Guidelines for User-oriented Software Feedback Systems

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Ilja Golland

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Bernhard Rumpe

Registration date: 01.11.2015
Submission date: 29.02.2016

Eidesstattliche Versicherung

Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum	Unterschrift

Contents

List of Figures

Abstract

People nowadays use software on both desktop computers and on mobile phones more than ever before. Especially on mobile phones apps are being developed for nearly every purpose in everyday life. When they first came out, they were mostly interesting gadgets for technically versatile users. At the same time, due to the increase in popularity, former target groups extend to more and more users of different technical knowledge. Many of them do not only have a different technical skill set, but also are not as technically savvy as technical experts. In addition to this, even with the increasing popularity of electronic devices in general and on mobile phones, frustration is still a persistent phenomenon. While technical experts may be aware of how to handle technical problems and solve them on their own, everyday users might have more problems. In the same matter, communication between users and developers is key when solving frustration and problems. Our work sets a novel approach considering this group of everyday users with no prior technical background and no higher technical skill set. We have studied both developers and users to analyze this gap in communication. Our results showed that developers have a certain set of essential information they need to help users. In addition to this, we saw partial evidence, that these users might be able to deliver this kind of information for certain problems. With these findings, further investigations can be conducted to establish extended results and gain better insight into the users' ability of reporting problems.

Überblick

Heutzutage benutzen Menschen Software sowohl auf Computern als auch auf Handys mehr als je zuvor. Gerade auf mobilen Geräten werde Apps für nahezu jeden Zweck im Alltag entwickelt. Als Apps gerade erst neu waren, waren Sie zunächst interessante Gadgets für technisch versierte Benutzer. Zur gleichen Zeit haben sich, auf Grund der steigenden Beliebtheit, die Zielgruppen mehr und mehr erweitert auf Nutzer mit unterschiedlichem technischem Wissen. Viele von ihnen haben nicht nur unterschiedliche technische Fähigkeiten, sondern sind auch nicht so technisch versiert wie ein Experte in dem Gebiet. Dazu kommt, dass auch mit der zunehmenden Beliebtheit von elektronischen Geräten im Allgemeinen und von mobilen Geräten, Frustration immer noch ein bestehendes Begleitphänomen ist. Während sich Experten bewusst sind, wie sie mit technischen Problemen umgehen und sie auch selbstständig lösen, hat der alltägliche Nutzer vermeintlich mehr Probleme. In diese Kontext ist die Kommunikation zwischen Entwicklern und Benutzern entscheidend, wenn es darum geht Frustration und Probleme zu bewältigen. Unsere Arbeit stellt einen neuen Ansatz hinsichtlich dieser Gruppe von alltäglichen Nutzern ohne technischen Hintergrund und ohne fortgeschrittene technische Kenntnisse. Wir untersuchten sowohl Entwickler als auch Nutzer um diese Lücke in der Kommunikation zu analysieren. Unsere Ergebnisse zeigten, dass Entwickler einige gewisse Daten und Informationen benötigen um Nutzern helfen zu können. Zusätzlich haben wir teilweise Belege dafür entdeckt, dass diese Nutzer in der Lage sind diese Informationen beschreiben und liefern zu können. Mit diesen Grundlagen können zukünftige Untersuchungen getätigt werden, für weitere Resultate und ein besseres Verständnis für die Fähigkeiten der alltäglichen Nutzer bezüglich des Beschreibens von Problemen.

Acknowledgements

First and foremost I want to thank my supervisor Florian Heller for letting me write a thesis about this particular topic. His guidance and mostly his own interest in my topic continuously encouraged me to work and research further, which was further supported by his trust towards me and I am very grateful for.

Moreover I want to thank Prof. Dr. Jan Borchers and Prof. Dr. Bernhard Rumpe for being my first and second examiner. In this regard I also appreciate that I was able to work on my thesis right at the chair at any time I wanted to.

In addition, I would like to thank Christian Corsten for helping me beforehand on deciding what kind of topic I should choose to work on.

During our user studies I encountered lots of participants who not only showed interest in my work but also encouraged me to pursue this research. Therefore I would not only like to thank all the participants for taking their time and being part of my studies, but also for keeping me motivated and showing me that this work is and will be appreciated.

Likewise I appreciated the help of my friends who gave me critique and advised me during the writing and helped proofreading it.

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in Canadian English.

Download links are set off in coloured boxes.

File: myFile[a]

[a]http://hci.rwth-aachen.de/public/folder/file_number.file

Chapter 1

Introduction

"It is [. . .] a mistake to assume that being a consumer or being a designer would be a binary choice"

—Gerhard Fischer [2002]

It is often heard and read that technology is constantly evolving, nowadays even faster than ever. What is often missed and forgotten though, is that technology is not only just created by people, it is more importantly also made for people. At first glance it might appear evident, but those who make something and those who use it do not necessarily have the best relationship, if there is any at all. Quality assurance is a regular part of the development process which checks the fulfillment of certain requirements. Together with UI/UX design and engineering, these methods can help improving the usability of certain products. But what comes after the development? Is the product ready to be used by end customers? Surely, this is not the case as we still see unfinished software, used by all kinds of people. People with different technical and educational backgrounds. This is also heavily supported by the fact, that oftentimes software has a certain deadline and is released, regardless whether it is finished or not.

Development of a product does not stop after the deployment

In any case, it ends up in the users' hands. But this does not complete the process. Software is a variable property, it can change all the time and needs to be maintained, made safe for all that will come. But what exactly will come? This is a tough challenge for developers. As mentioned before, technology is constantly evolving and developers have to keep up, not only with their fresh and new processes but also keeping their old work fresh. To a certain part this is done by staying up to date with new security issues and keeping up with new versions of operating systems, for example. These are general aspects of software, they apply globally and are at least to certain degrees foreseeable.

Keeping software up to date does not only rely on technological process but also on users

A different challenge are the users. It is said that the average user does not exist, making this everything but foreseeable [Galitz, 1997, p. 769]. Thus, the developers rely on these to give feedback in order to use it and to improve the product and satisfy their users. But still, these problems can be considered coming from the developers' perspective. Users have the problem, that they use software and get frustrated. This frustration unfortunately often stays inside. They tend to either get annoyed by it and continue working regardless, or ask others for help [Bessiere et al., 2003]. In some cases the cause for this is technical ignorance but oftentimes the fault lies within the software.

Communication is a crucial component in the solution

In those cases the developers are completely unaware of this, because no feedback is generated, and therefore it does not reach them. Analogue to the other problem, this is the problem from the users' perspective. It becomes clear that the key to solve these two problems and connect them is communication, more specifically feedback.

While the issue of lack in communication sounds simple, it comes with a lot of other problems. Nowadays, there are more and more platforms for which developers write software. With the boom of wearables or smartwatches, software gains new dimensions. But desktop computers and mobile devices still amount to a large percentage on the consumer market [Smart Insights].

An essential part is if different platform developers need different information

While they both are software platforms, in some aspects developers work completely different. This leads to the assumption that those developers might also require different kind of information when dealing with feedback. Which in turn gives enough reason to analyze this matter and see what developers do with feedback and what they require to be able to do so.

This matter would deal with one side. It would give an insight to how the requirements on different platforms differ, if they do at all. If there is no significant difference, universal feedback might be a feasible solution.

This process also raises other questions. It is quite certain that technically versatile people are able to describe the sources of their frustration and submit this to the developer. A different issue are the reasons for what is holding them back from doing this more actively. But what about the masses of people? What about the non-technically versatile users who also use software and also become frustrated (maybe even more than the others)? The question becomes this: Are average non-technically savvy users just unable to write useful feedback, or are they actually able to and their reasons for not doing this intersect with the reasons the technically skilled users have?

The other question being, if users are not able to provide useful feedback or have reasons similar to technically skilled users for not being active

The latter case would further help with the general problem. Research would only need to focus on the reasons which hold users back from giving feedback. In addition, this might also shift or extend the field of research more towards psychological areas.

This leads to the following questions, which were our main objectives:

- Are the requirements for helpful feedback different for mobile and desktop developers?
- Are everyday users, who are not technically savvy, able to write such helpful feedback?

1.1 Structure

Our work is structured as follows. In chapter 3 "Public Feedback" we present our literature review regarding all related works and studies. These come from similar areas of interests and motivate us in our approach and ideas.

Concerning our own work, our first analysis was done in scope of public feedback. We have studied the different App Markets from Google and Apple and give a first glimpse to how users give feedback in public. The results and discussion is concluded in chapter 3.

Most importantly, we have conducted our own studies with developers and users in order to approach solutions to our questions. Chapter 4 is divided into both studies and describes the particular methodologies and goals in detail as well as our results.

Finally we summarize our results and our own contribution in chapter 5. We discuss what impact our results can have and also what could motivate further studies to extend our own.

Chapter 2

Related work

Feedback in general is a broad term. In most cases research focuses on rather specific feedback, namely reports of different kinds. Especially bug reports are a wide topic as it offers different aspects and possibilities for approaches. We have seen that along language and automation, a wider area focuses on so called post-deployment usability and also the behavior of passive and active users.

2.1 Automation

This seems reasonable as bug reports may address major issues with the software. For this matter, there are different attempts at automating the process of working with them. This ranges from extracting data ([Bettenburg et al., 2008]) to using the content in order to predict fixing time and the severity ([Zhang et al., 2013], [Guo et al., 2010], [Lamkanfi et al., 2010], [Bhattacharya and Neamtiu, 2011], [Chaturvedi and Singh, 2012]). Those particular studies analyzed how the content of bug reports can be used to determine the fixing time. They found out that using machine learning and enough data, accurate time predictions indeed can be done ([Lamkanfi et al., 2010], [Chaturvedi and Singh, 2012]) and that not only the quality of the content plays a role but also the persons involved affect the time ([Guo et al., 2010]). A person who has had success in getting their bugs fixed in the past, will more likely have new bugs fixed as well. At the same time, bugs with a higher interest also have an increased likelihood of getting a fix, which also holds true for bugs which have a higher chance of "hurting the company financially or in terms of reputation amongst customers". While this is helpful for developers as they can use this to estimate the time without skimming through loads of bug reports, this does not completely solve the problem of improving the quality of bug reports in the first place. Instead, it gives insight into what is important to have in a bug report in order to get it fixed in a reasonable time.

2.2 Language

Furthermore, Ghose and Ipeirotis [2011] studied the content of product reviews and their helpfulness in regard to sales of the product. While those are not as formal as bug reports, it can still be useful to see how product reviews are perceived. Their studies showed that there indeed is a correlation between an increase in sales and reviews with higher readability and also subjectivity. It might be interesting to see if there is a similar connection with software, especially in App Markets.

It seems that readability is an important factor but at the same time a problem, considering how to formulate or describe problems. For this matter, Ko et al. [2006] investigated this in a linguistic analysis. They concluded that tools would greatly help users when creating reports. Those could capture the observed behavior and then "they would only have to supply a description of the expected behavior", making end users "able to generate precise reports with little effort". This would also help solving the "Vocabulary Problem" [Furnas et al., 1987], which describes the problem that many users use many different words when referring to one particular entity.

2.3 Post-deployment Approaches

In their work, Nichols et al. [2003] pointed out that "once a piece of software has been released users have reduced opportunities to communicate with developers". This way "individual users just get frustrated and develop personal work-around tactics. From the point of view of the developers these incidents are invisible [...]". Evidently, frustration is a known issue, which is also subject of different studies ([Bessiere et al., 2003] and [Lazar et al., 2003]). The results in the latter also showed that "one-half to one-third of the time spent in front of the computer was lost, due to frustrating experiences", which makes this also a time issue. And one of our focuses is to see what hurdles exist, preventing (average) users to turn it into feedback and communicate it to the developers. Finally, they almost call for "empowering end-users to proactively contribute to usability activities is a valuable, and under-explored, technique". These outcomes add to our own believe, that feedback is a crucial point when improving software after its deployment.

