User Feedback in the AppStore: An Empirical Study

Dennis Pagano
Technische Universität München
Munich, Germany
pagano@cs.tum.edu

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

***Abstract*—Application distribution platforms - or app stores - such as Google Play or Apple AppStore allow users to submit feedback in form of ratings and reviews to downloaded applications. In the last few years, these platforms have become very popular to both application developers and users. However, their real potential for and impact on requirements engineering processes are not yet well understood. This paper reports on an exploratory study, which analyzes over one million reviews from the Apple AppStore. We investigated how and when users provide feedback, inspected the feedback content, and analyzed its impact on the user community. We found that most of the feedback is provided shortly after new releases, with a quickly decreasing frequency over time. Reviews typically contain multiple topics, such as user experience, bug reports, and feature requests. The quality and constructiveness vary widely, from helpful advices and innovative ideas to insulting offenses. Feedback content has an impact on download numbers: positive messages usually lead to better ratings and vice versa. Negative feedback such as shortcomings is typically destructive and misses context details and user experience. We discuss our findings and their impact on software and requirements engineering teams.**

***Index Terms*—user needs, user feedback, mobile requirements**

I. Introduction

Application distribution platforms such as Apple AppStore[1], Google Play[2], and Windows Phone Store[3] enable users to find, buy, and install software applications with few clicks. Their popularity is growing at high speed. As of March 2013, over 800,000 applications are available in the AppStore and Google Play, and over 130,000 in the Windows Phone Store. The download numbers are astronomic, with around one billion application downloads per month in the AppStore. The growing popularity of these platforms, the ease of sale and deployment, as well as their huge communities of registered users make then very attractive for software organizations.

These platforms (also called app stores) also provide a user feedback feature, which is particularly interesting from the software and requirements engineering perspective. Users who buy an application can *rate* it with a number of "stars" and post a *review* message. Both ratings and reviews are public and visible to other users and developers. Such feedback allows for a user-driven *quality* assessment and marketing. Applications with higher ratings also rank higher in "top lists", which in turn increase the application visibility and its download numbers. Feedback also allows users to give and get *recommendations* on applications, similar to hotel booking sites.

Interestingly, users also seem to provide feedback to the application vendors and developers, which might help to improve software quality and to identify missing features [12]. A quick browse through, e.g., the AppStore shows that users publish feature requests and other messages on improving the application. The three following questions arise:

1) Can app stores serve as communication channel among users and with developers?
2) How can developers use app stores to better and quicker understand evolving user needs and requirements?
3) Which methods and tools should support software organizations to *analyze*, *aggregate*, and *use* this feedback?

This paper reports on an empirical study, which represents the first cornerstone for answering these questions. We aim at understanding the status quo of user feedback in application distribution platforms. The paper's contribution is threefold. First, it explores how and when users provide feedback, analyzing correlations between feedback metadata and its impact on the application popularity. Second, our study identifies and classifies topics in the user reviews, their co-occurrences, popularities, and impacts. Third, the paper gives insights into how researchers and tool vendors can systematically gather and use constructive feedback and how to integrate it into requirements and software engineering infrastructures.

We first introduce the study questions, method, and data (Section II). Then, we summarize the results along the three research questions: feedback usage, content, and impact on the user community (Section III). After discussing our findings (Section IV) and presenting the study limitations (Section V), we survey related work (Section VI), and sketch future research directions (Section VII).

II. Study Design

A. Research Questions

We study the *usage* of the feedback feature by the users, the *content* of feedback, and its *impact* on the user community.

Feedback usage describes *how* application users provide feedback. For that, we study the following questions:

- *Feedback frequency*: When and how often do users provide feedback?
- *Feedback meta-data*: What is a typical user feedback in terms of length, ratings, and helpfulness?

[1]https://itunes.apple.com/us/genre/ios/id36?mt=8
[2]https://play.google.com/store/apps
[3]http://www.windowsphone.com/en-us/store

978-1-4673-5765-4/13/$31.00 © 2013 IEEE

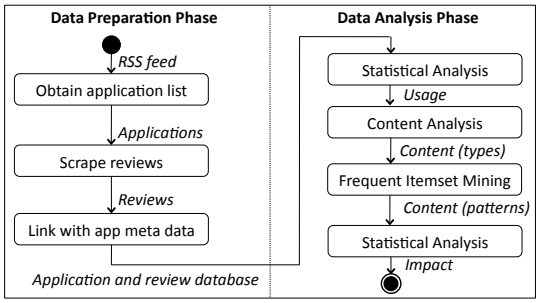

Figure 1. Research method.

Feedback content describes *topics* (i.e. semantic entities) provided in the feedback information and their frequencies. In particular, we investigate the following questions:

- *Feedback type*: Which types of user feedback exist?
- *Feedback patterns*: Are there patterns in the co-occurrences of feedback types?

Feedback impact describes whether user feedback *influences* the rating of other users and their decision to use the application. We study the following questions:

- *Market impact*: Do specific feedback types or feedback patterns influence the ratings?
- *Community impact*: Is specific feedback more appreciated by the user community than other?

With the term feedback, we refer to ratings, reviews, or both. When answering these questions we distinguish between *free* and *paid* applications, analyzing whether the pricing has an impact on the studied research questions.

B. Research Method

Our research method consisted of two phases: a data preparation and a data analysis phase, as depicted in Figure 1.

1) Data Preparation Phase: In the data preparation phase, we collected the data and created a database from it. We started by investigating the three biggest application distribution platforms and the available data: Apple's AppStore, Google's Play, and Microsoft's Windows Phone Store. All three include comparable data, which we are interested in: information about applications and associated reviews written by users together with a rating. We decided for Apple's AppStore because we had prior experience with the technology and applications, and because we found a possibility to receive the data programmatically. We leave comparisons between different application distribution platforms for future research.

In the AppStore applications belong to one of 22 disjoint categories[4], to help users finding applications more quickly. Further, applications are distinguished by their price in *free* apps and *paid* apps. Because of these classifications, we decided to draw a stratified sample of the data, including an equal number of free and paid applications for each category.

On September 16, 2012, we queried *a list of the top 25* free and paid applications in each category utilizing an RSS feed generator provided by Apple[5]. This list includes the most downloaded applications and is updated daily. Next, we parsed the list to extract the identifiers of included applications. We then used an open source scraping tool[6], which we had modified to scrape the list of reviews by iterating through all application identifiers. In the last step, we linked each application's meta data such as application name, release date, price, etc. with the obtained list of reviews and inserted the result into a MySQL database.

