Which Version Should be Released to App Store?

Maleknaz Nayebi, Homayoon Farrahi, Guenther Ruhe
SEDS Laboratory
University of Calgary
Calgary, Canada
Email: {mnayebi, homayoon.farrahi, ruhe}@ucalgary.ca

***Abstract*—*Background*: Several mobile app releases do not find their way to the end users. Our analysis of 11,514 releases across 917 open source mobile apps revealed that 44.3% of releases created in GitHub never shipped to the app store (market). *Aims:* We introduce "marketability" of open source mobile apps as a new release decision problem. Considering app stores as a complex system with unknown treatments, we evaluate performance of predictive models and analogical reasoning for marketability decisions. *Method:* We performed a survey with 22 release engineers to identify the importance of marketability release decision. We compared different classifiers to predict release marketability. For guiding the transition of not successfully marketable releases into successful ones, we used analogical reasoning. We evaluated our results both internally (over time) and externally (by developers). *Results:* Random forest classification performed best with F1 score of 78%. Analyzing 58 releases over time showed that, for 81% of them, analogical reasoning could correctly identify changes in the majority of release attributes. A survey with seven developers showed the usefulness of our method for supporting real world decisions. *Conclusions:* Marketability decisions of mobile apps can be supported by using predictive analytics and by considering and adopting similar experience from the past.**

***Keywords*-Release management; Mobile apps; Empirical study; Marketability; Analogical reasoning; Survey**

I. Introduction

Market and user characteristics of mobile apps make their release management different from proprietary software products and web services. By focusing on mobile apps, we introduce the notion of release marketability. "Marketability" is part of release management for platform-mediated software products [2]. Developers release a version of the app on a platform which was also known as the app store or marketplace. Users can download the app from the platform instead of getting it directly from the software owner as it is done traditionally. We illustrated this fundamental difference in offering software app products in Figure 1.

While an increasing number of software products are designed and developed for platform-mediated environments, in this paper we only study releases of open source mobile apps. In our former study [10] we found that a substantial number of releases never reach end users through the app store. A *marketed release* is a release that was introduced in a Git repository as well as in the app store. However, a *not marketed release* is a release that was only introduced in the Git repository of the app [10]. "Marketability" refers to the question if a new release should becomes marketed or not.

Based on our former studies [9], [10], we characterize a release with a set of code, release timing, and market attributes and predict release marketability. Once we find that a release is not marketable, we perform analogical reasoning to retrieve similar not successfully marketable releases in the past. We observe how the attributes changed while a release transitioned into a successfully marketable release. Following the idea of case-based reasoning [15], these *analogical changes* give approximation of effort, nature of changes, and code quality needed to be achieved for a release transition. Offering these analogical changes is not meant to be prescriptive and should be adopted and revised considering the context of the app. However, as a form of software engineering knowledge management [13] this is supposed to support developers' decision-making. This paper has four main contributions:

First, we introduced and confirmed the importance of marketability decision for mobile apps which has not been investigated so far.

Second, we compared three machine learning methods to predict release marketability.

Third, to transition a not marketable release into a successfully marketable one, we performed analogical reasoning using the experience of the same app and looking across apps.

Fourth, we evaluated results of analogical reasoning with actual changes that transitioned an under question releases (internal validation over time). We also performed a survey with app developers for external validation.

The research conducted in this paper is motivated by a

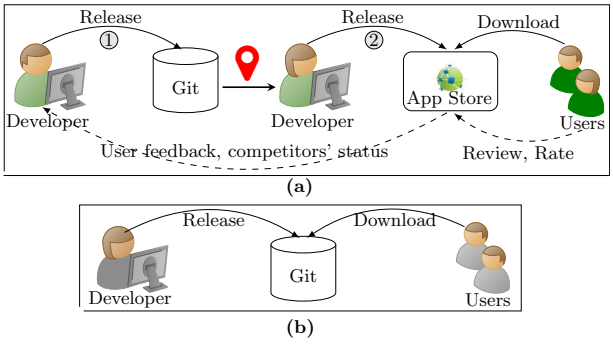


Fig. 1. Distribution of a product release to the end user for (a) app versus (b) non-app open source software products.

978-1-5090-4039-1/17 $31.00 © 2017 IEEE
DOI 10.1109/ESEM.2017.46

survey with 22 release experts discussed in the next section. In Section III, we formulate the research questions. We provide a motivating example in Section IV and describe our methodology (Section V). We outline the design of the empirical evaluation in Section VI, followed by empirical results given in Section VII. We discuss threats to validity (Section VIII), report about the analysis of related work (Section IX), and provide conclusions (Section X).

II. Significance of Mobile App Marketability Decisions

From analyzing 917 F-Droid open source apps, we found that 35.4% of 11,514 releases were never shipped to the app store. In Figure 1 we visualized a simplified release process for open source mobile apps in comparison to traditional open source applications. Considering the process shown in Figure 1, a developer releases into a Git repository (① in Figure 1(a)). Once the release is in the Git repository, the decision (Figure 1 - 📍) to be made is if this release should be shipped to the app store or not. While the release might not be shipped into the app store (② in Figure 1 (a)), a user can still install the app from its Git repository. This is different from the analysis of release readiness [1], which is limited to the release on a Git repository as it is shown in Figure 1 -(b). Marketability is solely differing between *releasing into a Git repository* or *releasing into the end-user and app store*.

We performed a study with participants of the 4^{th} International Workshop on Software Release Engineering (RELENG 2016) to evaluate the impact of "the market" on release decisions for mobile apps. Participants self-evaluated their level of expertise in release engineering practice, research and release engineering of mobile apps on a 3-point scale (one indicated the lowest and three indicated the highest level of proficiency). The majority of the participants had higher expertise in release engineering practice (with Average 2.1). Our participants had an average expertise of 1.8 in release engineering research and 1.6 in release engineering of mobile apps. To evaluate the importance of marketability, we performed a comparative survey study between mobile apps and non-mobile apps (desktop and web applications) using the factors found by AlAlam et al. [1].

