Online Reviews as First Class Artifacts in Mobile App Development

Claudia Iacob[(1)], Rachel Harrison[(1)], Shamal Faily[(2)]

[(1)] Oxford Brookes University
Oxford, United Kingdom
{iacob, rachel.harrison}@brookes.ac.uk

[(2)] Bournemouth University,
Bournemouth, United Kingdom
sfaily@bournemouth.ac.uk

Abstract. This paper introduces a framework for developing mobile apps. The framework relies heavily on app stores and, particularly, on online reviews from app users. The underlying idea is that app stores are proxies for users because they contain direct feedback from them. Such feedback includes feature requests and bug reports, which facilitate design and testing respectively. The framework is supported by MARA, a prototype system designed to automatically extract relevant information from online reviews.

Keywords: Mobile apps engineering, app stores, online reviews.

1 Introduction

The mobile application market has grown considerably over the past five years: mobile application downloads are expected to reach 48 billion by 2015 [1]. Surprisingly, however, there is a dearth of work on what the implications of mobile app development might be. Arguably, the theory and practices underpinning classical software and usability engineering need to change. In many cases, individual developers act as both designers and developers as they bring their apps to market. Also, they may share characteristics with end-user developers [2], in that they may lack formal education in software engineering and HCI and build apps that initially satisfy their own requirements before bringing these apps to market. Once available on the market, apps are subject to reviews from their users. App stores provide a straightforward way for users to give feedback on the apps they use, reviews proving to be much more trusted by users than the descriptions that developers provide [3].

Several studies on the impact that online reviews have on users have been conducted [4, 7, 8, 9, 10], but there has been little interest on how online reviews could benefit developers. A tool for retrieving information relevant for developers from online reviews is described in [5], but no framework to incorporate the tool and make it directly available to developers has been presented. This work aims to fill this gap and introduce a framework for developing mobile apps, which relies heavily on app stores and, particularly, on online reviews from app users. The paper is structured as follows: Section 2 introduces the framework, while Section 3 describes supporting tools and their role in the framework; Section 4 briefly points out the scarcity of

research in related areas. Lastly, the paper ends with conclusions and ideas for future work.

2 A Framework for Mobile App Development

App stores now have a place in the app development cycle. They collect direct feedback from users, including them indirectly in the whole development loop. In [5], the authors identified several recurring themes users report through online reviews. Such themes include feature requests and bug reports.

Feature requests inform design processes and support designers in eliciting users' requirements. In classical software engineering, such processes are usually supported by techniques such as focus groups, interviews, or questionnaires where users need to be present and are subject to questioning or analysis. App stores change this by providing direct feedback from users without them interacting with the actual designers. Even though such feedback is less structured, it is provided with little bias and in copious amounts. Writing a review does not require a user to be in laboratory settings and hence does not risk the possible bias, which may come from that context.

The deployment of apps is done on app stores. Users can use the app and evaluate it, and provide direct feedback on any possible malfunctions of the app. Such feedback is mostly expressed as bugs. Usually, evaluating an interactive system involves laboratory usability testing. In the case of apps, however, bugs get reported through reviews. At their own pace, users make use of apps and then report on the problems they encounter, feeding such data back to the designers and developers and also supporting the maintenance processes of apps. Thus app stores can be thought of as proxies for users.

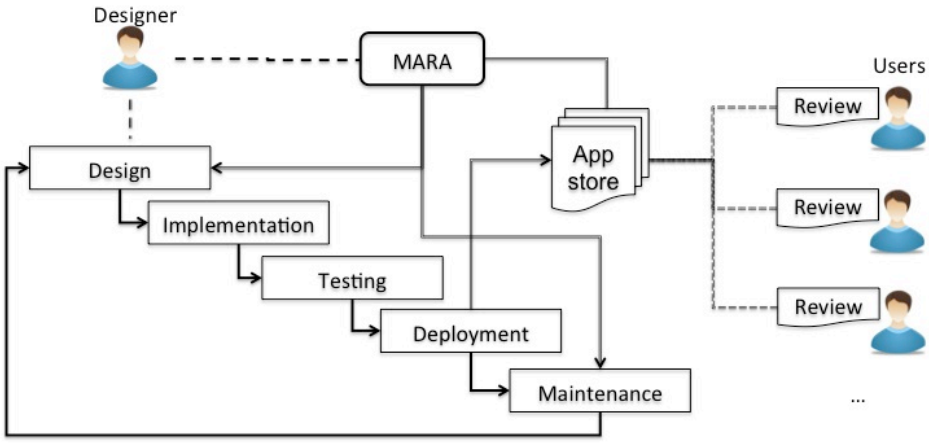

Figure 1 – A new model for app development

There are two challenges that this approach faces. On the one hand, the number of online reviews for an app may make it impossible for one developer to go through them all and manually identify relevant information for the development process or

trends in such information. On the other hand, the language used in reviews rarely respects grammar norms and rules, making it difficult to apply Natural Language Processing techniques to automatically extract information from reviews.

