have missed particular versions of an app. However, since these crawling APIs are used extensively by others too, and since our frequency of data collection is higher than the frequency of new releases of an app, we believe that our results do not suffer significantly from such threats. The last author is NSERC/RIM Industrial Research Chair in Software Engineering of Ultra-Large-Scale (ULS). Even though BlackBerry is a competitor of Android, we hope to have controlled for this threat, since all analyses were interpreted and validated jointly by all authors.

4.3 Construct Validity

In the present Google Play app-store, it is not clear when app-users rate an app. App-users may rate an app at any time independently of the current version released on Google Play. Hence, the version-rating that we calculate may not be perfectly accurate. However, the version-rating is not inherently present in any of the app-stores (except the Apple App Store), and since we downloaded the apps at

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

frequent intervals, we believe that we captured the version-rating as accurately as possible. Finally, only a small number of people might rate a particular app, yielding a relatively unreliable rating. Since we filtered out apps with a low number of raters, the effect of anomalous raters has been dealt with.

5 Related Work

We consider that our work is related to review systems. Hence, we present work related to review systems, and discuss the differences with our work.

(Galvis Carreño and Winbladh, 2013; Iacob and Harrison, 2013) have each analysed the textual content present in the user reviews of mobile apps for the purpose of requirements elicitation. In our study we examine the ratings assigned to the app by the users and not the textual content of the reviews. The goal of our study is also different - 'Is the current ratings system in the Google Play store appropriate?'.

Amazon.com has also been the subject of several studies about its review sys- tem. (Chevalier and Mayzlin, 2006) studied behavioral patterns of reviewers on the two online booksellers Amazon.com and Barnesandnoble.com. They found that positive reviews on a book result in the increase of sales of that book.

(Wang et al, 2008) proposed a system to improve Amazon's review system adding two factors: review credibility and time decay. (Mui et al, 2001) proposed a Bayesian probabilistic technique to weight a rating based on the rater's personalization. (Cui et al, 2010) studied the effect of early electronic word-of-mouth (WOM) in Amazon from data collected during a period of nine months. Surprisingly, they found that negative reviews have a strong positive effect on the sales of new products (i.e., negative reviews help increase the sales) as long as most reviews are positive. The negative review's effect however, decreases with time. They posit that once a substantial number of reviews are in (critical mass of adopters reached), it will positively affect sales. Thus the early reviews are critical for a product's success. We find a similar trend in our study too. If the initial ratings of an app are poor and has been rated by a substantial number of raters, then it is very difficult for the store-rating to ever recover.

Our work differs from the aforementioned works mainly in the following aspect: we use a different object of study, the Google Play app-store, where the type of product is software (apps), which changes frequently over time (and can be updated easily) compared to books and other products. Additionally, newer versions of books and other products get released as a new product and not an updated version of the same product.

6 Conclusion and Recommendations

This study examined 32,596 app versions across 10,150 apps from the Google Play store collected over a time span of one year in order to analyze the dynamics of

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

store-ratings of mobile apps that are displayed next to the app in the store. Since such ratings are one of the primary means for apps to attract users and generate revenues, we studied the impact of using the store-rating for an app, and how the ratings of apps vary from one version to another.

We find that due to the cumulative nature of the current store-rating, it is very difficult to climb up from an initial poor rating after a large number of raters have rated an app. From this, we conclude that the system of store-rating for apps is insufficient.

Therefore, our recommendations are:

App-store-owners

1. Should display both the current store-rating as well as version-rating (for at least the current version, which the Apple App Store currently practices). Otherwise, app-developers have no incentive to maintain or improve the rating (perceived quality) of the app. Therefore the app-users risk not having access to the best quality apps.
2. Moreadvancedmethodstogeneratearatingshouldbeexplored,oneoptionbeing a rating system that exponentially decays the older ratings – hence putting more emphasis on recent version-ratings over much older (and most likely outdated) version-ratings.

App-developers

1. If the practice of only displaying the current store-ratings is followed, then there is limited benefit from releasing a new and improved version of a low-rated app. Instead, we recommend that the app-developers release the new version of an app that has a poor store-rating as a new app, and redirect their current user base to the new app.
2. The release strategy of "Release early, Fix later" is not an optimal strategy for an initial or early app version with respect to the current system of store-ratings. An initial version of low quality could get enough users to rate the app poorly, and make it difficult to bring the store-rating back up to 4 stars or more. This underlines the importance of prioritization of requirements as well as sufficient quality assurance of (at least) early versions of a mobile app.

References

AppBrain (2013) Comparison of free and paid android apps. URL http://www.appbrain.com/stats/free-and-paid-android- applications

Chevalier JA, Mayzlin D (2006) The effect of word of mouth on sales: Online book reviews. Journal of Marketing Research 43(3):345–354

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Cui G, kwong Lui H, Guo X (2010) Online reviews as a driver of new product sales. In: Management of e-Commerce and e-Government (ICMeCG), 2010 Fourth International Conference on, pp 20 –25

Dienst S, Berger T (2012) Static analysis of app dependencies in android bytecode. Tech. Note, available at http://www.informatik.uni-leipzig.de/ ~berger/tr/2012-dienst.pdf

Galvis Carreño LV, Winbladh K (2013) Analysis of user comments: An approach for software requirements evolution. In: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pp 582–591

Google (2013) Ratings and comments. URL http://support.google.com/ googleplay/android-developer/bin/answer.py?answer= 138230

