App Store Effects on Software Engineering Practices

Afnan A. Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, Mark Harman

Abstract—In this paper, we study the app store as a phenomenon from the developers' perspective to investigate the extent to which app stores affect software engineering tasks. Through developer interviews and questionnaires, we uncover findings that highlight and quantify the effects of three high-level app store themes: bridging the gap between developers and users, increasing market transparency and affecting mobile release management. Our findings have implications for testing, requirements engineering and mining software repositories research fields. These findings can help guide future research in supporting mobile app developers through a deeper understanding of the app store-developer interaction.

Index Terms—Empirical Software Engineering, Mobile App Development, App Store Analysis

1 Introduction

There has been much recent progress in Software Engineering for App Stores using techniques that have drawn on many areas of software engineering research including, for example, software testing, software repository mining and software requirements elicitation [1].

In this study, we interview and survey app developers, regarding their interactions with app stores. Our aim is to better understand developers' practices when making apps. This understanding will help us determine the extent to which information from app stores affects developers' decision making and observe how the app store ecosystem influences engineering tasks during the app's development process. Moreover, our findings may guide future software engineering research in app development, maintenance and evolution.

Our study is the first to closely investigate how the app store ecosystem affects mobile software engineering during all life-cycle stages. Indeed, previous work has mainly focused on mobile developers' perspective regarding engineering aspects and implementation challenges introduced by the mobile platform [2] [3] [4] [5], briefly alluding to a few app-store-specific findings. For example, Nayebi et al. [5] found that developers are aware of how the app appears to potential users in the app store based on certain release strategies. Lim et al. [6] surveyed app store users reporting the importance of packaging decisions for mobile app success. A previous study by Rosen and Shihab [7] about the questions asked by app developers on Stack Overflow revealed that the most popular topic asked was app distribution, i.e. the requirements imposed by the app store owner for publishing apps.

- *A. Al-Subaihin, F. Sarro, S. Black, L. Capra and M. Harman are with the University College London, London, United Kingdom. E-mail: {afnan.alsubaihin.14, f.sarro, s.black, l.capra, mark.harman}@ucl.ac.uk.*
  A. Al-Subaihin is also with College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. E-mail: aalsubaihin@ksu.edu.sa.
  Mark Harman is also with Facebook, Inc.

Manuscript received February 1, 2018; revised XYZ 00, 20xx.

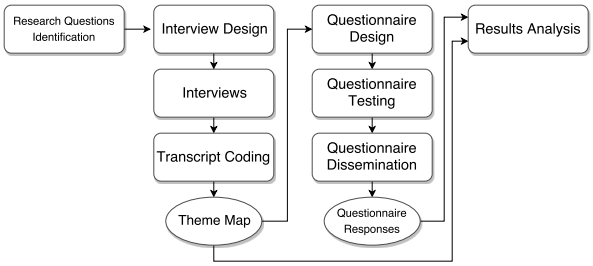

Fig. 1: The various stages of the study.

Our paper reports a survey of app developers' software engineering practices through an interview-and-questionnaire approach, allowing us to highlight open issues and challenges for the growing App Store Software Engineering community with a focus on relationships between app stores and software engineering research in several research areas including requirements, testing and software repository mining.

Our methodology combines an empirical study technique with a thematic analysis approach [8] [9], which is commonly used in behavioural sciences to analyse qualitative data [10] [11]. The stages of our methodology are illustrated in Figure 1. After formulating the research questions, data collection was conducted in two main stages. The first stage was a series of interviews with mobile development team managers and members. The interviews were then analysed and coded using deductive thematic analysis [11] and the results were used to design a questionnaire that was subsequently disseminated (stage two) to a wider audience in order to collect further quantitative data. Both the qualitative results of the interviews' thematic analysis (i.e., theme map) and the quantitative results of the questionnaire were used to explore and deduce the findings reported herein.

The findings of our study make several contributions that give evidence to support the perceived differences between app store development and more traditional software development. The principal findings about app stores

themselves are:

1) **Closed loop:** The gap between developers and their users is closed by the facilities app stores provide. They denote a channel of communication that directly connects users to developers. For example, our study findings indicate that 51% of respondents frequently perform perfective maintenance based on user's public feedback in the app store. This may be introducing new communication channels and approaches through which requirements are gathered and acted upon.
2) **Transparent market:** The ability for developers to, not only experiment with competitors' products, but also to be able to witness, in real time, the performance of these products in the marketplace, constitutes a considerably more transparent market for deployment of software systems compared to more traditional software markets. For example, our study reveals that 56% of respondents frequently elicit requirements by browsing similar apps in the app store. This is one of the motivations for app store mining and analysis: to better understand the marketplace, and emerging trends from competitors and overall user and developer behaviour within app stores.
3) **Tailored release strategy:** The gap between releases is shorter in app stores than for more traditional software systems deployed such as shrink-wrapped software [12]. It is also governed and constrained by a third party: the app store. Our results report that 54% of respondents adopt a release strategy that is influenced by the app store's regulations. This has implications for innovation and rapid response to technical and market developments.

One of the interesting properties of app stores is the way in which these stores cut across different software engineering concerns, raising inter-related questions and research problems for different software engineering activities [1]. More specifically, the findings of the survey have actionable conclusions for researchers and practitioners from several software engineering sub-fields, including:

1) **Requirements engineering research:** We report evidence that suggests that developers are almost as concerned about reported features (both wanted and unwanted) by their users, as they are with, for example, bug reports. This finding highlights the way in which app stores provide a direct communication channel between developers and their users. It is also particularly interesting to note that feedback is used by developers, to also identify *un*wanted features, hinting at the growing prevalence of the need to remove/modify features as well as the continuous need to identify new features to add.
2) **Mining software repositories:** Our study reveals that app store developers place importance on screenshots (in order to gather features for their apps). Therefore, although existing work on mining textual information from feedback and reviews and ratings is valuable (and also used by developers according to our survey), the current lack of studies on the use of images as a source of information needs to be addressed. Mining user interfaces is typically undertaken in communities such as the Computer-Human Interaction (CHI) [13] and User Interface Software and Technology (UIST) [14] communities, which may also find novel research challenges in this new area of application. We envisage that 'user interface mining' may find new applications in app store mining and analysis.
3) **Other software engineering disciplines:** We find a strong belief among developers that app stores contain information that help developers maximise the chance of success for their products, thereby motivating and partly validating app store analysis and mining as a research area. However, the quality of code is not identified as being the strongest influencing success factor; other aspects such as user experience, visibility, novelty and brand are accorded notably higher importance by developers. This indicates, for example, that work on code 'smells' in app store code, while important, needs to be complemented by, and combined with, work on user experience and other human- and business-facing aspects.
4) **Business Community:** Our study has important findings for those working at the interface between software engineering and business considerations. That is, while developers claim it is important to have someone in their team concerned with marketing and business intelligence, they also report that this person tends to be self-taught and relies on experience rather than formal training.

Our study also has other findings for these sub-fields, and several findings concerning app testing of relevance to the wider software testing branch of research [15]. The observation that a study on app store developers can have actionable conclusions for so many different software engineering communities, highlights the crosscutting nature of this relatively recent phenomenon in software development and deployment. Clearly, as app stores develop further, there will be a need for further studies and surveys. The authors hope that the findings from this survey will provide a useful reference point for such further studies and analyses.

2 Methodology

To study developers' practices when developing for mobile app stores, we followed a mixed method drawing from **survey and case study empirical research methodologies** [16].

There are two reasons supporting this choice of methodology. Firstly, case study research is a way of analysing contemporary phenomena that are difficult to separate from their natural context [17], which is the case for app stores. However, as this is a global phenomena affecting the majority of the population (app developers), it is not strictly a case study, so we also followed a survey technique to collect data from a sample of the affected population using two data collection methods (namely, interviews and questionnaires).

This particular research will aim to be both an exploratory qualitative empirical study as well as a descriptive one [18] [17]. To ascertain and gauge the level to which app store affects developers' view and practices, we conduct exploration activities first. Specifically, we take a comprehensive view of the aspects in which app stores may affect developers' activities. This study is designed following the

case study research guidelines by Runeson et al. [17] and survey research guideline by Kitchenham and Pfleeger [19].

We coupled our methodology with **deductive thematic analysis** to analyse qualitative data [8] [9] [11].

Thematic analysis is a method of analysing textual content and deriving patterns of thematised meaning from it. Similar to grounded theory [20], thematic analysis originated from the social sciences and has since been utilised in computer science empirical research involving human subjects. Wohlin and Aurum [21] report it as one of the qualitative analysis methodologies in their decision making structure for empirical software engineering research; Cruzes and Dyba [22] formalize an extension of thematic analysis to thematic synthesis in software engineering research.

We have selected thematic analysis due to its flexibility, ease of understanding and independence from theory. While grounded theory allows for theory-agnostic analysis of data, thematic analysis can be conducted within a theoretical framework [11]. In this study, we operate under the software engineering life cycle stages (as per the software engineering body of knowledge areas 1-5 [23]) as our theoretical framework, hence we follow deductive thematic analysis [8] [11] (as opposed to inductive analysis or grounded theory [20]), using semantic themes as developers are expected to have sufficient domain knowledge eliminating the need for latent theme inference. Furthermore, our thematic analysis method follows the essentialist/realist method and not a constructionist one as we assume a simple relationship between participants' answers and meaning [8] [11]. In conducting thematic analysis for this study, we follow the guidelines provided by Braun and Clarke [11].

Data collection was conducted by surveying main stakeholders of this research who are mobile app owners and developers. Similar to the scientific method of gathering information via surveys, gathering initial insights was done using interviews. Then, based on the interviews' initial findings, we designed a questionnaire and disseminated it in developers social circles. The questionnaire is important in order to validate the findings with larger consensus. Through analysing the interviews we identified areas of interest on which the paper focuses and sheds more light. Certain patterns of responses (whether with more consensus or disagreement) that pertain to the research questions and promise valuable and deeper understanding of the software engineering practices were highlighted when writing the survey questions. The survey was designed in order to investigate in more detailed and systematic way all that relates to the specified research questions.

3 Study Design

The stages of our study are depicted in Figure 1. As both empirical research and thematic analysis studies rely on proper identification of research questions, the first stage is setting the questions to be emphasized and answered. Phase two is dedicated to the exploration and preliminary gathering of information. This is done by interviewing app developers and discussing their views and current practices. During this phase the interview structure is designed with a set of potential topics and questions to be explored in light of the research questions; then the transcripts of the interviews are analysed and coded using deductive thematic analysis resulting in a theme map. The third phase consists of collecting data by disseminating a questionnaire (designed in light of both the research questions and the insights gathered from the interviews and the theme map) to communities of interest. In the following subsections we discuss in greater details each of the phases depicted in Figure 1.

3.1 Research Questions

The research questions we aim to answer in this study cover two main areas of interest: app store's effects on software engineering processes and possible success criteria and skill sets that emerge due to app stores.

App stores are now major application deployment portals that drive the user's application discovery process. A large scale study of mobile users' tendencies by Lim et al. [6] unveiled that the majority of users rely on the app store to discover new apps: 73% of more than 10,000 respondents visited an app store at least once a month; whereas only 9% did not rely on an app store to download apps. Another major aspect of the app store is users' ability to rate the quality of apps, post feedback, comments and reviews; thus effectively opening a channel of communication between developers and users. Therefore, we believe user feedback in the app store may go beyond its recognised benefit in general markets in establishing trust of the seller's ability to deliver on their product's promise [24]. Furthermore, app stores are designed to collate similar apps together. Developers and managers are able to find apps in the same application domain including their specifications and performance in the app store environment.