First, participants evaluated the importance of six attributes on planning for mobile versus non-mobile releases. The box plot distributions of the results are shown in Figure 2-(a). In this figure, "1" shows the least and "5" shows the highest perceived importance of a factor. The results indicate that *Customers' expectations* and *Market and competitors* are significantly more important for mobile in comparison to non-mobile apps (Mann-Whitney test p-value = 0.032 and 0.001). Also, the results revealed that *Implementing a new feature* has a higher importance in planning for a release of a mobile app compared to other attributes (p-value= 0.002).

Second, the participants evaluated the importance of factors for measuring the success of a release. Our results in Figure 2-(b) show that, *customers' feedback* and *sales status* have significantly higher importance for mobile apps in comparison to the non-mobile applications. 20 out of 22 participants (90.9%) believed that *customer feedback* has the highest importance ($= 5$) for evaluating success and failure of mobile apps.

Third, we asked participants if they think market acceptance is a more important criterion for releasing a mobile app version than it is for other software products.

> 95.4% *of participants (21 out of 22) believed that market acceptance is more important for mobile apps than it is for any other software product releases.*

Fourth, we asked the survey participants about the perceived reason for this difference by evaluating the importance of three factors extracted by Nayebi et al. [7] on a five-point scale (see Figure 2-(c)). On top of that, participants could openly add factors. Survey participants ranked the importance of *customers' feedback* significantly higher than *reluctancy to update* (p-value = 0.032) and *stronger competition* (p-value=0.043). Three participants added below responses:

"Unlike desktop, mobile apps do not seamlessly update. So, this missile some of the effects of public feedback."

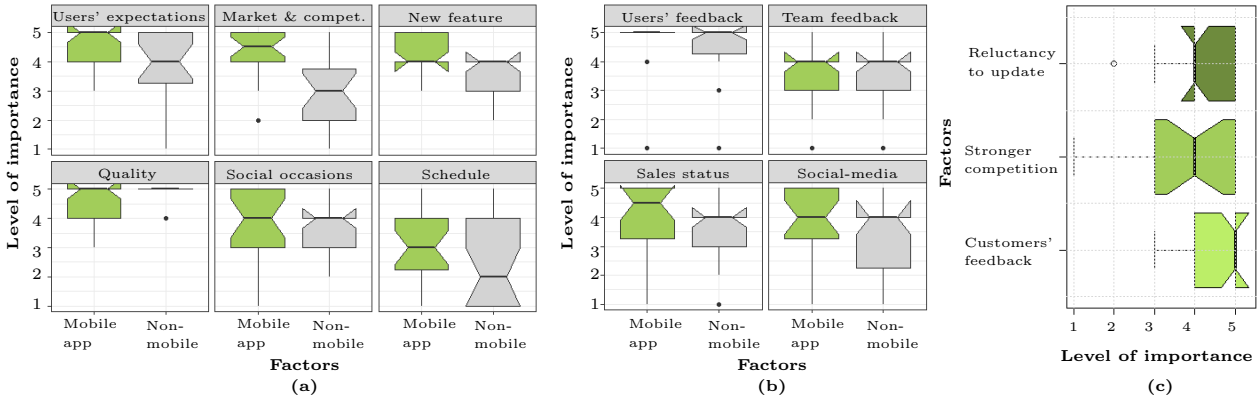


Fig. 2. Results of survey with 22 release engineers (a) Importance of factors for release planning (b) factors for evaluating success and failure, and (c) reasons of difference in release planning mobile apps versus traditional software.

"The market is more important for mobile apps as the path between developer and user is much shorter."

"There are more choices in mobile than in web."

The importance of market and the high number of not marketed releases which we observed in open source mobile apps motivated us to investigate on methods for assisting mobile app developers in their decision making. Based on the results showed in Figure 2-(b)), we analyze users' feedback to classify releases into successful and non-successful ones.

A marketable release could be successful or not. "Marketable but unsuccessful" is the gray area of marketability decision, saying that similar releases were delivered into the market however they were not successful. In this paper, *transitioning of a release* refers to transitioning a not marketable or marketable but unsuccessful release into a successfully marketable release. We describe the related research questions in the next subsection.

III. Research Questions

We investigate on three research questions. We compare classifiers to evaluate if marketability is predictable (**RQ1**) and if so, how can we use past experience to support marketability decisions (**RQ2**). On top of that, we evaluate the usefulness of this support for app developers (**RQ3**):

RQ1 - *(predictability)*: Which classifier algorithm works better for marketability prediction when comparing Decision Tree, SVM, and Random Forest models?
Why and how: We study how accurately we can predict marketability of a release by analyzing a set of release and app attributes. By identifying a success criteria, we classify each incoming new release of a mobile app as being exactly one of *marketable and successful*, *marketable but unsuccessful*, or *not marketable*. We compare Decision Tree, Random Forest and Support Vector Machine (SVM) classifiers using two sets of attributes.

RQ2 - *(internal validation)*: How useful is analogical reasoning in suggesting changes for transitioning a release?
Why and how: Learning from experience is an established concept as a form of guidance for decision-makers, not being anyhow prescriptive in its nature. For an incoming new release, we find the most similar releases from the same app and similar apps. Then, we identify changes in release and code which resulted in transitioning a not successful or not marketable release, into a successfully marketable one. Having data over time, we compare analogical changes with the actual changes of a transitioned release.

RQ3 - *(external validation)*: To what extend app developers consider the analogical reasoning useful for making release marketability decisions?
Why and how: Besides the formal and internal validation addressed in RQ2, we performed a survey with actual app developers. This is a very first step in the challenging domain of external validation and design of a decision support tool.

IV. Motivating Example

We briefly illustrate the main idea of the paper by an example. Through this example, we provide a sneak peek into the prediction and recommendation methods and sample results. For the purpose of illustration, we selected `BatteryXu` which is an app for battery management.

`BatteryXu` has seven releases between $January\ 31^{st}, 2014$ and $October,\ 31^{st}, 2016$ as shown in Figure 3. Among them, two releases are exclusively kept in the GitHub repository (not on Google Play). Two of the marketed releases were unsuccessful, meaning that they received reviews with negative sentiments. Beginning of 2016, `BatteryXu` has a $Release$ ④ in GitHub. Using release and app attributes (Table I) to build a Random Forest classifier, with 78% precision we predict that $Release$ ④ is not marketable and should not be released into the app store (**RQ1**). Looking into the releases later in 2016, we can observe this prediction was correct.

