2017 43rd Euromicro Conference on Software Engineering and Advanced Applications

An Automated Feedback-based Approach to Support Mobile App Development

Simon Scherr, Frank Elberzhager, Konstantin Holl
Fraunhofer IESE
Kaiserslautern, Germany
{first.last}@iese.fraunhofer.de

Abstract—**The acceptance of mobile applications is highly dependent on the realized set of features and on the quality of the application. Information about their acceptance can be gained quickly by collecting and analyzing user feedback such as explicit textual reviews provided by an application's users or implicitly provided usage data. With an approach based on developing a minimal set of functions in order to realize a minimum viable product (MVP), it is possible to put a product on the market within a short amount of time. Currently, the elicitation, analysis, and processing of user feedback is unfocused and takes too much time and effort to mitigate the poor quality of the application. Hence, we outline an approach named Opti4Apps, which is aimed at tailored quality assurance as part of MVP development and enables and expands the benefits of an MVP by providing a semi-automated feedback elicitation, analysis, and processing framework. This is intended to raise the effectiveness and efficiency of early user feedback consideration during further development in order to assure the quality and acceptance of an app as an MVP. We will present the overall structure as well as the process behind the Opti4Apps framework. As proof-of-concept, we implemented an initial prototype of our idea, focused on textual user feedback.**

Keywords—mobile; user feedback; automation; MVP

I. Introduction

The relevance of using mobile apps has heavily increased in recent years, which is also reflected by the number of devices: 353 million laptops and desktop PCs versus 563 million smartphones and tablets sold during this period [1]. This pervasiveness of mobile devices makes it possible, among other things, to extend existing business processes with new innovative functionality (e.g., context-sensitive input assistants) to further support such processes. A defect-free process is a prerequisite for the acceptance of a mobile application. New technology fields, such as smart ecosystems or the Internet of Things, also require high quality of mobile apps. High quality is often cited as the most critical factor in these areas [2]. One typical example of a quality issue in the mobile area is the change from one mobile network to another [3]. Disconnection in a critical situation, e.g., during a mobile process control activity [4], can stop the respective process and may even lead to monetary loss. Consequently, a quality assurance approach is needed that explicitly covers mobile-specific issues and considers mobile-specific characteristics. Furthermore, the typical development process and goals of this area also need to be considered, such as agile practices or the common goal of short time to market.

For the successful establishment of innovative products and services, the right timing is essential. When trends are not addressed or innovative software products enter the market too late, the risk is high that the investment will be wasted. In order to address this problem, more and more companies follow a lean startup process [5]. Instead of releasing a fully functional software product whose development time and cost might be very high, an early product is deployed. Such a minimal viable product (MVP) contains an initial set of features at an acceptable quality. Based on customer feedback, the software product is further developed and optimized step by step.

In this contribution, we will focus on such customer or user feedback and investigate how such feedback can be gathered in an automated way, and how a company can benefit from such feedback. The two research questions are phrased as follows:

RQ1: How to capture user feedback automatically?
RQ2: How to use captured user feedback automatically as part of the development process?

This article is structured as follows: Section 2 presents related work on MVP and user feedback. Section 3 provides an overview of the Opti4Apps framework, which was defined to capture feedback and analyze feedback in order to optimize mobile apps. The approach is evaluated in Section 4, and Section 5 concludes this article.

II. Related Work

A. App Development as Minimum Viable Product

There are two ways of publishing products or services. First, using traditional development, a product is developed fully, meaning it is released with a large number of features. The drawbacks of this approach are: establishment on the market takes place very late; the costs until market entrance are very high; and a product's marketability and/or its idea or vision is determined only at this late point in time.

Second, a product can be placed on the market in an iterative manner, using many small development cycles. Often a clear, long-range vision linked to the product is missing in this approach. Better is iterative development by creating a so-called MVP – a minimum viable product. Here, the goal is to position an initial, viable product on the market as quickly as possible, then successively extend and optimize it based on experience and customer feedback. In this regard, Kromer mentions four aspects: customers, value proposition, channels, and relations. Value propositions are made to customers using channels and feedback about established relations is collected [6].

Initially, the majority of the customers belong to the group of so-called early adopters, people who recognize trends early and are willing to explore innovations. This allows reducing the risk of developing an unmarketable product. Additionally, maximizing the insights by learning about customers can be performed with low effort.

This learning is related to those characteristics that are essential for a product's or service's long-term success on the market and for creating loyal customers. A strategy including an MVP is usually associated with rapid introduction of features that are released quickly, including some that may still be in a prototype state [7]. By doing this, their usefulness can be determined without the need to fully implement them. A prominent example for this is the file hosting service Dropbox. Dropbox initially released a video as its MVP, which described the functionality of the product.

However, after a product or its underlying service has been validated successfully for a certain amount of time, a broader market entrance has to be made, addressing more than just early adopters. While early adopters consciously accept the deficiencies of the product because innovation is the most prominent aspect and appropriate communication takes place, this is not the case for the majority of users. One example of this is the introduction of the iPhone in 2007. Early adopters consciously accepted the lack of functionality such as the missing copy-and-paste function as well as missing 3G connectivity in order to be among the first users [5].