2) Data Analysis Phase: The data analysis phase consisted of three steps, which respectively answer the usage, content, and impact questions. To analyze the feedback *usage* we applied descriptive statistics. We also conducted statistical tests to exclude hazard factors and report on the error rates. To explore the feedback *content* and find included topics, two researchers independently performed a manual content analysis [11] of a random sample from our data set. After that, we applied frequent itemset mining [18] for identifying latent patterns among the topics. Finally, to study the *impact*, we combined feedback meta-data and content and again employed a statistical analysis of the resulting data. We detail on each of these analysis steps in the corresponding result section.

Table I
USER FEEDBACK BY CATEGORIES. $N = 1126453$.

#	app category	# feedback free apps	# feedback paid apps	mean price	max price
1	Books	23,962	8,641	2.43	9.99
2	Business	35,265	23,997	4.11	16.99
3	Catalogs	9,517	5,725	1.35	4.99
4	Education	16,628	16,577	1.99	3.99
5	Entertainment	45,761	38,201	1.47	4.99
6	Finance	36,182	15,259	2.99	14.99
7	Food & Drink	19,066	8,318	2.27	9.99
8	Games	38,923	43,602	1.43	6.99
9	Health & Fitness	32,845	29,657	2.39	7.99
10	Lifestyle	39,954	12,607	1.51	4.99
11	Medical	17,203	4,160	2.39	5.99
12	Music	42,001	32,218	2.91	7.99
13	Navigation	15,961	10,528	5.35	49.99
14	News	28,041	19,822	2.07	4.99
15	Photo & Video	37,786	31,770	1.67	4.99
16	Productivity	37,426	32,695	3.15	9.99
17	Reference	28,269	16,393	1.91	4.99
18	Social networking	51,899	26,691	1.59	3.99
19	Sports	22,374	7,173	3.55	29.99
20	Travel	24,350	10,939	2.91	9.99
21	Utilities	46,984	45,021	1.51	3.99
22	Weather	20,709	15,353	2.87	9.99
		Σ=671,106	Σ=455,347	Ø=2.27	$\leq$49.99

C. Research Data

Table I shows an overview of our data set. In total, we obtained 1,126,453 reviews from 1,100 applications (550 free, 550 paid). In the AppStore, reviews for a specific app are reset with every release of the app. Therefore, the reviews in our data set were exclusively issued after the last release of the respective application. Less than half of the reviews (518,041 or 45.99%) specified the reviewed application version. We could not explain this clearly, but we hypothesize that users

[4]excluding "Newsstand", which is an additional tag rather than a category.

[5]http://itunes.apple.com/rss/

[6]https://github.com/oklahomaok/AppStoreReview

are able to enter feedback via a browser or via the AppStore software. Only, the latter has access to the installed version.

Similarly, some reviews did not specify their publishing dates. But as we obtained the feedback in the order it appeared and since feedback dates only include the day, we could calculate all missing dates from predecessor and successor reviews. The oldest feedback was entered on 10 July 2008. Our data set therefore spans more than 4 years.

Most reviews for free apps were written in the category "Social networking" (51,889 – 7.73%), least in the category "Catalogs" (9,517 – 1.42%). Most paid apps reviews in our data set belong to the category "Utilities" (45,021 – 9.89%), least reviews were published in the category "Medical" (4,160 – 0.91%). On average, the most expensive applications belong to the category "Navigation" ($5.35 mean), while applications in the category "Catalogs" are the cheapest ($1.35 mean). Overall, the average paid application in our data set costs $2.27, while the maximum price is $49.99 in the category "Navigation". The average application price across our complete data set including free apps is $0.92.

III. Results

A. Feedback Usage

1) Feedback Frequency: Our data contains 1,126,453 reviews from 918,433 distinct reviewers, i.e. around 1.23 reviews per reviewer. Of these reviews, 671,106 apply to free apps, while 455,347 are written for paid apps. This difference is significant (two-sample t-test, $p<0.001$, CI=0.99). We thus conclude that, in total, more feedback is given for free apps than for paid apps. Most probably the difference results from the larger user communities of free applications.

In our data set we counted 568,599 distinct reviewers for free apps and 389,563 distinct authors of reviews for paid apps. This means, the user ratio of free to paid apps (1.46:1) matches the review ratio of free to paid apps (1.47:1).

We first studied the number of reviews by individual reporter. We found that 84,567 reviews (7.51%) were made by "Anonymous". In addition, we observed several other anonymous usernames, such as "????" (353 reviews) or "???" (330 reviews). In total, 57 of the top 1,000 users in our data have such anonymous usernames, leading to a total of 87,282 anonymous reviews (7,75%). The remaining reviews are authored by other non-anonymous usernames, even if it is questionable if they really identify the users. Overall, the issued reviews per user seem to follow a power-law distribution. 826,874 (90.03%) of the reviewers in our data set have written only 1 feedback. In contrast, only 1,183 (0.13%) account for more than 5 reviews. This shows that a small group of users is very active and continuously giving feedback.

On average, each user community of an app provided 22.09 reviews per day. Again, more users provide feedback for free than for paid apps. We found 36.87 daily reviews for free and only 7.18 daily reviews for paid apps (two-sample t-test, $p<0.001$). Based on medians, the number of daily feedback in the different price categories varies only slightly, between 0.38 for apps that cost $14.99 and 3.32 for free apps.

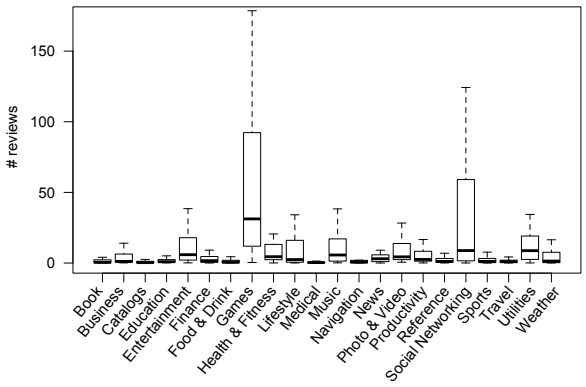

Figure 2. Daily feedback per app category.