Once we anticipate that the release is not successfully marketable, we retrieve experience to guide developers in future release decisions. In **RQ2**, we search for changes that in the past transitioned a similar not successfully marketed release. Following *case-based reasoning*, by presenting analogical changes to developers, they can adopt, revise, and reuse them [12] to make the release marketable and successful. We summarized the key steps for answering **RQ2** in Figure 4.

First, we calculate the Euclidean distance of $Release$ ④ of `BatteryXu` from all the other not successfully marketed releases from our sample data set which includes open source mobile apps hosted on F-Droid and GitHub (see Figure 4-(a)). We pick the three most similar releases to $Release$ ④. As an instance, Release ⓕ of `BatteryBot` was an unsuccessfully marketed release that was similar to $Release$ ④ of `BatteryXu`. As demonstrated in Figure 4-(b), for each of these three releases we mine which changes were applied to transition the release. These changes are measured as the Δ of the 12 release attributes presented in Table I. To add context and make other releases comparable to $Release$ ④, we move from the absolute degree of change in each attribute to relative changes using frequency based discretization. As a simple example, closing six issues in a release of an app which usually closes 20 issues per release is low. However, closing six issues in a release of an app that usually closes two issues per release is high. Hence, we map the degree of change (Δ) of each attribute relative to the overall change of the attribute across all releases of the app using frequency based discretization. Release ⓕ of `BatteryBot` was not marketed and was the

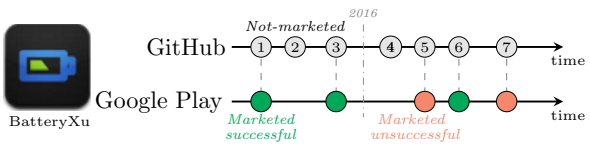
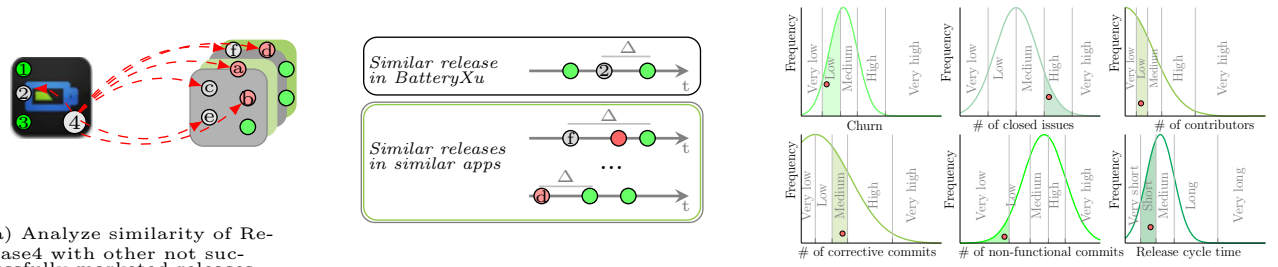


Fig. 3. Marketed and not marketed release of a sample app "BatteryXu". The color of marketed releases show the success status inferred from review sentiments.



(a) Analyze similarity of Release4 with other not successfully marketed releases (i) within BatteryXu, and (ii) across similar apps.

(b) Find changes in attributes of each similar release that transitioned it

(C) Discretizing relative changes in attributes of similar releases for sample release f

Fig. 4. Main steps of retrieving past experience for transitioning Release 4 of BatteryXu.

most similar to *Release* ④ of `BatteryXu`. As an instance, we calculated the churn between each two consecutive releases of `BatteryBot`. We calculated the twentieth percentiles of churn for `BatteryBot` and observed that the churn of Release ⓕ is in the 40% percentile, thus considered as "low" (see Figure 4-(b)).

As a result, we abstracted three analogical changes that are potentially applicable for *Release* ④ of `BatteryXu`. These changes provided a successful transition of former releases similar to *Release* ④:

Analogical change 1: Low churn ; low # of contributors ; medium # of corrective , very low # of non-functional commits; high # of closed issues ; within a short release cycle .

Analogical change 2: Low churn ; low # of contributors ; high # of perfective, medium # of corrective commits; within very long release cycle.

Analogical change 3: Low churn ; very low # of contributors, high # of corrective, low # of non-functional commits; high # of closed issues ; within a short release cycle .

Analogical change 1 shows that Release ⓕ of `BatteryBot` transitioned into a successful release by involving low churn, low # of contributors, and within short release cycle (proxies for low effort). However, the changes were focused on quality enhancement (nature of changes) as medium # of corrective commits were involved and high # of issues were closed (quality enhancement). These were done along with few non-functional changes.

To internally validate the usefulness of analogical changes, we compared them with the actual changes that transitioned *Release* ④ to successfully marked *Release* ⑥ later in 2016:

Actual change in `BatteryXu`: Low churn ; low # of contributors ; medium # of corrective , low # of non-functional commits; high # of closed issues ; within a short cycle .

The maximum # of valid changes (= 5) occurred in *Analogical change 1* which was retrieved from the release most similar to *Release* ④. The minimum # of valid changes (= 3) occurred in Analogical change 2. Overall, two analogical changes, *Analogical change 1* and *Analogical change 3*, were needed to cover all the valid changes.

V. Methodology

We discuss the process and different techniques used for our empirical investigation. We used `Python` language and Scikit-learn[1] package throughout the project to gather data, and implement predictive models and perform analogical reasoning.

A. Marketability prediction (RQ1)

A considerable amount of releases fail to attract attention and satisfaction of users. The initial survey in Section II showed that "customers' feedback" is significantly most important factor to assess success and failure of an app release. In this paper, each release belongs to exactly one category of:

Marketable successful: The release is in a proper status to be released on the app store and likely will be a successful release (attracts positive reviews). This is a *"yes"* answer to the question of marketability.

Marketable but unsuccessful: Similar releases were shipped to the app store however they were unsuccessful. This is a *"maybe"* answer to the question of marketability.

Not marketable: The release is not in a proper status to be shipped to the app store. This is a *"no"* answer to the question of marketability.

To define the success of an app, we relied on users' review sentiments. Among different forms of user feedback in app store, reviews appeared to be the most informative [6], [8]. Each marketed release can be classified as being either successful or unsuccessful:

Release is successful if the average sentiment of reviews after a release and before the next consecutive marketed release is positive;

Release is unsuccessful otherwise.