B. *Capturing of User Feedback*

Software add-ins like the tool OpenProposal [8] and mobile applications like iRequire [9] and ConTexter [10] enable large groups of users to describe requirements and requests for a product and provide the ability to substantiate these with additional artifacts, such as pictures. Solutions for problems can be elicited even in mobile and distributed work environments. These solutions assume that the stakeholders have the intrinsic motivation to contribute their problem descriptions, which is not required in the Opti4Apps approach. However, clustering such data reveals tendencies of the development to dynamically identify feedback. Categorization of problem descriptions through semantic similarities (e.g., nearest-neighbor heuristics and lexical characteristics with cosine or distance measures [11]) makes it possible to detect similar requests based on the coherence of messages. In this way, dispensable messages can be excluded. Besides, the comparison of users by means of a k-nearest-neighbor heuristic helps to group contributors in order to predict who may possibly have similar needs and thereby identify a global need [12]. Such heuristics may support the intent of Opti4Apps by transferring the heuristics to quality assurance.

User comments on Internet platforms, in app stores, message boards, and bug trackers can be collected and analyzed automatically. To do so, scientific approaches using algorithms from the area of natural language understanding have been developed to analyze the comments (e.g., AR-Miner [13]). Some applications also use statistics, like the popularity of a suggestion in relation to the number of reactions to it. Providers like UserVoice [14] combine the feedback of an application with a bug tracker. Moreover, there are techniques for revealing creative solutions as well as typical problems based on the behavioral patterns resulting from usage mining. Lessons learned from these approaches can be used for the implementation of Opti4Apps. MyExperience [15], e.g., is a system that enables semi-automated elicitation of the usage data of mobile devices. This comprises the interpretation of sensors, the duration of calls, and other events. Also, the user can provide feedback using a form, such as rating the speech quality of a phone call. Unlike Opti4Apps, MyExperience only comprises certain events of the smartphone, not events of new applications under development.

Appsee [16] provides automated analysis of the dwell times of screens within the developed application as well as touch heat maps, videos showing its usage, navigation diagrams, tracking of clicks, and other events. According to Appsee, the integration into one's own application takes place within a single line of code. The insights are illustrated on a web-based platform. Acquisition of usage data (usage frequency, duration, misentries) is similar to the concept of Opti4Apps.

III. The Feedback-based Development Approach

Figure 1 provides a conceptual overview of the Opti4Apps approach: We assume an application deployed as an MVP on a mobile device and users who use the app and can give feedback. This feedback comprises app usage data (e.g., usage frequency, duration, or misentries, similar to Appsee [1]), state (e.g., installation and online state), and explicit user feedback (reviews, bug reports). Feedback can be provided automatically (e.g., via a specific agent running on the mobile device), semi-automatically (e.g., some data is tracked by the mobile device and has to be sent manually to a backend), or manually (e.g., users provide some written feedback in the app store). In other words, feedback can be provided explicitly by the user or implicitly through measurement by technical means.