Figure 2 shows the number of feedback submitted for the various application categories. Since the distributions are positively skewed, we use medians to report on average daily feedback. Users publish most feedback per day in the category Games (median 31.24), followed by Social Networking (median 8.82) and Utilities (median 8.75). Least feedback per day is provided in the categories Catalogs (median 0.30), Medical (median 0.34), and Books (median 0.53). An obvious interpretation of this result is that popular categories like Games with large user communities get more feedback, while niche categories with small communities get less. Another potential reason is that apps in categories like Catalogs and Books mainly offer information rendering features, while games and utilities offer more complex features sets, which stimulate more feedback. Finally, we might hypothesize that specific social domains such as gaming and social network encourage the participation of the users, who spend more time with the app and develop a special relationship to it.

On application level, most feedback was given for the free application Facebook, which ranks 6th in the category Social Networking. For this application, users published 4,275 reviews in just one single day. The average of these ratings is 3.95 stars. Least feedback in our data set was provided for the application Packers Radio & Live Scores that ranks number 16 in the category Sports and costs $0.99. In this case, 2 users provided feedback in 303 days. Both of them gave 5 stars.

Last, we investigated the user feedback behavior over time. We first estimated the timespan between each feedback and the first feedback. From the timespans, we calculated the distribution of the feedback over time. Figure 3 shows that users quickly give less feedback over time. Although the distribution is not exponential (Kolmogorov-Smirnov test rejects this hypothesis with $p<0.001$), users give most feedback in the first few days after a release, leading to a long tail over time. This suggests that user feedback is triggered by new releases.

2) Feedback Meta-Data: We first studied the **length of feedback** in our data set. Overall, feedback length ranges from 1 character to 6,000 characters. The median feedback length across all applications is 61 characters (mean is 106.09). 2,802 (0.25%) reviews comprise only one character, while 6 app

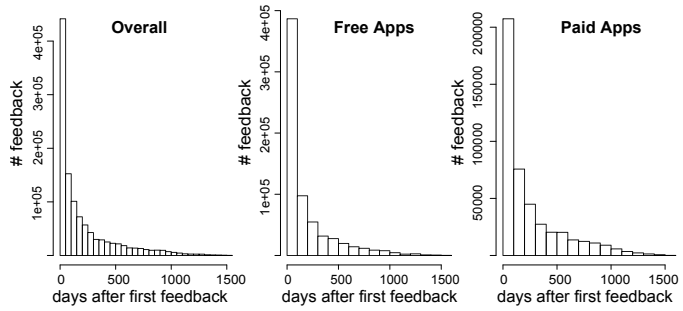

Figure 3. Relative distributions of feedback over time.

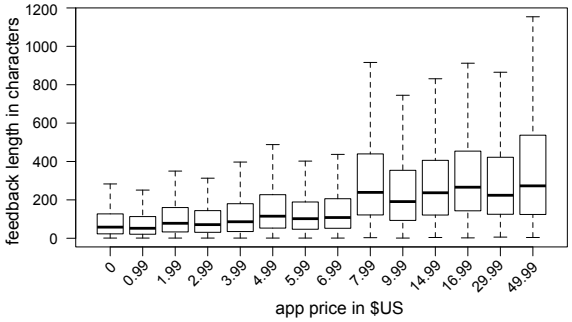

Figure 4. Feedback length by application price.

reviews contain 6,000 characters. None of these very long texts contains useful information. Rather, the corresponding users seem to have written random characters or repeated sequences of characters and white space to generate visual patterns. 76.7% of all reviews (863,951) include less than 140 characters, which is the length of a Twitter message, while 80.4% of the reviews (905,974) contain less characters than an SMS text message (160). Over 99% of the feedback contains less than 675 characters, which corresponds to around a third of a printed page. We therefore conclude that application feedback mostly consists of short messages, more similar to a tweet than to other communication artifacts such as email.

From the application vendor perspective, one possible interpretation of this result is that a lot of the messages – but not all – are useless, motivating systematic means to filter irrelevant feedback. From the user perspective, this result reveals that users do not allocate much time for giving feedback. It is also well known that mobile users use multimedia in addition to text to communicate complex content [13], [14].

As shown in Figure 4, feedback length seems to increase with application price. Although we could not directly find a significant linear correlation between app price and feedback length, we were able to show a significant increase in feedback length between lower-price and higher-price applications. To this end, we divided our data into two disjoint sets. The *lower-price* set includes applications with a price up to $6.99, while the *higher-price* set contains all apps which are more expensive. A two-sample Wilcoxon rank sum test showed that users write significantly longer feedback for higher-price than for lower-price applications (p<0.001).

Table II
FREQUENCY OF DIFFERENT RATINGS.

rating (stars)	1	2	3	4	5
frequency	130,940	46,943	69,193	181,441	697,932

Feedback length seems to be related to the rating in stars. By using a series of two-sample Wilcoxon rank sum tests, we could show the following relationships for the median length of feedback according to its rating: $m_1 < m_2 \wedge m_2 > m_3 > m_4 > m_5 \wedge m_1 > m_3$ (p<0.001). We think that this result is quite interesting. One interpretation is that users tend to write less the more they like an application – a sign that there is less to improve. Conversely, it indicates that written feedback is mostly used for improvement requests.

Ratings in our data set are overall very positive, leading to an average rating of 4.13 stars. 697,932 (61.96%) of the reviews contain a 5 star rating, 879,373 (78.07%) give at least 4 stars. Only 130,940 (11.62%) reviews rate the application with the lowest possible rating (1 star). Table II shows an overview of the underlying frequency distribution. We obtained our data set by querying the top downloaded applications in each available category. Although our data sample contains a large number of low ratings (177,887 or 15.79% reviews with less than 3 stars), we cannot exclude an influence of the applications' rank on the overall ratings (and obviously vice-versa). A χ^2-test of independence confirmed the hypothesis that rank and ratings are not independent in our data (χ^2=2,661,333, p<0.001).

In the AppStore, users can assess the quality of an existing feedback by rating its **helpfulness**. In our dataset, only 67,143 (5.96%) reviews are rated by other users regarding their helpfulness. From these, 38,519 (57.37%) are considered 100% helpful. Only 19,118 (28.47%) rated reviews are rated useless by more than half of the rating users. Interestingly, 16,671 (24.83%) are rated completely useless. This means that the user community agrees in over 82% of the cases about their opinion. Using Hartigans' dip test [7] we could show that the distribution of helpfulness is bimodal (p<0.001). As it turns out, if feedback is rated, it is regarded either very helpful or very useless by the user community.