This involves so-called remote usability evaluation, which is the focus of works by Hartson and Castillo [1998] and Castillo et al. [1998] and is defined as methods "wherein the evaluator, performing observation and analysis, is separated in space and/or time from the user". While this primarily refers to methods of conducting usability methods from afar, their studies give interesting insight to the capability of users. For their work, Hartson and Castillo [1998] conducted a study with "users with no background in software engineering or human computer interaction" but with a "barest minimum of training in critical incident identification". Their objectives were to find out if those users were able to report critical incidents and if so, how well. During their experiments they had users participating and also trained usability experts (the experimenter) in order to compare their results. These results indicated that the users identified 66 out of the 97 critical incidents, which were reported by the experimenter. Following up they hypothesized that users will rate reports in regard to severity similar to how the experimenter would rate. "Across all 24 user subjects, the experimenter′s rankings agreed with those of users for 55 out of 66 (83%) of the critical incident reported", showing another positive result which confirmed their hypothesis. Further results showed that the participants were also able to report incidents with different severities: between the users and the experimenter there was a 75% matching for high severity, 79% for medium severity and 33% for low severity. Their studies summarize that "users with a minimum of training can identify, report and rate the severity level of their own critical incidents". Similar results concerning the severity were also shown in [Castillo et al., 1998]. However, for their studies the users had 15 minutes of training in order to be able "to recognize critical incidents during their own usage". While these results that just mere training can help, their results would find most application in (custom) software used in companies or other closed groups, which can then be improved by having the employees write reports. The focus of our study are everyday users. Those will also have no background, but they will receive no training or guidance, making them a different target group.

In their work, Chilana et al. [2011] surveyed 333 usability professionals in order to investigate a similar matter, post-deployment usability. They found that "the role of usability appears to diminish in the post-deployment phase and usability professionals are rarely involved in postulated post-deployment activities". Through their studies their results show a clear indication that after software has been developed, the usability is not an important factor anymore as it was during the development (cf. Figure 2.2). This shows that testing and (usability) maintenance of software persists only to a certain time. Again, this rises cause for us to investigate if (everyday) users can participate and help.

2.4 Active participation

One of our motivations was to try and understand what keeps users back from submitting feedback. According to our literature review this appears to be a field of interest in not only HCI but also general online communities.

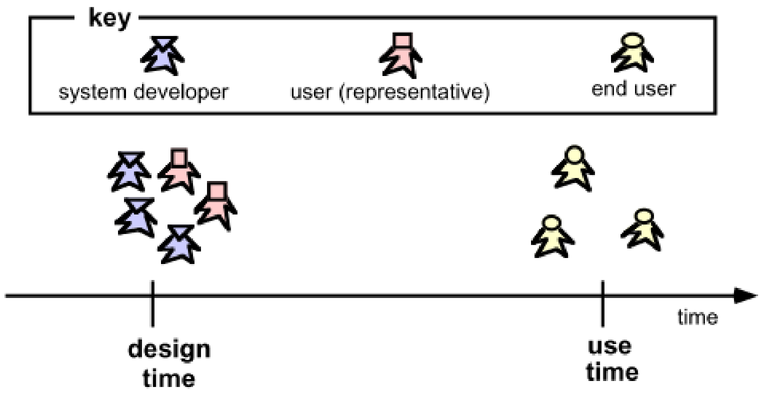


Figure 2.1: Design and use time from [Fischer, 2002]

Fischer [2002] even describes ”[t]he fundamental challenge for human-computer interaction (HCI) [as] to invent and design a culture in which humans can express themselves and engage in personally meaningful activities.”. This study discusses the different roles of users, namely the consumers and designers, and analyses the relation they have. He states that there has not been much research in HCI addressing the issue of creating ”computational environments, in which humans of all backgrounds would feel in control, get into the systems easily because of their low threshold”. Furthermore, he puts emphasis on how users do not necessarily have to be either a consumer or a designer.

> It is also a mistake to assume that being a consumer or being a designer would be a binary choice: it is rather a continuum ranging from passive consumer, to active consumer, to end-user, to user, to power users, to domain designer, to medium designer, all the way to meta-designer.

According to him one of the hindrances for users is the impression, that computer systems are unfriendly and uncooperative, and they ”spend more time fighting the computer than solving their problems”. This is also consistent with the findings of Lazar et al. [2003] regarding the time

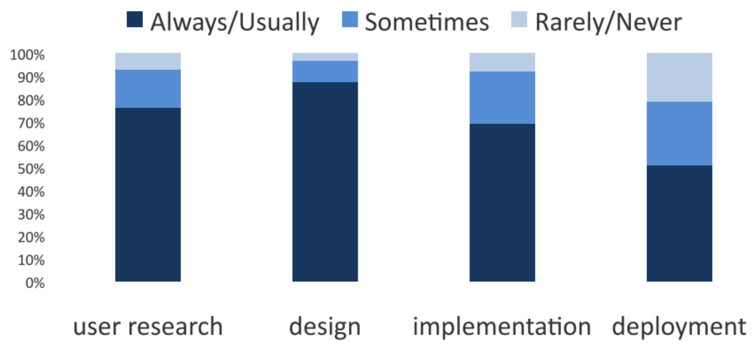


Figure 2.2: Usability involvement in different phases of development from [Chilana et al., 2011]

aspect as mentioned before. Figure 2.1 shows the process of development and the usage afterwards. During the development process, the only roles who engage with the usability of the product are usability professionals and some representative users. This is also a problem according to the works of Chilana et al. [2011] (cf. Figure 2.2, as discussed before).

As shown in Figure 2.2, only 50.9% of the asked usability professionals stated to be involved in the product in the deployment phase. This shows that the developers (meaning all the workers involved in the development process) only participate to a certain point in time. Fischer [2002] further states in his article that ”Software users and designers will not fully determine a system’s desired functionality until that system is put to use” and ”Software systems must evolve at the hands of the users. End users experience a system’s deficiencies; subsequently, they have to play an important role in driving its evolution”. This is also emphasized in [Hartson and Castillo, 1998], stating that ”[t]he need for usability improvement does not end with deployment, and neither does the value of lab-based usability evaluation” which also highlights the further need for involving the real users after a product is finished. These results back our motivation to see how users can become active and indeed contribute.

“Generally speaking, most members are Passive Users. For example, about 99% of people who use Apache are Pas-

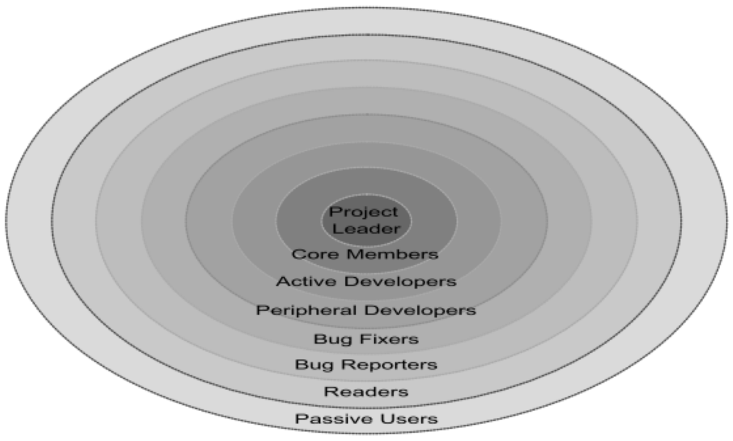


Figure 2.3: General structure of an OSS community from [Nakakoji et al., 2002]

sive Users" according to Nakakoji et al. [2002]. In their work, this was referring to open-source software communities, with passive users being defined as those who "just use the system in the same way as most of us use commercial software". A complete categorization can be seen in Figure 2.3. When we partially map this to our interests, the passive users should also be the largest group. We are not interested in the several developer roles, rather we wanted to see if the gap between passive users and bug reporters can be bridged, similar to the article by Fischer [2002].

2.5 Frustration

As mentioned above, frustration is a leading aspect when discussing user feedback and is also an active field of research. Nyer [2000] tried to find out if, aside from improving the chances of solving the issue, there is also a "more direct beneficial effect" when complaining. Their methodology used customers of a fitness center and those were asked according to their satisfaction. The results of their study concluded that unhappy customers who were asked to let out their complaints had a significant increase in satisfaction compared to those who were not. Additionally,

A study showed that the act of complaining also increases the satisfaction customers who were least satisfied initially had the greatest increase. While the domain in their study is different, it still rises motivation and incentivizes research about how to help users deal with their frustration, showing that there can also be benefit in the act of giving feedback, regardless of possible rewards.

2.6 Feedback criteria

In their work, Zimmermann et al. [2010] surveyed both developers and users to see how the information that developers require intersects with the information that users provide. Further, they also provided a tool, which is able to rate the quality of written reports and also encourages them in improving those. Their findings showed that there was a mismatch between both parties, more specifically "between what developers consider most helpful and what users provide". They achieved those results by asking the developers, inter alia, to provide information about criteria for good and bad feedback and also what information they need in order to be able to fix the issue. At the same time, users were asked to state which information they usually provide when reporting an issue. Despite their interesting findings, their focus lied mainly in the mismatch of information for a special case of these parties. The users (in their work the *reporters*), were experienced (bug) reporters, who had already submitted at least 25 bug reports and were not assigned to fixing any bugs. Meaning that those are already experienced and would have a certain knowledge in how to write reports. Apart from this, the developers came from "APACHE, ECLIPSE, and MOZILLA projects". While they do not mention it explicitly, if they considered both desktop and mobile applications, from these projects, MOZILLA would be the only candidate to provide mobile applications and therefore we assume they made no differentiation.

Furthermore, their tool CUEZILLA provided interesting insight to how users can be informed about the quality of their report and also encouraged to improve it by providing missing information. In addition, they provided general recommendations on "how to engage users and build better tool support". We used the information criteria and some of these recommendations they have used in their work for our studies to see how both desktop and mobile developers think is important (cf. 4.1 "Developer Survey").

In their conclusion they list possible motivations in regard to how their study can be extended. One of them being "Do reporters with more experience and higher reputation submit better bug reports?". Our approach and focus is indeed also an extension of their study (and to a certain degree also a generalization) but in the opposite direction as we wanted to see if reporters with *no* experience can submit bug reports.

Chapter 3

Public Feedback

In this section we discuss the general feedback, which can be accessed and read by every user, in order to get a wide grasp of how users openly discuss problems or compliment software. Besides, this allows a first comparison between desktop and mobile systems. For the mobile area we used different kinds of sources such as the Play Store on the Android platform, the App Store for iOS devices and also Mac systems, whereas both were chosen primarily for feedback for mobile device applications. Choosing apps only on one platform might lead to results being biased by the platform itself, therefore we chose to review feedback on both of them. Furthermore, we wanted to avoid other possible app or developer specific factors and hence chose one app, which is available on both platforms.

An overview of general feedback of users and different platforms

In order to get representative and qualitative data for users' response to mobile apps, it was necessary to find out which apps or kind of apps would be eligible. By eligible, in our case we used the metric "Most Downloaded". The reason we did not choose "Most Ratings" or "Most reviews" was, that in this study we would only discuss a small set of ratings and reviews (meaning the amount would not matter that much) and "Most Downloaded" should be enough of an indicator for a good insight to what people think.

Using a smartphone, we assumed that web browsing and native web apps would take the most usage percentage on smartphones. Considering the results of Flurry Insights[1], in the year 2013 browsers were used just 20% of the time, unlike (native Web-) Apps with 80%. Hence it should be more representative to use a native app for our studies. To avoid a bias of the platform (and also because they are structured differently), we wanted to discuss the findings of the same app for both Android and iOS. As there are no official explicit download numbers for iOS apps, we searched for websites or official statements. Further, we only looked at free apps, since those attract more users and therefore more ratings. According to Yahoo[2], Gadgets360[3], and MacRumors[4] the most downloaded (free) apps on iOS in the year 2014 was Facebook Messenger, followed by Snapchat and YouTube on second and third place. For Android, the site Androidrank[5] offers qualitative statistics about the Play Store. However, choosing "Installs" as a sorting criteria, the list itself only shows install numbers of very rough estimates such as "1000 M". Going to the app specific site itself, it showed estimated install numbers. Using these, the top apps (using manual comparison) seemed to be Facebook, WhatsApp Messenger, and (Facebook) Messenger. Although the Android results are from the current date and the iOS list from 2014, we came to the compromise to use the common app Messenger for our studies.