This gives us ground to suspect that the app store's configuration may have an effect on the evolution of apps. which motivates our first research question:

RQ1. How does the app store ecosystem affect the software development life cycle processes? For this research question, and to set the scope of this paper, we consider the Software Engineering Body of Knowledge (SWEBOK) [23] areas 1 through 5 as the software engineering life cycle stages; namely software requirements, design, construction, testing and maintenance. The Software Engineering (SE) research community has indeed highlighted the opportunities and challenges introduced by such an ecosystem [15] [25]. Software engineering researchers sought to leverage information found in the app store to guide mobile developers during requirements engineering [26] [27] [28] [29] [30] [31] [32], testing [33] [34] [35] [36], maintenance [37] [38] [39] [40] [41] [42] and release management [43] [44] [45] [46] [47]. In posing this research question, we aim to observe the current involvement of information extracted from the app store in guiding the software engineers' effort in each of the aforementioned stages.

The study by Lim et al. [6] also reported that the market is dominated by a handful of app stores, chiefly Google Play and the iOS App Store. This accounts for high density of potential users, exposure and total downloads for apps. Furthermore, the ecosystem offers low barrier to entry giving rise to the number of offered apps making it an increasingly competitive marketplace. We investigate

whether such high competitiveness and emphasis on user acquisition and retention increases the types of considerations that the development team takes. Hence, we ask this research question:

RQ2. What new sets of best practices and skill sets emerge due to app stores, if any? The low barrier to entry also facilitated smaller teams of developers (2-5 developers) to publish apps that are still deemed viable and competitive [4] [48]. We investigate the types of activities that mobile development teams carry out and the skills required that are influenced by the app store ecosystem and are outside of the recognised software engineering life cycle activities considered in RQ1.

App stores do not only provide browsing and search capabilities to users, but also employ quality measurements to provide curated content and refined 'lists' to users. Lim et al. highlighted that, in order to discover new apps, 37.6% of respondents browse the app store randomly, 34.5% check the top downloads charts and 25.8% look at featured apps. This highlights the major role that app stores play in driving success to mobile applications. To observe the involvement of app stores in success and its measurement, we ask the following research question:

RQ3. How is success perceived and measured by developers in the app store environment? Previous research seems to regard app rating as a proxy for quality (and therefore success), thus investigating the relationships between user rating scores and apps' user reviews content [49] [50], release plan [45] [51] [46], features [32], software metrics [52], security [53], code churn [54], faultiness [55] [56], underlying hardware/architecture [57] [58] [59] and many other software factors [60] [61]. By contrast, we shift the focus to app developers and owners' view on what defines 'success', thereby uncovering other app-store-specific metrics which developers monitor. Furthermore, we aim to uncover the relationship between success and the developers' perceived quality of the app. This research question does not look into the role of the market for success, but rather investigate whether the market introduced new metrics through which developers perceive the success and quality of their app.

3.2 Interviews

Interviews were conducted to initially explore developers' interaction with app stores before and after release. The interview protocol is described in the upcoming section followed by a description of the participating sample and data analysis method.

3.2.1 Protocol

The interviews were semi-structured and followed a funnel model where questions are generic at the beginning and become more specific as the interview progresses. The funnel approach was selected to permit the conversation to flow naturally instead of controlled question-answer cycles. This allows the interviewees to be put at ease thus talking freely about what they deem important and pertinent with regards to the general topic. Then, taken from the current topic of conversation, the interviewer refocuses the conversation to a more precise subject of interest. This method suited the exploratory and observational goals we required of the interviewing process.

The interview questions were brainstormed and meant to be near-exhaustive in nature. They, we believe, cover most aspects of contact between developers and app stores. All interviews were conducted by one of the authors except for one which was attended by a second interviewer.

The interview plan contained 40 questions that the interviewer, ideally, sought to cover. The set of drafted questions are in Table 1. Since the interviews were semi-structured, this plan served only as a reference for the interviewer and was not enforced. The plan highlighted some of these questions as suggested conversation starters within broad topics. Using this way of conducting the interview, interviews typically flowed smoothly and the developer answered most questions without interrupting the flow of the conversation.

3.2.2 Participants

The selection of interview participants relied on purposive sampling where participants had to be individuals involved in the production of an app in the app store. In selecting participants, we sought a broad set of sources for opinions with regards to team roles including developers, managers and app owners. Since this is an exploratory step, we are not aiming to make any generalizable discoveries and, therefore, relied on convenience recruitment[1] of participants.

We have interviewed a total of 10 app development team members. The interviewees were recruited through UCL Advances[2] and via social contacts. From there, a snowballing recruitment technique was carried out in which the developer was asked to recommend other colleagues and connections for the interview.

Table 2 reports the interviewed sample along with their respective experience demographics. Among the 10 interviewees, 7 had formal education in an engineering/computer science related field. Fields that are outside of the faculty of engineering were dubbed non-technical. The team sizes of participants were between 1 and 17 developers. The interviewed sample had between 4 and 27 years of experience in software development. The number of apps they have developed spans from one app to 20 apps. The degree of exposure of the sample's apps also ranges between apps that have been downloaded 100 times to apps downloaded 800,000 times.

3.2.3 Data Analysis

After transcribing recorded interviews, data analysis was carried out to identify emerging concepts from the corpus. This was conducted using Thematic Analysis [11]. Thematic analysis, as the name suggests, employs the concept of themes when analysing textual data. Thematic analysis requires reading the scripts intensively before coding the responses in light of the research questions. The codes are tags that interpret certain responses and help identify their

1. Convenience sampling is the most common sampling technique especially in laboratory psychology research where participants are mostly volunteers [62]. It is a non-probabilistic sampling method that is used for preliminary exploration of a phenomenon since it is less costly than probabilistic methods.

2. UCL Advances is a project by the UCL Economic Challenge Fund that contains a large contact base of entrepreneurs and app owners through its UCL testing app lab.

TABLE 1: The set of interview questions.

***First* Background and demographics**	***Fourth* App Features**
Mobile platforms / app stores	What do you think are app features?
Other development experience (desktop, web, etc.)	Do you think it is easy to find something to implement?
Years of Experience (Development and Mobile)	How do you decide which features to include at the beginning?
Application Domain	How do you decide which features to add/remove later on?
Independent or corporate?	How do you gauge the success of a certain feature?
Number of developed apps (How many of these app were released in an app store?)	How do you decide which app features to include in the app description?
Dedicated marketing team (or person?)	Do you look into competitors features to identify technical trends?
Team Size	***Fifth* User Feedback**
Number of total downloads, ratings, feedback..	How do you know why users downloaded your app?
Revenue model	(Android): How do you know why users uninstalled your app?
***Second* Generic Views**	How/why do you encourage users to rate/review/share your app?
How would you describe your experience in dealing with app stores?	Do you actively respond to user reviews and Feedback? How? Why?
How are app stores different from any other deployment platform/method?	How do you respond to user reviews and feedback? What are the reasons as to why?
How does it make development easy how does it make it difficult?	To what extent does user feedback affect next releases?
What are the important factors of success in app stores?	***Sixth* Tools and Metrics**
***Third* App Packaging**	How do you measure your success over competitors (metrics)?
What do you think the most important criteria when selecting screenshots/ description/ tag line?	Do you find analytical tools provided by the app store enough to support your decisions (previously discussed)?
(Android) Do you think app permissions matter? Why?	What extra tools do you use, if any?
What do you think causes users to download your app? (same: to uninstall your app?)	What procedures do you take to advance your competitive advantage?
How do you decide on a revenue model?	..and what metrics do you think useful to enhance the app?
Did you ever have to change your app's price/revenue model? Reasons behind the change?	Do you think you need analysis and statistics that encompass the entirety of the app store?
How do you decide in which categories to release your apps? Do you think it matters?	

TABLE 2: Demographical data of the developers interviewed.

Participant	Formal Education	Years of Experience	Team Size	Team Role	Number of Apps	Success Metric
P1	Technical	4	1	All	6	-
P2	Non-Technical	-	6	Owner/Manager	1	2,000 Ratings
P3	Non-Technical	2	6	Owner/Manager	1	100 - 500 Downloads
P4	Non-Technical	7	9	Owner/Manager	1	32 Ratings
P5	Technical	6	1	All	3	200 - 250 Downloads
P6	Technical	17	6	Owner/Manager	2	140 Ratings
P7	Technical	27	17	Developer	1	1,000 Ratings
P8	Technical	10	5	Owner/Manager/Marketing	20-30	800,000 Downloads
P9	Technical	-	4	Owner/Tester	3	10,000 Downloads
P10	Technical	-	3	Developer	3	15,000 Downloads

topic. The codes are then clustered to form a theme map that serves as the visual representation of the main findings of the interviews.

Transcript coding: In light of each research question, the data is scanned in order to be assigned a code. Interview codes represent a certain area and tags certain attitude, opinion or knowledge expressed by the respondent regarding that area. Due to the extensive length of the interviews, the coding process helps tag only relevant responses with regards to the research questions. The first author initially tagged the interview transcripts, then compiled a list of all the codes and example instances of each of the codes. The list was then revised by two other authors to ensure their representativeness with regards to the research questions with consensus. After the revision, the first author revised the tagged corpus, this has been done in two iterations. Due to the nature of the codes, they were allowed to overlap and merge/divide throughout the revision process.

Figure 2 shows the final list of codes for each research question.

For the **first research question**, the codes are the typical software development phases according to the Software Engineering Body of Knowledge (SWEBOK) [23] as the research question investigates practices related to the software lifecycle processes. This practice of using the processes as transcript codes is part of the deductive nature of the thematic analysis methodology in which the codes follow a certain pre-known taxonomy. It is also a practice done by other similar software engineering qualitative research [4] [63] [64].

The **second research question** centres around new emerging skill sets and roles required of mobile app development team members. The codes selected for RQ2 pertain to the possible assigned tasks for team members and to what degree do they deviate from those of a classical software team roles. The second code (implicit know-how) highlights exhibiting knowledge or certain app store-specific best practice that was not formally learned or part of the respondent's education. The final code (non-technical activity) is for highlighting any skill that the respondent is exhibiting or discussing that are not engineering-related.

The **third and final research question** pertains to performance measurements that are particularly important for mobile apps distributed through app stores. The first code (determination of success goal) highlights the clarity and determination of a success goal for the release of the app and whether that goal is app-store-specific such as being featured in the app store's main page. Other codes tag the

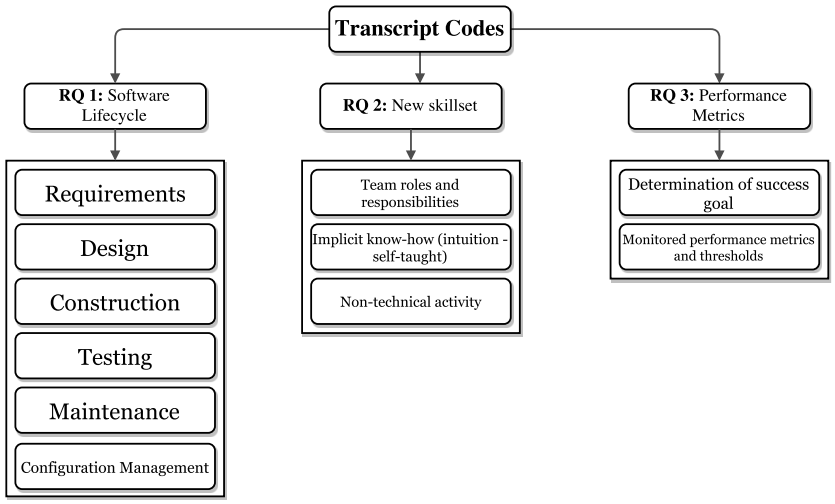

Fig. 2: Transcript raw data codes used to tag responses: These codes represent recurring topics and certain responses of interest. This is the first stage of interview analysis. The codes and their content are then used to deduce themes in Figure 4

various app store analytics and set thresholds.