Following Smedt and Daelemans [16], a user's sentiment in a review could be positive or negative, measured as *polarity*

[1]http://scikit-learn.org/

TABLE I
ACCURACY OF DIFFERENT MODELS FOR PREDICTING MARKETABILITY OF OPEN SOURCE APP RELEASES.

Release attributes		
ID	**Attribute**	**Definition**
att_1	Churn	Lines of code that changed (added or deleted) between two releases.
att_2	# of changed files	Number of files that were changed between the former release and the current release.
att_3	# of contributors	Number of people contributed to the current release.
att_4	Release cycle time	The number of days between the former release and current release.
att_5	# of open issues	Number of open issues at the time of release.
att_6	# of issues opened following a release	Number of issues opened between the former release and the current release.
att_7	# of closed issues	Number of issues closed between the former release and the current release.
att_8 - att_{12}	# of corrective, adaptive, perfective, implementation, and non-functional commits	Nature of changes categorized based on categories defined by Hindle et al. [4].
App attributes		
att_{13}	# of app releases	Number of marketed release of the app by the time of current release.
att_{14}	Release cycle variance	Variance of marketed release cycle times by the time of current release.
att_{15}	Average sentiment	average sentiment of all reviews across all releases at the time of the current release.

which is a number in the range [-1, 1]. For each marketed release, we used average sentiment of all the reviews following the release and before the next marketed release as the success criteria. We analyzed sentiments' polarity for app store reviews using Python's `Pattern` [16] module. We applied and compared three different machine learning techniques to classify a new release into one of these categories. To compare the performance of three classifiers Decision Tree, Random Forest, and Support Vector Machine (SVM), we used both 10-fold cross validation and Leave-One-Out (LOOCV) cross-validation. We used two sets of attributes (see Table I):

Release attributes: Code, time, and developer based attributes to reflect status of a release.

App attributes: Release attributes of the app as well as average users' review sentiment across all the releases.

We selected **release attributes** using the results of our former study [7] where we described the characteristics of marketed and not marketed attributes. In that study, we demonstrated the impact of the number and cycle time of releases (*When?*), nature of changes (*What?*), reported issues and change requests (*why?*), and extent and domain of changes (*Which?*). We introduced all the release attributes in Table I. Our previous study [10] showed significant differences between marketed and not marketed releases concerning all the 12 attributes listed in the table. Our initial study showed that not marketed releases have a significantly greater change velocity (att_1, att_2). However, marketed releases have a higher speed in introducing and resolving issues (att_5, att_6, att_7). Our descriptive statistics showed a significant difference between release cycle time for marketed versus not marketed releases (time between each two consecutive releases). With regards to nature of changes (att_8 to att_{12}), it became apparent that not marketed releases have significantly higher number of corrective and implementation commits, while marketed releases put the focus on non-functional changes.

For predicting marketability of releases from Github, we also used **app attributes** and selected them based on the results of our former study [9]. Therein, we investigated which attributes create better entropy between clusters of similar mobile apps. These attributes are also introduced in Table I. Without compromising on entropy we selected a minimum number of possible attributes (three) for building our classifiers [9]. We used average sentiment of the reviews over all releases as the market and user attribute, for simplicity and consistency throughout the paper.

B. Analogical reasoning to support marketability decisions (RQ2)

In **RQ1** we predicted marketability of mobile app releases. In **RQ2**, we provide analogy-based support for transitioning a not successfully marketable release into a successful one. Analogical reasoning has been successfully used in different fields of software engineering [15] and beyond [12]. Reuse of software knowledge is known to be inherently difficult because, in a strict sense, each software project is different from other in details. However, reuse of experience is essential, and it is intuitive to learn from experience. To assist decision making about release marketability, we retrieve related experience. The input for **RQ2** is a not successfully marketed release. We illustrated the process of analogical reasoning in Figure 5 and describe the three steps below:

Step ❶:We find most similar releases to the incoming new release within our knowledge base. The knowledge base is the repository of not marketed releases that have been successfully transitioned into a successfully marketed release. We determine the Euclidean distance between the incoming not marketable release r and all releases in the knowledge base. We pick the m top releases with shortest distance to r.

Step ❷: For each of the m releases similar to r, we analyze their transition into next successfully marketed release. The aim is to guide the transition of r into a successfully marketed release denoted as r^+. We compute the changes in all the 12 release attributes (att_1 to att_{12}) for each pair of a similar release r_j and the transitioned r_j^+ ($j = 1...m$) release. More formally, for an app a, the change $\Delta att_i(a, r_j)$ of a release attribute i is defined by Equation (1):

$$\Delta att_i(a, r_j) = att_i(a, r_j^+) - att_i(a, r_j)$$
$$\text{for app } a \text{ and release } r \text{ for all similar releases} j = 1...m \quad (1)$$

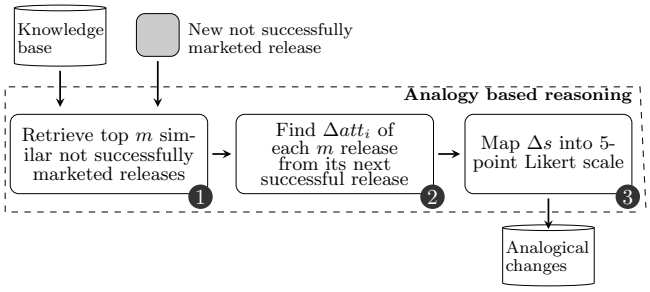


Fig. 5. The process of creating analogical changes for release transition.

Each value $\Delta att_i(a, r)$ is the absolute value of attributes' change when release r_j transitioned to release r_j^+.

Step ❸: To abstract from detailed numbers, we map the values of $\Delta att_i(a, r)$ into five-point scale using frequency based discretization. We calculate the change of each attribute during a transition (between the not successfully marketed release under analysis and the first successfully marketed release after that). We descritize $\Delta att_i(a, r)$ considering the frequency distribution of all the changes of att_i in different releases of app a. We calculate five percentiles (at 20^{th}, 40^{th}, 60^{th}, and 80^th levels) of att_i across all releases of app a. This way, each $\Delta att_i(a, r)$ is mapped to exactly one category. We visualized an instance of this process in Figure 4.