We have developed a tool called MARA (Mobile App Repository Analyzer) to answer these challenges. In its first iteration (described in [5]) MARA was designed to: 1) retrieve all the reviews of an app as HTML pages directly from the app store, 2) parse the raw description of the reviews and extract their content and meta-data (e.g. the date the review was posted, user information, device information associated with the review, scores from users etc.), 3) store the content and meta-data, and 4) mine for feature requests in the review content. To better support the framework described in this paper, we extended the tool to extract other types of feedback as well.

3 MARA, Version Two

We extended MARA to extract other types of feedback as well as feature requests. We are particularly interested in mining for bugs. Therefore, we adapted and extended the tool's architecture to answer such needs. The mining algorithms (4 in Figure 2) takes as input: a) the content of a set of reviews (for example, all the reviews of a given app) and b) a set of linguistic rules defined to model the language used (e.g. the language for feature requests, bugs, customer support feedback, usability feedback, etc). It then outputs all the sentence fragments in the review content which match at least one linguistic rule in the set used as input (i.e. all feature requests, or bugs).

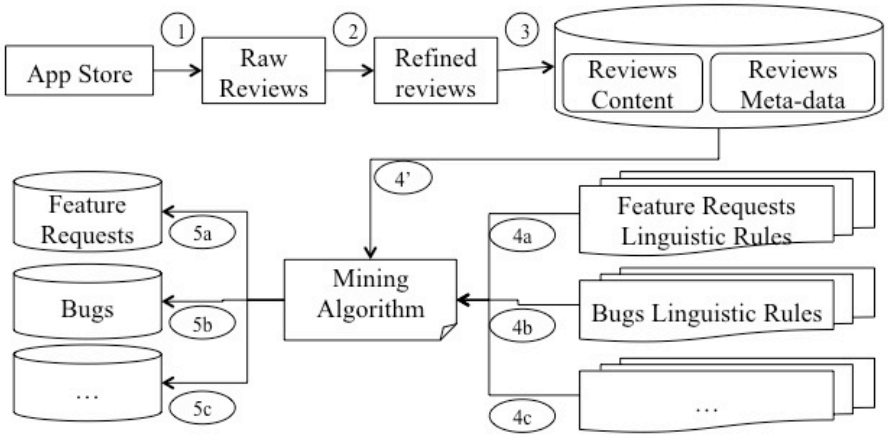

Figure 2 – MARA Architecture

3.1. Extracting Bugs

Based on the sample of 3279 randomly chosen reviews described in [5], we defined the set of linguistic rules associated with bugs. To do this, we manually extracted bug reports from the sample. We identified three categories of bugs: *major*, *medium*, and *minor* bugs and two main differences between these. First, they differ in the extent of the malfunctioning caused by the bug. *Major* bugs make it impossible to use the app. *Medium* bugs affect just one specific function of the app leaving the rest to function properly. *Minor* bugs are issues of only slight severity, which do not prevent the user from using any of the app's functions. Second, they differ in the level of frustration the users express. *Major* bugs are often expressed as "*forced shutdown many times*", "*crashes at the end of the race*", "*fails and loses entire workout*", "*app is not working*", or "*it won't open*". *Medium* bugs are less intense and more specific, focusing on a particular aspect of an app - "*does not show any tracks of previous workouts*", "*the Slovenian dictionary is missing even the basic words*", "*miles do not add right*", "*I did not create a password. Yet it asks me for my password*". *Minor* bugs are merely observations related to a feature of an app - "*slight issue with GPS data*", "*a little bug on the Persian keyboard*", "*text overlaps for lower percentages*". The majority of the bugs identified were major bugs (38.10%), while medium severity bugs accounted for 28.33% of the entire feedback reporting usage issues. 20.5% of the reported bugs were minor.

We aimed to identify linguistic rules defining each type of bug. Following the process described in details for feature requests [5], we associated each sentence labelled as a bug in the training sample with a keyword.

closes, can't/cannot/couldn't, don't/doesn't/does not/didn't, not, fail, crashes, lag, error, stopped, freeze, won't/will not, not able to, locks, unable to, erased, eliminated, impossible to, impossible to, glitches, reboot, annoying, problems with, bugs, causes, lost, restart, horrible, slows down, reloads, switches off, timeout, wouldn't, malfunction

Table 2 – Keywords for expressing major bugs

Linguistic Rule
stopped {downloading, running, syncing}
{don't, doesn't, none, didn't} *work (since the update)*
impossible to <action>
will not (even) {open, start, execute, activate, install, show up}

Table 3 – Linguistic rules for defining feature requests

We then identified all the keywords associated with more than 3 sentences in order to exclude accidental associations of keywords with sentences. The number of keywords we filtered in the case of medium and minor bugs was not conclusive enough to support further analysis. However, in the case of major bugs we identified 33 keywords (Table 2). The keywords filtered do not always point to bugs; therefore,

we went through all the sentences associated with these keywords and identified the contexts in which these keywords are used. We further abstracted these contexts into 74 linguistic rules. A fragment of this set is shown in Table 3.