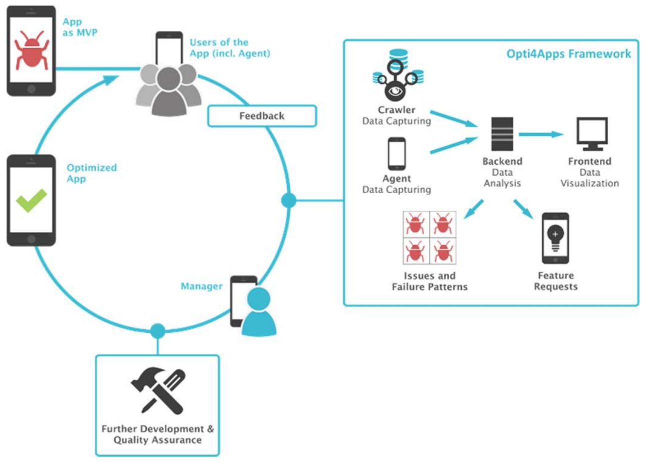

Figure 1 Opti4Apps Approach Overview

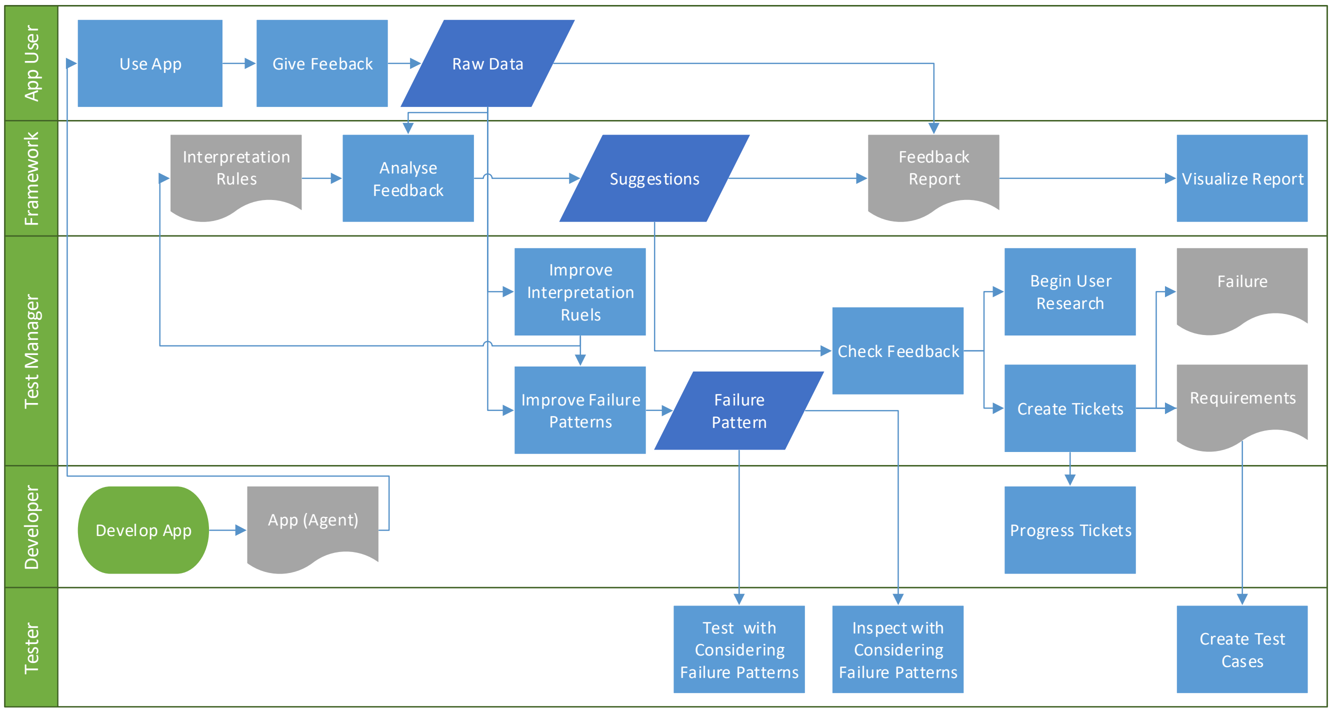

Figure 2 Overview of the Opti4Apps process

The feedback then has to be analyzed and provided in a suitable way so that the company developing the app gets the information it needs to improve the app. For this purpose, we propose the Opti4Apps framework. It uses, for example, the data provided by the agent integrated into the mobile application, direct feedback, or feedback gathered by performing data mining analyses to detect existing deficiencies or reveal improvement ideas. This huge amount data is consolidated by a backend and classified to generate a human suitable overview.

The analysis of the collected information in the backend via data mining is intended to enable a fast learning effect with respect to existing deficiencies. Thus, it can provide a baseline for effective further development as well as for focused quality assurance. Identified failure patterns [17] can increase the effectiveness of the quality assurance of the current development or lead to new features. The improved app is then again distributed to users for further feedback.

A. Process

In the following, we describe the process (see Figure 2) behind the Opti4Apps approach. A part of the Opti4Apps framework is integrated into the app during development time. This integration means that an agent is placed in the app, which is later used for collecting data for usage mining. When the app is deployed to the end user, it is possible to gather feedback from the user. The approach includes two kinds of feedback. The user can provide feedback for improving the app passively or actively. This is done in the following ways: The agent within the app is able to gather data for usage mining in the background, without any additional effort for the user, as soon as the gathering process has started. The user himself can actively give feedback within the app by using dedicated feedback features and contact forms, which directly send data to the app provider. Furthermore, users typically provide valuable information by posting app store reviews, ratings, statements in social networks, forums, and other places.

This raw data is used for gathering insights about the app using text mining methods. For this purpose, the text mining component crawls dedicated sources like the corresponding app data in the app stores, social media sources, and texts sent directly to the app vendor, such as customer support requests. For example, raw text feedback could be the following: “The button is too small so that clicking is difficult.” The collected raw data is then analyzed afterwards. The same is done with the data captured via usage mining. For both cases, interpretation rules are used within the data miner component.

Performing this analysis allows creating improvement suggestions for the app and getting new proposals for interpretation rules. Furthermore, it is possible to complement and overhaul the failure patterns used until that time. Updating the rules and patterns allows evolving the quality of the mining as well as that of the app, as rules and patterns are adapted to the needs of the app. Considering the example above, the improvement suggestion would be to make the buttons larger.

The test manager checks the improvement suggestions and creates new tickets based on the logged misbehavior as well as new requirements. These tickets are addressed by the developers. The testers consider these insights when creating and executing test cases. Similarly, reviewers consider them for their review. This again improves the quality of the app.

B. Components of the Opti4Apps Framework

The conceptual architecture (see Figure 3) of the Opti4Apps framework consists of two main parts: application and analysis. In the first part, usage data is generated by an app user or textual feedback is given by the app user via various media, while the second part consolidates, analyzes, and presents the gathered data.