People spend most of the time with native web apps, therefore we also chose one

As for desktop application we chose to use publicly available feedback for the Mozilla Firefox application [Mozilla, 2015 (accessed December 12, 2015], although we will also use the data about Firefox for mobile phones from the same source. We noticed, that there seems to be discrepancy between open source and closed source projects, as we could only find open sourced software, which had public feedback. Without explicit analysis we assume that closed source developers and companies tend to not have their feedback data be public due to privacy and mainly security reasons. The reasons for choosing Mozilla Firefox as our candidate for the desktop environment was that according to NetMarketShare the top three browsers used in the last

[1]http://flurrymobile.tumblr.com/
[2]https://www.yahoo.com/tech/
[3]https://gadgets.ndtv.com/apps/news/
[4]https://www.macrumors.com/
[5]http://www.androidrank.org/

year (December 2014 - November 2015) were Microsoft Internet Explorer, Google Chrome, and Mozilla Firefox and according to Statcounter[6] for the same time period, those were Google Chrome, followed by Apple Safari and Internet Explorer, with Mozilla Firefox on the fourth place. Unfortunately, the other three browsers could not be considered for our studies because they do not offer a publicly accessible database for their feedback, making our choice the only one we could use for analysis. Another advantage of using Mozilla Firefox is that as mentioned above, it also lists the feedback received for mobile devices.

Firefox has the advantage of Firefox Input, a database of feedback for all versions

3.1 App Markets

Both of the app markets offer the users the possibility to write a review for an app and also rate it. The significant part of this kind of feedback is, that it is public, available for everyone to read. And, unlike the other mainly used method of feedback, that is sending it from within an application, in this case the users have to go explicitly to the respective App Market of their operating system to do this. Many apps have notifications or pop-ups while they are running, asking the user to rate the app and have him automatically switch to the App Market to do so, but it still is a method involving more steps, also requiring a certain willingness from the user. As part of our user studies we also tried to investigate how the ratio is of giving feedback on App Markets or sending it directly to the developer and what the reasons are for this (see section 4.2.1 "Pre-session Questionnaire" (p. 48)).

3.1.1 Play Store

Google's app market is accessible both via Android smartphones and using the website (Play Store[7]). After opening the page of a certain app, there are 3 or 4 reviews shown

[6]http://gs.statcounter.com
[7]https://play.google.com/store

(3 on mobile and 4 on the website), with the default sorting criteria "Usefulness" and - in case one is logged in with their Google account - they have an option to only show ratings from users with their current device or other devices they connected with their account. The latter option only appears on the website while on mobile it is only possible to either view the ratings for the newest version or not, and in terms of devices it is only possible to choose the current device. It has also to be noted that users can give a star rating without writing a review on the Play Store.

The Play Store offers a rather restricted overview

The sorting works only in one way as it is not possible to choose either descending or ascending. Likewise, it is impossible to choose the most helpful bad rating or most helpful good rating, a feature offered by online stores such as Amazon, which could have given more insights.

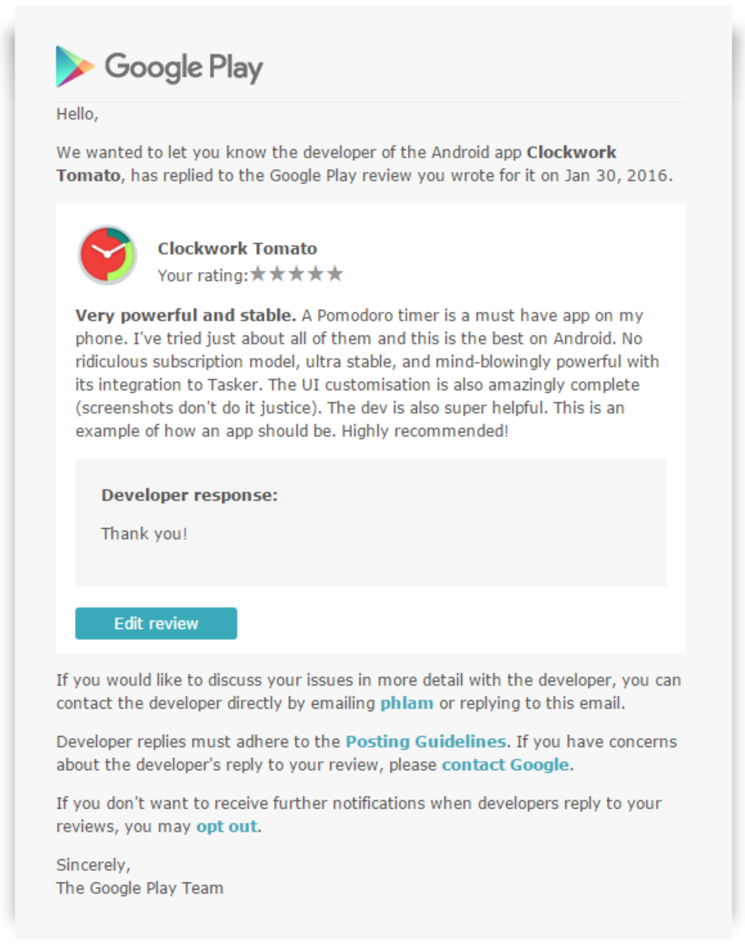
The fact that the initial 3/4 shown ratings have different star ratings, ranging from 1/5 to 4/5 might suggest, that their display is in fact related only to their usefulness. One can choose a rating and either report it, or click "Useful" or "Not useful" although the current "value" of usefulness of a rating cannot be seen.

Taking a closer look at the spectrum of the different ratings, we might assume that there is no spam or random characters appearing in the ratings. But this is probably due to the fact, that those just might not receive enough votes, and therefore do not appear on the first places. Considering the actual value of ratings, the information or feedback provided, we observed that users wrote both detailed and also short descriptions.

There was no clear tendency as both rather explicit and also short reports were written

Sorting by *Most helpful first* gave a list of reviews, but without further indication of how many users considered these to be helpful. Considering only the comments, which address problems users have with the application, most of them referred to the app not starting or crashing. Some explained that the app crashes when something explicitly happens while most just mentioned the problem, without any steps for the developers to reproduce. An interesting finding was that some users, who gave a high rating, were referring to problems mentioned by those, who gave a low

Some reviews were referring to others, creating a kind of dialogue



Google Play

Hello,

We wanted to let you know the developer of the Android app **Clockwork Tomato**, has replied to the Google Play review you wrote for it on Jan 30, 2016.

Clockwork Tomato
Your rating: ★★★★★

Very powerful and stable. A Pomodoro timer is a must have app on my phone. I've tried just about all of them and this is the best on Android. No ridiculous subscription model, ultra stable, and mind-blowingly powerful with its integration to Tasker. The UI customisation is also amazingly complete (screenshots don't do it justice). The dev is also super helpful. This is an example of how an app should be. Highly recommended!

Developer response:

Thank you!

If you would like to discuss your issues in more detail with the developer, you can contact the developer directly by emailing **phlam** or replying to this email.

Developer replies must adhere to the **Posting Guidelines**. If you have concerns about the developer's reply to your review, please **contact Google**.

If you don't want to receive further notifications when developers reply to your reviews, you may **opt out**.

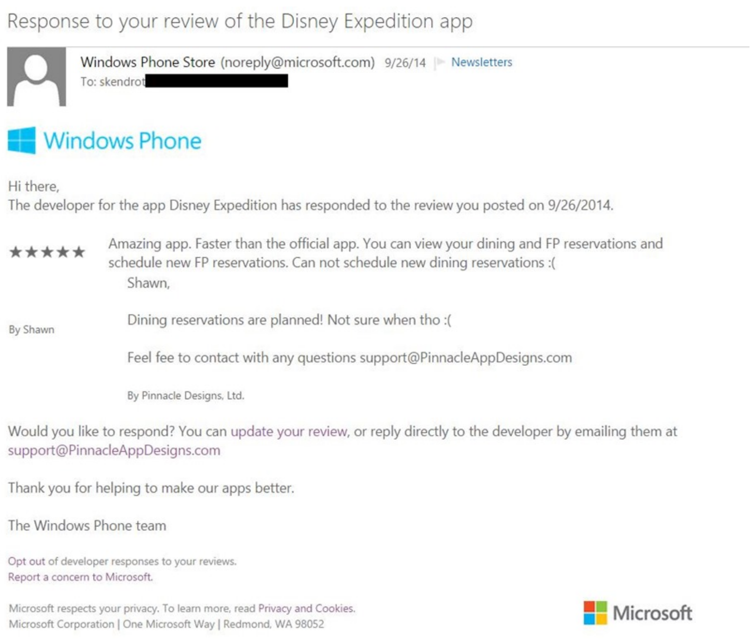
Sincerely,
The Google Play Team

Figure 3.1: Notification that an Android developer replied to a review

rating, stating they did not have those on their device.

In the Play Store and Windows Phone store, developers are also able to reply to reviews. This means, that the Android app market is not only a channel for active participation of customers but also a bidirectional way between customers and developers. Further, a user will get an email notification in case a developer replied, which also offers an option to edit the review (see Figure 3.1).

Google's Play Store and Microsoft's Windows Phone store allow developers to respond to users' reviews, who get notified


Response to your review of the Disney Expedition app

Windows Phone Store (noreply@microsoft.com) 9/26/14 Newsletters
To: skendrot[redacted]

Hi there,
The developer for the app Disney Expedition has responded to the review you posted on 9/26/2014.

★★★★★ Amazing app. Faster than the official app. You can view your dining and FP reservations and schedule new FP reservations. Can not schedule new dining reservations :(

By Shawn

Shawn,

Dining reservations are planned! Not sure when tho :(

Feel fee to contact with any questions support@PinnacleAppDesigns.com

By Pinnacle Designs, Ltd.

Would you like to respond? You can update your review, or reply directly to the developer by emailing them at support@PinnacleAppDesigns.com

Thank you for helping to make our apps better.

The Windows Phone team

Opt out of developer responses to your reviews.
Report a concern to Microsoft.

Microsoft respects your privacy. To learn more, read Privacy and Cookies.
Microsoft Corporation | One Microsoft Way | Redmond, WA 98052

Figure 3.2: Notification that a Windows Phone developer replied to a review
Image taken from http://www.visuallylocated.com/

3.1.2 App Store

Similar to the Play Store the App Store is accessible via iTunes and the web pages of apps. Accessing a page for a certain app opens up an overview with screenshots for different platforms (iPhone/iPad/Apple Watch) and the rating on the left side. This rating is the average rating of the current version, unlike the average rating for all like in the Play Store. However, it is possible to switch to the reviews and ratings for all versions.

Initially more sorting options compared to the Play Store

At the time we investigated version 56.0 of the newest version of the Messenger (posted on 3 February 2016). Concerning the reviews, there are also sorting options: *Most Helpful*, *Most favourable*, *Most critical*, and *Most Recent*. While the Play Store did not offer the possibility to see e.g. filter negative reviews, in this case this can be done by sorting by *Most critical*. This showed 32 results starting with 1 star reviews up to 3 stars and then the (less critical) 5 star reviews one sees with the default sorting option.

Regarding the content we were able to see a similar picture as in the Play Store comments. There are comments which express complaints about certain lack of functionality, which can be short and not explicitly describing the issue, but at the same time also rather elaborated variants. This especially holds true for the critical comments.

Complaints consist of either short or long descriptions

Sorting by *Most Helpful* we assumed to see the comments which received a certain amount of votes regarding the question "Was this review helpful?", which is displayed below each comment. But out of those, which were shown, only a few displayed that there were customers who "found this review helpful". Most of them had 4 or 5 star ratings and those at the top were also very detailed and explicit. Still, this makes it unclear and not transparent for the customer to understand, why those reviews are helpful. Some referred to issues mentioned and wrote about their own experience, stating that they did not have this issue. This is a similar behavior to discussions on BugZilla or other bug tracking systems, which allow comments on reported bugs.

The *Most Helpful* section is unclear why it contains such reviews

Similar to the Play Store comments, many of the *Most helpful* reviews referred to the critical comments, creating a sort of dialogue. This is a distinguishing aspect of public feedback, as the ones directly sent to developers cannot be accessed and read by others. As of right now, this discussion remains between customers, as developers currently have no option to respond to reviews unlike Android and Windows Phone developers.

3.2 Firefox Input

As discussed before, we chose Firefox mainly because Mozilla offers an easily accessible way to their feedback data on their site for Firefox Input. Different filters are applicable such as the time range, the product (Firefox, Firefox for Android, Firefox OS, Firefox for iOS), the operating system which was used (Android, Windows 7, Windows

10,...), or the language of the system from which it was sent. While those are properties one might expect for filters, there is another criterion which is about the sentiment. When users want to send feedback, independent of the device or system, the first thing that pops up is the question "How does Firefox make you feel?", making them choose between "Firefox makes me happy" and "Firefox makes me sad" with an additional option to change the device for which the feedback should apply to. This choice is obligatory in order to get to the next step, in which they would type in a text box what they like or do not like in particular. Other fields are only asking for an optional website, a check if they want to provide technical information about the browser and also an email address for possible follow up feedback, leaving the text box the only required field to fill out.