Deducing Themes: Codes are then collated into a group of potential themes. A theme represents a unit of an emerging pattern of responses with regards to a certain topic and/or research question. Themes do not certainly perform a one-to-one mapping to research questions and they go through rigorous revisions as the researcher goes through the data in several passes. From analysis, thematic coding results in a theme map. The theme map reflects the main findings observed from coded responses and their relationships with one another. This process produces a rich, detailed description of the data without hindrance by data that are not relevant to the research questions. This has been carried out by the first author and then revised by three more authors in a collaborative session until a consensus has been reached (over three iterations).

3.3 Questionnaire

Based on the emergent topics of interest extracted from analysing the interviews (Figure 4), and in light of the research questions, a questionnaire was used to ascertain findings, explore new ideas and measure the prevalence of some practices. The following subsections describe the questionnaire, its design and the participating sample.

3.3.1 Design

In designing this survey, we followed the guidelines provided by Kitchenahm for personal opinion surveys [65]. The initial design comprised of 118 statements and questions that have been drafted in several collaborative sessions among four of the paper's authors. The survey is divided into subsections representing themes of activities in addition to the demographics section. The survey draft went through several revisions where we have removed questions that we deemed to be open to interpretation based on the respondent's experience and may be ambiguous or misunderstood. An example of an unclear question was: 'I prefer releasing an alpha/beta version of my app on the actual app store rather than one specific for testing.' (developers may not understand what is meant by 'one specific for testing' and may interpret it differently). Furthermore, we have prioritised questions with higher relevance to the software engineering community and so did not include ones such as: 'When I find an app that has a similar main functionality, I still can have a competitive advantage." After eliminating repetitive, unclear and questions deemed irrelevant (22 in total), the questionnaire ended up with 96 questions. The questionnaire is divided into these sections: Demographics, Software lifecycle (Idea conception and requirements gathering, design, construction, testing and maintenance), Emerging new skill sets and finally, Metrics. We have first conducted a pilot study where we invited developers to fill the survey in read-aloud sessions in which they read questions out loud as well as externalized their thinking process. A total of six developers reviewed the questionnaire questions and gave feedback. First of which was their complaints regarding the length of the questionnaire. Based on that, we removed a few more questions that were lower in priority; in addition to merging the last section of the questionnaire with previous sections. Additionally, we arranged the questions such that demographics only appear at the end of the questionnaire except for two easy questions that serve as a warm-up. Another valuable insight from how developers filled the questionnaire was their consistently mistaking 'design' in a software engineering process sense with the process of graphic design and building user interfaces. This and other inconsistencies in the meaning of certain terms were observed in the read-aloud sessions and were thus corrected in the survey. Two questions were deemed totally unclear and were therefore rephrased. The final questionnaire contains 11 short sections and 79 statements grouped into 42 questions[3]. The questions' answers are 5 Likert items on the Likert scale that represent degrees of agreement, frequency, interest or importance. The survey also includes multiple choice and open ended questions. We have elected to make all the questions optional in order to mitigate the challenge of the length of the survey. This means that each question has its own sample that can be a subset of the surveyed sample. By making the questions optional, we ensure the

3. The final survey is available at http://afnan.ws/survey. A pdf version of the survey can be downloaded from http://afnan.ws/survey/survey.pdf

certainty of the response since no respondent has to reply in order to progress further in the survey. Another approach we used to mitigate the length of the questionnaire was branching: Based on the respondent's answers to certain questions, the control flow will skip questions that are irrelevant. For example, we ask the participant if they ever released more than one version of their app, if the answer is no, we skip questions relating to release management and perfective maintenance and proceed to the subsequent section.

3.3.2 Participants

The survey was disseminated via posters and flyers around UCL campus, email to interest groups as well as social media. The flyers were also passed around 2 research conferences. Cold calls were also posted to several mobile developer groups in the professional social network LinkedIn[4].

The total number of respondents to our questionnaire is 186. However, since all questions are optional, each question has its own sub-sample of respondents. Of the 186 respondents, 103 have completed the questionnaire. The maximum number of respondents for a question is 185 and minimum is 107, average number of respondents over all questions is 133 with a median of 119 (barring open-ended questions and those in a branch). Of all those who entered the survey, 57% answered 100% of the questions.

The survey responses came from developers based in 36 different countries. The majority of the respondents are aged between 25 and 34 (50%). We consider the responses of any of the mobile application development team member regardless of their role. The majority of the responses originated from developers (57%), remaining roles include: managers (14%), marketers (8%), and those who assumed multiple team roles (18%). The years of experience in software development ranged from less than a year to 20 years, with an average of 7 years and a median of 5 years. The respondents reported an average of 4.2 years of experience in developing mobile apps specifically; with a median of 4 years, a maximum of 15 years and minimum of 1 year. The majority (84% of a total of 101 respondents) reported having a formal education in a technical/engineering field whereas 21% had a business-related formal education. The average size of teams reported was 5 working full time with as low as 1 and as high as 66 team members and a median of 2. A total of 103 respondents informed us of the platforms they develop for: 72% publish in iOS app store, 75% in the Android app store (Google Play), 12% in Windows Phone store and 11% published to other platforms: Amazon, Blackberry and Samsung stores, Apple TV and others. The respondents' apps had varying degrees of exposure, the largest had 10 million active users and the lowest had 14. The sample had a median of 1,500 active users and a mean of approximately 300,000 active users.

3.3.3 Data Analysis

In reporting the results of the questionnaire, we merge the number of respondents of the two extreme Likert items to simplify interpretation. For example, in the Likert agreement scale we merge the number of those who agree and strongly agree to report the overall agreement rate and also merge the number of those who disagree and strongly disagree to report the overall disagreement rate. Moreover, we report the weighted average response in order to differentiate higher agreement/disagreement (or its equivalent in other scales), especially when ranking popularity of answers. The weighted average response of each statement is the average of scores assigned where strong agreement (or its equivalent) is scored 5, agreement is 4, neutral is 3, disagreement is 2 and strong disagreement is 1.

In summary, the agreement percentage gives the ratio of respondents who agree/strongly agree (or their equivalent) with a statement; whereas the weighted average score gives insight on how strongly respondents agree with this statement.

The questionnaire's quantitative findings are reported augmented with relevant qualitative ones extracted from interview transcripts to help aid the reader in understanding some developers' point-of-view regarding certain patterns of responses or opposing opinions. The quotes were selected by backtracking through pertinent themes and their codes.

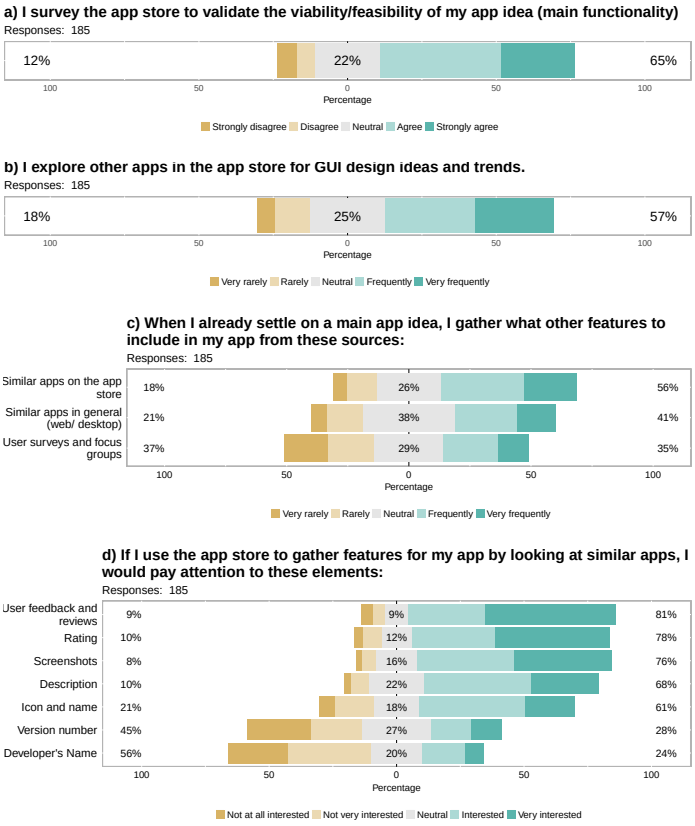

Fig. 3: **RQ1.** Responses to questions regarding the initial phases of development.

4 Findings

This section reports the findings of the empirical study. The findings from the interview phase are presented in Subsection 4.1 in the form of the resulting theme map, which guided the design of the survey. Since the main goal of conducting interviews was to guide the survey design and due to the limits of the interviewed sample, we do not

4. LinkedIn: http://www.linkedin.com

discuss the results of the interviewing process in isolation but discuss them across Sections 4.2–4.4 in conjunction with the quantitative results of the questionnaire[5] to augment it with qualitative insights.

4.1 Interview Analysis Results

Figure 4 shows the results of the theme map deduced from the interview analysis. This theme map helped us construct an insight into the state of interaction between developers and app stores. Its main purpose was to pave the way towards designing the questionnaire.

The first theme map component is **Software Process**. In terms of the **requirements gathering** phase, the app store has been proposed as a method of exploring an application domain, validating ideas, checking ideas against redundancy and exploring the possibility of reuse. User expectations of features required of apps in a certain domain seem to be particularly of interest.

In terms of **design**, designing user interfaces is important as it will translate to a screenshot in the app's page. Screenshots are viewed as a big determinant of whether the user decides to download an app. Throughout discussions regarding designing user experiences, developers expressed exasperation regarding following strict OS vendor and app store owner's guidelines especially since the changes are often out of the team's control and interfere with the team's plan.

At the **construction** phase, developers include specific pieces of code that ask the user to rate the app and redirect them to the app store for that purpose. App permissions are a worrying factor during development as importing unnecessary APIs might bloat the permissions list thus making an app less desirable. Furthermore, during the construction phase, developers settle on tracking strategy in order to implant tracking code within the app.

During alpha and beta **testing** developers sometimes choose to distribute testing versions via the app store which gives them more feedback and exposure. Developers expressed interest in gauging users' interaction with the app within the app store ecosystem as part of the beta testing phase.

Maintenance has been found to be the most affected by the app store ecosystem. Developers expressed interest in users' feedback and rating as a major driving factor for new releases. Many of the interviewed developers mentioned that the app store ecosystem has enforced a certain release plan for their apps.

One aspect of interest was the practice of monitoring similar competing apps, especially during gathering requirements and perfective maintenance. Interviewed developers responses were divided regarding that particular practice. Those who declared it dangerous quoted addiction towards constant comparison and the eventual uselessness of having an app that is a copy of another. Opposing those views are developers who said that keeping an eye on competitors is necessary in such a competitive environment as the app store. However, they said that it is important to monitor in order to differentiate the app from similar apps and gather certain features from similar apps for perfective maintenance.

The second theme is related to the interviewees' app store know-how, aptitude, general **best practices** and other activities outside of the well known software lifecycle practices. These are patterns of knowledge that are not evidence- or theory-based. This knowledge appeared as **intuitive** and not the result of formal training and in some cases heavily relied on observation of other apps in the app store. When a respondent shows propensity for **evidence-based** knowledge of app store management it was either the effect of formal training or the outsourcing of such tasks.

Finally, the last theme is app's **success and performance monitoring** in the app store environment. We have detected variation in terms of perceived success of an app in the app store. A large number of developers did not emphasize the quality of code, documentation, or overall architectural design for building a good software product. On the other hand, app store analytics are an often mentioned topic in our interviews. The respondents quoted many metrics they deem important to monitor to gauge the success of the app and its perceived quality by users. There were no global threshold for any of the metrics but an upward trend is certainly desirable.

4.2 RQ1: Lifecycle Processes

App stores, as they reach an almost-monopolization of mobile app distribution with regards to a particular operating system, are prone to introduce some changes to how developers carry out software engineering tasks. For example, we anticipate, due to the app store regulation, for it to change the way developers plan releases. Additionally, as app stores provide a rich environment in which users leave feedback, including reporting bugs and requesting features, for it to affect developers' requirements elicitation activities. The following sections go through our findings affecting requirement elicitation, testing, maintenance and release management.