B. Feedback Content

To investigate the content of user feedback in our data set, we employed an iterative content analysis technique. We first drew a stratified random sample [4] of our data. Stratified sampling divides the population into mutually exclusive groups, corresponding to the app categories and the pricing model in our data set. We randomly selected 12 reviews from each of the 22 application categories for both free and paid applications, leading to 528 reviews. We felt this threshold is large enough to conduct statistical tests while a manual analysis is still feasible. Next, two researchers *independently* "coded" the random sample to identify topics included in the feedback. We allowed the assignment of multiple topics since most feedback contained more than a single topic.

Table III
TOPICS IN USER FEEDBACK. $N = 1100$.

#	topic	description	freq.
t1	praise	expresses appreciation	75.36%
t2	helpfulness	scenario the app has proven helpful for	22.45%
t3	feature information	concrete feature or user interface	14.45%
t4	shortcoming	concrete aspect, user is not happy with	13.27%
t5	bug report	bug report or crash report	10.00%
t6	feature request	asks for missing feature	6.91%
t7	other app	reference to other app, e.g. for comparison	3.91%
t8	recommendation	suggests acquisition	3.82%
t9	noise	meaningless information	3.27%
t10	dissuasion	advises against purchase	3.27%
t11	content request	asks for missing content	2.91%
t12	promise	trades a better rating for a specific improvement	2.00%
t13	question	asks how to use specific feature	1.27%
t14	improvement request	requests improvement (e.g. app is slow)	1.18%
t15	dispraise	opposite of praise	1.18%
t16	other feedback	references or answers other feedback	1.09%
t17	howto	explains other users how to use app	0.91%

The coding process included two iterations on the single reviews. We started with an empty set of topics. We then read each review and identified a topic describing the information included. If the corresponding topic was in our list, we used it, otherwise we added it with a short description and an example. This description was shared between the coders, allowing them to "interpret" the written feedback. We did not look for strictly syntactic entities, but for semantic entities. For instance, the topic "praise" could be assigned to feedback including "cool!". "great!", or "awesome". On the other side, sometimes we had to interpret sarcasm or ironic feedback, e.g., "I lost all my phone contacts. Great, thank you!". This was the major motivation for using manual analysis instead of text mining.

We obtained two sets of topics, a set T_1 with 16 items and a set T_2 with 21 items. By discussing about the identified topics, we discovered that T_1 was a strict subset of T_2, but that T_2 allowed more specific distinctions regarding certain information entities. We therefore decided to take T_2 as base for further coding, with one modification: We removed the 4 least frequent topics, as they had been found only in 1 or 2 reviews. The resulting 17 topics served as our coding guidelines for the rest of the content analysis.

Before the final coding round, we doubled our random sample to 1,100 reviews. This allows us to make predictions about our data set at the 95% confidence level, accepting an error margin of 3%. Both researchers then coded this larger sample, again independently from each other. Finally, we discussed about feedback for which the assigned codes did not match and decided together for a final code.

1) Feedback Type: The final results are shown in Table III. We found that the most popular topic is "*praise*", which denotes any kind of praising the application. This topic is predominant in over 75% of the analyzed samples. The second most popular topic is "*helpfulness*", which describes a use case or situation where the application proved to be helpful to the user. It is predominant in over 20% of the feedback. Further the topics "*feature information*" as well as "*shortcoming*" are predominant in over 13% of the analyzed feedback sample.

Table IV
USER FEEDBACK TOPIC CATEGORIES. $N = 1100$.

category	topics	frequency
rating	t1, t12, t15	856 (77.82%)
user experience	t2, t3	359 (32.64%)
requirements	t4, t5, t6, t11, t14	340 (30.91%)
community	t7, t8, t10, t13, t16, t17	146 (13.27%)

While reading the sample feedback, we made three interesting observations. First, the **quality of feedback** in our random sample varies quite strongly. On the one hand, there are app reviews with a high quality, which suggest interesting new features and justify their suggestions profoundly. On the other hand, some feedback does not add any value to the numeric rating of the application. Second, users tend to become insulting quickly. Especially when they have spent money, they seem to forgive the developers nothing. Feedback like "Fire the idiot who designed this app!" is far from constructive and explains developers' disenchantment with user feedback. Third, users seem to complain frequently about removed or changed features. We see this as an indicator for users' getting used to specific workflows with an application. Changing an application in a way that affects users' workflows seems to be a source of dissatisfaction.

To estimate information diversity in the feedback, we calculated the distribution of number of topics per feedback. Overall, 6 (0.55%) reviews contained 5 topics, 22 (2.00%) reviews contained 4 topics, 116 (10.55%) reviews included 3 topics, 427 (38.82%) reviews 2 topics, while 528 (48.00%) of the reviews contained only one topic. In other words, the majority of feedback (52%) contains more than one topic.

To further interpret and compare the information in feedback, we grouped the resulting topics into four themes:

1) **Community**. These topics represent community and social aspects. Specifically, references to *other feedback* and *other apps*, *questions* to other users, *howtos* explaining how to use the application, as well as *recommendations* and *dissuasions* are included in this theme.
2) **Requirements**. This theme captures topics related to the improvement of an application. All requests – *feature*, *content*, and *improvement requests* – as well as *shortcomings* and *bug reports* belong to this theme.
3) **Rating**. This theme includes topics that are related to a judgement of an application, i.e. *praise* and *dispraise*, but also *promise*, which expresses the user's intention to change her judgement given certain improvements.
4) **User experience**. This theme comprises topics related to descriptions of the app in action. These are *helpfulness*, which captures use cases where the application has proven helpful, and *feature information*, which includes descriptions of application features and user interface.

Table IV summarizes the identified themes together with the included topics, and shows their frequency across the random sample. Frequency denotes the number and percentage

of feedback that includes at least one of the associated topics. In our random sample, "*rating*" is by far the most frequent theme with a frequency of over 77%. This corresponds to the main intention behind application distribution platforms after the distribution itself, which is giving other users indicators for good applications and thus guaranteeing a high quality among the applications. The second most frequent theme is "*user experience*" which is predominant in nearly one third of all feedback. We think that the high popularity of this theme is interesting. It suggests that users tend to share their experiences with other users and developers, presumably to justify their statements about an application such as ratings, recommendations, or dissuasion. The theme "*requirements*" is predominant in around 30% of all feedback. This suggests that despite the overall quite positive ratings, users often externalize requests for improvement. Last, the theme "*community*" shows up in around 13% of the analyzed samples. We think that this number is quite high, given the original purpose of the AppStore. It might suggest that users naturally tend to form communities, i.e. to react to each other, ask questions, or publish possibly helpful information.