Every report comes with a sentiment, indicating if the user is happy or not

For our study we always applied the time frame of one year from 10 December 2014 to 10 December 2015 in order to have a static source of data and enough data to see representative percentages, resulting in roughly 206517 messages (despite this time range being constant, the amount of messages still decreases over time, for which we could not find an explanation). We would also like to note that these findings were done manually, although this process may have easily been done automated for a rather precise and deeper analysis.

The period considered was December 2014 to December 2015

An immense classification was done with the sentiment criterion, showing 90% ($\sim$194458) of all messages being unhappy and only 10% ($\sim$21980) happy. Without further research we assume this may be the case because those who are unhappy have the incentive that the developers might read it and improve the product which would satisfy those users again while happy users might think that they do not "gain" anything by telling the developer that they are happy with their product.

For the different product versions of Firefox, the Android version is the most widespread one with 47% ($\sim$102436), followed by the Windows desktop version with 43% ($\sim$93563) and Firefox OS and Firefox for iOS with 8% ($\sim$18148) respectively 1% ($\sim$2284). There is also an "Unknown" entry with 0% (∼7). The low spread of Firefox for iOS is also matched by the results of Statcounter[8] and NetMarketShare showing a usage of 1.91% (as Firefox is not listed by its name we assume it belongs to "Other") respectively 0.69% for the same time period.

Firefox on mobile and Windows PCs are the most widespread versions

To analyze the content of individual messages we set the language filter to "English" for the following studies, which accounts for about 39% (∼434).

As mentioned before we manually investigated some of the messages submitted to see what kind of information users are willing to provide. Filtering only those categorized as "Happy", in terms of length the messages showing appreciation are kept rather short with few words (e.g., "makes me happy", "Thanks", "i like firefox for its speed") while those which additionally point out negative aspects tend to get rather detailed (as quote 1 below demonstrates).

Appreciation often shown with a few words unlike negative aspects

[8]http://gs.statcounter.com

> Quote 1: "Firefox has become fast and more stable over the years. It was and remains my default browser. Great work!
>
> However, here is a feature request that could improve the overall user experience (IMHO):
>
> Please integrate the "Page/Security Info Button" ->"More Information" dialog into the main browser window; either as a side bar for each page or as a new tab like the preferences and addons tabs.
>
> These currently floating windows are really annoying; just like popups were in the early days of web browsers ;)"

Some highlight particular features

Whereas some users mention what features they like and sometimes also give scenarios in which those appear to be useful (e.g., the volume icon indicating that sound is played in this tab), many tend to just mention they are happy without referring to anything in particular or just compare it to other browsers and emphasize the advantages over those (e.g., "Add-on(s) and the freedom to change my browser as I wish. I think it's better than Google Chrome!", "works great. had a lot of problems with explorer ").

Longer and detailed texts were also found when switching the filter to "Unhappy", although we also found many messages describing problems with only a few words. In most cases those did also not mention where they encountered these problems such as "bad performance", having crashes regularly without any comment about the website or section in which the user was, or Firefox being slow and using too much memory, albeit the two latter could be meant as general problems. Other complaints issued problems with pages being not correctly displayed or loaded very slowly. We noticed though, that users complaining about these issues were sometimes providing information about how they were not having these problems using other browsers. Others were describing detailed feature requests.

> It would be nice to have the option to scroll up to reload pages over having to press tabs button

> every time. Also maybe add the option to swap between tabs by swiping left/right. Otherwise 9/10 Firefox nailed it again.

Before, we have discussed the general feedback for the most general classifications. Looking at the platform specific messages we were able to notice some interesting findings. For Firefox for Android and the "Happy" filter, the majority seems to be spam or random strings of characters. This also ranges from messages containing just email addresses to random numbers. As a result, on each page which displays 20 entries, on average ∼74% were spam (tested with 10 randomly selected pages out of 31). It is hard to find and tell the reason for this behavior as, unlike to App Markets there should be no benefit in having many positive messages. Although interestingly enough, the rate of spam is vastly reduced when changing to "Unhappy" messages, accounting for only ∼2.5% (tested with 10 randomly selected pages out of 1422). It should be noted that there are also messages, which we did not categorize as spam, but are still considered useless due to vulgar use of language or ranting.

Very high rate of spam on the "Happy" filtered messages, unlike the "Unhappy" filtered ones

The actually useful content containing messages could be considered qualitatively ranging from few words indicating the main source of frustration (similar to before mentioned problems such as crashes, slow loading, etc.) to longer paragraphs reporting in detail.

Chapter 4

User Studies

Since feedback involves two parties, one who submits the feedback and the other party receiving it, we conducted studies with two different groups representing each party. In order to gain as much insight as possible, this was done according to the following approach. Considering the aforementioned relationship between both parties, the surveys to be conducted also should be similarly related. Using both questionnaires and an experiment, the studies were divided into a 2-phases-approach:

Feedback occurs between two disjunctive parties

1. Create a questionnaire for developers
2. Use the results to derive a survey for users

The most part of the first survey is based primarily on the survey done by Zimmermann et al. [2010], which was directed towards both developers and users. While their questionnaires were independent of each other (since they put emphasis on the matching of the results), our approach developed consecutively. As such, we started with a questionnaire for developers to find out about their perspective on feedback and factors which they consider important. Deriving these formed the basis for the survey conducted with the users, which is split into multiple parts (see 4.2 "User Study"). As the second phase depended on the first one, the latter needed to gather enough information to give

We divided the studies according to a 2 phases approach

The information in bug reports considered important by the developers formed the basis for the user study qualitative results for the second one and therefore induce overall qualitative information throughout the whole studies.

4.1 Developer Survey

Developers were asked to fill out a questionnaire

As mentioned above the information gained by this survey is crucial, therefore it was important to make sure the acquired data set would not only be as much as possible but also has a certain relevance which would help with the follow up studies. This survey consisted of a questionnaire, which gathered data from developers online and served as a source for qualitative and quantitative data. It was designed with both close-ended and open-ended questions, which we explain in detail in the following section 4.1.2 "Questionnaire".

4.1.1 Interviews and Experiences

4.1.2 Questionnaire

The questionnaire for developers was designed with 23 questions, having both open-ended and close-ended types, which as aforementioned would allow us to gain a mixture of quantifiable data and data which has an open space for answers, giving us possibly new kinds of information. We attached the full questionnaire in Appendix A "Questionnaire for Developers".

First half of the developer questionnaire asks about background and skills as well as feedback workflows

Age is our only relevant demographic information, which we can use to determine if certain groups of ages correlate to different expectations in terms of feedback. Additionally, this question combined with the other first seven questions serve the purpose of classifying the different developers into groups such as their main development platform and information about their working environment (in case they do not just develop as a hobby but as a job). Question #2 (*For which platform do you develop mainly?*) will be used as an important filter in order to gain a view on how the results of the the rest of the survey differ depending on each platform. There are many different metrics that can be used to assess a developers' skills and experience in his expertise, such as *years of experience, how skilled would you describe yourself (with options like beginner or expert), how much time do you spend on development* with each having advantages and disadvantages (e.g. years of experience and skill level do not necessarily have a causal relationship). For our studies we chose *How many years have you been developing software?* as question #3 together with question #4 (*Are you developing as a hobby or professionally?*) to #7 (*What is your job role in this project?*) so developers could assess their own time in which they actually developed software. The intention behind question #8 (*Is your software free?*) is that users might have different expectations of and demands on the app depending on if they downloaded it for free or paid for it, which then can impact and influence their feedback behavior.

We used multiple questions to assess the software development skills metric

Following up questions #9, #10, and #11 allow developers to describe the feedback work-flow in their software. Information about the best (in terms of usability and time and effort) way for a user to submit feedback would be quite useful, unfortunately this is not easy to gain just from asking a developer. One possibility would be if a user submitted feedback about the feedback system itself, but for our study we make the fair assumption that this does not happen on a regular basis and therefore these three questions should be enough for the time being. In order to get a rough overview of the kind, type, and actual helpfulness, questions #12 to #14 give information on if users tend to criticize or give complements, in what form they do that, and how helpful that is.

What ways can the users use to give feedback and how is it used?

The next part of the questions deals with specific and technical details. Question #15 and #16 deal with the information given. While the former is about information which the developers think is the most important when responding to feedback and in case of bug reports, which is the most useful in regard to fixing those, the latter is about characterization of bad feedback. The possible answers we used were almost the same options which were used in the question-

Which information is important and what are properties of bad feedback naire done by Zimmermann et al. [2010] with an additional option to add criteria which has not been listed. The reason being that - based on our own experience and knowledge - in their study the given options covered all possible criteria and technical aspects one would provide when using software in general. We did not use the criteria *components*, *build information*, and *code examples*, considering their highly technical nature, which an average user most likely would not know about. This is also further supported by our discussion that at least the first two properties are not required to be filled in by the user (see 4.1.3.2 "Feedback information"). Their results also indicated that these information are not among the most important, which is also consistent with the criteria used for the study done in [Hartson and Castillo, 1998]. These two questions are very essential since they are the main contributors for the user study, insofar as the outcome of these will be used to assess the ability of the users to submit these types of information.

Some rather technical properties from the works of Zimmermann et al. were not used for our study

Afterwards, we wanted to know if developers also know or at least had any assumptions regarding the reasons for this, hence question #17 (*What do you think are the reasons for that?*) covers some possible options which we thought of, but again including a free text option for all other cases.

The rest of the questionnaire deals with ideas from a developers perspective, how feedback system in general could be improved.

Bad feedback: Why is it bad and how can it be improved?

#19 serves the purpose for us to see if the kind of feedback which is not considered spam but also does not give a lot of information (as seen in 3 "Public Feedback") is still valuable to a developer, as it might at least give an indication (however information-rich it is) that their software is good or bad.

Using previous specific results in order to generalize for mobile and desktop developers

In their work, Zimmermann et al. [2010] came up with recommendations on how to improve those systems. Question #22 (*What do you think of the following methods?*) serves two purposes, first to see if developers in our sample agree on these and second, since in their study only developers for desktop/server applications were considered, we wanted to see if mobile platform developers also consider these recommendations to be improvements. This makes the two preceding questions (*If you could change anything in the whole system[...]* and *What do you think about rewarding good bug reports with methods like[...]*) seem similar to #20 but such order and making the former two open-ended and the latter close-ended, we can use the advantages of both types. Those being giving enough freedom and not to bias with predetermined options, but it is also, also in case of ignorance of ideas, a way of evaluating existing ideas.

We further asked the developers about the recommendations which originated from Zimmermann et al.

We would also like to note that despite asking about possible rewarding systems, we will not discuss this possibility in detail in our study. Rather, this was inspired by systems such as the Amazon Vine program Amazon Vine[1] and Stack Exchange, and the possibility of beta access for apps on Android (see [Google Alpha/Beta Program] and Figure 4.1) and iOS (TestFlight[2]) as well as Windows Phone (Microsoft Beta Testing[3]. In addition, recent studies suggest that gamification indeed can help encouraging people in ways such as learning new things or providing feedback [Gamasutra], [Lotufo et al., 2012].

Gamification and beta access as rewarding systems

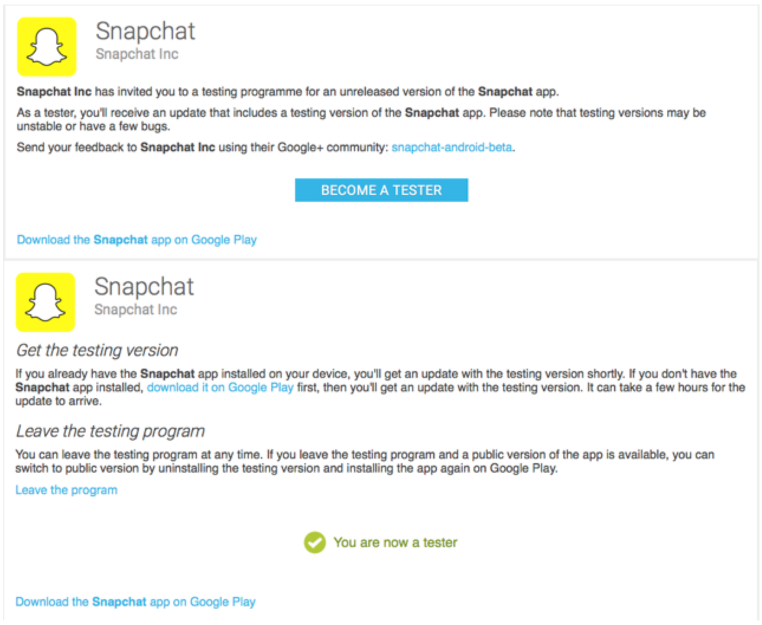
4.1.3 Results

4.1.3.1 Basic information

As a survey tool we chose Google Forms and posted it on Reddit, more specifically subreddits which topics are about programming or development. We asked users to fill out the questionnaire only if they have developed some kind of software for either desktop or mobile platforms and have also received at least some feedback so far. Furthermore, in case they were working on multiple applications we asked them to choose just one, which would be the best fitting for such a study. We also designed the form so that multiple submissions were possible, allowing them to send in data for different platforms.