4.2.1 Requirements Elicitation

To developers, not only can the app store serve as a distribution channel, it is also a large repository of apps. In this repository, access to similar and competing apps have never been easier. Not only can developers see how are other apps presented, but users' reaction to them. By sifting through user comments, they can identify common bugs, appreciated features, requests and usage scenarios of apps in an application domain of interest.

We hypothesized that, naturally, developers follow and observe similar and competing apps. However, during the interview process, we observed polarised results regarding this particular activity. Certain developers expressed negative connotations with such practices *"You'll never win if you are stuck playing catch-up"* one developer expressed. On the other hand, others stated it as a necessity for survival. Among those one who said: *"I think it's vital to know what else is out there. You have to get a sense not just of what you are competing with but how it is delivered. Looking at [competitors'] reviews is something that we did to see if the features we included were appreciated by people or whether they were just not mentioned or actually thought to be waste of time. So, the app store*

5. The questionnaire responses can be viewed online at: https://www.surveymonkey.com/results/SM-83LPDRNW8/

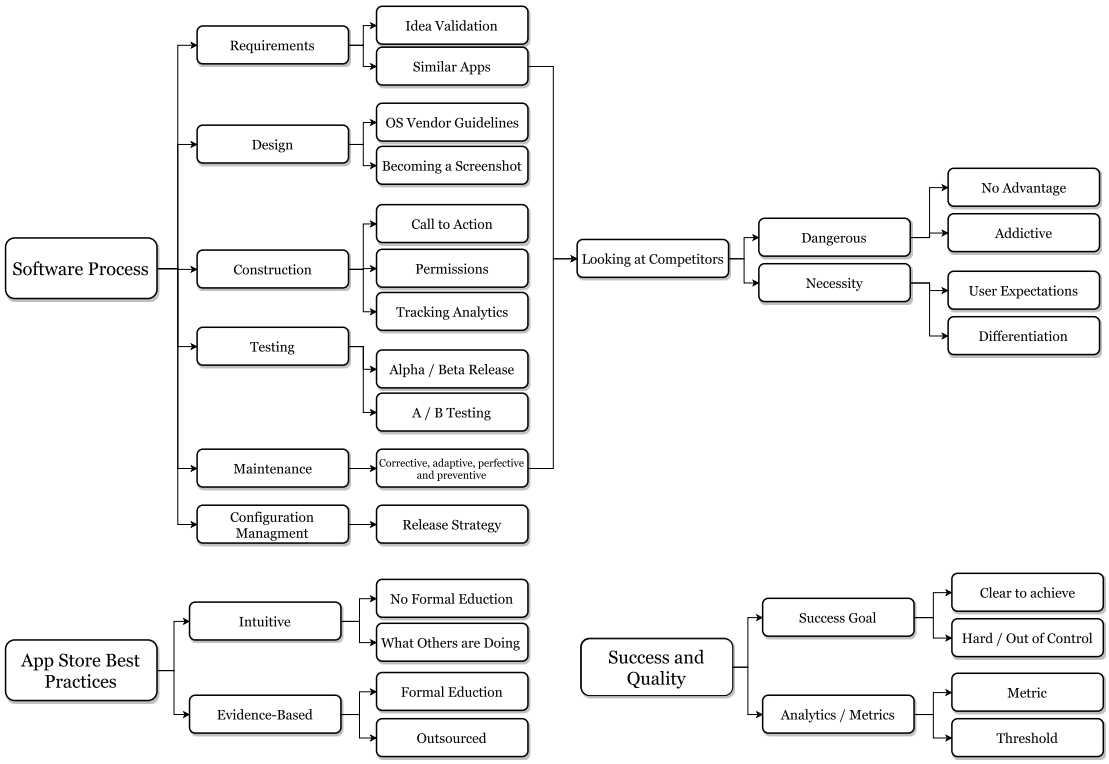

Fig. 4: Thematic analysis findings in the form of a theme map. The theme map summarizes the data patterns found in the interview transcripts relating to the research questions.

provide a rich stream of information about what works and what people think of the app itself."

In the questionnaire, more than half respondents surveyed the app store at the initial phases of development for both validating the app's idea (65% answered agree/strongly agree) or for user interface inspiration (57% answered frequently/very frequently). Of those gathering requirements for their app, the most frequent source has been other similar apps (56% answered frequently/very frequently scoring a weighted mean of 3.55) followed by similar desktop and web apps (41%, 3.30) and user surveys and focus groups (35%, 2.92). Figure 3 shows a breakdown of answers.

When asked about which elements of other similar apps are investigated, the three most popular were: user feedback (81% answered interested/very interested scoring a weighted mean of 4.19), rating (78%, 4.09), app's screenshots (76%, 4.05) and description (68%, 3.83). On the other hand, over half of respondents did not find the developer's identity of interest (56% not interested, 2.51) and 45% were not interested in how many versions competing apps released (2.69) (Figure 3-d).

Some developers clarified that this is not done just for the purpose of comparison with other apps, but for understanding a specific market and the user's expectations for a particular application domain. One developer clarifies: *"I focus on understanding the experience of the users and customer development more than comparing my idea to other apps. If I'm browsing other apps I'm either looking for inspiration in design or other ways to solve my problem."*; A survey respondent further clarifies: *"I found that app-users (especially social media) have*

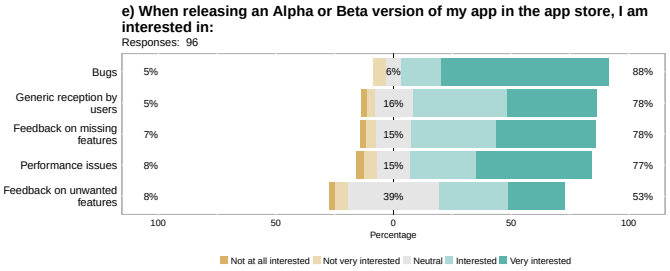

Fig. 5: **RQ1.** Responses to questions regarding the testing phase.

been accustomed to a bunch of features that become de facto a must for a new project."

> *For **requirement elicitation**, app stores provide a large stream of information and historical data to software engineers. The majority of surveyed developers use it to explore apps related to their application domain to gain an understanding of the expected user experience and anticipate features.*

4.2.2 Testing

App stores provide developers with a rich channel to conduct pre-release testing. Additionally, the rating and comment/review sections can give developers much to process. In this section we review developer responses regarding intent when pre-releasing the app in the store.

When a sample of 171 developers were asked if they indeed release alpha and/or beta versions to the app store, 59% answered yes. Among those who answered yes, we further investigated what they hope to uncover by pre-releasing the app. The distribution of the answers is depicted in Figure 5.

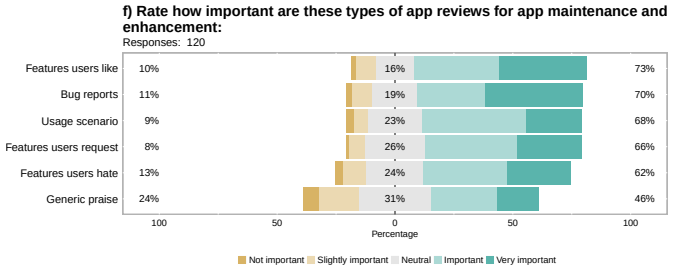

Fig. 6: **RQ1.** Responses to questions regarding the usefulness of user feedback.

Perhaps unsurprisingly, finding bugs garnered the highest interest. The finding with least interest was unwanted features; however, a previous study [66] reports that 78% of 106 surveyed developers rated functionality deletion as important and/or more important than adding new features. This may suggest that, while developers deem the removal of functionalities important, they might not necessarily discover which features to remove during alpha/beta testing.

More interestingly, we observe that 78% of developers who release alpha/beta versions of their apps in the app store, are also interested in the generic reception of the app and the type of ratings, reviews and social hype it would garner (4.11 weighted average).

While the large amount of users that find and download a pre-release of the app is a good thing, some developers warned that over-exposure of the app might negatively impact the app's image if it has major issues. One developer wrote: *"We release the app in a staggered way so that a subset tests it and if something goes wrong we can early roll back to a stable version and fix any major bugs."*. A survey respondents also concurs: *"Premature social hype could doom the project."*

> *For **testing**, many of the developers use the app store to publish pre-releases. In addition to finding bugs and discovering enhancements, 78% of those developers also stated that they release the alpha/beta version to test the general reaction of users in the form of ratings and social hype.*

4.2.3 Maintenance

When the app is published in the app store, developers come to maximum contact with users. The ratings, reviews and recommendations start coming in. We investigate the extent to which developers incorporate user input from the app store into their maintenance strategy.

During the interview process, we have detected that developers regard user reviews posted in app stores as a bug reporting and feedback collection tool in addition to a marketing tool. Several developers informed us that having a healthy proportion of negative feedback is an important nudge in the right direction *"[Positive Feedback] doesn't really help me. It should contain some information to help me improve the app, either something is wrong, something is missing, something they want,"* one developer expressed. Due to the rapid iterations typical of mobile apps release plans, one developer informs us that *"those bad reviews is what makes a really successful product."* To some developers, the app store is just another bug reporting and user engagement channel, albeit a prolific, public one: *"We see what is being asked the most, regardless of the channel, we get the feedback from the different channels and aggregate them."*

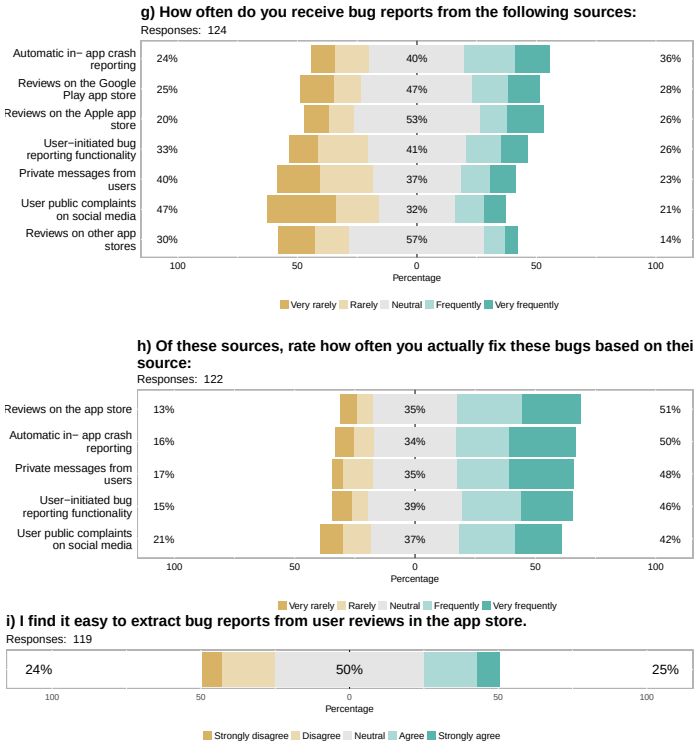

Fig. 7: **RQ1.** Responses to questions regarding the usefulness of user feedback for corrective maintenance.

Another developer highlights the importance of feedback coming through the app store rather than any other channel: *"Because whenever you're frustrated you want to voice your frustration immediately. And the only way to communicate with the developer, people think, is the app store."*

Since user reviews in app stores contain large diversity of information including complaints, praise, usage scenarios, feedback on features and bug reports (a taxonomy devised by Guzman et al. [67]), we asked developers about the types of feedback that they deem particularly important for app maintenance and enhancement. Developers rated high all of the suggested types as seen in Figure 6. Scoring highest (according to weighted mean) are bug reports (70% agreed/strongly agreed scoring 4.01 weighted mean), features users like (73%, 3.99) was next, followed by feature requests (66%, 3.86), usage scenarios (68%, 3.81), features they hate (62%, 3.74), and generic praise (46%, 3.35).