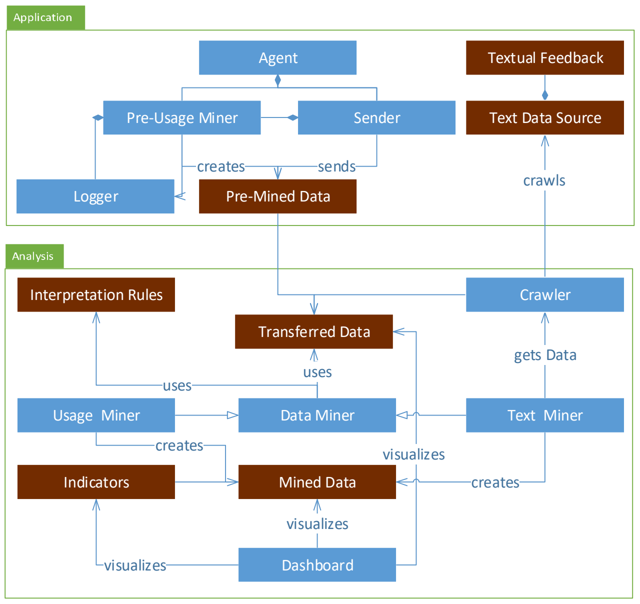

Figure 3 Opti4Apps Conceptual Architecture

1) The Application Part

When an app is deployed and distributed as an MVP, an agent is running on the mobile device. An agent is a software library that can be implemented directly in the app, but can also run as an independent library component. It consists of three main components: a logger, a pre-usage miner, and a sender. The logger tracks several actions of the app user, such as which screens in the app are used for how long in which order, how often an app crashes, what the user types in, which gestures are performed, etc. Besides such interaction data, device data, such as sensor data or performance data, can also be logged. A pre-usage miner decides whether every data point is considered, or whether only such data is captured which was configured as relevant and therefore interacts with the logger component. Finally, a sender component is able to send data packages to the Opti4Apps framework. This can be done continuously or at specific points, depending on factors like Internet connectivity. Besides the app on the mobile device, further data sources are considered that usually provide textual feedback from the user. These may be app stores, forums, and social media channels such as Twitter, in-app feedback functionalities, or even transcribed phone calls from a hotline.

2) The Analysis Part

While data from the mobile device is sent to the system at defined times, a crawler component is necessary to gather textual data from the different sources. Here, the framework captures feedback from the users who are giving it away freely.

A central storage environment for textual feedback information is provided, which would otherwise be scattered around in different communication channels and therefore would only be available partly to the relevant stakeholders. The framework provides a common and shared view on all the different feedback sources. As the APIs of such textual sources are usually different, the crawler has to cope with these requirements and has to be flexible. This means that an infrastructure needs to be in place that allows the integration of more specific crawler components following the exact specification of an abstract one. This provides the flexibility of diverse data sources being unified by a general crawler concept. As a concrete example, we have an abstract crawler for app store feedback that is specialized by concrete incarnations for gathering feedback from the Apple App Store and from the Google Play Store.

3) View on the Data

All kinds of data that arrive at the system are part of the transferred data. These can be raw or already pre-mined usage data and device data from the mobile device, or text data from textual sources. Within the backend, another data miner, usually a more powerful one compared to the mobile version, is applied on the gathered data. This data miner serves as the analysis component and checks the gathered data, resulting in mined data. Basically, pre-mined data from the mobile device is also part of this mined data. The data miner is also divided into a usage and a text miner component to address the different kinds of data analysis adequately. The respective miners then use interpretation rules (either for text or usage) to come up with concrete improvement indicators, which may be feature wishes or qualitative improvement suggestions, for instance.

4) Examples

Consider the following two examples, which describe the same issue, but once taken from textual feedback and once from usage feedback:

- A user writes feedback in the app store: “The start button is not large enough!”

  This feedback is crawled and stored in the backend, i.e., it belongs to the ‘transferred data’. The text miner then uses the interpretation rule ‘usage of the term “button” and synonyms for “small” results in the issue that the button is too small’. The resulting improvement indicator is that the button has to be enlarged.

- A user taps several times around the start button before the app is started; these actions are logged.

  The pre-usage miner uses the interpretation rule that when a user taps several times besides a button before hitting it, the button is too small. This defect pattern can then result in the same improvement indicator that the button should be enlarged.

Such improvement indicators have to be checked by a human such as a test manager or a product owner, or a confidence level has to be added so that the validity can be assessed.

The results can ultimately be shown in some frontend dashboard, where different widgets can be used for visualization purposes. Filters might be used to focus on specific aspects of the analyses.

IV. Evaluation of the Opti4Apps Framework

A. *First Version of the Implemented Opti4Apps Framework*

Based on the conceptual framework described above, we created an initial version to get insights into the opportunities the Opti4Apps framework will bring to mobile application quality assurance. The first version described here is related to the user feedback analysis part of the framework. It consists of two main parts: a backend for data collection and data analysis, and a web-based dashboard for visualizing the raw data and the insights found by the data analysis (see Figures 4 and 5).

For the backend, we started with the crawler components to be able to capture raw feedback. Here we make use of three different crawler components for collecting app store reviews from the Apple App Store and the Google Play Store. Relevant aspects of the collected feedback are data source, title, text, date, and the optional rating between one and five stars. As the data model for review feedback is shared among the stores, the approach is also feasible for capturing cross-platform feedback in the analysis phase.

For the initial analysis, we created a quantitative overview of the captured feedback. Furthermore, we integrated first aspects of text mining into our Opti4Apps framework. The data overview is mostly based on the captured raw data itself. It provides an overview of the amount of data captured in which period of time and from which data source. Besides, an overview of the distribution of the ratings is created, which provides information the kind of data the analysis is based on.