Participants should have already developed an application for desktop or mobile system and received at least some feedback

[1]https://www.amazon.com/gp/vine/help
[2]https://developer.apple.com/testflight/
[3]http://bit.ly/1QqIVG9



Snapchat
Snapchat Inc

Snapchat Inc has invited you to a testing programme for an unreleased version of the **Snapchat** app.

As a tester, you'll receive an update that includes a testing version of the **Snapchat** app. Please note that testing versions may be unstable or have a few bugs.

Send your feedback to **Snapchat Inc** using their Google+ community: snapchat-android-beta.

BECOME A TESTER

Download the **Snapchat** app on Google Play

Snapchat
Snapchat Inc

Get the testing version

If you already have the **Snapchat** app installed on your device, you'll get an update with the testing version shortly. If you don't have the **Snapchat** app installed, download it on Google Play first, then you'll get an update with the testing version. It can take a few hours for the update to arrive.

Leave the testing program

You can leave the testing program at any time. If you leave the testing program and a public version of the app is available, you can switch to public version by uninstalling the testing version and installing the app again on Google Play.

Leave the program

✓ You are now a tester

Download the **Snapchat** app on Google Play

Figure 4.1: Beta access example using the Snapchat app

Our total data set consisted of 54 samples

In total we received 54 submissions with ages ranging from 14 to 47 and a median of 24.5. The platform with the highest occurrence was Android with 28 developers followed by 13 iOS developers. Furthermore, there were six developers for Linux, three for Windows and two for Mac. There were two developers who inputted their platform on their own, which are one who developed Chrome apps and one developer who worked on both iOS and Android. We also listed Windows Phone as a possible platform but none of the participants chose it. For our further discussion we dichotomize between *Desktop* and *Mobile*, whereas *Desktop* consists of Linux, Mac, and Windows ($n = 12$) and Android together with iOS are regarded as *Mobile* ($n = 42$). Hereafter, we will also refer to these terms.

Dichotomization between *Desktop* and *Mobile*

Considering the background and our metric in regard to development knowledge, most (22.22%) of the participants had 5 years of experience, followed by 6 and 10 years (both 9.26%). 20 of them develop as a hobby while 34 develop professionally. Further, we asked to fill in the job role the participants have taken in their software project. Most were assigned as developers and (software) engineers. Some developers (25 stated to have less than ten employees working in their company in case they work professionally) seemed to be the only worker involved in their project, resulting in doing the jobs of all. This was also the most occurring company size; only eight stated to work in a company with 10-49 or 50-49 employees (six respectively three) and five worked in bigger companies of between 250-499 and over 500 employees (one respectively four).

Most of our subects were sole developers or software engineers

4.1.3.2 Feedback information

In the following we will often use our dichotomization for comparing the different results. First, we asked about the ways of giving feedback in their particular software and how it is received and used.

In total we saw that sending mails was the method which was implemented in most applications (75.93%) applications for both desktop and mobile (75% and 76.19%). Unlike desktop software, mobile apps have their respective App Markets as a sole source for the software and their reviews. The results showed that 59.52% were using reviews in the Play Store and App Store for feedback, suggesting that public feedback is another important source for developers to see how users feel. On the other hand a Windows developer and a Linux developer stated they would also receive reviews in stores (making 16.67% for the desktop section). Without further information we presume those stores to be the Windows Store[4] and a software center used in Linux distributions. In a matter of technical submission of feedback we saw that different tools for mobile software were mentioned such as Bugsnag[5] and Crashlytics[6] (7.14%) as well as the general term *(bug) report* which amounted to 8.33% on mobile and 11.9% for desktop. An interesting outcome was that social media such as Facebook and Twitter were named five times but although those per se are not something restricted to mobile, no desktop developers listed those. This also holds for Github (issues) with two

Many developers rely on App Markets to see how users feel

No Desktop developers used social media or Github

[4]https://www.microsoft.com/de-de/store/apps
[5]https://bugsnag.com/
[6]https://try.crashlytics.com/

listings for mobile and none at all for desktop.

For desktop systems, aside from store reviews, mail was the most common and also most used method

The first follow-up question asked about the usage of these ways. We saw that mail was the most common method and these results showed that for mobile applications this was with 40.48% also the most used method and on desktop systems with 16.67% the second choice. The first choice or rather all the other choices such as writing reports and writing reviews were used equally with 8.33%. Unlike for mobile applications, for which writing reviews was the second most popular method (21.43%) followed by using support (4.76%) and writing reports (2.38%). This requires further studies as the term *support* was used in general, without further indication of what technical way (mail, forum, etc.) was meant. We mentioned that Crashlytics together with Bugsnag were tools used as well, but those differ in the way that they are automatically used without the users explicit intention. In this regard, it becomes evident that those who listed to use these tools in their applications also listed that those are used as well.

Further findings provided evidence that mail was also the most useful one for developers

Having these outcomes we also wanted to see which were the most helpful for developers to see if there is a possible balance between what people use and their particular helpfulness. The results for Desktop still persist in regard to having the same method with the highest percentage, that is being the mail with 33.33%. This is a positive result because this means the most used method is also the most helpful one. In addition, the same also holds for the mobile platform as sending mails seems to be the most used way and also the most helpful (42.86%). For the second most helpful way, writing reviews and using the App Markets also have a matching with 14.29% of the developers perceiving it as essential and as mentioned before it was also the second most used method.

As for a tendency of positive and negative feedback in general, our findings suggest that both desktop and mobile developers received more positive than negative feedback (75% respectively 66.67%). This is no indicator for how helpful the feedback itself is, but it reveals an interesting observation of the willingness of people to send feedback in general, more precisely people seem not only to criticize

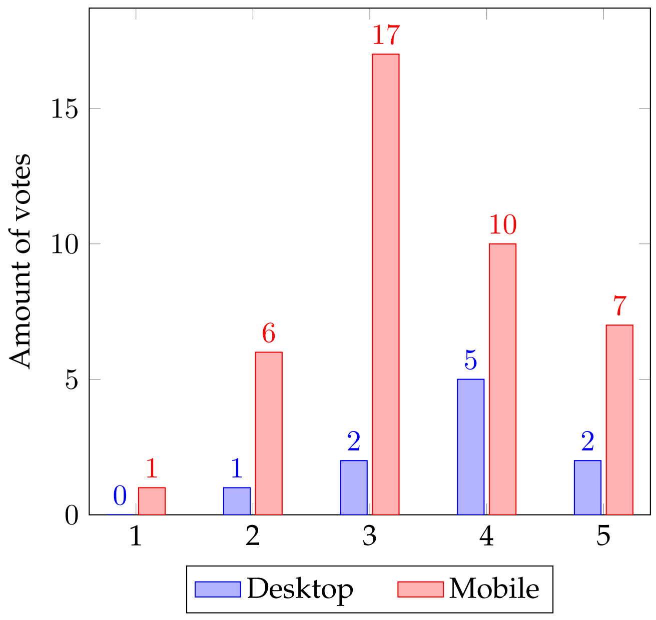


Figure 4.2: General helpfulness of feedback

but also to compliment the development.

In regard to the content itself we saw another positive tendency for both platforms. The graph above shows the results of the Likert scale from 1-5 concerning the helpfulness of feedback (with 5 indicating *most helpful*). It shows that developers rather chose higher values to indicate how helpful the overall feedback is, meaning that there is space for improvement.

Another interesting finding showed that every desktop developer received feature requests/removal issues and only half receive manually written bug reports (while automatically generated bug reports took 41.67%). For mobile there were slightly less occurrences of issues requesting additional features or removal (85.71%) while the other types were about equally common (61.90% - 66.67%). These results show a very particular tendency for feature issues for desktop software and a strong tendency for the same type on mobile.

Every desktop developer received feedback concerning features, on mobile almost every developer

We explained earlier that the crucial part of the developer study are the questions regarding the explicit information

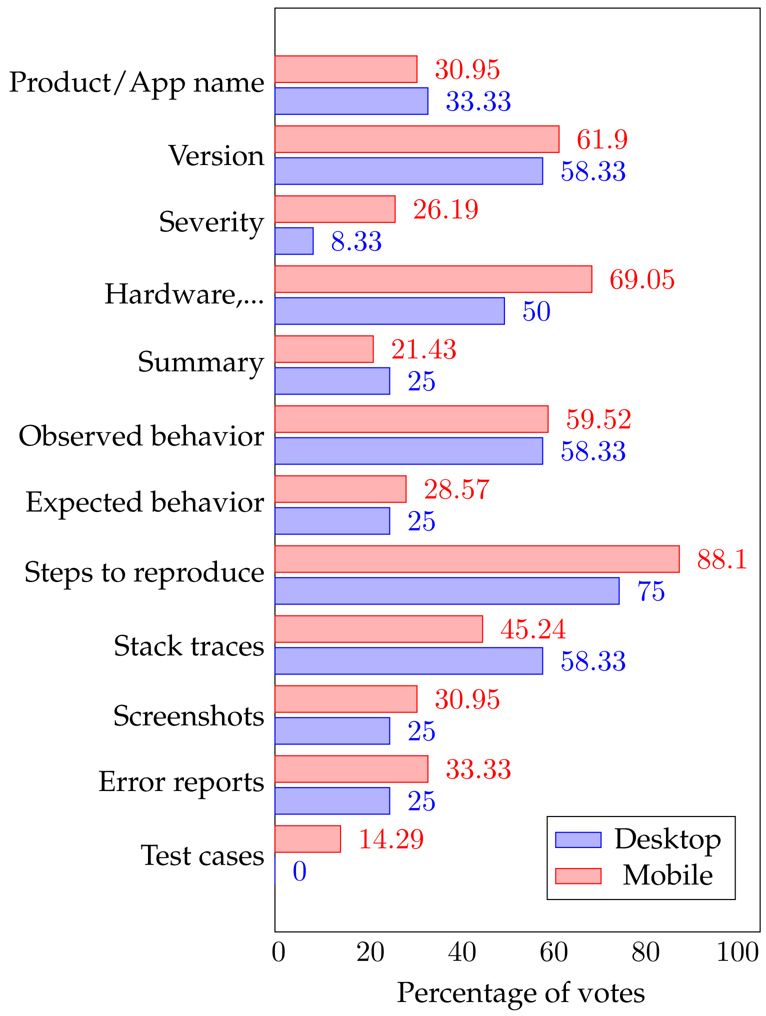


Figure 4.3: Comparison of important information

The important criteria were almost the same for both platforms

submitted to the developers and how important it is for them. Figure 4.3 illustrates the results from both Desktop and Mobile developers. From this data we can see that surprisingly almost no differences were found. This applies to basic information such as *Product/App name* (mobile 30.95% and desktop 33.33%), *Version* (61.9% to 58.33%) but also to behavior data such as *Observed behavior* (59.52% to 58.33%) and *Expected behavior* (28.57% to 25%) and the related *Steps to reproduce* (88.1% to 75%) as well as *Screenshots* (30.95% to 25%) and *Error reports* (33.33% to 25%). The importance of steps to reproduce is also consistent with the results of Guo et al. [2010].