Corrective Maintenance: Developers were asked to rate the frequency of receiving bug reports based on the channel. Figure 7-g, depicts the results. In general, it shows an equal distribution with no channel prevalent in frequency. The highest in agreement, in terms of frequency is automatic in-app crash reporting, followed by the app store user reviews. User public complaints on social media was rated the least frequent (47% of respondents rated rarely/very rarely, 2.5 weighted average) followed by private messages from users (e.g. via email) which was rated rarely/very rarely by 40% of respondents scoring a weighted mean of 2.74.

On the other hand, when developers were asked which issues are frequently prioritised based on these sources,

there is a trend towards favouring user reviews in the app store (51% of respondents prioritise it frequently/very frequently scoring a weighted mean of 3.61) tied with user's private messages (48%, 3.61) followed by automatic in-app crash reporting (50%, 3.6). We noticed that although private messages were less common, they were prioritised more frequently. This has been expressed during the interviewing process; especially vehemently by one developer: *"There's something more direct about an email [opposed to user public reviews]. A person has gone through the trouble of writing an email. It's more in-depth about it as well, I appreciate that."*

When it comes to prioritising user feedback coming from the app store, 51% of respondents reported frequently/very frequently fixing issues coming via that channel; whereas only 13% rarely/very rarely did it, as depicted in Figure 7-h. In that regard, we were interested to gauge whether developers found it challenging to extract actionable feedback from the app store. Figure 7-i shows that, 25% of respondents agreed/strongly agreed that it's an easy task, while 24% professed to finding it hard.

By analysing interview content, we find three main obstacles preventing developers from fully leveraging user feedback in app stores, despite its perceived importance. First is the frequency with which users post into the app store can make it challenging to catch up with those comments. Second, reviews can be largely repetitive and mixed with noise obscuring finding a distinctive list of requested fixes and enhancements. As one developer puts it: *"The problem is we get 4-5 reviews a day. And because they're largely similar and positive we don't read them in any depth. It would be really useful to have a way of aggregating the things that people most often asked for and the things that they said annoyed them the most. I certainly know what the highest things are as they get repeated often. But within there are sort of 'second tier' stuff that I'm not clear about what we should prioritize. so we have to choose and understand what makes users happy is the thing that would be useful."* The third challenge is a general distrust over the content found in app store reviews. A developer informs us *"I don't rely on comments coming from app store] because the comment system on the app store is completely broken. It's full of fake reviews, people leaving reviews because they are working for the competition and people leaving bad reviews because they're angry they didn't get the point of your app."*

Perfective Maintenance: In an app's journey, developers seek to grow the app by providing more value to users in the form of functionality and performance enhancements. This type of perfective maintenance is typically planned around user engagement in test sessions and focus groups in addition to the application's vision and roadmap. App stores provide rich communication channels in which users are able to submit their requests for new features and possible enhancements. Developers believe that delivering on those requests carry large marketing value for the app. Research by Martin et al. [45] showed that 33% of releases from a sample of 26,339 had an impact on user rating, and that impact is likely to be positive in free apps (59%), most importantly, they report that these significant releases are bug fixes and new features. While a study by Palomba et al. [68] showed that on average, developers include feedback from 49% of informative reviews into the new release, they also report that responding to user reviews has a positive

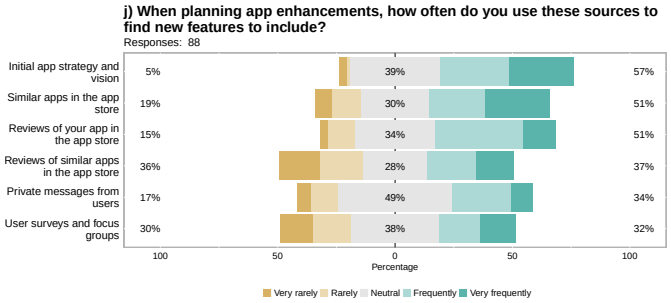

Fig. 8: **RQ1.** Responses to questions regarding the usefulness of user feedback for perfective maintenance.

effect on subsequent app rating ($\rho = 0.59, p-value < 0.01$). This highlights the important role app stores play as a communication channel and a source for planning app evolution.

To gauge the role user feedback play in perfective maintenance, we asked developers to rate how frequently do they use feedback from app stores as opposed to other sources. Figure 8 shows the tendency of the results. The results reveal that the most popular one (ranked by weighted mean) is initial app strategy and vision (57% use it frequently/very frequently scoring 3.79 weighted mean) while viewing the features of similar apps in the app store comes in second in frequency (51%, 3.53), next is user feedback of the app itself (51%, 3.48); 34% of respondents agreed to viewing private messages of users (3.21 weighted mean) whereas 32% frequently/very frequently looked at user surveys and focus groups (3.04 weighted mean). On the other hand, rated least frequent was user reviews of similar apps in the app store (37%, 3.0 weighted mean).

This indeed agrees with our interview observations. As one developer informed us regarding their practice when trying to find new features to enhance the app: *"[I keep] an eye on competitors and eye on my customers and community."*

> *In **maintenance**, classical channels for user engagement and bug detection endure. However, developers seldom ignore those enhancement requests posted by users in the app store. For perfective maintenance, developers employ user reviews of their app and the features of similar apps for enhancements and possible reuse.*

4.2.4 Release Management

App stores are managed by large firms who are usually the ones managing the platform and/or operating system. These organizations tend to prioritise raising the quality of apps marketed in their stores and thus enforce certain criteria on apps prior to granting them access to the store. This review procedure introduces delays that mobile developers usually plan around. Mobile developers expressed exasperation at losing a certain degree of control when it comes to release planning. *"And what that means is, you try to get rid of all the bugs before you launch. And this slows things down"*, a developer complains, *"So you try to avoid this horrible situation, which we've been in a few times, where you release something and then it breaks and then you have 11 days of letting your users down and getting negative reviews. You can't do anything about it because Apple takes a long time."*

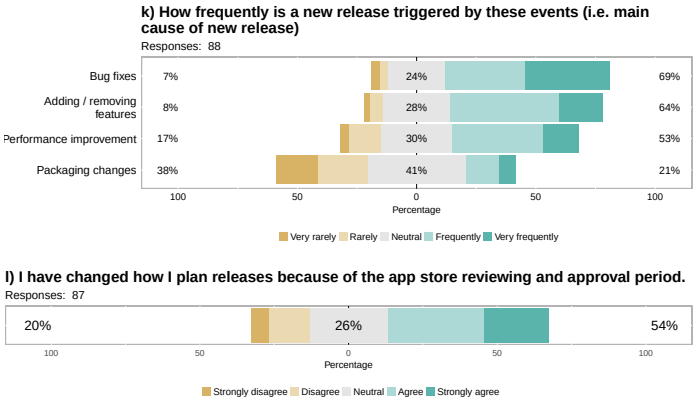

Fig. 9: **RQ1.** Responses to questions regarding the effect of app store to release strategy.

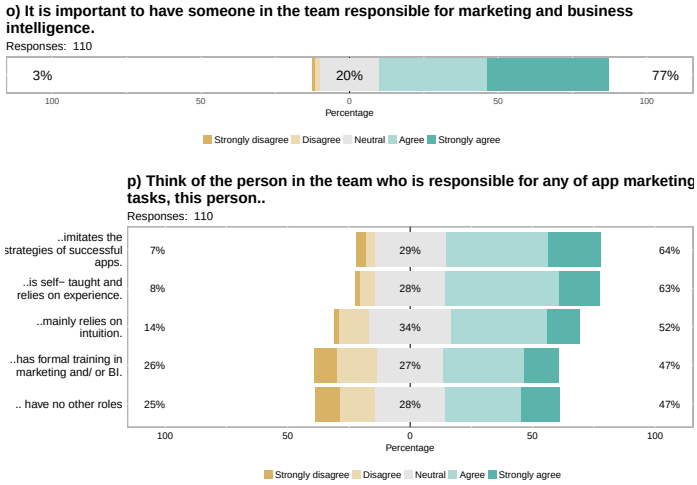

Fig. 10: **RQ2.** Responses to questions regarding the type of activities and skills required in a development team.

This is also mirrored in the questionnaire results as 54% of respondents agreed that they changed the way they plan releases because of the app store review and approval period. And this indeed is a worrying concern when our results reveal that the major reason motivating a new release is a bug fix as apparent in Figure 9.

*In **release management**, more than half of the developers reveal that they indeed change how they plan releases based on the app store's approval period. In general, developers expressed a need to conduct more rigorous testing as the gap between submitting a bug fix and it being published increases.*

4.3 RQ2: Emerging Skill-sets and Best Practices

We investigated if the app store ecosystem introduced new types of activities carried out by the development team. We conjecture that due to the way app stores lowered barriers to entry, smaller development teams had to carry out non-technical tasks and demonstrate app-store specific know-how for their app to thrive.

During our interviews, developers highlighted that it is paramount to the success of an app the way it is presented in the app store. In app stores, the competition is very high as a great deal of apps compete for the user's attention. To developers, quality of the product does not only come down to good software, other factors regarding presentation in the app store environment come into play. *"I can be very cynical and say that it's the only thing that matters. From experience I say that great communication and normal app works better than great app with bad communication. And for communication I mean everything: packaging, marketing, PR, etc. So it's crucial."* a developer informs us.

We have observed throughout the interviewing process that respondents exhibited confidence in their best practices knowledge: Application strategy, implementation, and mostly, app store culture and know-how. For example, one developer elaborated on best ways to post a screenshot in the app store *"Do not put a boring screenshot that's not wrapped in a phone: wrap it in a phone and put some text above it."* when asked how did they know this technique is effective, they said it was by looking at other apps. Another developers informs us: *"It's just trial and error, looking at what other people are doing, what I like and what works."* Something that prevails many of their practices.

This is reflected in the questionnaire responses as the majority of surveyed developers acknowledged that it is important to have a team member who is responsible for carrying out marketing and business intelligence tasks (77% agreement, 4.13). However, 63% report that their marketing team member is self-taught and relies on experience (3.75 weighted average). Of those surveyed, 64% agree that whomever is carrying out these tasks imitates the strategies of successful apps (3.80) whereas 25% report that the team member responsible for marketing decisions is not dedicated to that role (3.31). Figure 10 depicts a breakdown of the answers.

*We observed a number of developers who needed to address many non-technical issues. The **new skill sets** required by engineers developing for app stores include facilitating app discovery for users in addition to understanding the competitive environment and user expectation when selecting core functionality and supporting features, custom release strategy for mobile app stores, and several practices leading to a better brand for the app.*

4.4 RQ3: New Success Criteria and Performance Measures

As the app is released into perfective and corrective maintenance cycles, developers have access to immediate feedback regarding the quality and performance of the app. This feedback takes many forms. In addition to user rating and reviews, developers have access to a large number of metrics including app downloads (rank), user retention rate, revenue and number of reviews. We investigated the extent to which developers monitor these metrics and the role they play in decision making.

To gain a better idea of perception of success, we asked respondents to write what they define as success in the app store. Several developers restricted their definition of success to the app correctly delivering its functionality. *"When the user is able to do the core features of the application*

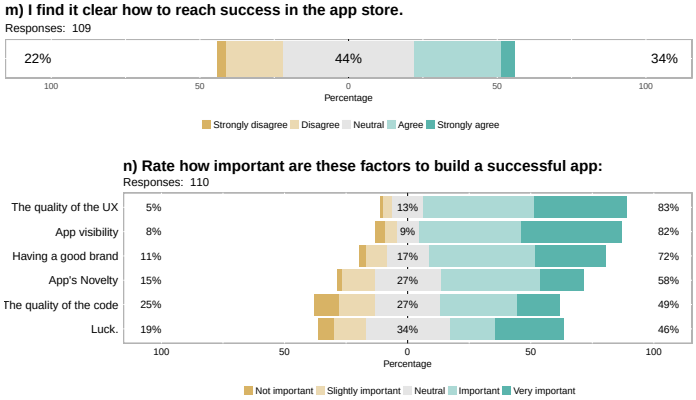

Fig. 11: **RQ3.** Responses to questions regarding the knowledge of success factors in the app store.

quickly and without much trouble," a developer wrote. Other developers answered similarly: *"[When app solves a real problem."* and *"Providing a real great solution to an existing problem."* However, and more interestingly, the majority of their answers quoted a measurable, app store metric. Of our sample, 52 informed us of the metrics they observe to evaluate the success of the app. The most popular was the number of downloads/installs (37%), followed by rating (28%), active users/retention rate (27%), revenue (15%), then application's ranking and number of installs (tied at 3.8%). Few respondents (6%) mentioned application's validity/verification (i.e. the app delivering the needed functionality without faults).