To evaluate the usefulness of analogical reasoning in this context, we applied analogical reasoning for a release and compared analogical changes with actual ones. To do so, we used all the releases of F-droid apps before 2016 as our knowledge-base. This represent 75% of our whole data set. We demonstrated this process in Figure 6. First, we found releases with a successful transition in 2016 for our evaluation purpose (Step 1). For each release with a successful transition (Step 2), we applied the three steps of Figure 5 to extract analogical changes. For each release in our test set we found the Δ of all attributes between test release and its actual next successfully marketed release (Step 3). Then we discretized and mapped Δs (Step 4). Finally we compared the actual changes with analogical changes of release attributes (Step 5).

To internally evaluate the usefulness of analogical reasoning in this context (**RQ2**), we compared analogical changes with the actual ones. We used four evaluation criteria:

Coverage: Number of valid attribute changes that were covered by at least one of the analogical cases.

Distribution: Minimum number of analogical cases needed to cover most number of valid attribute changes.

Max # of valid changes: Maximum number of matches between actual and and analogical changes in a similar case.

Min # of valid changes: Minimum number of matches between actual and analogical changes in a similar case.

C. External validation of analogical reasoning (RQ3)

In **RQ3**, we performed a survey with mobile app developers to understand usefulness and intuitiveness of the retrieved analogical changes. We asked developers to define the extent

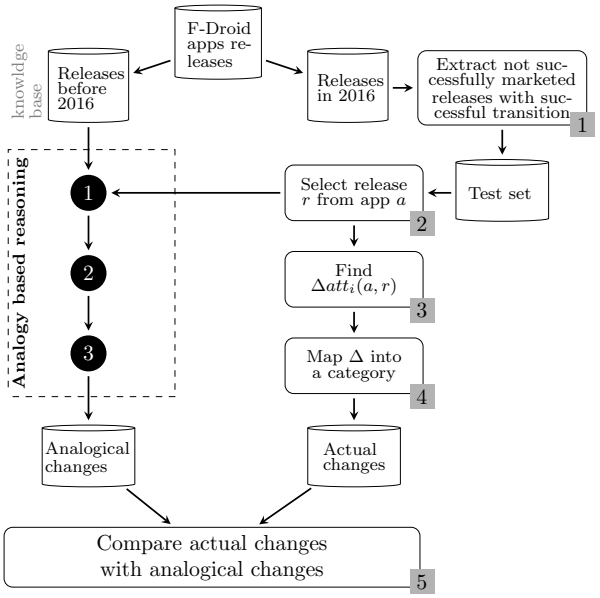


Fig. 6. The process of analogy based reasoning for transitioning a release and evaluating its results (**RQ2**).

(on a 5-point scale with five being highest) they rely on their own experience, versus the product team experience versus experience from similar products for making marketability decision.

We presented to them "app attributes", a link to apps' GitHub repository, and a link to apps' Google Play page. We showed the attributes of the release under analysis (att_1 to att_{12}) to the survey participants. We presented to them transitions we have mined from similar releases in the past and asked each participant, to classify and rank the usefulness of analogical changes. Each participant could classify extracted transition into **useful** (the retrieved transition is useful to transition the release under question), **maybe** (the retrieved transition may be helpful to transition the release under question), and **useless** (the retrieved transition is not useful to transition the release under question). Participants ranked the transitions in a way that the first transition in class "useful" is the most helpful and the top transition in the class "useless" is the least useful transition.

VI. Case Study Design

In what follows we describe the design of our case study.

A. Mining open source apps from F-droid and GitHub

We gathered all the apps from the F-Droid repository of the open source Android apps over a period of 12 months. By monitoring and crawling over that period, we increased our sample size from 1,273 apps at the start to the 1,844 apps as of $October$ 2016. Most of them (917 apps) were hosted on GitHub. To get similar data attributes for all the apps (such as release tags), we focused our analysis on the 917 open source Android apps hosted on GitHub. For all 1,844 apps from F-Droid, we parsed the HTML data from their respective F-Droid

page and gathered the apps' *package name*, *source control repository*, and *issue tracker* information. We filtered data so that only apps hosted on GitHub open source repositories remained. This process resulted in 917 open source Android apps. We gathered data for each app using their GitHub URL as stated in F-Droid. We analyzed GitHub log of each of these apps to extract release attributes (see Table I).

B. Collecting data for marketed and not marketed releases

Using the *package names* obtained from F-Droid, we retrieved the Google Play page of each app. We then parsed the HTML data from Google Play (for reviews) and `Searchman.com` (for release dates). `Searchman.com` is a third party analytics platform which gathers data of Google Play over time and hence, keeps all the release dates of an app. We also gathered release identifier and dates from GitHub repository of each app. While mapping releases between GitHub and Google Play, four different cases occurred as shown in Figure 7;

Type 1: The GitHub release date and identifier match the Google Play release date and identifier. We call these releases being both in Github and Google Play as *"marketed"*

Type 2: The GitHub release identifier or release date is not an exact match with the Google Play release. We manually inspected several releases to find an approximation schema for mapping two releases. We analyzed release identifiers as well as date of release to map releases between Github and Google Play.

Type 3: The GitHub release is not available in Google Play. This means that neither the identifier nor the release date approximately match with any release in the app store. We call these releases "*not marketed*".

Type 4: The Goole Play release is not available in the GitHub repository of the app. These cases were rarely observed (less than 1% of releases). Often, following this type of releases, open source development on GitHub was discontinued. We excluded these cases from our data.

The most challenging inconsistency was **Type (2)**. In those cases, we called two releases *identical* if they had a distance of not more than five days in their release date. We considered two release identifiers the same if one or both had "v" or "version" in the beginning, or one or both had ".0" at the end. For sub-string matching of release identifiers, we used the regular expression "[0-9]+(\.[0-9]+)(\.[0-9]+)?". As an example, if "*V 1.3.0*" were released on March 17^{th} in GitHub and release "*Version 1.3*" released on 18^{th} in Google Play, we considered these two versions the same and called this release "marketed".