Significant differences between both platforms could be seen in the *Severity* with 26.19% from mobile developers and just 8.33% for desktop developers. Without further details and information it is hard to tell the exact reason for this difference. An expected difference could be seen in the *Hardware, OS and version, App version* property with a difference of 19.05%. This result may be explained by the fact that mobile developers require more information about hardware and software due to the fact that there is a wide range of different smartphones available with another wide range of different operating systems. Test cases had another severe difference with 14.29% of the mobile developers listing them as important while no desktop developers saw them as necessary information when dealing with issues. It should be noted that test cases are one of the more technical information users can provide as they would need to create test cases or scenarios and be quite specific or explicit. Nevertheless, this might only explain the low percentage for both platforms but not the difference. Nevertheless, developers can gain similar data from other information such as those about observed/expected behavior and steps to reproduce, which might be sufficient.

Discrepancies could be seen in hardware/software information as well as test cases

Looking at the negative aspects we found out that the most occurring characteristics for bad feedback happened to be formulation problems (bad grammar, unstructured text, non-technical language, or just too long text) with 75% on desktop systems and 83.33% on mobile systems. This alone has no significance though, since developers should still be able to understand the core message even when it is badly formulated. Which is also true for spam, which amounts to 58.33% on desktop and 45.24% on mobile, it evidently is a problem, but is also most likely happening automatically in most cases and not meant as feedback in the first place (also see 4.2.4 "Results"). Rather, characteristics such as wrong or missing information should be focused on. First of all, wrong/missing basic information (about the *Product/App name*, and *Hardware, OS and version, and app version*) took about 66.67% on desktop and 57.14% on mobile. Again, despite these high percentages, those are not necessarily bad results since developers should be able to easily and au-

The biggest issue with bad feedback were formulation problems tomatically detect this kind of information without requiring the user to find out this kind of information (which can be hard task depending on the technical knowledge of the user) and submit it. Nichols et al. [2003] also mentioned in their work that the input of information, which is known to the system, "raises the cost of reporting and contributes to inaccuracy in the reports".

Formulation problems were checked as often as problems with describing behavior by mobile developers

Other findings concluded that mobile developers have checked wrong/or missing description of behavior as often as formulation problems. Reporting about the behavior is, without further assistance by tools (or other technology), heavily relying on the user and especially on mobile devices, not easy in terms of usability (that is due to today's input methods in those devices). However, users with concerns regarding privacy may not want to use recording software, thus this problem will require a novel approach. Our findings in the previous section (**??** "**??**") partly confirm this result regarding missing descriptions of behavior.

Shortage of motivation as a leading indicator for giving bad feedback

We asked the developers to give their impression of why users might give feedback with these bad properties. To have a certain range of ideas we gave some options instead of making this open ended, leaving only the *Others* option for additional information. According to them the most probable reason on both platforms (83.33% on desktop and 78.57% on mobile) is the lack of motivation to write out all the necessary information. In addition some commented that they think there might be a "lack of understanding", "No knowledge on how to give helpful feedback, or "Users are not technical". This gives us the impression that a lack of both, motivation and technical knowledge are eligible reasons from the developers perspective. These are also results we tried to confirm in our user study (see 4.2 "User Study"). The findings in this part gave us information about the information and problems with current feedback. The next part, therefore, moves on to discuss possible improvements from the perspective of the developers.

4.1.3.3 Improving feedback

Before asking the users themselves, we wanted to see if developers had some ideas on how to incentivize users to give useful feedback or how to give any feedback at all. One individual suggested to use different forms of incentives such as paying for feedback and one (Android) developer reported about their own attempt: In their app they were exchanging ″premium features for free in return for feedback″ but apparently there was still a lack of feedback because ″there is no way to make them actually put thought into it.″ However, the majority of those who responded to this question felt that it is impossible to make incentives in order to get more or more helpful feedback. Another mentioned aspect was that users, whenever writing a report or want to address any issues, should be aware that developers will use this particular report and the written content to try and fix the issue. This issue of users being unaware was also mentioned by some developers before, regarding the reasons for bad feedback. This is a critical issue since just mentioning it or writing it will not necessarily result in making users aware of the fact, that developers will use this information. One possible approach to this and the previous issue with motivation might be something like Zimmermann et al. [2010] have used in their tool CUEZILLA, which is providing empirical facts about fixing bugs to encourage the user.

The majority of the developers expressed the belief that encouraging users is impossible

A few pointed out that they are concerned users are not aware of how important their data is

The previous results showed that there is indeed a problem with formulation and missing information of both basic system data and also behavior. This further raises the question if the kind of bad feedback, which often just names the problem and does not give any further information is helpful at all. 41.66% of the desktop developers chose that it is not helpful while the other 58.33% said it is indeed still helpful. We saw a similar pattern coming from mobile developers with 45.24% to 54.76%. Based on these results it is hard to make any assumptions or implications, therefore it is necessary to look further into this and investigate the exact reasons.

About the same percentage of developers see bad feedback as helpful and not helpful

In order to solve these problems developers had different approaches. Most developers, regardless of their platform, suggested that a major issue is the communication between developers and users who rate and write reviews. They mentioned several reasons for this case, the one that stood out the most was that it seems to be either hard or impossible for developers to reach out to those users. We saw before (see 3.1 "App Markets") that the Google Play Store and and the Windows Phone App Market allow developers to reply to reviews while the Apple App Store does not offer that feature (cf. Figure 4.4 and Figure 4.5). This certainly makes it hard for developers who can only see the user's profile but no other data they can use to communicate.

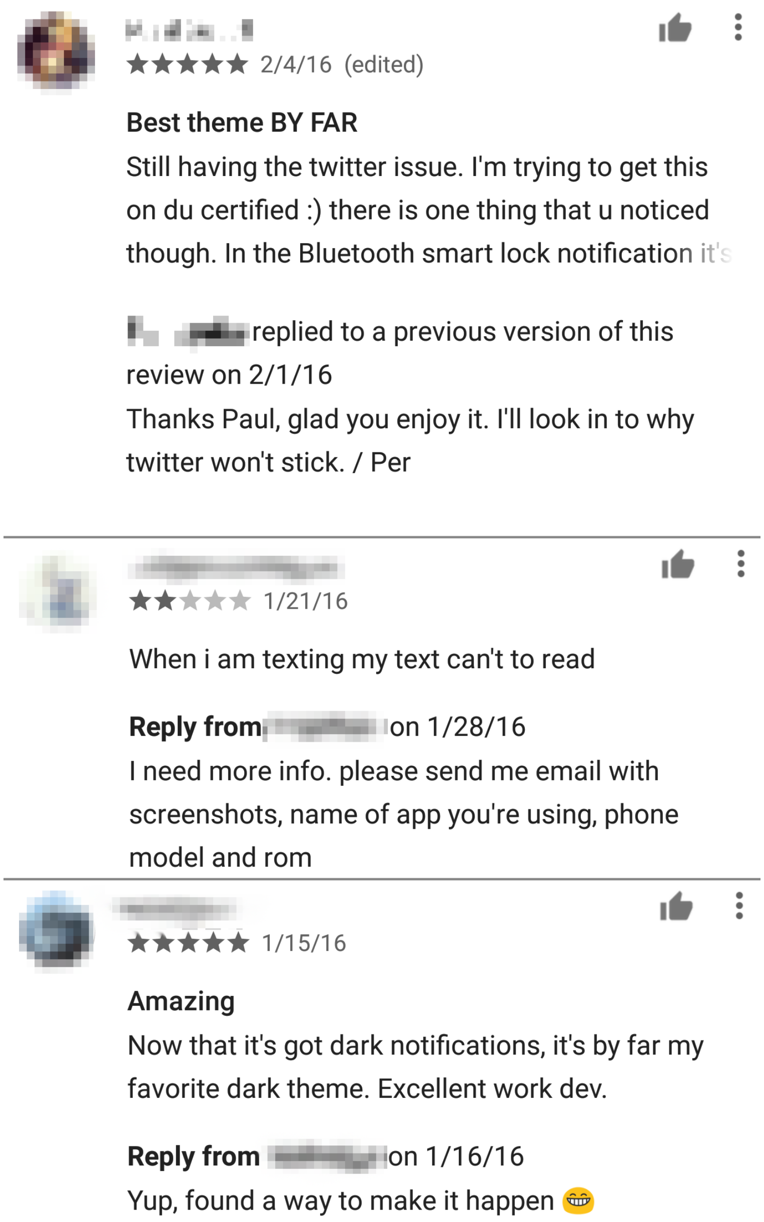
Leaving reviews and ratings does not necessarily allow a two-way communication

Another issue with reviews was that, depending on the platform, they can be written for a specific version of the software at some point in time and they will remain in the market regardless of updates. This can indeed be a problem for developers who update their software and fix issues people were addressing. While the bad ratings remain, other people, who are interested, might read those and refrain from installing them in the first place. Further, those bad ratings persist and can decrease the overall rating of the software, resulting in the same, which makes this a problem for both developers and (new) users. For written reviews and ratings, the Play Store shows the author that this was done for an earlier version. Unfortunately, there is no notification which allows the user to directly go the app page to check this.

Bad ratings in the app markets persist and can drag down ratings, regardless of updates

Furthermore, a few (mobile) developers addressed the problem of being able to give a star rating but not write anything about the reasons. In other words, users can give a very low star rating but due to the lack of text, developers would not be able to know what the problem is. Therefore, some suggested that when giving a lower (star) rating, either a bug report must be written or at least the reason for this has to be provided. This might work in a similar way to what eBay is using for feedback for buyers and sellers. After purchasing or selling an article and providing feedback one can choose between positive, neutral, and negative feedback. Choosing one of the two latter will bring up a text which tells the user to "contact the seller to see if any-

Mobile developers suggested to force users to write reasons in case they give bad ratings



[redacted]
★★★★★ 2/4/16 (edited)

Best theme BY FAR
Still having the twitter issue. I'm trying to get this on du certified :) there is one thing that u noticed though. In the Bluetooth smart lock notification it's

[redacted] replied to a previous version of this review on 2/1/16
Thanks Paul, glad you enjoy it. I'll look in to why twitter won't stick. / Per

[redacted]
★★☆☆☆ 1/21/16

When i am texting my text can't to read

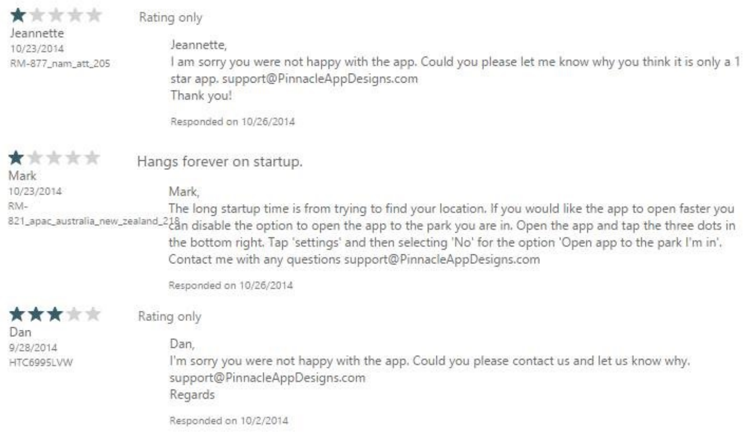
Reply from [redacted] on 1/28/16
I need more info. please send me email with screenshots, name of app you're using, phone model and rom

[redacted]
★★★★★ 1/15/16

Amazing
Now that it's got dark notifications, it's by far my favorite dark theme. Excellent work dev.

Reply from [redacted] on 1/16/16
Yup, found a way to make it happen 😁

Figure 4.4: Replies of developers to a review in the Play Store



★☆☆☆☆ Rating only
Jeannette
10/23/2014
RM-877_nam_att_205

Jeannette,
I am sorry you were not happy with the app. Could you please let me know why you think it is only a 1 star app. support@PinnacleAppDesigns.com
Thank you!