Through interviewing developers, we detected a certain uncertainty and lack of control towards reaching success in the app store. This is confirmed by the questionnaire responses. Respondents, when asked if the path to achieving that success is clear and easy to follow, showed reluctance with only 34% of respondents agree that, indeed, they find it clear (3.14 weighted mean) as seen in Figure 11-m. We further explored their opinion regarding the most important factors to build a successful app and were surprised to see that the lowest rated was the quality of code and documentation while the one rated highest in importance was the quality of the user experience (UX) as shown in Figure 11-n.

*Surveyed developers reported a unique perception of **quality measurement** giving low ranking to code quality and documentation in determining an application's success. Developers tend to quote more app-store-specific **quality measurements** than classical software engineering ones with number of downloads surpassing user's rating.*

5 Discussion

The results presented reveal implications that can inform relevant research spanning several scientific sub-fields. Through our findings, we summarise implications under three top-level categories: developer-user interaction, market transparency and application release cycles.

5.1 Developer-User Interaction

App stores, in their current format, have further **bridged the gap** between potential users and developers. We detect that developers view the app store as a channel of communication. Though not the only one, this channel has two distinct properties: It increases the prolificacy of users and can affect the success of the app. Pagano and Maalej [69] found that free apps received an average of 36.87 daily reviews in 2013; and more recently, Mcilroy et al. [70] analysed the reviews of 12,000 apps in the app store and reported that free apps receive 7 reviews per day on average; in addition, McIlroy et al. [71] and Palomba et al. [68] report that responding to user reviews has a positive effect on subsequent app rating ($\rho = 0.59, p - value < 0.01$); whereas Lee and Raghu [72] found that continuous updates increase the applications success. Developers seem to be aware of this to some extent. This is reflected by our survey respondents professing interest in gauging public reception and social hype of the app in alpha or beta testing stages (78% agreement); this is particularly important in light of the finding by Ruiz et al. [73] that mobile app stores ratings fail to adapt to the actual current satisfaction levels and are resilient once they reach a certain number of user base.

While automatic in-app crash reporting is the most prolific channel of reporting bugs, the one mostly prioritised by our respondents is user reviews in app stores. Additionally, 51% of respondents frequently use user reviews for app features enhancement. Our results reveal that, while developers and researchers point to the benefits of using reviews for app evolution, 24% reported experiencing difficulty in extracting bug reports from user reviews. Reasons hindering proper utilisation of user reviews include its noise and volume.

Requirements engineering research have directed their attention to solving the problem of analysing user reviews for the benefit of app evolution. Research has been carried out to classify user reviews according to their type and actionability to the developer [38] [39] [43] [44] [67] [74] [75], review summarisation [40] [76] [77] as well as feature-specific analysis [30] [32] [78] [79]. Further research employed user reviews and the app's extracted features to localise change requests within code and to couple natural language with source code patterns [41] [80]. A systematic review of the literature relating to opinion mining from user comments in the app store is provided by Genc-Nayebi and Abran [81]. However, research remains scarce on the problem of detecting fraudulent reviews. Such reviews, not only increase the amount of noise when extracting useful information from user reviews, they also introduce errors regarding app ratings and subsequently in any analyses that incorporate the app's rating score (e.g. [50] [53]). While the field of "opinion spam" detection advances in other areas of research, its transference to app store analysis is necessary. Xie and Zhu [82] and Li et al. [83] reveal the existence of a fake rating black market and provide in-depth analysis of their characteristics and its effect on the app store.

5.2 Market Transparency

One of the major contributions of a centralized mobile application marketplace is the significant increase in **trans-**

parency and availability of information for content creators. The applications' price, features, reviews, ranking and release strategy are publicly available. Our results reveal that over half respondents monitor similar and competing apps at the stage of requirement elicitation (56% frequently view similar apps). Requirements engineering research can help further investigate the effect of this practice and facilitate it further.

In performing perfective maintenance, frequently investigating features of similar apps is as common as considering the feedback of the developer's own app (51% frequently/very frequently for both). This insight can help guide further software repository mining work that collates information from various applications that share functionalities or are in the same application domain. This insight also attests to the viability of app store performance predictions based on the past evolution of other similar applications.

To this end, Vu et al. [84] provided a keyword-based approach to mining reviews of apps and Shah et al. [85] detected similar apps based on feature overlap and merge reviews of those app for feature-specific sentiment analysis. Sarro et al. [32] showed that similarity of app features/descriptions can successfully lead to accurate prediction of app success (i.e. rating). We believe this vein of research can be further extended to incorporate automatic detection of similar useful apps while using the evolution of these apps (and their reviews) to recommend possible feature inclusion and other strategic decisions for developers.

However, we draw the attention of the community to possible pitfalls when analysing reviews and ratings of mobile apps as a 'rating call-to-action' gains popularity among developers. A rating call-to-action operates within the app to request users to rate the app and subsequently directs them to the app store. Of our respondents, 38% embedded a rating call-to-action into their apps. The majority of those (56%) admitted to ensuring a call-to-action is activated when the user appears sufficiently engaged and having a positive experience with 35% ensuring the app first prompts the user for their rating, then only directing them to the app store when their rating is high enough. This may carry certain implications towards the bias of app rating and reviews. In their empirical study, Pagano and Maalej [69] analyse 1,126,453 reviews from 1,100 applications from the Apple app store, half of which were free, and reported that the overall average rating of all reviews is 4.13 with 61.96% of reviews having a 5 star rating, while such a high average rating value is not observed analysing the content of app stores in 2011 when such call-to-action may not have been as popular [26].

Among similar applications' attributes that are made available for developers to observe, user feedback garner the most attention (81% are interested/very interested scoring 4.19 weighted mean) closely followed by ratings (78%, 4.09) and screenshots (76% , 4.05). This promises significant contribution to developers were researchers to employ image processing to mine applications users interfaces to extract actionable information as is done with applying natural language processing over applications descriptions [30] [29], UI text [86] and user feedback [1].

5.3 Release Planning and Quality

Our questionnaire reveals that the app store regulations and approval periods affected 54% of respondents' **release strategy**. The research by Nayebi et al. [5] reveals that a majority of their sample (36 developers) adopt a time-based strategy (80%) with 45% releasing weekly or bi-weekly. Furthermore, they find that 36% of respondents will change their release plan to accommodate user feedback and that 61% agree that a time-based release strategy affects the application's success in terms of feedback and user rating.

Several studies reported that frequent releases cause an increase in user engagement (i.e. reviews and ratings) [69] [70] and that certain types of releases have significant impact on user ratings [45] [51]. This supports the idea that release strategies in app stores may not only be influenced by vendors' guidelines but also by users' public reaction in the form of downloads, reviews and ratings. Adams and McIntosh [87] emphasize the need for software engineering research to further investigate the implications of the industry's recent trends in adopting certain release engineering practices including rapid delivery and mobile app release cycles.

These release practices enable mobile development to adopt rapid adaptation and fast route to market that closely resembles that of web application development. This adaptation model may inform further research in the software engineering community to extract and transfer experiences and techniques from similar platforms such as web development. One example is the ease of adoption of A/B testing which thus far has been predominantly applied in web based applications. Of our respondents, 39% already perform A/B testing.

In addition to change in release practices, the perceived quality criteria of mobile apps seem to shift. Surveyed developers deemed user experience design of higher importance than code quality. This is in line with the findings of Nayebi et al. [46] in which they surveyed 22 mobile developers and found that 'customers expectations' and 'market and competitors' were deemed more important in mobile development compared to other platforms, whereas 'quality' was rated higher with more consensus for other platforms than mobile apps. This is confirmed by the study of Minelli and Lanza [88] where they report that open source Android apps showed high complexity with smaller sizes, large reliance on third party libraries and overall neglect of development guidelines.

6 Threats to Validity

During the design of this study, potential threats to its validity have been addressed in an effort to minimise their risk.

6.1 Construct Validity

Construct validity in qualitative studies mainly pertains to a unified understanding between what the researcher has in mind and what the respondent eventually understands [17]. Prior to building the questionnaire, the interviewing process with several app developers served to orient the researcher towards the culture and type of knowledge to which the

mobile app community adheres. Several books were suggested by the developers that the researcher has read to familiarise themselves with the terminology and the process (a good example is The Lean Startup by Eric Ries [89]). Due to the nature of the interviews, misunderstandings were detected and cleared up. The questionnaire was built upon the insight provided by the interviewing process in addition to transcript analyses. The read-aloud pilot study of the questionnaire ensured the detection and elimination of incompatible terminology and other misunderstandings.

6.2 Internal Validity

Internal validity is at risk when causal factors are examined and reported. As this study is mainly data-driven with first and second degree collection methods (interviews and questionnaire), we present the results as observed. In interpreting the data, we make clear our conjectures are aligned with those recorded during the interviews. Causal analysis is limited as to this type of study. Another aspect that is a threat to internal validity is proper analysis by the researchers of the interview transcripts. While one author has done the thematic analysis, it has been revised and validated by three other authors in more than one collaborative session till consensus was reached. We limit the threat by augmenting our findings with questionnaire responses.

6.3 External Validity

Although the initial information gathering technique only aims to interview a low number of developers, the interviewing process terminated when responses to all questions were pre-observed in previous interviews. The developers selected for interviews, though with varying backgrounds and sizes of teams, represent a limited sample. However, it is common for interviews to limit the sample as they only serve as an exploration device rather than seeking generalizable answers. Afterwards, all possible findings are augmented by disseminating a questionnaire to developers in order to measure the extent to which developers adhere with the findings. Through the questionnaire, we were able to reach an even more diverse set of mobile developers overseas. Having both methods of collecting data, we employ triangulation that can help us in affirming the validity of the results.

The questionnaire garnered 186 responses, this number, thought fairly large and in line with several similar research as shown in Section 7, cannot be claimed to be representative of all types of development teams, applications domains or characteristics other than the ones reported in our sample.

7 Related Work

Smartphone adoption grew rapidly in the past years. As mobile operating systems and underlying hardware form-factor differ from their desktop counterparts, it naturally follows that mobile applications are also distinctive [90]. In investigating mobile app development from a software engineering perspective, research generally took one of two themes: (1) works investigating **how mobile app development differs from classical software development** and (2) uncovering **software engineering challenges rising from the mobile development paradigm**.

In investigating the distinction between mobile application and classical software development, Wasserman summarises the differences in 8 areas including the hybrid nature of applications, platform fragmentation and new user interface requirements. Minelli and Lanza [88] show that open source F-Droid mobile applications are distinct from classical software system in terms of size, ease of comprehension and degree of reliance on external libraries. This is confirmed by Syer et al. [12] as they report that mobile apps tend to have less lines of code, smaller development teams, and rely more heavily on the underlying platform. Additionally, they find that, regardless of the size of the project, mobile developers tend to fix bugs faster than desktop/server software teams. Other aspects in which mobile application software engineering have been found to differ from classical one is in testing: [48] [91], release management [5] and size/effort estimation [92] [93] [94] [95] [96] [97].