The text mining analysis starts by adding information to the raw texts in a form that allows text-mining techniques to be applied more easily. First of all the texts get a lemmatized representation to enable analysis of just the root forms of the words. Based on the lemmatization, we remove typical fill words. We are just adding this information and always keep the initial raw user feedback. Furthermore, we use dictionaries to replace Internet slang terms with their common equivalents e.g. “coz” would be replaced by “because”. We are approach capable of doing this in English and in German.

Having reached this more suitable form for an analysis, we create a bag-of-words model of the transformed text. This gives insights on how often a word occurs in a text by ignoring grammatical forms and positions in the text. To find more similar expressions and feedback, we also create a bigram model based on word units. This kind of model can be used to refine insights found via the bag-of-words technique. This allows identifying similar reviews more easily. For example, in the bag-of-words representation, “drains battery” would be identified as two items “drain” and “battery”; however, our bigram would be “drain battery”. When accessing the data for “battery”, we can immediately identify “drain battery” as a potential subcategory, but we also see that feedback for “drain” might be related to the battery. It is also possible to relate “improve battery” to “drain battery” due to this representation.

A concept that is part of our framework is emotional analysis. Texts on the Internet are enriched with emoji and emoticons. Newer mobile operating systems for touch devices even suggest follow-up emoticons while typing on the keyboard. This makes emoticons present throughout app store reviews. Many emoticons are used by people to express a certain mood, feeling, or attitude. Such an emotion can be captured by analyzing the emoticons present in the reviews. This is a huge advancement compared to traditional text analysis, which can only gather emotional feedback by doing a semantic analysis of the text. In the case of Internet texts, this emotional feedback is already included as metadata within the text in the form of emoticons. Therefore, we create a bag of emoticon representation per text by using our emotional classification scheme (see Table 1). Having captured the emotional feedback allows us to make statements about the mood a certain review is based on, e.g. whether a user is only dissatisfied that a feature is missing or whether he is angry about this. Furthermore, we are able to make a statement on the overall attitude towards the product. This allows deeper insights than just looking at app store ratings.

Table 1 Emoji Classification Scheme

Type	Subtypes
Positive	Excited; Happy; Funny
Neutral	Judgment depends on context; Without any judgment
Negative	Angry, Bored, Sad, Scared
Surprised	-

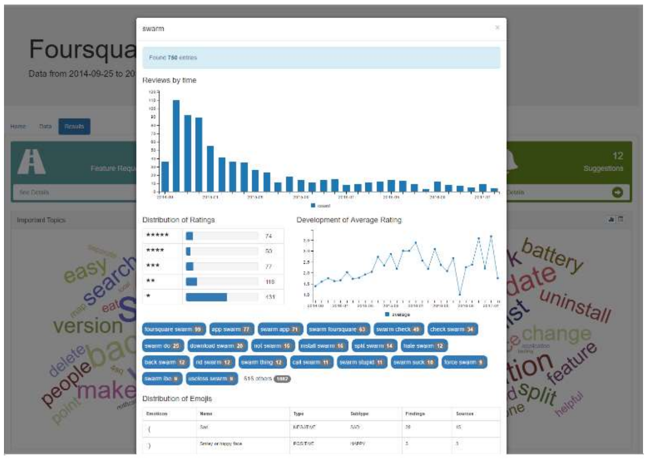

Figure 4 Detailed data for feedback relating Swarm

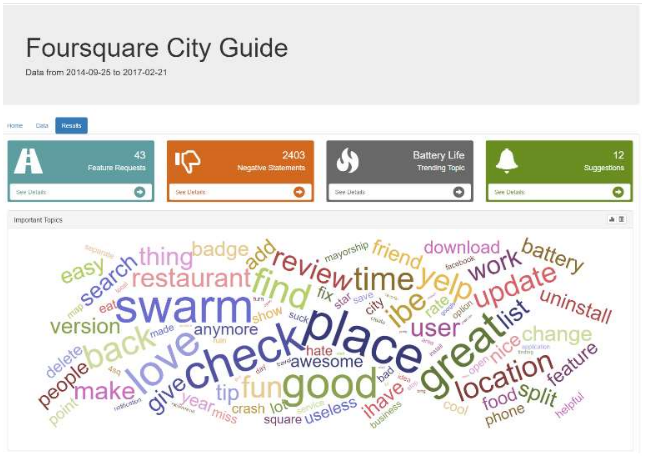

Figure 5 Word Cloud for Foursquare City Guide

The quality of the result depends strongly on the amount and quality of the user feedback. It is important to get a critical mass of users to capture a representative amount of feedback. Having too few user statements might lead to wrong conclusions not representing the wishes and problems of the majority of the users. From a technical perspective, capturing and storing textual feedback from data sources works pretty well. The subsequent analysis procedure builds on standard text analysis methods adapted to the specialties used for our texts like Internet language, but also considers information like ratings, the date, app store, etc. and not only the text itself.