2) Feedback Patterns: We used the frequent itemset mining algorithm ECLAT by Zaki [18] to identify co-occurrences of topics in our data sample. Itemset mining [1] is a data mining method for discovering relationships between different variables based on their co-occurrence in databases. It takes as input a database containing at least two different variables as well as a parameter specifying the minimum support σ for the relationships to discover. The output includes frequent itemsets, i.e. sets of values which co-occur in at least σ percent of the data. Our goal was to find sets of topics which co-occur in our data set with a higher frequency than others. Consequently, we built a database, which contained the topic codes for each feedback in our random sample. We ran the ECLAT algorithm with a minimum support of $\sigma = 0.01$ and a minimum pattern length of 2, which means that results should only include itemsets with at least 2 topics.

Table V shows the 20 patterns, which we obtained with these thresholds. The most frequent pattern is {helpfulness, praise}, which is present in more than 20% of all feedback. It describes the usefulness of an application together with a positive rating. A concrete example in our data set is "Great for uploading receipts on the go. Easier than reconciling on the computer." The second most frequent pattern {feature information, praise}, which applies to over 14% of our random sample is similar to the first, with the difference that it describes more concretely a positive feature or functionality of an application. A concrete example in our data set is "I love that this app takes less than ten seconds to let you know where your battery life is!!! I love it." The third pattern {feature request, praise} is predominant in nearly 5% of our random sample. It illustrates positive feedback which also includes a feature request. From such a pattern we would expect a lower rating of the application than we would for instance from the first pattern. To test this hypothesis, we investigate regularities between feedback content and ratings in the following section.

Table V
FREQUENT TOPIC PATTERNS IN USER FEEDBACK. $N = 1100$. ASTERISKS MARK PATTERNS WHICH ARE NOT CLOSED.

#	pattern	support	Ø rating
p1	{helpfulness, praise}	22.18%	4.86
p2	{feature information, praise}	14.18%	4.83
p3	{feature request, praise}	4.64%	4.37
p4	{helpfulness, feature information, praise}	4.27%	4.87
p5	{helpfulness, feature information}*	4.27%	4.87
p6	{praise, recommendation}	3.73%	4.90
p7	{other app, praise}	2.64%	4.79
p8	{praise, shortcoming}	2.64%	4.24
p9	{content request, praise}	2.27%	4.60
p10	{dissuasion, shortcoming}	1.82%	1.45
p11	{helpfulness, praise, recommendation}	1.73%	4.95
p12	{helpfulness, recommendation}*	1.73%	4.95
p13	{bug report, dissuasion}	1.27%	1.21
p14	{bug report, shortcoming}	1.27%	1.57
p15	{bug report, praise}	1.18%	4.23
p16	{feature information, praise, recommendation}	1.09%	4.83
p17	{feature information, recommendation}*	1.09%	4.83
p18	{improvement request, praise}	1.00%	4.18
p19	{feature information, other app, praise}	1.00%	4.91
p20	{feature information, other app}*	1.00%	4.91

Table VI
DISTRIBUTION OF RATINGS ACROSS TOPICS IN USER FEEDBACK. – □ 1 STAR, □ 2 STARS, ▒ 3 STARS, ■ 4 STARS, ■ 5 STARS.

#	topic	Ø rating	rating distribution
t8	recommendation	4.88	
t2	helpfulness	4.85	
t3	feature info.	4.81	
t17	howto	4.80	
t1	praise	4.78	
t11	content request	4.25	
	Ø sample rating	*4.08*	
t14	impr. request	3.92	
t7	other app	3.91	
t6	feature request	3.89	
t9	noise	3.67	
t16	other feedback	3.67	
t13	question	2.86	
t12	promise	2.27	
t4	shortcoming	2.10	
t5	bug report	1.84	
t15	dispraise	1.69	
t10	dissuasion	1.39	

C. Feedback Impact

1) Market Impact: To study the market impact of feedback, we explored relationships between application ratings and feedback topics as well as feedback patterns. We first tested the independence of the identified topics and the final rating of the application by the user with a number of χ^2-tests. The results show that the topics "*other app*" (p=0.90), "*other feedback*" (p=0.69), and "*howto*" (p=0.56) are independent from the rating, while all others are not (p<0.05).

To further study the impact of specific feedback types on the app rating, we calculated for each topic the distribution of the associated ratings as well as the average rating across all feedback where it occurs. Table VI illustrates the results and relates them to the overall average rating of the feedback in our random sample. The topic leading to the most positive reviews is "*recommendation*", followed by "*helpfulness*" and "*feature*

information", while the topic with the most negative reviews is "dissuasion", followed by "dispraise" and "bug report".

These results allow for two interpretations. First, we can order the *requirements* topics according to their impact on user ratings. *Content requests* (4.25 stars on average) are the least critical requirements feedback. Their average rating even lies above the overall average rating of the sample. *Improvement requests* (3.92 stars on average) and *feature requests* (3.89 stars on average) are more critical, but their average rating still lies above the theoretical middle of 3 stars. *Shortcomings* (2.10 stars on average) have a definite negative impact on rating. *Bug reports* (1.84 stars on average) are most critical. Second, the results suggest that *user experience* topics are mainly included in positive reviews. In other words, users do not tend to include their experience with an application in negative feedback. Since this information is critical for corrective maintenance [20], this finding indicates that user feedback in current form will unlikely help developers to improve their applications.

To break the impact of topics on ratings further down, we calculated the top five topics per rating. Table VII shows the results which confirm our findings. *Shortcomings*, *bug reports*, and *feature requests* have high influence on negative ratings, while *helpfulness* and *feature information* on positive ratings.

Next, we investigated the relationship between feedback patterns and specific ratings. To this end, we calculated the average rating for each of the identified patterns. The results are included in Table V. The average ratings vary largely across the different patterns. The most positive (closed) pattern is {helpfulness, praise, recommendation}, which has an average rating of 4.95 stars. As we had hypothesized, the pattern {feature request, praise} has a lower average rating (4.37 stars). The pattern {bug report, dissuasion} accounts for the lowest average rating (1.21 stars), which corresponds to the negative message of reporting a bug and dissuading other users from buying the application.