Looking into the challenges that are introduced by mobile software development, Joorabchi et al. [3] followed a grounded theory approach in interviewing 12 app developers followed by a survey of 188 respondents. Among the challenges they find is platform fragmentation, lack of testing tools, closed source underlying platforms, data management intensity, frequent changes of underlying platform and third party libraries, hybrid nature of mobile apps, limited hardware capabilities, difficulty of code re-use from other platforms and strict HCI guidelines. These findings were confirmed by survey study conducted by Flora et al. [98] in addition to suggesting new challenges: The high quality expectations of users augmented with big competition and the insufficiency and uncertainty of requirements gathering for such markets. These last two challenges were also observed by Lim et al. [6]. They identify the need for newly emerging packaging requirements with price sensitivity and managing a large space of potential features even when domain-specific. Rosen and Shihab [7], by employing topic modelling over StackOverflow data, report a set of 32 main topics concerning distribution, third party APIs, data management, sensors, tools and user interfaces.

The qualitative study by Francese et al. [4] gathered information by interviewing 4 technology managers in addition to surveying 82 mobile app professionals. They report that their surveyed developers perceive mobile platform fragmentation and inadequate testing support as the two main difficulties. They also report that mobile developers concede that developing software for mobile devices is different than that of other type of software development.

In testing, Kochhar et al. [48] investigated the adequacy of mobile app testing practices. They report that 86% of open source Android applications do not contain test cases, and those that do have poor line coverage (median 9.33%). To that end, they survey 127 Microsoft developers and found that the majority (114 out of 127) use manual testing rather than automated testing tools. They compile a list of challenges that prevents developers from adopting said tools, including time constraints, poor documentation and emphasis on development. These limitations are confirmed in a study by Linares-Vàsquez et al. [91] comprising 102 respondents of open-source mobile developers.

In maintenance, Linares-Vàsquez et al. [99] investigated how mobile app developers detect and fix performance issues. They collected the responses of 485 open-source Android developers and analysed their Github repositories. They found that, in order to detect performance bottlenecks, developers are aided by user reviews and mostly rely on manual execution. Salza et al. [100] found that developers do not promptly update third-party APIs (especially non GUI-related ones) and that 89% of apps with up-to-date APIs are highly rated.

Nayebi et al. [5] shed light on possible changes in release practices of mobile applications due to the OS vendor's quality assurance process. Surveying 674 mobile app users and 36 developers, they find that about half of the surveyed developers follow a rational release strategy. Those who do, are more likely to be experienced developers with higher success. More interestingly, it seems that how the app is perceived by users when considering downloading it affects how developers make release strategy decisions. About 44% of the surveyed developers believe that their release strategy affect the perception of users regarding that app. On the other hand, they find that end users do indeed prefer apps that are more recently updated. They also report that the second most common release strategy in the app store is a marketing-based strategy. Further in support of this finding, we measure and report respondents' perceived importance of having someone on the team solely dedicated to the task of app 'marketing' (77% agreement); we additionally report the strategies used by team members to make marketing decisions. The previous paper presents interesting findings and an in-depth view of the release strategies of mobile app developers; in our study, we shift the focus to address the role of the app store itself with a wider basis of scientific evidence (186 respondents).

Nayebi et al. subsequently investigated open-source app versions that are not shipped into the app store [46], introducing the concept of release 'marketability'. To this end, they surveyed 22 developers, the majority of which (95%) state that market acceptability of a mobile app release is more important than that of traditional software. Our study further investigates the effects of app-store-specific criteria to the success of the app as a whole (not just a specific release [46]) finding that while the quality of the user experience is the most agreed upon, it is closely followed by the application's visibility in the app store.

Villarroel et al. [43] and Scalabrino et al. [44] conducted a semi-structured interview with 3 project managers of software companies developing mobile apps in order to evaluate the usefulness of their tool (which extracts and clusters user reviews from the app store into bug report or new feature request). The tool was first demonstrated to the managers, then they were asked about the usefulness of reviews (do you analyse user reviews when planning a new release?), to which they answer yes. Our study confirms this prior finding with a wider basis of scientific evidence (186 respondents), it also extends it by reporting the frequency with which developers receive bug reports from user feedback in app stores and how this affects its priority (compared to other channels). Our paper additionally investigates further stages at which developers refer to the app store (idea conception/validation, requirement elicitation, GUI design inspiration and feedback of competing/similar apps).

While these studies partially address the changes introduced by app stores, research studies fully addressing the potential effects of the mobile application distribution model are few and far-apart. Holzer et al. [2] identify the app store as a two-sided market. A Two-sided market [101] is a market model consisting of two groups of clients participating in one platform where the increase of one side attracts the increase of the other and thus the market is in growth loop; where in the case of mobile app stores, one client group is users and the other is developers. Holzer et al. further discuss the various trends such a market may introduce and their implication on developers. The centralised sales portal model, for example, carries the implication that developers have immediate access to the entire consumer base and lowers distribution costs; but on the other hand, imposes limits on the freedom of the developers. Our paper addresses this gap by taking software development life cycle phases as the point of analysis when surveying industry practitioners to uncover the involvement of app store in developers' practices. We believe conducting this type of research is important as more platform-mediated application markets rise in popularity (e.g. wearable apps, voice assistant skills) introducing the need to investigate whether and how deployment portals for platform-specific software affect software development practices.

8 Conclusions

This study investigated aspects of app store developers' software engineering activities revealing overarching themes of importance to app store software development. The three main themes that emerged were market transparency, user-developer gap reduction, and release cycles.

App stores exemplify market transparency in which app description, features, price, rating and user feedback are public. Our survey found that developers do, indeed, refer to similar apps when designing their own. Our results reveal that developers are interested in monitoring similar apps for maintenance and evolution. We also found that other apps' user feedback, rating and screenshots and are the three most important aspects of information gleaned from the open market by developers. Our results highlight the way in which app stores have become a communication channel between users and developers. Our findings confirm that developers seldom neglect user feedback posted on app stores; user feedback was the third strongly agreed-on source of app improvement after the initial strategy of the app and monitoring similar apps on the app store. Our survey respondents also rated user reviews as the second most prolific channel of bug reporting after automatic in-app crash reports but regardless was scored highest in prioritisation. User feedback, in addition to being informative to developers, also determines the overall rating of the app. Previous research on this subject showed that release frequency correlates with increased user feedback [69] [70]. More than half of our respondents reported changing their release plan in accordance with perceived constraints imposed by the app store ecosystem. These findings have actionable conclusions for software engineering practitioners

and researchers, including requirements engineering, testing and mining software repositories research communities, and also business communities.

REFERENCES

[1] 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*, pp. 1–1, 2016.

[2] A. Holzer and J. Ondrus, "Mobile application market: A developers perspective," *Telematics and Informatics*, vol. 28, no. 1, pp. 22–31, Feb. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736585310000377

[3] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real Challenges in Mobile App Development," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, Oct. 2013, pp. 15–24.

[4] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora, "Mobile app development and management: results from a qualitative investigation," *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, pp. 133–143, 2017.

[5] M. Nayebi, B. Adams, and G. Ruhe, "Release Practices for Mobile Apps – What do Users and Developers Think?" in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Mar 2016, pp. 552–562.

[6] S. L. Lim, P. J. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, "Investigating Country Differences in Mobile App User Behavior and Challenges for Software Engineering," *IEEE Transactions on Software Engineering*, vol. 41, no. 1, pp. 40–64, Jan 2015.

[7] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, Jun 2016.

[8] R. E. Boyatzis, *Transforming qualitative information : thematic analysis and code development*. Sage Publications, 1998.

[9] K. Roulston, "Data analysis and 'theorizing as ideology'," *Qualitative Research*, vol. 1, no. 3, pp. 279–302, Dec 2001.

[10] B. E. Whitley, M. E. Kite, and H. L. Adams, *Principles of research in behavioral science*, 3rd ed. Routledge, 2012.

[11] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[12] M. D. Syer, M. Nagappan, A. E. Hassan, and B. Adams, "Revisiting prior empirical findings for mobile apps: an empirical case study on the 15 most popular open-source Android apps," pp. 283–297, 2013.

[13] A. Miniukovich and A. De Angeli, "Computation of Interface Aesthetics," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. New York, New York, USA: ACM Press, 2015, pp. 1163–1172.

[14] T.-H. Chang, T. Yeh, and R. Miller, "Associating the visual representation of user interfaces with their internal structures and metadata," in *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. New York, New York, USA: ACM Press, 2011, p. 245.

[15] M. Harman, A. Al-Subaihin, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Mobile app and app store analysis, testing and optimisation," in *Proceedings of the International Conference on Mobile Software Engineering and Systems*, ser. MOBILESoft '16. New York, NY, USA: ACM, 2016, pp. 243–244.

[16] I. Benbasat, D. K. Goldstein, and M. Mead, "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly*, vol. 11, no. 3, p. 369, Sep 1987.

[17] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Dec. 2008.

[18] C. Robson and K. McCartan, *Real world research : a resource for users of social research methods in applied settings*, 4th ed. John Wiley & Sons, 2015.

[19] B. Kitchenham and S. Pfleeger, "Principles of survey research part 2," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 1, pp. 18–20, Jan 2002.

[20] B. Glaser and A. L. Strauss, *Discovery of Grounded Theory Strategies for Qualitative Research*. Taylor and Francis, 1967.

[21] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1427–1455, dec 2015. [Online]. Available: http://link.springer.com/10.1007/s10664-014-9319-7

[22] D. S. Cruzes and T. Dyba, "Recommended Steps for Thematic Synthesis in Software Engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, sep 2011, pp. 275–284. [Online]. Available: http://ieeexplore.ieee.org/document/6092576/

[23] A. Abran, P. Bourque, R. Dupuis, and J. W. Moore, Eds., *Guide to the Software Engineering Body of Knowledge - SWEBOK*. Piscataway, NJ, USA: IEEE Press, 2001.

[24] I. Bohnet, H. Harmgart, S. H. (Ucl), and J.-R. Tyran, "Learning Trust," *Journal of the European Economic Association*, vol. 3, no. 2-3, pp. 322–329, may 2005. [Online]. Available: https://academic.oup.com/jeea/jeea/article/2281111/Learning

[25] A. A. Al-Subaihin, A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store mining and analysis," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, DeMobile 2015*, 2015, pp. 1–2.

[26] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Investigating the relationship between price, rating, and popularity in the blackberry world app store," *Information and Software Technology*, vol. 87, no. Supplement C, pp. 119 – 139, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095058491730215X

[27] ——, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics," *UCL - Research Note RN/14/10*, September 2014.

[28] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," pp. 108–111, Jun. 2012.

[29] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '16. New York, NY, USA: ACM, 2016, pp. 38:1–38:10.

[30] F. Sarro, A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang, "Feature Lifecycles as They Spread, Migrate, Remain and Die in App Stores," in *2015 23rd IEEE International Requirments Engineering Conference*. IEEE, Aug. 2015.

[31] Y. Liu, L. Liu, H. Liu, X. Wang, and H. Yang, "Mining domain knowledge from app descriptions," *Journal of Systems and Software*, vol. 133, pp. 126–144, nov 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121217301784

[32] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer rating reactions can be predicted purely using app features," in *Proceedings of the 26th IEEE International Requirements Engineering Conference*, ser. RE'18. IEEE, 2018, p. to appear.

[33] D. Han, C. Zhang, X. Fan, A. Hindle, K. Wong, and E. Stroulia, "Understanding Android Fragmentation with Topic Analysis of Vendor-Specific Bugs," in *2012 19th Working Conference on Reverse Engineering*. IEEE, Oct 2012, pp. 83–92.

[34] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan, "Prioritizing the devices to test your app on: A case study of android game apps," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 610–620.