We visualized the analysis results in a web-based dashboard. This was done for several reasons. First, it allows ease of access through a simple website; furthermore, we can easily present the results visually in multiple ways. We decided for the beginning to have three major tabs providing information to the user: Home, Data, and Results. At the top of each tab, we present relevant key data in order to provide even valuable information with just a glance at the dashboard. The home tab serves as a summary of the data tab, focusing on raw data, and the results tab focuses on the analysis results.

The data tab shows insights about the amount of data currently in focus, the distribution over time, the sources, and the distribution of the ratings. In the results tab, you will find a word cloud (see Figure 5) based on our bag of words, as well as the result of the overall emotional analysis. We always provide a trace to the raw data on which an analysis result is built. This gives the user the opportunity to check on which data any aggregated or calculated element is based. Furthermore, the results contain suggestions regarding what should be changed in the app. Currently, these suggestions are created manually.

B. Evaluation

Before evaluating our framework, we evaluated our emotional feedback model (step 1). This showed a homogenous view on how people perceive emoticons in relation to emotions. Then we evaluated our framework using app store reviews of three apps (step 2). For this purpose, we captured reviews from Foursquare City Guide [18], IKEA Catalog [19], and WhatsApp Messenger [20] from the Apple App Store and the Google Play Store.

Step 1: We evaluated the emoticon classification by means of a survey. At the time of writing, this study was still in progress and in the following, we present the results based on the first 24 participants. We asked the participants to map emoticons to emotions. The participants used our classification scheme (see Table 1) to classify them. Our participants were on average 31.5 years old. There were 13 males and 11 females in the group. We later used the data from the survey to improve our classification for the framework. It turned out that most people had a common understanding of the emoticons presented, and perceived and used them in the same way. There was a very clear separation between the main categories and for most of the emoticons, the separation between the subcategories was very clear. We will continue to evaluate these results to understand these correlations even better.

Step 2 – Foursquare: We had 6147 reviews for Foursquare, given between 2014-09-25 and 2017-02-21. The data for IKEA catalog consisted of 5841 reviews, made between 2012-10-17 and 2017-03-10. The captured feedback from both apps covered several years, which allows detecting trends in the reviews as well checking relations to released updates. Due to the technical limitations for the Google Play Store, the feedback about WhatsApp was only captured for a short period. Nevertheless, it is very interesting as we were able to capture feedback regarding a controversial change that WhatsApp performed regarding its status functionality. We captured 13,204 reviews between 2017-03-14 and 2017-03-20.

Besides analyzing the quantitative data, we did an investigation of the bag of words and bigrams by manually identifying problems and checking the time, they were reported. We were able to detect several issues from the reviews. Those

Table 2 Example Issues found

App	Issue	Description	Reports	Solution
Foursquare City Guide	Battery drain	The app caused battery consumption that was too high for some users.	153	App updates seem to have improved battery life as complaints decreased a lot after the peak.
Foursquare City Guide	Swarm split	Swarm was introduced as a new app; some features were moved to it and some deleted.	750	Not solved, people just got used to it over time.
Foursquare City Guide	Crash	Some users experienced crashes immediately after starting the app.	102	App update solved the issue, as it was a temporary peak.
Foursquare City Guide	Check-in	The check-in functionality was removed and moved to Swarm.	421	Not solved, people got used to it over time.
Foursquare City Guide	Badges	The feature was removed from Foursquare and was later introduced in Swarm.	183	Not solved, people got used to it over time.
Ikea Catalog	Catalog Download	Catalog download is too slow.	463	-
Ikea Catalog	Catalog Functionality	The catalogs are more like a PDF version and do not offer any interactive features.	1174	-
Ikea Catalog	Crash	Some users experienced crashes immediately after starting the app.	229	App update solved the issue.
WhatsApp	Status	WhatsApp changed the permanent user status to a short time status.	1529	WhatsApp announced that it bringing the old status back.
WhatsApp	Contact List	The status update removed the contact list from the app.	393	App update solved the issue, as it was a temporary peak.
WhatsApp	Game Ratings	Users gave ratings to WhatsApp and commented on it, as it was a game.	345	No WhatsApp issue as these are fake reviews.

issues were later compared to release notes and press articles for the three apps. You can find the example issues in Table 2.

In the following, we will show example findings we encountered within the Foursquare, IKEA, and WhatsApp app feedback.

Foursquare had an average rating of 3.19 on the usual five-star rating scale typically used in app stores. It is important to mention that this rating does not reflect the overall rating of the app. It uses the ratings that also include an app review. It turned out that 37% of the reviews were five-star reviews and 30% were one-star reviews. The huge number of very positive and very negative aspects makes this an interesting example, as improving the MVP means keeping the good things and improving the bad things.

Foursquare was split into Foursquare City Guide and Foursquare Swarm in May 2014 [21]. Some features were moved to Swarm, while others were removed altogether. This was a hot topic in the reviews as it caused many changes. Important features no longer offered were the ability to check in to places, get badges, and become mayor of a city. According to our data, many people were upset that two apps were now needed to get less than before. This negative feedback mainly occurred at the end of 2015. We also saw that the app had much lower ratings in the time following these changes. There was a high correlation between one-star ratings and the occurrence of Swarm, mayorship, and check-in in a review. Also, the low ratings seem to have more negative emoji than other reviews.