2) Community Impact: To study the impact of feedback on the user community, we investigate relations between feedback helpfulness and feedback length, content, as well as rating. At first glance, we could not observe any linear or polynomial correlation between the feedback length and its helpfulness for other users. However, a χ^2-test showed that feedback length and helpfulness rated by other users are statistically dependent (χ^2=2043547, p<0.001). To further study this phenomenon, we split feedback into three groups according to its helpfulness. *Low helpfulness* indicates that up to 33% of the users who rated the feedback, found it helpful. *Medium helpfulness* indicates that 33–66% of the users who rated the feedback found it helpful. *High helpfulness* includes feedback considered helpful by more than 66% of the users.

By using a series of two-sample Wilcoxon rank sum tests, we were able to show the following relationships for the median feedback length according to its helpfulness: $m_{low} < m_{medium} > m_{high} \wedge m_{low} < m_{high}$ (p<0.001). This result means that feedback with low helpfulness is the shortest, while feedback with medium helpfulness is the longest in our data set. When going beyond the descriptive level, the significance of the length difference between low and high helpfulness feedback is open to dispute, as the median lengths differ only by 7 characters ($m_{low} = 114, m_{medium} = 144, m_{high} = 121$). In contrast, the difference to feedback with medium helpfulness lies between 23 (19.01%) and 30 (26.32%) characters.

Table VIII
HELPFULNESS OF TOPICS IN USER FEEDBACK. $N = 74$.

#	topic	avg. helpfulness	N
t6	feature request	90.33%	7
t3	feature information	86.36%	11
t1	praise	83.93%	51
t16	other feedback	83.33%	3
t2	helpfulness	81.50%	18
t11	content request	75.00%	4
t14	improvement request	70.83%	4
t7	other app	67.33%	7
t17	howto	65.14%	5
t4	shortcoming	63.18%	13
t5	bug report	53.47%	14
t10	dissuasion	0.00%	1
t12	promise	0.00%	2
t8	recommendation	0.00%	1

Remarkably, all feedback with rated helpfulness is significantly longer than other feedback. The 67,143 reviews which have been rated in our data set have a median length of 121 characters, while the median length of the remaining 1,059,310 reviews is 58 characters, which is *less than half*. This difference is significant (two-sample Wilcoxon rank sum test, p<0.001), confirming that longer feedback is more likely to be rated by others, possibly as it contains more information.

Next, we studied how feedback helpfulness is related to feedback content. Unfortunately only 74 reviews (6.73%) in the random sample used to explore feedback topics, have been rated according to their helpfulness. This number does not allow us to generalize the relation between topics and helpfulness. Nevertheless, to explore possible relationships, we calculated the average helpfulness per topic for this set. Table VIII shows that the topics, *feature requests*, *feature information*, and *praise* are included in the most helpful feedback, while *recommendation*, *promise*, and *dissuasion* are included in the least helpful feedback.

Last, we investigated relationships between feedback rating and helpfulness. A χ^2-test showed that these variables are statistically dependent in our data set (χ^2=11952.97, p<0.001). To further analyze this relation, we calculated the average helpfulness per feedback rating. Figure 5 shows the result in the form of a sequence of violin plots, one for each rating. Violin plots are similar to box plots, but also illustrate the probability density of the underlying data. The mean helpfulness is marked with a red line for each rating. A series of two-sample t-tests showed the following significant relationships for the mean feedback helpfulness according to its rating: $m_1 < m_2 < m_3 < m_4 < m_5$ (p<0.001).

This result means that feedback which rates the application better is considered more helpful by other users. One possible interpretation of this result lies in the main use case of app stores, i.e. allow users to find good applications. We think that users browse feedback in order to understand if an application

Table VII
TOP FIVE TOPICS PER RATING.

#	1 star ($N = 138$)	2 stars ($N = 56$)	3 stars ($N = 58$)	4 stars ($N = 166$)	5 stars ($N = 682$)
1	shortcoming (50.00%)	shortcoming (55.36%)	shortcoming (31.03%)	praise (92.77%)	praise (97.07%)
2	bug report (46.38%)	bug report (33.93%)	bug report (22.41%)	helpfulness (18.67%)	helpfulness (31.23%)
3	dissuasion (18.84%)	dissuasion (12.50%)	feature request (22.41%)	feature request (18.07%)	feature inform. (19.21%)
4	promise (7.25%)	feature request (7.14%)	praise (20.69%)	feature inform. (15.66%)	recommendation (5.43%)
5	other app (5.07%)	promise (5.36%)	noise (12.07%)	shortcoming (10.84%)	feature request (3.67%)

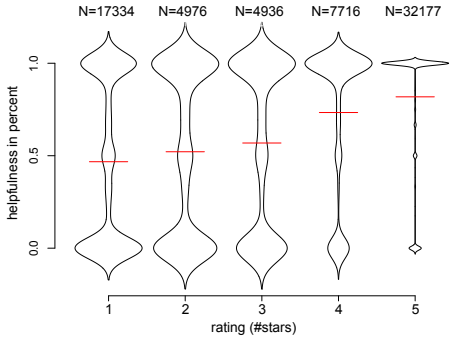

Figure 5. Feedback helpfulness and rating. $N = 67139$.

is perceived to have a high quality by the community. In this case, their risk of buying a pig in a poke is low. Therefore, users might consider better reviews, which typically describe application features and use cases as we found out, as better decision support, and consequently rate them more helpful. In contrast, it is hard to imagine that users will take the time to rate feedback as helpful, which gives the application a low rating. In this case users would rather leave the download area and browse for an alternative.

IV. DISCUSSION

We discuss our findings and their implications by revisiting the main motivating questions from the introduction.

1) Can app stores serve as communication channel among users and with developers? Our results show that the answer is yes. App users continuously post dozens of comments per day, addressing developers and the user community. Comments to the developers range from praise and thankfulness, e.g. "great app, just what I needed." to bug reports e.g. "since update I can't open links from email, please fix." and feature requests, e.g. "I wish it would notify me if I go below a certain dollar amount". Community topics include recommendations to other users, direct questions on specific features, answers to those questions, or explanations on how to use the app.

Current app stores lack *bidirectional* communication features such as replying to or referencing posts (as known in social media). This prevents developers from contacting specific users to ask clarifying questions or inform them that their problem has been addressed. As most users post comments shortly after new releases, one possible way to enhance the communication is to link changes and new features to user feedback, which influenced or led to these changes. This would increase the *involvement* of users and help to understand the change rationale. We also think that users should be personally involved with personal accounts and profiles. This prevents unqualified content, increases users' motivation, and allows developers to contact specific users.