Harman M, Jia Y, Zhang Y (2012) App store mining and analysis: MSR for app stores. In: Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on, MSR'12

Iacob C, Harrison R (2013) Retrieving and analyzing mobile apps feature requests from online reviews. In: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, pp 41–44

Kellogg D (2011) 40 percent of u.s. mobile users own smartphones; 40 per- cent are android. URL http://blog.nielsen.com/nielsenwire/ online_mobile/40-percent-of-u-s-mobile-users-own- smartphones-40-percent-are-android/

Louis Columbus (2013) Roundup of mobile apps and app store forecasts, 2013. URL http://www.forbes.com/sites/louiscolumbus/2013/ 06/09/roundup-of-mobile-apps-app-store-forecasts-2013/

Mui L, Mohtashemi M, Ang C, Szolovits P, Halberstadt A (2001) Ratings in distributed systems: A bayesian approach. In: Workshop on Information Technologies and Systems

Pettey C, van der Meulen R (2012) Gartner says free apps will account for nearly 90 percent of total mobile app store downloads in 2012. URL http: //www.gartner.com/it/page.jsp?id=2153215

Wang BC, Zhu WY, Chen LJ (2008) Improving the amazon review system by exploiting the credibility and time-decay of public reviews. In: Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on, vol 3, pp 123 –126

Whitby A, Josang A, Indulska J (2004) Filtering out unfair ratings in bayesian reputation systems. In: Proceedings of the 7th Int Workshop on Trust in Agent Societies

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Zacharia G, Moukas A, Maes P (1999) Collaborative reputation mechanisms in electronic marketplaces. In: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8 - Volume 8, IEEE Computer Society, Washington, DC, USA, HICSS '99, pp 8026–

Tweets:

1. Cumulative nature of the current store-rating prevents apps from improving their initial poor rating.
2. App-devs could benefit by releasing new version of an app that has a poor store-rating as a new app due to current store-rating system.
3. "Release early, Fix later" is not an optimal strategy for an early app version with respect to the current system of store-ratings.

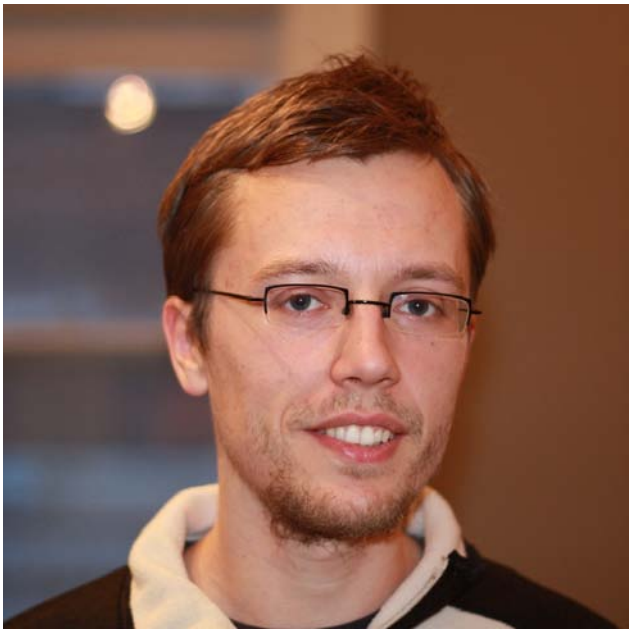
Author Bios:

Israel J. Mojica *is a software engineer at McAfee. His research interests include mobile software and empirical software analysis. Mojica has an MS in computer science from Queen's University. Contact him at Israel_Mojica@McAfee.com.*

Meiyappan Nagappan *is an Assistant professor at the Software Engineering Department of Rochester Institute of Technology. His broad research interests include deriving solutions for software system stakeholders and using large-scale software engineering data to address the concerns of software developers, software operators, build engineers, and project managers. More specifically he focuses on mining mobile app data to provide the various stakeholders of mobile apps with actionable recommendations. Nagappan has a PhD in computer science from North Carolina State University. Contact him at mei@se.rit.edu.*

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Bram Adams *is an assistant professor at the École Polytechnique de Montréal, where he heads the Maintenance, Construction, and Intelligence of Software (MCIS) lab. His research interests include software release engineering, software integration, software build systems, software modularity, and software maintenance. Adams has a PhD in computer science engineering from Ghent University. He was an organizer of the First International Workshop on Release Engineering (RELENG 13) and is a member of IEEE. Contact him at bram.adams@ polymtl.ca.*



Thorsten Berger *is a postdoctoral fellow in the Generative Software Development Lab at the University of Waterloo. His research interests include model-driven development, variability modeling for software product lines and software ecosystems, variability-aware static analyses of source code, and mining software repositories. Berger has a PhD in computer science from the University of Leipzig. Contact him at tberger@gsd.uwaterloo.ca.*

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Steffen Dienst *is a doctoral student in business information systems at the University of Leipzig. His research interests include using machine-learning techniques to help monitor the operation of renewable power plants. Dienst has an MS in computer science from the University of Leipzig. Contact him at sdienst@informatik.uni-leipzig.de*

Ahmed E. Hassan *is the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan has a PhD in computer science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community, and he serves on the editorial boards of* IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering*, and* Springer Journal of Computing*. Contact him at ahmed@cs.queensu.ca.*

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE

Digital Object Indentifier 10.1109/MS.2015.56 1070-986X/$26.00 2015 IEEE