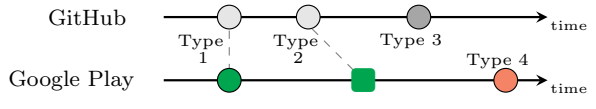


Fig. 7. Mapping GitHub releases to Google Play releases resulted in four different types of transition.

As the result of this process, we gathered a pool of 11,514 releases over 917 apps. Among them, 7,435 releases were marketed and 4,079 releases were not marketed. For identifying success of a marketed release, we gathered users' reviews following each release of an app. We analyzed reviews' sentiment [3], [8] by calculating polarity [16]. In total, we analyzed 78,304 reviews. Among the 7,435 marketed releases, we identified 3,734 releases as *successful* and 3,701 releases as not.

C. Set-up of knowledge base and test set (RQ2)

For the purpose of evaluation in **RQ2**, we mined not marketed or marketed-unsuccessful releases that transitioned into at least one successfully marketed release. For establishing our knowledge base, we limited our search into releases from 2016. The 917 open source mobile apps of our sample set had 2,498 releases on GitHub between *January* $1^{st}, 2016$ and *October* $31^{st}, 2016$. Among them, 58 releases across 52 apps satisfied the conditions for this evaluation. For a subset of these 58 releases, we could also retrieve similar experience from the same app. This means that for 19 releases of 19 apps, at least one former instance of not successfully marketed releases with a transition to successfully marketed release existed.

VII. Results

In this section, we present the results of the three stated RQ's sequentially.

A. Evaluation of classifiers for marketability (RQ1)

We compared Decision Tree, Random Forest, and Support Vector Machine classifiers for predicting marketability. We trained these models first by considering release attributes and second by considering app and release attributes jointly (see Table I for the attributes). To reduce bias, we evaluated the accuracy of the prediction models by both 10-fold and Leave-One-Out cross-validation (LOOCV).

As the result of data pre-processing in Section VI we ended up with a dataset of 3,734 (32.4%) marketed successful, 3,701 (32.1%) market but unsuccessful, and 4,079 (35.4%) of not marketed releases. We used this data set to build and evaluate our prediction model for **RQ1**. The precision, recall, and F1 score of the three classifiers are shown in Table II. The results of 10-fold cross-validations are the average values of 10 runs of cross-validations. The performance of the three prediction models on our data set is about the same. We used the minimum number of attributes to build our model by excluding correlated variables from our attribute set considering the results of our former study [9].

Decision Tree: We did not weight the decision tree as our sample had approximately the same size of labeled data in each category. We used Gini index to decide on splits of the tree and tuned the parameters to ensure that each leaf node in the tree has the minimum of four samples and prevent over fitting.

Random Forest: For this model, we limited the number of features each tree can use at each split by using square root and 20% threshold. This reduced the F1 score between 4% to 7% comparing to the models without feature restriction.

SVM: To tune parameters we performed an exhaustive search over $Kernel$ (function to transform data), $Gamma$ and C values (for non-linear data transform) in a way to maximize the score of the left out data. This tuning was done using `Scikit`'s `GridSearchCV` package which optimizes parameters by cross-validated grid-search.

Random Forest had a slightly better F1 score. Besides, when using app attributes in conjunction with release attributes, this results in better recall with precision being about the same.

B. Internal validation of analogical reasoning results (RQ2)

To internally validate the usefulness of the analogical reasoning following the method described in Section V and Figure 6 we compared the analogical changes with the actual changes and applied the four criteria discussed in Section V-B. For each release, we retrieved similar releases from the same app (if available) and from five ($m = 5$) similar apps. We used $m = 5$ as we observed most of our 58 releases have five similar releases with normalized Euclidean distance < 0.25. As we discussed in in Section VI-C, we evaluated cross-app analogical reasoning on 58 releases and within app analogical reasoning for 19 releases.

Figure 8 shows the evaluation results for 58 releases by only looking into similar apps. Eight release attributes have been changed in the majority of the transitioned releases. The analogical reasoning could cover on average 70% of actual changes, however, these were mostly distributed in three separate pieces of retrieved experience. For 21.1% of transitioned releases, we could correctly retrieve changes in all attributes. For 44.8% of cases, we could correctly retrieve above 90% of attribute changes. The minimum and a maximum amount of change show the most ($median = 50\%$) and least ($median = 18\%$) attribute changes among five similar cases for each of the 58 releases.

Table III presents the comparison between analogical changes and actual changes, within and across apps. Only for one release (R14), the within app analogy reasoning outperformed the cross app analysis. However, it appeared that reasoning within and across apps are complementary. In the majority of releases (63.1% of 19 releases), looking into the within app reasoning increases the coverage of the cross app analysis by 20% to 37.5%. In other words, we can correctly infer changes in some attributes analyzing former releases of the same app.

The results showed that analogical reasoning is useful for guiding marketability decisions. However, the valid changes are distributed across several analogical changes. This is aligned with the general methodology, as in this context, there is no expectation that experience could be reused without adaptation by the domain expert.

TABLE II
ACCURACY OF DIFFERENT MACHINE LEARNING TECHNIQUES FOR PREDICTING MARKETABILITY OF OPEN SOURCE APP RELEASES.

Classification technique	**10-fold cross validation**			**LOOCV**		
	Precision	Recall	F1	Precision	Recall	F1
Use release attributes only						
Decision tree	0.75	0.74	0.74	0.81	0.80	0.81
Random Forrest	0.77	0.78	0.78	0.85	0.86	0.85
SVM	0.68	0.66	0.66	0.83	0.80	0.81
Use app and release attributes						
Decision tree	0.76	0.79	0.74	0.86	0.88	0.81
Random Forrest	*0.78*	*0.79*	*0.78*	0.83	0.89	0.85
SVM	0.71	0.77	0.73	0.73	0.77	0.74

C. External validation of analogical reasoning (RQ3)

We identified apps with frequent marketed and not marketed releases in 2016 (more than 2 of each) and identified developers with a considerable number of commits (more than 5) from each project and invited them to participate in our survey (16 developers in total). Seven developers from four different open source apps participated in our survey for external evaluation of analogical reasoning.