2) How can developers use app stores to better and quicker understand evolving user needs and requirements? Even if most feedback repeats the ratings in natural language, users also frequently share their needs, ideas, and experiences in the AppStore – an important source of information for app designers and analysts. Other studies showed that developers embrace user feedback as important information to improve software quality and identify missing features [12].

About one *third* of the feedback includes topics on software requirements and user experience, varying from shortcomings and feature requests to scenarios in which the app was helpful and descriptions of specific features. Requirements analysts can immediately profit from this feedback, gaining insights on how their applications are *actually* being used, beyond simple download and sales numbers, and being able to assess the importance of features, which might facilitate release planning. Moreover, driven by their own needs, some users develop and share extensions, workarounds, and ideas, which others might profit from, and which might deliver inspirations and ideas for new features [16]. Finally, feedback like feature descriptions or howtos can be used as starting-point for documentation.

We found that reviews are generally brief messages, similar to tweets. In other domains like movies or hotels, feedback tends to be 3–4 times longer [9]. Users seem to *not* be willing to spend much time on writing reviews, and rather focus on the actual task they are performing with the app. However, the amount, frequency, and content of feedback show that users are willing to share their comments, experiences, and ideas. This suggests that software vendors should investigate means for minimizing the feedback submission effort, e.g. by suggesting textual descriptions and pro-actively encouraging users to share their experiences. In the same direction, we found that user experience is often missing in negative reviews such as shortcomings and bug reports. This hinders developers from improving their apps only from this textual feedback [20]. One promising approach to overcome this "deadlock" is to automatically collect user experiences by instrumenting the app and monitoring the interactions of users at runtime [10]. Contextual data will enrich textual feedback and help developers to better understand the needs and the concrete situations (i.e. context) in which they have emerged.

3) Which methods and tools should support software organizations to *analyze and aggregate* user feedback? App stores include useful feedback but also noise. We found that feedback quality varies widely, from helpful messages

for other users and developers to insulting offenses. Thus, analysts and developers would profit from tools and methods that systematically filter and aggregate feedback. In particular, we think that researchers should investigate predictor models to classify feedback (e.g. bug reports and shortcomings), filter important "warnings" to which developers must react, or evaluate project success early on. There is already considerable research on mining bug reports which can also be tested on app stores e.g. to detect duplicate or similar feedback.

We also found several regularities, which can be used to build such models. First, our results show that most feedback is provided shortly after new releases, with the frequency decreasing quickly over time. This means that the feedback is related to new features and changes in the release. Analyzing the feedback over several releases would allow for an early detection of problematic releases or features. Second, an important finding is that the community typically agrees about the helpfulness of feedback, which allows for aggregating it. Third, we identified several correlations between the numerical ratings, helpfulness, and the textual feedback. We hypothesize that there would be more correlations if more metadata about users, versions, and usage data would have been available. Last, the result of our manual analysis is an annotated corpus of real data[7], which can be used to train a classifier, e.g. to identify actionable data in newly submitted feedback [5].

We found that users tend to give reasons for their ratings. One implication of this finding is that users are aware of the rating potential and willing to provide helpful explanations to the vendors. Indeed, our study confirms that feedback content has a real impact on the market, since more positive messages usually also lead to better application ratings and vice versa.

In general, positive feedback is more appreciated by the user community. This can be explained by the main use case of application distribution platforms, i.e. to support users in finding good applications. Likewise, longer feedback is more likely to be regarded helpful by other users, probably because it contains more information. From a methodological perspective, we are convinced that it is important to integrate user feedback gathering and analysis in the software development lifecycle. We think that analyzing feedback in app stores can complement other requirements engineering activities such as workshops and interviews in particular when time is critical and many incremental releases are planned.

V. Results Validity

Our study was neither designed to be generalizable nor representative for all app stores. However, we think that the results have a high degree of generalizability, in particular for Google Play and Windows Phone Store. Although the entire population is publicly unknown, we think that our results are representative for user feedback in the AppStore. First, our dataset includes all user feedback from the 1,100 most downloaded applications. Second, we conducted statistical tests, checking the results' significance and excluding hazard factors. Third, the results of descriptive statistics and content analysis are similar. Nevertheless, there are three limitations, which should be considered when interpreting the results.

First, we obtained our data by scraping feedback from the AppStore based on a list of *most downloaded* applications. This might have resulted in a relationship between download numbers and other variables, such as the ratings. But since our data sample still contains a large number of low ratings (177,887 or 15.79% reviews with less than 3 stars), we feel confident with the results. Specifically, we were interested in how ratings relate to other variables like content and helpfulness, rather than finding absolute numbers within the population such as the overall average rating of feedback.

Second, in order to explore topics in user feedback, we sampled 1,100 reviews from our data set, as a manual analysis of the complete data set is unfeasible. The statistical evidence is less strong than for the complete data set. We tried to mitigate this threat by choosing a sample size, which allows to make a generalization to our data set at the 95% confidence level, accepting an error margin of 3%. Moreover, we selected a stratified random sample, which guarantees that the resulting sample equally considers different categories in the data set.

Third, we studied relations between feedback helpfulness and content based on a small number of reviews. However, other analyses, e.g. of relations between ratings and helpfulness indirectly confirm our findings. Nevertheless, we encourage other researchers to replicate our content analysis study, purposefully selecting feedback based on its helpfulness.

Finally, we made two simplifying assumptions during our analysis, which might partly limit the construct and internal validity of the results. A number of reviews did not explicitly specify the publishing date, so that we needed to interpolate these values. Since we obtained the feedback in the order it had appeared online, we could compensate for the missing values by investigating the date of the reviews that were published directly before and after. We are confident that this does not influence the resulting data, since feedback date does only include the day and the data gap was never larger than 1 day. In particular, this algorithm never changes the order of feedback.

To analyze feedback content, we relied on manual analysis. The results are subject to experimenter bias. To reduce this risk, *two* researchers conducted pair analysis *independently* from each other. We iterated this activity by refining the rating criteria to improve the inter-raters agreement. We only reported on results where the agreements were over 90%.