In May 2015, Foursquare brought “mayorship” and other features to Swarm in a similar form as before [22]. After making those changes to Swarm, it was mentioned much less compared to the time before. Furthermore, the store ratings for Foursquare went up because people got used to having two apps instead of one. Nevertheless, the number of ratings itself decreased a lot in the analyzed time. This could be seen as a hint that Foursquare City Guide is not so popular anymore.

Due to the changes made to the app, many people started comparing the app to Yelp [23]. As it turned out, many people saw large similarities between the two apps and some believed that Yelp would be the better alternative.

Step 2 – IKEA Catalog: The versions in use in 2014-12 / 2015-01 and 2015-06 seemed to have bugs at app startup. The first of those two app versions also appears to have caused battery consumption issues for some users as more people than usual reported battery drain issues.

The 5841 reviews of the IKEA catalog had an average rating of 2.95. For this app, too, many of the ratings were either very positive (five stars: 33%) or very negative (one star: 37%). While the median number of reviews did not decrease as for Foursquare, the ratings seem to go down during the observation period. Starting with a monthly average of over four stars, the ratings subsequently dropped to an average of less than three stars starting from July 2015, including months with less than two stars.

One important topic for the IKEA catalog app are the catalogs itself. In 62% of the cases, the feedback related to downloading catalogs had only a one-star rating. People were complaining that the download times for the catalogs were too high, and that it was sometimes not working at all. In addition, critical views were expressed regarding the usability of the catalogs. People reported that it appeared slow to use and they saw no benefit, often considering it a slow and cumbersome PDF look-alike version of the paper catalogs. Therefore, most feedback containing the word slow is related to download times and the use of the catalogs.

In addition, the IKEA Catalog app appears to have had two versions containing considerably more crashes than other versions. One was in use in January and another one in August of 2013. 65% of the user stating a crash gave a one-star rating.

Step 2 – WhatsApp: We also examined the reviews for the WhatsApp Messenger, which is one of the most popular messenger apps. Therefore, it gets hundreds of reviews per day. To align the timespan between the Google Play Store and the App Store, we were only able to analyze a few days.

Of the 13,204 reviews, the average rating was 3.95 stars. 11% of the feedback was related to WhatsApp’s brand new status functionality at that time. This makes it a good example of how responsive user feedback is. These reviews provided an average rating of 2.5 stars, resulting in a significant influence on the overall rating. Looking at the emoji used for this topic, most were negative, with the most frequent subtype being sad or angry. The reviews show that the people did not like the short-time status and wanted the old status functionality back. As this feature was introduced with the version in use at that time, it is interesting to analyze the feedback regarding the keyword *version*. 11% of the feedback belongs to this group, with an average rating of 2.4 stars. A similar result can be found for the feedback related to *update*. Many people felt that WhatsApp tried to copy Instagram and Snapchat with this feature. Knowing that the status functionality had just been changed and looking at the results presented in the dashboard, a quality assurance team could extract the information that the users responded very negatively to the change introduced in this update. In response to the huge amount of criticism, WhatsApp has announced that they will reintroduce the old status feature [24]. The same update also removed the contact list from the app itself. Of the 393 reviews mentioning *contact*, 43% gave a one-star rating. Most people complained that they did not find their contacts any more.

Besides, we noted several hundred ratings for WhatsApp regarding the keyword *game*. More than 95% of these ratings were four or five stars and their content was totally unrelated to the messenger app. Spam bots might created these reviews as they praise the app, e.g. as an excellent casino or gaming app.

Discussion: Considering the results for only three apps shows that by applying the text-mining approach from Opti4Apps we are able to gather deep insights into the users’ attitude towards an app, its features and problems within a short period due to our automated capturing and analysis process. It furthermore provides knowledge about critical changes between two updates that might be loved or hated by the users. The approach was feasible for detecting issues within apps by analyzing the long-term results for two apps and the short-term results for one app. It showed that some issues remain strongly visible over a long period of time, which means that they are not solved, but also that it is possible to get insights on short-time issues that affect a significant number of users. We also observed that the users' view on an app may still change over time even if it app is not often subject to huge changes, like the IKEA Catalog app. Things that might have been considered great in the past might just be okay now, and in the future users might even see them as negative. On the other hand, we saw that users can also get used to once unpopular changes like the split of Foursquare City Guide. The vendor reacted to user feedback by performing changes and, in the end, the users learned to cope with the situation. We also saw that users give comparative feedback. They are not only giving feedback about the app itself, but often also give hints regarding what competitors seem to do better. Our approach was feasible for detecting and clustering app issues out of the masses of user feedback, and was able to identify the reasons for this as well as possible improvements.