First, we asked the developers to what extent (on a five-point Likert scale) they rely on their "own experience", "status of the apps' former releases", and "lessons learned from other apps" to make release decisions. Only two developers rely on their personal experience stronger than former experience and lessons learned. Reuse of experience within app context were slightly more important to developers comparing to cross-app experience. it appears that reuse of former experience is acceptable along with personal knowledge for surveyed developers. The developers valued their own experience and within- and across- app experiences all on average or above.

Second, for each of the 19 releases we discussed in Table III, a developer defined if the retrieved case is *useful*, it *might be* helpful or is *useless*. Overall, 121 analogical changes for 19 releases were evaluated by seven developers. Each release was evaluated by one developer and no developer evaluated analogical cases of more than four releases. The results showed that 48.8% of all the retrieved cases were considered as "useful" by developers, while 42.9% were considered as "might be useful", and 8.3% as "not useful". Also, 84.2% of *the most similar* cases to each of the 19 releases are "useful" and the rest (three of them) were evaluated as "might be

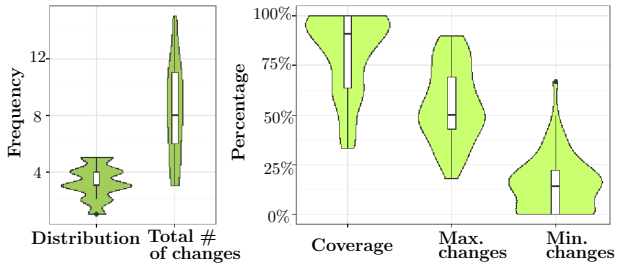


Fig. 8. The distribution, min and max actions across 58 similar cases.

TABLE III
COMPARISON OF CROSS APP AND WITHIN APP EXPERIENCE FOR OPEN SOURCE MOBILE APPS.

	Reasoning based on similar apps					Reasoning based on same app releases					
ID	**Coverage**	**Distribution**	**# of changes**	**Min. Valid changes**	**Max. valid changes**	**# of cases**	**Coverage**	**Distribution**	**# of changes**	**Min. Valid changes**	**Max. valid changes**
R1	100%	3	8	12.50%	50%	2	100%	9	2	22.20%	33.30%
R2	100%	4	8	25.0%	50.0%	1	75.0%	4	1	75.0%	75.0%
R3	100%	5	11	18.2%	45.5%	1	55.6%	9	1	55.6%	55.6%
R4	100%	3	9	11.1%	55.6%	2	50.0%	4	1	50.0%	50.0%
R5	100%	3	15	26.7%	46.7%	1	41.7%	12	1	41.7%	41.7%
R6	100%	5	6	33.3%	50.0%	1	40.0%	5	1	40.0%	40.0%
R7	100%	2	5	40%	80%	1	25%	8	1	25%	25%
R8	92.80%	4	14	14.20%	64.20%	2	36.30%	11	1	9%	36.30%
R9	85.7%	3	5	20.0%	40.0%	1	57.1%	5	1	57.1%	57.1%
R10	85.7%	4	8	25.0%	62.5%	2	42.9%	7	1	42.9%	42.9%
R11	84.6%	5	15	13.3%	46.7%	1	53.8%	7	1	53.8%	53.8%
R12	80.0%	3	14	21.4%	28.6%	1	40.0%	11	1	40.0%	40.0%
R13	71.4%	2	7	28.6%	85.7%	1	57.1%	5	1	57.1%	57.1%
R14	62.5%	4	11	18.2%	27.3%	2	75.0%	14	2	14.3%	41.7%
R15	60.0%	4	7	14.3%	42.9%	1	40.0%	5	1	40.0%	40.0%
R16	42.9%	3	11	9.1%	27.3%	1	41.7%	12	1	16.7%	25.0%
R17	41.7%	3	6	16.7%	50.0%	1	33.3%	6	1	33.3%	33.3%
R18	41.7%	4	12	16.7%	33.3%	2	23.1%	13	2	7.7%	16.7%
R19	36.4%	4	13	9.1%	27.3%	2	25.0%	12	2	16.7%	16.7%

useful". Developers ranked the retrieved experience based on the usefulness. We compared developers' ranking with the ranking based on similarity between analogical releases and the under question release. The positive 0.63 Spearman's rank-order correlation between these two showed that there is a positive relation between releases' similarity and usefulness of them for developers.

Third, we asked developers if and to what extent they would like to reuse experience from other apps to decide on release marketability. Six of the seven developers stated their definite willingness to know about similar cases while making marketability decision. One developer were uncertain about ease of using this information. Four of the developers suggested more structure, context, visualization and a tool support to make the results more usable:

"Availability of multiple choices is the best. But, looking into the app repository with URL eats my time."

"The choices are not extremely different. Visualizing them or making them side by side or color them."

"With a tool I will use it better. This was not easy to follow."

"I like more organization of info like differences, category, why they are similar to my release, etc. Also, app store had millions of apps not only five."

VIII. THREATS TO VALIDITY

As of any emperical study, we need to consider several threats to judge the validity of the results.

RQ1 - Prediction models for marketability: Parameter tuning (e.g., the number of trees, the size of the trees) affect the performance and accuracy of machine learning techniques. The same applies to the definition of the training and test sets as used in the experiment. We used exhaustive search for tuning SVM in conjunction with extra benchmarking for other two classifiers to tune parameters. While the variation of these parameters and time component is of potential interest, the detailed results of this type of analysis were not considered essential for this study. We also evaluated the accuracy of classifiers using ten times of 10-fold cross-validation as well as Leave-One-Out technique. Mapping releases between GitHub and Google Play for one of the four types (Type (2)) are uncertain. We used manual inspection of multiple releases to infer time and textual filters to mitigate the risk of mapping. The same is true for analyzing the success of marketed releases by analyzing the sentiment of reviews. Multiple factors in app store could be used as a success factor. However, reviews have been used more widely in former research.

RQ2- Usefulness of analogical reasoning: We validated the results of analogical reasoning by comparing the changes retrieved with actual changes over time to reflect how good was our reasoning. The measures we used to evaluate the conformance between former experience in the context of a release and actual changes applied in the under question release are proxy measures. We abstracted attribute changes by mapping them into discretized categories. While this added context to our reasoning, it might create construct validity in the process. We also could evaluate 58 releases using cross-app data and 19 releases using the same app information. We used almost $4/5$ of our data as the knowledge base and only used data of 2016 for evaluation purposes. We can be sure that the results of **RQ2** reflect the performance of our reasoning in most recent data. However, a more comprehensive evaluation by changing the size of the knowledge base and test set might give more insight. We also looked only into top 5 most similar releases considering the distribution of Euclidean distance but tuning this parameter for other contexts should be considered.