3.2. Evaluation

To evaluate this revised version of MARA, we replicated the evaluation approach described in [5], but for bugs rather than feature requests. We used precision (P)[1], recall (R)[2], and Matthews Correlation Coefficient (MCC)[3] to measure the performance of the algorithm when extracting bugs. We defined: a) true positives (TP) as the correctly returned bug reports, b) false positives (FP) as the returned results which are not the actual bug reports, c) true negatives (TN) as the non bugs not returned as results, d) false negatives (FN) as actual bug reports not returned as results. We evaluated the tool with the same testing sample described in [5]; this contained all the reviews of half of the non-free apps available on the Google app store. We selected paid apps mainly because they tend to receive more reviews. The sample consisted of 136,998 reviews in total. We ran the mining algorithm on the sample using the linguistic rules defined for the bug reports. For measuring precision, we randomly selected 3000 outputs of the algorithm and a human coder went through them to identify the true positives. Based on that, we computed a value P = 0.91 for bugs. For measuring recall and Matthews Correlation Coefficient, we randomly selected one app and we considered its reviews as our sample. The results we obtained are summarized in Table 4.

Inputs	R	MCC
778	0.89	0.91

Table 4 – Recall and MCC metrics

4. Related Work

To the best of our knowledge, little research on online reviews has focused on mobile apps. However, there has been a keen interest among researchers in studying the impact of online reviews on both customer behavior [6] and product sales [4, 6, 7, 9, 10]. Based on an analysis of customer reviews of books from two different online bookstores, Chevalier et al. [6] found that a) in general, reviews are overwhelmingly positive, b) the popularity of different types of books is very similar across the two sites studied, c) the impact of 1-star reviews is greater than the impact of 5-star

[1] Precision is the ratio between the returned results which are actual feature requests (TPs) and the total number of returned results (TPs + FPs)

[2] Recall is the ratio between the returned results which are actual feature requests (TPs) and the total number of feature requests in the input (TPs + FNs)

[3] MCC value is between -1 and 1, with MCC = 1 for a perfect predictor, MCC = 0 for a random predictor, and MCC = -1 for a perfect inverted predictor.

reviews, and d) an improvement in a book's reviews leads to an increase in sales of that book on the site. Bounie et al. [4] analyzed the effect of online reviews of video games on purchasing decisions and compared them to personal and expert reviews. They found that offline information sources such as specialized magazines and trial versions as well as online information have a significant positive effect on video game purchases. When it comes to online reviews, half of the respondents claimed to always consult such reviews and 57% of them reported purchasing a video game after consulting the reviews.

The impact of a review's content and length on its helpfulness to a decision process are analyzed in [8]. The authors suggest that extremely negative reviews are perceived as less helpful than moderate ones and they classify products as search goods (i.e. a good for which it is easy to obtain information on product quality prior to interaction with the product) and experience goods (i.e. a good for which interaction with the product is necessary to obtain information on the product). Moreover, they found that the depth of a review has a positive effect on the helpfulness of the review, whereas the length of a review increases the helpfulness of a search good review more than that of an experience good review. We are not aware of studies looking into how manufacturers/developers make use of reviews to improve their products.

5 Conclusions and Future Work

In this paper, we argue that classic software engineering models and techniques may not be suitable for the development context of mobile apps. Online reviews of mobile apps hold a place in app development. The growth in the number of apps, the demographics of app developers, and the emergence of the app store distribution model mean that online reviews need to step forward and take a more productive role in the engineering of apps. In a move towards evidence based app engineering we introduce a framework which supports the full integration of user feedback reported through app stores in various phases of the app development process. During the design phase, feature requests from users are automatically extracted from reviews and the designer can then use them to guide future iterations of the app. During the testing and maintenance phases, bug reports coming from users are automatically extracted and used to help improve the apps.

As future work, we are designing studies for evaluating MARA involving app developers over longer periods of time. We are mainly interested in addressing the usability issues the tool might have and the impact its use has on the overall app development process.

References

1 http://cdn.idc.com/research/Predictions12/Main/downloads/IDCTOP10Predictions2012.pdf.

2 Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N. 2004. Meta-design: a manifesto for end-user development. Commun. ACM 47, 9 (Sep. 2004), 33-37.

3 http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf.

4 Bounie, D., Bourreau, M., Gensollen, M., Waelbroeck, P. Do Online Customer Reviews Matter? Evidence from the Video Game Industry, Telecom ParisTech Working Paper No. ESS-08-02.

5 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), 41-44.

6 Chevalier, J.A., Mayzlin, D. 2006. The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research,* 43(3), 345-354.

7 Dellarocas, C., Awad, N.F., Zhang, X. 2004. Exploring the Value of Online Product Ratings in Revenue Forecasting: The Case of Motion Pictures, *Proc. of ICIS2004*, 379-386.

8 Mudambi, S.M., Schuff, D. What makes a helpful online review? a study of customer reviews on amazon.com. *MIS Q.* 34(1), 185-200 (2010).

9 Zhang, Z., Varadarajan, B. Utility scoring of product reviews. *Proc. of CIKM '06*. ACM Press (2006), 51-57.

10 Zhang, X., Dellarocas, C. The Lord of the Ratings: How a Movie's Fate is Influenced by Reviews?, *Proc. of ICIS2006*, 1959-1978, (2006).