Responded on 10/26/2014

★☆☆☆☆ Hangs forever on startup.
Mark
10/23/2014
RM-821_apac_australia_new_zealand_218

Mark,
The long startup time is from trying to find your location. If you would like the app to open faster you can disable the option to open the app to the park you are in. Open the app and tap the three dots in the bottom right. Tap 'settings' and then selecting 'No' for the option 'Open app to the park I'm in'.
Contact me with any questions support@PinnacleAppDesigns.com

Responded on 10/26/2014

★★★☆☆ Rating only
Dan
9/28/2014
HTC6995LVW

Dan,
I'm sorry you were not happy with the app. Could you please contact us and let us know why.
support@PinnacleAppDesigns.com
Regards

Responded on 10/2/2014

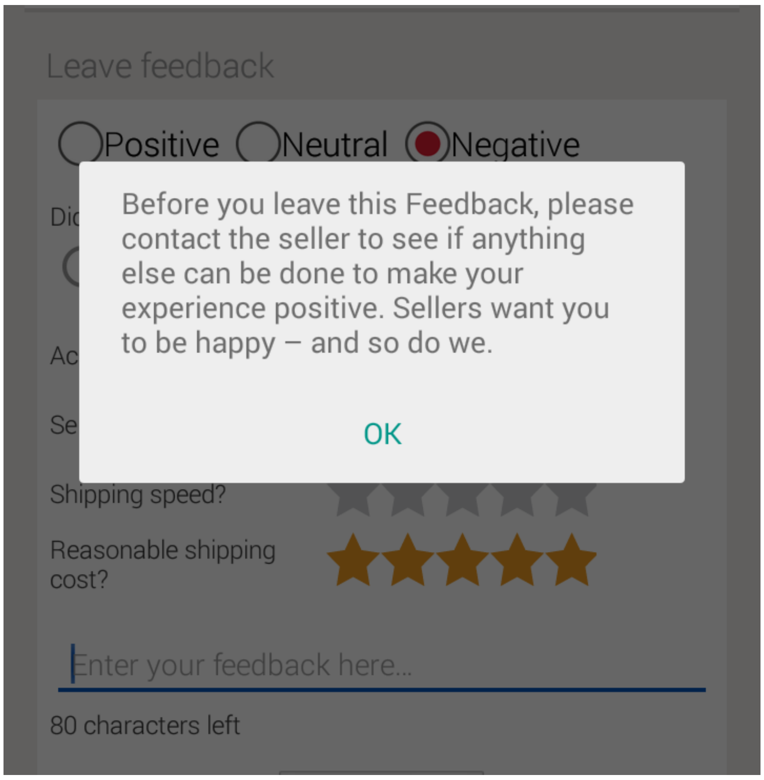
Figure 4.5: Replies of a developer to reviews in the Windows Phone store

eBay's approach of encouraging users to write text might also work for giving lower ratings for apps

thing else can be done to make [their] experience positive." (cf. Figure 4.7 and Figure 4.6). Additionally, the developer should be able to contact the user who gave a review and also be able to reply to it. For this, the user could get a notification and be asked to change his rating or review. As discussed in 3.1.1 "Play Store", the Play Store and Windows Phone store already offer such an option. Another possibility would be to notify when an update for the app is available and remind him of his given feedback.

More than half of the developers supported the idea of both gamification and beta access

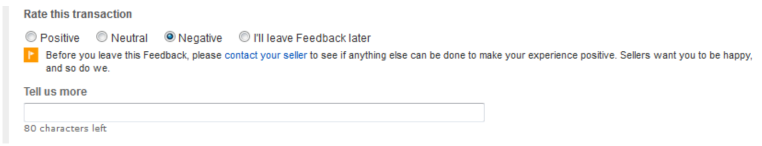
In addition to this completely open question we also gave a concrete example on how users can be encouraged. We asked what developers think of approaches such as gamification or giving access to beta versions. Both of these approaches can already be found in different applications (see 4.1.2 "Questionnaire"). More then half of the developers supported both of these ideas (saying it would be interesting and would also give developers the chance to have users "test either fixes or features they requested, without interrupting other users"). An iOS developer stated that he "give[s] users access to betas when [he] add[s] a feature they requested. It helps to build a relationship with the user". Some also brought up ideas of incentives, using money for bug hunting or give access to beta features. In fact, several companies provide so called Bug bounty programs, such as Github [Github Bounty], Google [Google


Leave feedback

Positive Neutral [✓ Negative]

Before you leave this Feedback, please contact the seller to see if anything else can be done to make your experience positive. Sellers want you to be happy – and so do we.

Figure 4.6: eBay warning text in the app


Rate this transaction

Positive Neutral [✓ Negative] I'll leave Feedback later

Before you leave this Feedback, please contact your seller to see if anything else can be done to make your experience positive. Sellers want you to be happy, and so do we.

Tell us more

80 characters left

Figure 4.7: eBay warning text on the website

VRP], or Facebook [Facebook White Hat], offering payment to those who submit issues.

Those who showed skepticism mentioned different reasons. It could make users feel entitled or make them give false feedback and abuse the system; one developer mentioned he had personal experience with false feedback due to such a system but thinks about using such approach again in a different way. Another one stated that gamification in general ”would be a mess” and instead ”having a

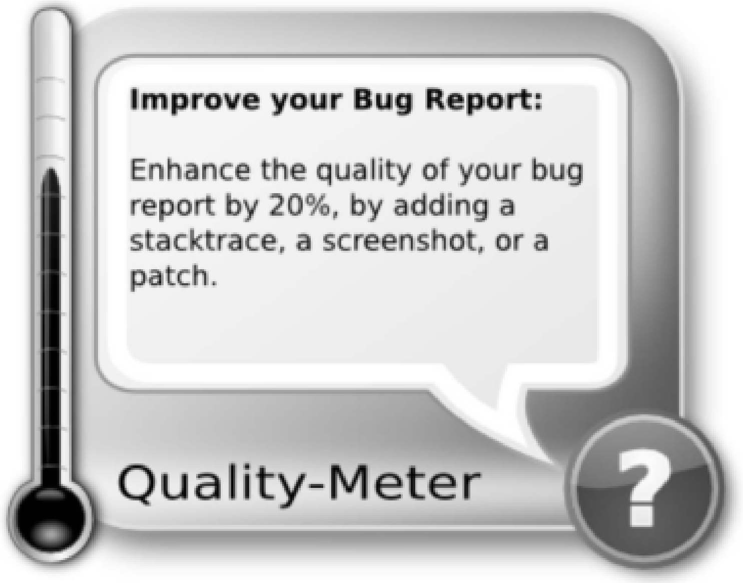

Improve your Bug Report:
Enhance the quality of your bug report by 20%, by adding a stacktrace, a screenshot, or a patch.
Quality-Meter
?

Figure 4.8: Mockup of CUEZILLA's quality measurement from [Zimmermann et al., 2010]

At the same time others gave valid reasons why these types of incentives might lead to different problems in feedback

system like stackoverflow would be perfect". Partly, these reasons are justified. Users might send in reports and hope to get something in return, and get upset in case the will not get any reply. This might be solved by different rewarding regulations and rules. Regarding false feedback, it should be clear that when having such systems an instance is necessary which assesses feedback in terms of quality. One developer mentioned that "[f]iguring out how to judge "good bug reports" and writing features to reward good behavior is a terrible waste of time [...]". This is only true insofar as a human worker will do this task. But Zimmermann et al. [2010] showed in their work a tool which measures the quality of bugs, proving that at least to some extent this work can be automated (cf. Figure 4.8). We extended this discussion with our last question, which listed five recommendations from the same study and asked the participants to evaluate those. The first one involved implementing different interfaces for novices and experts, allowing new users to give basic information and not overstrain them while experienced users could submit further data. The participants seemed indifferent, only 24% would

Most developers looked critically at having two different interfaces welcome this idea and 19% think it is a bad idea. This certainly is not easy to assess in general and might be more interesting in an explicit discussion involving reputations and rankings. Further, this would either require rules on how to classify reporters or let them choose, which could lead to other problems, that is people would need to know themselves and a developer also commented that people would not want to identify themselves as something "inferior" like a novice or beginner.

Using user profiles and individual reputations did not receive much approval

This leads to the second recommendation, discussing individual profiles and reputations for each user. Interestingly, this rather "global" approach lead to less approval and disapproval with merely 15% respectively 11%. While the reasons are probably the same as before, this leads to the assumption that not many developers believe a classification or grouping of reporters would lead to better results. This also partly corresponds with the results from the results of the question regarding gamification.

Giving users feedback about the status was an approach, which was supported by almost half of all developers

We discussed earlier that the communication between reporters and developers is key. Hence it can help to give feedback about the working status of reported issue. In this case, the results concluded that 44% of the participants agreed to this and merely 4% were against it. It should be noted though, that this does not necessarily mean that users (and further average user) agree as well. This gave us reason to test this in the user study (see 4.2 "User Study").

Recording assistance received the most support, but the actual realization is not easy

Earlier, we also mentioned that the system is able to detect basic information about the used hardware and software. This leads to the proposal that it would certainly help further (especially those who are not technically savvy) if the system could also record the steps to reproduce the addressed issue. As Zimmermann et al. [2010] mentioned in their studies, a user received an award for his bug because he "used a flash movie to demonstrate the rather complicated steps to reproduce [...]". This recommendation also received the most positive reception from both desktop and mobile developers with 41.67% and 50%. An actual implementation of such tool assistance proves to be difficult though. Depending on the issue (i.e., one that leads to a crash) such a tool would need to work globally and not in the application itself. In such a case the respective system developers need to provide such a functionality and allow it to be sent or embedded into a report for the developer.

4.2 User Study

We decided to make a calculator because it does not require adaptation

As explained before, phase 2 of our approach involved active participation of users in an experiment along with a survey. We split the whole survey into smaller questionnaires and the practical experiment itself. Since one of our main focuses was to see how the average user deals with feedback, we designed a small application to test this. For this purpose we needed to decide on different matters such as what kind of application and for which platform. The application should not force the user to adapt to completely new things and therefore look familiar, also for those who are not even using technology in their everyday life. With this, we reached the decision to make a simple and basic calculator. Most people have used one at at least some point in their life and would manage how to use one.

One was a UI bug and another bug lead to a crash

To determine whether the participants were able to submit the information a developer requires, we built in flaws. More precisely, we implemented three bugs with different severities. The first one would cause no harm to the functionality and is purely a UI bug while the second one leads to a crash (cf. Figure 4.9 and Figure 4.10). Our calculator offered the ability to be used with both the numpad and also the mouse clicking the onscreen buttons. The third bug prevented the division and multiplication button on the numpad from working correctly.

This way we could check three different kinds of feedback reports. Therefore, it also supports this approach that due to this fact, subjects would try and associate their previous experiences with other calculators when using ours.

The UI bug caused a slight misplacement of a digit on a button and colored the number red when hovering with the cursor. When computing some basic operations (and not using the numpad for this) the user should come across

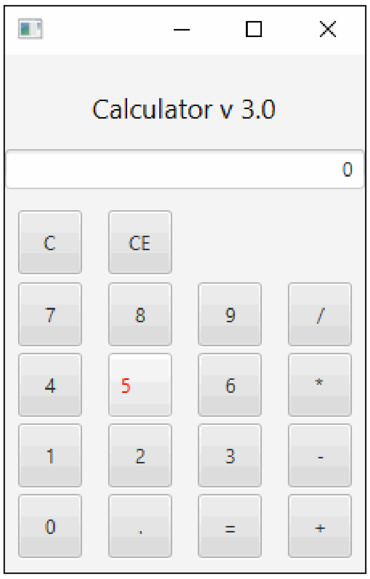
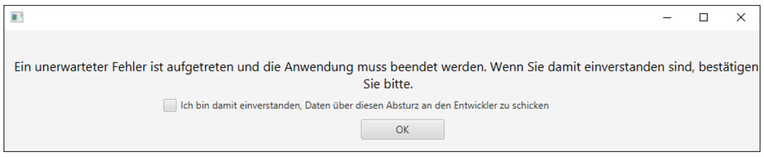


Figure 4.9: UI bug in our calculator application



Ein unerwarteter Fehler ist aufgetreten und die Anwendung muss beendet werden. Wenn Sie damit einverstanden sind, bestätigen Sie bitte.

Ich bin damit einverstanden, Daten über diesen Absturz an den Entwickler zu schicken

OK

Figure 4.10: Popup upon crashing in our calculator application

this bug fairly often. This was also a reason why we used a desktop PCs as a platform, as hovering allows easier possibilities for UI bugs. The other, rather major issue was used in the result/equality function. With a chance of 1/5, clicking the ″=″ button popped up a little window on top of the application, showing a faux error message, a second message and an ″OK″ button to close it. Further, the second text had a checkbox in front of it, which should be checked when the user wants to be contacted by the developer to help with this issue. Checking the box had no impact whatsoever, the purpose was for us to see if participants would read it and check it.

The UI bug concerned a bad text alignment while the other bug lead to a random crash

Before this experiment, the participants were asked to fill out the first questionnaire. Through the duration of the practical experiment, we asked the participants to also think aloud, meaning they should speak out their current thoughts. After the experiment we handed them a another questionnaire together with a report form, which they would use to fill out a feedback/bug report form. The following sections discuss the structure of these in detail.