[35] G. Grano, A. Ciurumelea, S. Panichella, F. Palomba, and H. C. Gall, "Exploring the integration of user feedback in automated testing of Android applications," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, mar 2018, pp. 72–83. [Online]. Available: http://ieeexplore.ieee.org/document/8330198/

[36] F. Sarro, "Predictive analytics for software testing," in *Proceedings of the 11th International Workshop on Search-Based Software Testing - SBST '18*. New York, New York, USA: ACM Press, 2018, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3194718.3194730

[37] C. McMillan, M. Linares-Vasquez, D. Poshyvanyk, and M. Grechanik, "Categorizing software applications for maintenance," in *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, Sep 2011, pp. 343–352.

[38] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, May 2013, pp. 41–44.

[39] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on*

Software Engineering (ICSE). New York, New York, USA: ACM Press, May 2014, pp. 767–778.

[40] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*. New York, New York, USA: ACM Press, 2016, pp. 499–510.

[41] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, May 2017, pp. 106–117.

[42] N. Jha and A. Mahmoud, "Using frame semantics for classifying and summarizing application store reviews," *Empirical Software Engineering*, pp. 1–34, mar 2018. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9605-x

[43] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*. New York, New York, USA: ACM Press, 2016, pp. 14–24.

[44] S. Scalabrino, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Listening to the Crowd for the Release Planning of Mobile Apps," *IEEE Transactions on Software Engineering*, pp. 1–1, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8057860/

[45] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis for app releases in google play," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*. New York, New York, USA: ACM Press, 2016, pp. 435–446.

[46] M. Nayebi, H. Farahi, and G. Ruhe, "Which Version Should Be Released to App Store?" in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, nov 2017, pp. 324–333. [Online]. Available: http://ieeexplore.ieee.org/document/8170119/

[47] S. Shen, X. Lu, and Z. Hu, "Towards Release Strategy Optimization for Apps in Google Play," 2017. [Online]. Available: https://arxiv.org/pdf/1707.06022.pdf

[48] P. S. Kochhar, F. Thung, N. Nagappan, T. Zimmermann, and D. Lo, "Understanding the Test Automation Culture of App Developers," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, apr 2015, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/7102609/

[49] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What Do Mobile App Users Complain About?" *IEEE Software*, vol. 32, no. 3, pp. 70–77, may 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6762802/

[50] H. Khalid, M. Nagappan, and A. Hassan, "Examining the Relationship between FindBugs Warnings and End User Ratings: A Case Study On 10,000 Android Apps," *IEEE Software*, vol. PP, no. 99, pp. 1–1, 2015.

[51] W. Martin, F. Sarro, and M. Harman, "Causal Impact Analysis Applied to App Releases in Google Play and Windows Phone Store," University College London, Research Note, RN/15/07, Tech. Rep., 2015.

[52] E. Shaw, A. Shaw, and D. Umphress, "Mining Android Apps to Predict Market Ratings," in *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*. ICST, 2014. [Online]. Available: http://eudl.eu/doi/10.4108/icst.mobicase.2014.257773

[53] D. E. Krutz, N. Munaiah, A. Meneely, and S. A. Malachowsky, "Examining the relationship between security metrics and user ratings of mobile apps: a case study," in *Proceedings of the International Workshop on App Market Analytics - WAMA 2016*. New York, New York, USA: ACM Press, 2016, pp. 8–14.

[54] L. Guerrouj, S. Azad, and P. C. Rigby, "The influence of App churn on App success and StackOverflow discussions," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, mar 2015, pp. 321–330. [Online]. Available: http://ieeexplore.ieee.org/document/7081842/

[55] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk, "The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, apr 2015. [Online]. Available: http://ieeexplore.ieee.org/document/6945855/

[56] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "Api change and fault proneness: A threat to the success of android apps," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 477–487.

[57] H. Hu, C.-P. Bezemer, and A. E. Hassan, "Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform Android and iOS apps," *Empirical Software Engineering*, pp. 1–34, mar 2018. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9604-y

[58] H. Hu, S. Wang, C.-P. Bezemer, and A. E. Hassan, "Studying the consistency of star ratings and reviews of popular free hybrid Android and iOS apps," *Empirical Software Engineering*, pp. 1–26, apr 2018. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9617-6

[59] E. Noei, M. D. Syer, Y. Zou, A. E. Hassan, and I. Keivanloo, "A study of the relation of mobile device attributes with the user-perceived quality of Android apps," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3088–3116, dec 2017. [Online]. Available: http://link.springer.com/10.1007/s10664-017-9507-3

[60] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the characteristics of high-rated apps? A case study on free Android Applications," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, sep 2015, pp. 301–310. [Online]. Available: http://ieeexplore.ieee.org/document/7332476/

[61] I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Impact of ad libraries on ratings of android mobile apps," *IEEE Software*, vol. 31, no. 6, pp. 86–92, 2014.

[62] F. J. Gravetter and L.-A. B. Forzano, *Research methods for the behavioral sciences*. Wadsworth Cengage Learning, 2012.

[63] E. Murphy-Hill, T. Zimmermann, and N. Nagappan, "Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development?" in *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. New York, New York, USA: ACM Press, 2014, pp. 1–11. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2568225.2568226

[64] I. Manotas, C. Bird, L. Pollock, and J. Clause, "An empirical study of practitioners' perspectives on green software engineering," University of Delaware, Tech. Rep. 2014/003, 2014.

[65] B. A. Kitchenham and S. L. Pfleeger, "Personal Opinion Surveys," in *Guide to Advanced Empirical Software Engineering*. London: Springer London, 2008, pp. 63–92.

[66] M. Nayebi, K. Kuznetsov, P. Chen, A. Zeller, and G. Ruhe, "Anatomy of Functionality Deletion," in *Proceedings of the Conference on Mining Software Repositories (MSR'18)*, Gothenburg, Sweden, 2018. [Online]. Available: https://www.ucalgary.ca/mnayebi/files/mnayebi/anatomy-of-functionality-deletions.pdf

[67] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble Methods for App Review Classification: An Approach for Software Evolution (N)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Nov 2015, pp. 771–776.

[68] F. Palomba, M. Linares-Vasquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Sep 2015, pp. 291–300.

[69] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, Jul. 2013, pp. 125–134.

[70] S. Mcilroy, W. Shang, N. Ali, and A. E. Hassan, "User reviews of top mobile apps in Apple and Google app stores," *Communications of the ACM*, vol. 60, no. 11, pp. 62–67, Oct 2017.

[71] S. McIlroy, W. Shang, N. Ali, and A. Hassan, "Is It Worth Responding to Reviews? A Case Study of the Top Free Apps in the Google Play Store," *IEEE Software*, vol. PP, no. 99, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7325189

[72] G. Lee and T. S. Raghu, "Determinants of Mobile Apps' Success: Evidence from the App Store Market," pp. 133–170, dec 2014. [Online]. Available: http://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222310206

[73] I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Examining the Rating System Used in Mobile-App Stores," *IEEE Software*, vol. 33, no. 6, pp. 86–92, nov 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7045413/

[74] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, 2015.

[75] M. Lu and P. Liang, "Automatic Classification of Non-Functional Requirements from Augmented App User Reviews," in *Proceedings of the 21st conference on Evaluation and Assessment in Software Engineering, EASE'17*, Karlskrona, Sweden., 2017.

[76] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why People Hate Your App Making Sense of User Feedback in a Mobile App Store," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*. New York, New York, USA: ACM Press, 2013, p. 1276.

[77] E. Guzman, O. Aly, and B. Bruegge, "Retrieving Diverse Opinions from App Reviews," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, Oct 2015, pp. 1–10.

[78] E. Guzman and W. Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, Aug. 2014, pp. 153–162.

[79] T. Johann, C. Stanik, A. M. A. B., and W. Maalej, "SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, Sep 2017, pp. 21–30.

[80] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella, "Android apps and user feedback: a dataset for software evolution and quality improvement," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics - WAMA 2017*. New York, New York, USA: ACM Press, 2017, pp. 8–11.

[81] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *Journal of Systems and Software*, vol. 125, pp. 207–219, Mar 2017.

[82] Z. Xie and S. Zhu, "AppWatcher : unveiling the underground market of trading mobile app reviews," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks - WiSec '15*. New York, New York, USA: ACM Press, 2015, pp. 1–11.

[83] S. Li, J. Caverlee, W. Niu, and P. Kaghazgaran, "Crowdsourced App Review Manipulation," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*. New York, New York, USA: ACM Press, 2017, pp. 1137–1140.

[84] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach (T)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Nov 2015, pp. 749–759.

[85] F. A. Shah, Y. Sabanin, and D. Pfahl, "Feature-based evaluation of competing apps," in *Proceedings of the International Workshop on App Market Analytics - WAMA 2016*. New York, New York, USA: ACM Press, 2016, pp. 15–21.

[86] X. Chen, Q. Zou, B. Fan, Z. Zheng, and X. Luo, "Recommending software features for mobile applications based on user interface comparison," *Requirements Engineering*, pp. 1–15, jul 2018. [Online]. Available: http://link.springer.com/10.1007/s00766-018-0303-4

[87] B. Adams and S. McIntosh, "Modern Release Engineering in a Nutshell – Why Researchers Should Care," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Mar 2016, pp. 78–90.

[88] R. Minelli and M. Lanza, "Software Analytics for Mobile Applications–Insights & Lessons Learned," in *2013 17th European Conference on Software Maintenance and Reengineering*. IEEE, Mar 2013, pp. 144–153.

[89] E. Ries, *The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Publishing Group, 2011.

[90] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*. New York, New York, USA: ACM Press, Nov 2010, p. 397.

[91] M. Linares-Vasquez, C. Bernal-Cardenas, K. Moran, and D. Poshyvanyk, "How do Developers Test Android Applications?" in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, sep 2017, pp. 613–622. [Online]. Available: http://ieeexplore.ieee.org/document/8094467/

[92] G. Sethumadhavan, "Sizing Android mobile applications," in *6th IFPUG International Software Measurement and Analysis Conference (ISMA)*, 2011.

[93] T. Preuss, "Mobile Applications, Functional Analysis, and the Customer Experience," in *The IFPUG Guide to IT and Software Measurement*, IFPUG, Ed. Auerbach Publications, 2012, pp. 408–433.

[94] H. van Heeringen and E. Van Gorp, "Measure the Functional Size of a Mobile App: Using the COSMIC Functional Size Measurement Method," in *Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2014 Joint Conference of the International Workshop on*. IEEE, 2014, pp. 11–16.

[95] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, "Investigating functional and code size measures for mobile applications: A replicated study," in *Product-Focused Software Process Improvement - 16th International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings*, 2015, pp. 271–287.

[96] ——, "Investigating functional and code size measures for mobile applications," in *41st Euromicro Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2015, Madeira, Portugal, August 26-28, 2015*, 2015, pp. 365–368.

[97] G. Catolino, P. Salza, C. Gravino, and F. Ferrucci, "A set of metrics for the effort estimation of mobile apps," *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, pp. 194–198, 2017.

[98] H. K. Flora, X. Wang, and S. V.Chande, "An Investigation into Mobile Application Development Processes: Challenges and Best Practices," *International Journal of Modern Education and Computer Science*, vol. 6, no. 6, pp. 1–9, Jun 2014.

[99] M. Linares-Vasquez, C. Vendome, Q. Luo, and D. Poshyvanyk, "How developers detect and fix performance bottlenecks in Android apps," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, sep 2015, pp. 352–361. [Online]. Available: http://ieeexplore.ieee.org/document/7332486/

[100] P. Salza, F. Palomba, D. Di Nucci, C. D'Uva, A. De Lucia, and F. Ferrucci, "Do Developers Update Third-Party Libraries in Mobile Apps?" in *26th International Conference on Program Comprehension (ICPC 2018)*. ACM, 2018. [Online]. Available: https://doi.org/10.475/123{_}4

[101] M. Armstrong, "Competition in two-sided markets," *The RAND Journal of Economics*, vol. 37, no. 3, pp. 668–691, sep 2006. [Online]. Available: http://doi.wiley.com/10.1111/j.1756-2171.2006.tb00037.x