V. Conclusion and Future Work

In this article, we presented the Opti4Apps framework, which was developed to consider user feedback in order to improve mobile apps. This is important as apps have usually short release cycles as well as a short time-to-market. Our approach makes use of the concept of an MVP and integrates feedback directly into the improvement process of the iterative development. This is done in an automated way that is capable of handling huge amounts of feedback in a short period. As this approach is also able to cope with combinations of different kind of feedback types and feedback sources, we can capture a representative picture of the app users. These factors make it an efficient way to get insights on the users' mind. Furthermore, we can easily capture trends. Doing so with traditional methods would require huge efforts as apps are often not only developed for one single combination of devices and platforms. We distinguish between textual and usage feedback and consider several feedback channels, such as app store feedback, social media, or automatically captured data. Besides an overview, we presented the process and an architecture view (RQ1). Furthermore, we evaluated two aspects with a focus on textual feedback: First, we investigated whether emoji's, which are a major part of textual feedback nowadays, are assessed in the same way by most people, which is most often the case, so they can be assumed to be highly valid during the interpretation even if used by different people. Second, we analyzed textual feedback regarding three large apps to see whether we can derive trends and gain new insights. Our Opti4Apps framework turned out to be able to perform the analysis automatically, and trends could be identified. Such feedback can help developers and quality engineers (RQ2).

In the future, we also want to focus on usage feedback to get further feedback, and to combine the results from such an analysis with the results of the textual feedback analysis. For this scenario, maintenance of the users' privacy will also be considered deeply. We also want to deepen the textual feedback analysis. Finally, we have to date only analyzed about 100 emoji to identify an initial trend regarding the validity of interpretation by different people, but we want to extend this kind of analysis with more emoji.

Acknowledgment

The research described in this paper was performed in the project *Opti4Apps* (grant no. 02K14A182) of the German Federal Ministry of Education and Research. We would also like to thank Sonnhild Namingha from Fraunhofer IESE for linguistic support

References

[1] Statista GmbH, Prognose zum Absatz von Tablets weltweit in den Jahren 2010 bis 2019. http://de.statista.com/statistik/daten/studie/165462/ umfrage/prognose-zum-weltweiten-absatz-von-media-tablets, 2015

[2] Rombach, D., Smart Ecosystems: An Enabler for Future Innovations, Software Engineering and Business, Innovation at the Era of IoT, 2015

[3] Aguilera, R., Top 10 Reasons iOS and Android Apps Crash. Mashable. http://mashable.com/2012/07/20/why-ios-android-apps-crash, 2012

[4] Zühlke, K., Mit der "Android-App" von unterwegs die Produktion sicher überwachen. WEKA FACHMEDIEN GmbH. http://www.elektroniknet.de/elektronik fertigung/sonstiges/artikel/86749/0, 2012

[5] The Lean Startup, The Movement That Is Transforming How New Products Are Built And Launched, http://theleanstartup.com, last access 10/15/2014

[6] Kromer, T.: „The four parts of a Minimum Viable Product", http://grasshopperherder.com/the-four-parts-of-a-minimal-viable-product, 2014

[7] Venture Hacks: „What is the minimum viable product?" Interview with Eric Ries, http://venturehacks.com/articles/minimum-viable-product, 2009

[8] Rashid, A., Wiesenberger, J., Meder, D., Baumann, J., Bringing developers and users closer together: The OpenProposal story. In: A. Heinzl, H.-J. Appelrath, E. J. Sinz, et al. (Eds.), Proceedings of the PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (CEUR-WS 328), 9–26, 2008

[9] Seyff, N., Graf, F., Maiden, N., Using Mobile RE Tools to Give End-Users Their Own Voice. In: 18th IEEE International Requirements Engineering Conference, pp. 37–46. IEEE, Sydney, 2010

[10] Gärtner, S., Schneider, K., A Method for Prioritizing End-User Feedback for Requirements Engineering. In: 5th International Workshop on Cooperative and Human Aspects of Software Engineering. IEEE International, 2012

[11] Natt och Dag, J., Regnell, B., Gervasi, V., Brinkkemper, S., A linguistic-engineering approach to large-scale requirements management. IEEE Software, 22(1), 32–39, 2005

[12] Dheepa, V., Aravindhar, D. J., Vijayalakshmi, C., A novel method for large scale requirement elicitation. International Journal of Engineering and Innovative Technology, 2(7), 375–379, 2013

[13] Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B.: AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace. In: 36th International Conference on Software Engineering. In press, 2014

[14] UserVoice, Feedback & Online Help Desk Software, https://www.uservoice.com, letzter Zugriff am 03.11.2014

[15] Froehlich, J., Chen, M., Consolvo, S., Harrison, B., Landay, J., MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. InProceedings of the 5th international conference on Mobile systems, applications and services (MobiSys '07). ACM, New York, NY, USA, 57-70, 2007

[16] Appsee, http://www.appsee.com, Appsee Mobile Analytics, last access 11/04/2014

[17] Holl, K., Vieira, V., Focused Quality Assurance of Mobile Applications: Evaluation of a Failure Pattern Classification, 41st Euromicro Conference on Software Engineering and Advanced Applications, pp. 349–356, 2015

[18] Foursquare, https://foursquare.com/, Foursquare, last access 03/31/2017

[19] IKEA, http://info.ikea-usa.com/Catalog/, last access 03/31/2017

[20] WhatsApp, https://www.whatsapp.com/, last acess 03/31/2017

[21] Foursquare Blog, http://blog.foursquare.com/, last access 03/13/2017

[22] Foursquare Blog, http://blog.foursquare.com/, last access 03/13/2017

[23] Yelp, https://www.yelp.com/, last acess 03/31/2017

[24] Techcrunch, https://techcrunch.com/, last access 03/20/201