RQ3- Perceived usefulness of analogical reasoning by developers: We evaluated the results of analogical reasoning with relatively small number of developers. This is comparable with similar studies evaluating the initial results [17]. We

interpret these finding as being initial, considering them as the first step of a more comprehensive evaluation. The ultimate goal is to develop a tool to provide recommendations for developers and evaluate it within real world projects. We have made the first effort in this direction by asking them via a survey. Also, the survey results reflect the subjective opinion of the participants, which may be different from reality. We used convenience sampling to obtain responses which increase the risk of bias.

IX. Related work

Mobile app release planning: Release planning and feature prioritization is a decision-centric process and part of incremental and iterative software evolution. Release planning and management of mobile apps were studied. Nayebi et al. [7] performed a survey with app developers and users and found that almost half of the developers have a clear strategy for releasing mobile apps and the strategy impacts the end user. Villarroel et al. [17] introduced CLAP to assist prioritization of users' need using the app reviews. Three app developers evaluated positive usefulness of CLAP. Xia et al. [19] predicted crashing releases of mobile apps by in depth analysis of 10 open source applications with total of 2,638 releases. Beside a successful predictive model, they found that textual data of commit logs are the best predictors of crashing releases. Also, Martin et al. [5] analyzed releases which significantly changed user ratings and performed causal analysis over time.

Release readiness: Release readiness is a composite attribute of software products. In a recent study, AlAlam et al. [1] found that the definition of release readiness is not consistent among different publications. Ware et al. [18] defined release readiness as the attribute for systems' stability and maturity. Port and Wilf [11] related release readiness as the process of measuring uncertainty about the quality of the software to evaluate its fitness for distribution. Shahnewaz and Ruhe [14] also considered release readiness as the readiness of software to be released at a specific point in time. In contrast to the previous work, release marketability is solely focused on the market and customer acceptance using analogy with former releases.

X. Conclusions and Future Work

Marketability of mobile app releases is a new decision problem and we showed the practical importance of it. For a set of open source mobile apps and their releases, we applied three machine learning techniques to predict marketability of a release and the results showed the better performance of Random Forest performed best. For transitioning of a not successfully marketed release, we used analogical reasoning to retrieve experience of similar releases and extracted analogical changes. We internally validated applicability of analogical reasoning for guiding marketability decision by comparing analogical changes with actual changes. Moreover, our external evaluation of analogical reasoning with open source app developers showed the usefulness of this method. While results look promising but future research is needed to address some of the discussed threats to validity. Designing a support tool for more comprehensive empirical evaluation over time and by developers is of interest. The integration of this tool with phases of Case-based Reasoning for reusing, adapting and retaining experience to enhance the reasoning is highly valuable. In addition, identifying higher level proxies such as *effort* and *cost* to abstract release attributes such as *churn* or number of developers will provide tangible insight to plan for the app evolution.

ACKNOWLEDGEMENT

We like to thank the attendees of RELENG'16 workshop for their comments on the initial study of this research and for participating in our survey. Many thanks to the app developers who helped us evaluating the results. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12 and Alberta Innovates Technology Futures.

References

[1] S. M. D. Al Alam, M. Nayebi, D. Pfahl, and G. Ruhe. A two-staged survey on release readiness. In *EASE*, page To appear. IEEE, 2017.

[2] T. R. Eisenmann. Platform-mediated networks: definitions and core concepts. In *Harvard Business School Module Note 807-049*.

[3] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *RE 2014*, pages 153–162. IEEE, 2014.

[4] A. Hindle, D. M. German, and R. Holt. What do large commits tell us?: a taxonomical study of large commits. In *MSR*, pages 99–108. ACM, 2008.

[5] W. Martin, F. Sarro, and M. Harman. Causal impact analysis for app releases in google play. In *FSE*, pages 435–446. ACM, 2016.

[6] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, 2016.

[7] M. Nayebi, B. Adams, and G. Ruhe. Release practices for mobile apps–what do users and developers think? In *SANER, 2016*, volume 1, pages 552–562. IEEE, 2016.

[8] M. Nayebi, H. Cho, H. Farrahi, and G. Ruhe. App store mining is not enough. In *ICSE*. ACM, 2017.

[9] M. Nayebi, H. Farrahi, A. Lee, H. Cho, and G. Ruhe. More insight from being more focused: analysis of clustered market apps. In *WAMA*, pages 30–36. ACM, 2016.

[10] M. Nayebi, H. Farrahi, and G. Ruhe. Analysis of marketed versus not-marketed mobile app releases. In *RELENG 2016*, pages 1–4. ACM, 2016.

[11] D. Port and J. Wilf. The value of certifying software release readiness: an exploratory study of certification for a critical system at jpl. In *ESEM*, pages 373–382. IEEE, 2013.

[12] M. M. Richter and R. Weber. *Case-based reasoning: a textbook*. Springer Science & Business Media, 2013.

[13] I. Rus and M. Lindvall. Knowledge management in software engineering. *IEEE software*, 19(3):26, 2002.

[14] S. Shahnewaz and G. Ruhe. Relrea-an analytical approach for evaluating release readiness. In *SEKE*, pages 437–442, 2014.

[15] M. Shepperd. Case-based reasoning and software engineering. In *Managing Software Engineering Knowledge*, pages 181–198. Springer, 2003.

[16] T. D. Smedt and W. Daelemans. Pattern for python. *Journal of Machine Learning Research*, 13(Jun):2063–2067, 2012.

[17] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. Release planning of mobile apps based on user reviews. In *ICSE*, 2016.

[18] M. Ware, F. G. Wilkie, and M. Shapcott. The use of intra-release product measures in predicting release readiness. In *ICST*, pages 230–237, 2008.

[19] X. Xia, E. Shihab, Y. Kamei, D. Lo, and X. Wang. Predicting crashing releases of mobile applications. In *ESEM*, page 29. ACM, 2016.