VI. Related Work

Application distribution platforms are a recent phenomenon. Nevertheless, there are a few of studies about them already. Chen and Liu [3] present a preliminary study on the popularity of applications in the AppStore. They found that the top-ranked paid applications are not necessarily closely correlated with customer ratings, what is confirmed by our study. In contrast, Harman et al. [6] mined the Blackberry app store for technical, customer, and business aspects of applications. The authors found a strong correlation between an application's

[7]The study data sets are available at http://mining.socialse.org

ratings and its download numbers, while no correlation seems to be present between price and rating as well as price and download numbers. While we found that application ratings and ranks are not statistically independent in our data set, we were not able to show a correlation. Instead, we described relationships between price and feedback length, and showed that users write more feedback for more expensive applications.

Zhou et al. [19] investigated the phenomenon of repackaging third-party Android applications. The authors found that developers frequently repackage legitimate apps from the original store to distribute them on third-party marketplaces. Zhou et al. showed that as many as 5-13% of the apps hosted on third-party marketplaces are repackaged and that the main use of repackaging is to replace existing in-app advertisements or embed additional ones to "steal" or re-route advertisement revenues, which illustrates the increasing importance of application distribution platforms as a business model for software companies. Yamakami [17] analyzes app stores from the business perspective and presents the underlying business models and key success factors. The author illustrates that viral marketing depends on the fact that users talk about their *experience* in both the app stores and social networks. Our study confirms this finding and shows that descriptions of user experience are typically missing in negative reviews.

Chandy and Gu [2] aim at classifying spam in the AppStore, in the light of recent "bogus" reviews. Such reviews can deceive users to download spam apps or to ignore apps which are victims of negative review spam. The authors present a latent model which is able to classify apps, developers, users, and reviews into normal and malicious categories. Our study confirms that reviews have an impact on the market and the user community. Moreover, it confirms the importance of being in the top lists, since top downloaded apps are not necessarily rated better in the AppStore. Research is assessing the apps and the reviews quality. Researchers like Hong et al. [8] work on the automatic classification of reviews that allows to distinguish helpful and useless reviews.

Finally, Seyff et al. [15] and Schneider et al. [13] propose to continuously elicit user requirements with feedback from mobile devices, including information on the application context. Our study confirms the importance of these approaches, and suggests that app stores could serve as platform to collect, exchange, and manage user requirements and user experience.

VII. Conclusion

User feedback and user involvement are crucial for modern software organizations. Users increasingly rate and review apps in application distribution platforms, called app stores. While part of this feedback is superficial and at most has an impact on download numbers, others include useful comments, bug reports, user experience, and feature requests. This can help developers to understand user needs, extending the application in a "democratic" fashion, towards crowdsourcing requirements. However, current platforms do not allow developers to systematically filter, aggregate, and classify user feedback to derive requirements, or prioritize development efforts. Developers would also benefit from enriching textual feedback with usage and context data.

Acknowledgement

This work was supported by the EC (FastFix project, grant FP7-258109). We thank Bernd Bruegge for valuable feedback.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD Conference on Management of Data*, pages 207–216, Washington, DC, 1993. ACM.

[2] R. Chandy and H. Gu. Identifying spam in the iOS app store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality - WebQuality '12*, pages 56–59, Lyon, France, 2012. ACM.

[3] M. Chen and X. Liu. Predicting Popularity of Online Distributed Applications: iTunes App Store Case Analysis. In *Proceedings of the 2011 iConference*, pages 661–663, Seattle, WA, USA, 2011. ACM.

[4] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting Empirical Methods for Software Engineering Research. In *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer-Verlag London, 2008.

[5] M. A. Ferrario, W. Simm, J. Whittle, P. Rayson, M. Terzi, and J. Binner. Understanding Actionable Knowledge in Social Media: BBC Question Time and Twitter, a Case Study. In *ICWSM*, pages 455–458, 2012.

[6] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: MSR for app stores. In *9th IEEE Working Conference on Mining Software Repositories*, pages 108–111, Zurich, Switzerland, 2012. IEEE.

[7] P. M. Hartigan and J. A. Hartigan. The dip test of unimodality. *Annals of Statistics*, 13(1):70–84, 1985.

[8] Y. Hong, J. Lu, and J. Yao. What Reviews are Satisfactory: Novel Features for Automatic Helpfulness Voting. In *35th international conference on Research and development in information retrieval*, pages 495–504, Portland, Oregon, USA, 2012. ACM.

[9] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the Stars: Exploiting Free-Text User Reviews to Improve the Accuracy of Movie Recommendations. In *1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, 2009.

[10] W. Maalej and D. Pagano. On the Socialness of Software. In *Proceedings of the International Conference on Social Computing and its Applications*, Sydney, Australia, 2011. IEEE.

[11] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Pub., 2002.

[12] D. Pagano and B. Bruegge. User Involvement in Software Evolution Practice: A Case Study. In *Proceedings of the 35th International Conference on Software Engineering*, San Francisco, CA, 2013. IEEE.

[13] K. Schneider, S. Meyer, M. Peters, F. Schliephacke, J. Mörschbach, and L. Aguirre. Feedback in Context: Supporting the Evolution of IT-Ecosystems. In *Product-Focused Software Process Improvement, LNCS 6156*, pages 191–205. Springer-Verlag Berlin Heidelberg, 2010.

[14] N. Seyff, F. Graf, and N. Maiden. End-user requirements blogging with iRequire. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, page 285, New York, New York, USA, 2010. ACM Press.

[15] N. Seyff, F. Graf, and N. Maiden. Using Mobile RE Tools to Give End-Users Their Own Voice. In *IEEE International Conference on Requirements Engineering*, pages 37–46, Sydney, Australia, 2010. IEEE.

[16] E. von Hippel. Lead Users: A Source of Novel Product Concepts. *Management Science*, 32(7):791–805, July 1986.

[17] T. Yamakami. A Three-Dimension Analysis of Driving Factors for Mobile Application Stores: Implications of Open Mobile Business Engineering. In *2011 IEEE Workshop of International Conference on Advanced Information Networking and Applications*, pages 885–889, Biopolis, Singapore, Mar. 2011. IEEE.

[18] M. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[19] W. Zhou, Y. Zhou, X. Jiang, and P. Ning. Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. In *2nd conference on Data and Application Security and Privacy*, pages 317–326, San Antonio, Texas, USA, 2012. ACM.

[20] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss. What Makes a Good Bug Report? *IEEE Transactions on Software Engineering*, 36(5):618–643, Sept. 2010.