4.2.1 Pre-session Questionnaire

We structured the first questionnaire as four sections: Demographic data, general experience concerning feedback, IT experience, and two specific parts about feedback on mobile devices and on desktop computers. Each section ended with a commentary field in case the participants wanted to add something.

We asked about the users' background in terms of technology and feedback experience

Unlike the developer study, the demographic data is more common as in this case we were interested in what kind of average user the participant is. The following two parts deal with the user's background in terms of feedback and technology. Concerning general experience with technology we asked the participants about their behavior and experience with different electronic devices. Similarly, we wanted to see how and if they work with apps. While this could have been done for software on desktop computers as well, it was fair to assume that people, who are not technically savvy, would rather look for new applications on their mobile phone than on their PCs. Regarding more specific behavior about feedback and giving feedback, section three contained questions about actual action. These range from general satisfaction and frustration when using software on to asking what they do in case they get frustrated up to what could help.

Participants were also asked to tell us about their behavior when dealing with frustrating software

4.2.2 Post-session Questionnaire

After conducting the practical part of the study, we asked the participants to describe their impression. For this matter we asked about their relative success in using our application and in case they got frustrated, describe the reason for this. Further, we asked if there were any features they missed or were considered unnecessary. On the first page, we asked this in general in order to get the first ideas about this. The next page contained the same answer, but in addition we showed three calculators (software calculators and a physical one). This way participants could be reminded of features they could have forgotten or did not know would be technically possible. The first two calculators shown are software types. They show a calculation method and respectively a simple design. While the other software calculator does not show any similar feature, some participants might remember the usability and design and hence form thoughts and demands on (new) ideas for our application. Unlike those, the third is an often used physical calculator with a significant variety of different buttons. Since our own calculator lacked lots of features and its design could also be improved, these figures can give the participants some impressions of what is possible (or remind them of that) and allude them to demand similar features, which they might mention.

The participants were asked to describe their experience with the practical experiment

4.2.3 Feedback Report

We designed the feedback form for the users very minimalistic and simple. Although this is not necessarily a form which will look the same way in a real submission form, this should not overstrain users and give them a feeling that the application asks for lots of information. Given these preconditions we asked the users to fill out eight of the original twelve criteria for a report. We removed stack traces, screenshots, build information, and test cases as these are information which are not required to be put in by the user or - in case of test cases - not important. Further we did not ask about the hardware information because for one, the system used was unknown to the users and would therefore not show representative data, and also as discussed, such data is known to the system and does not need to be input.

The feedback form contained 8 criteria for the user to fill out

All questions were made open ended except #3 which asks

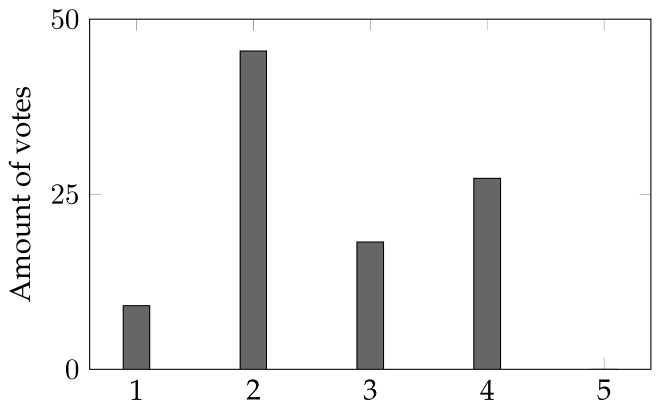


Figure 4.11: Degree of technical expertise

about the severity. In their studies, Herraiz et al. [2008] found out that three different severity grades are sufficient for developers when considering the time until a particular issue is solved. Therefore, we listed *low*, *medium*, and *high*.

4.2.4 Results

4.2.5 Background and Experience

In our user study eleven users participated, with ages ranging from 21 to 56 (with a median of 24). Five of them were students while the other six were working in companies or having their own. None were enrolled in technical study programs or working in a rather technical field. In addition to this, they had to specify their technical expertise on a Likert scale and a median of 2 indicated that most participants did not see themselves as very technically skilled (cf. Figure 4.11). Furthermore, all participants stated they use a desktop PC or notebook as well as a smartphone, both with internet. They use these devices either *quite often* or *extremely often* and all of them had already downloaded at least one app on their own.

Regarding their prior experience with feedback we saw that first of all, the general satisfaction with electronic devices was rather high with a median of 4 (cf. Figure 4.12. At

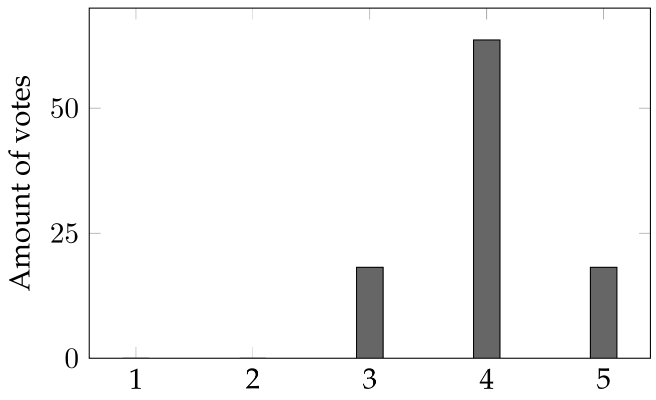


Figure 4.12: General satisfaction with electronic devices

the same time all but one participant stated to have been frustrated with those devices at least once. The frequency of getting frustrated was indicated by most as either *rarely* and *now and then* while one stated to get frustrated *very often*. Reasons for this are mostly that software wither crashes (45.45%), does not behave in a way the user intends to (54.54%), or is hard to use (36.36%). Out of these problems, only one participant rated his problems as a 4 on a Likert scale of 1-5. The rest rated their problems as 1-3 with a median of 2. These results indicate that while in general users are satisfied, they can still get frustrated by smaller problems (from their perspective).

In general users were rather satisfied with their electronic device

Further results showed that most participants approached the problems with their own methods: 81.82% stated to try and solve the problem on their own and 72.73% would search for help on the internet or in manuals. At the same time 18.18% stated to *get annoyed and continue using it*. In total, 81.82% were satisfied with their methods and indicated that they would help them.

4.2.5.1 Incentivization

During our survey we asked the participants to indicate on a Likert scale from 1-5 how likely it is that they would give feedback in general, in case they know that such possibility exists and also when some conditions are given. Figure

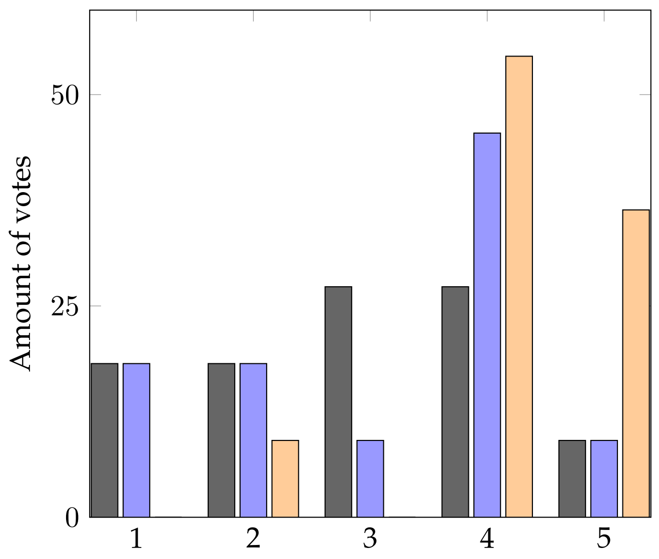


Figure 4.13: Likelihood of giving feedback

4.13 shows these outcomes. The black bars mark the likelihood of giving feedback in general, now that the participants are aware of the possibility to give feedback directly to the developer. The likelihood of reporting in case the method of doing so would be simple and intuitive is shown by the blue bars. Lastly, we asked about the probability if there would be some kind of rewarding system behind it and therefore encouraging the users, which is indicated by the orange bars.

While the first outcome shows a normal curve of distribution, there is a clear tendency for the latter two options. 54.54% chose either 4 or 5 in case the way of giving feedback is simple enough and 90.9% voted for 4 or 5 in case they would be rewarded. This shows that current feedback systems are not attractive enough and that certain changes, which involve incentivization, would inspire them to become more active.

Being rewarded and having a simple way of giving feedback are features which were supported by many users

Regarding the rewards in particular and other incentives, we observed that some options were listed more often than others, creating a certain taxonomy. Most participants wrote that they would like to receive in-app purchases or a discount/credit/coupon with 63.63% respectively 54.54%. Aside from these, participants also stated they would like to be able to have the paid version (in case there are both free and paid versions) temporarily or permanently (9.09%). Interestingly, one participant mentioned that having the issue fixed would be enough encouragement.

Looking at the current technical possibilities, these ideas are all admissible. For instance, Google (Providing Promo Codes[7]), Apple (In-app Promotions[8]) and Microsoft (Microsoft Promotional Codes[9]) offer so called promo codes, which allow developers to offer customers free versions of paid apps. Additionally, Android and Windows Mobile developers can also give out promo codes for in-app purchases. Previously we have also discussed the possibility for developers to provide beta access (see 4.1.2 "Questionnaire").

4.2.5.2 Terminology

Regarding the experiment, what stood out the most was that almost all participants had trouble filling out the feedback form without further explanation and oftentimes guidance. Most likely the reason is that we just used the terms as we have given them as options for developers, meaning developers will definitely understand their meaning but this does not necessarily apply to everyday users. But it was important to design the first version (see 5.2 "Future work") with a minimum of information in order to see, if this might be sufficient. Further iterations would simplify this, starting from this (see 5.2.1 "Feedback Form"). Our observations showed that after explaining the terms, participants stated that they would then understand.

The observation which stood out the most was that nearly all participants needed explanations of the terms used in the form

In addition to this, some particular fields caused the most confusion even after asking about it. Most of the participants were not sure what the difference was between the summary field and the fields for observed/expected behav-

[7]http://apple.co/1osILYj
[8]http://bit.ly/1PWTEc2
[9]http://bit.ly/20JnLJg

ior. It might be interesting to see in further iterations, if this breakup of these fields is necessary or, if the described behavior is enough.

4.2.5.3 Reports

The formulation problem could also be seen during the evaluation of the feedback forms

Throughout all results we were able to confirm the formulation problem, which we saw in the results of our first study, to a certain degree. As discussed before this might rest on the problem with the terminology used. It should also be noted for the results of the feedback form, that 66.67% addressed the crash issue and 33.33% regarded the computation issue with the multiplication/division button. In addition to this, from those who issued the crashing, two did not fill out the report form. As mentioned in the outcome of the post-feedback survey, they did not have the impression that there was a problem with the application worth reporting. One of them pointed out that they thought it was their own fault, thus a report was unnecessary. This results in our sample size of $n = 9$ for the reports.

Nearly half the subjects were not sure on how to rate the severity

Considering the basic information, all subjects formulated the name and version correctly. We have already considered the possibility of having this automated, in other words while this is positive, it is still not necessarily required to fill in. The severity ratings showed different results, as 33.33% chose the low priority and 55.56% chose the high priority, while only one participant rated the crash as a medium severity problem. Aside from these results it was rather interesting to see that almost all participants were not sure about how to rate it, either to what degree this problem annoys them during their usage or to what technical degree they could estimate it. During the experiment we told them they should rate it according to the former reason. This raises the need to make it more clear and describe how users should rate the problem.

The most important part, which also relies the most on the user input, were the fields about the behavior and steps to reproduce. For this part, we will refer to those users who addressed the crash issue group A and those, which information regarded the computation issue group B. Concerning the observed behavior, group A mostly described that a window appeared upon crashing and the content of it. One participant described that it popped up upon clicking the "=" button. All of group B reported the observed behavior correctly and described the incorrect information in the display. Similarly, in this case also one participant described it explicitly, giving a concrete example of what happened in the display using a specific input.

The information given in the expected results showed that most participants (of both groups) just noted that they would appreciate it if the application did not have the problem, which was already mentioned in the *observed behavior* field. As for the summary field, which came right before the block about behaviors, all participants wrote a short sentence about the problematic behavior of the problem.

For our experiment these results suggest that it should be looked into the relevance and importance of having both expected and also observed behavior, which is also supported by the fact that most participants were confused and thought the expected behavior would be enough.

While the expected behavior can be clear when concerning crashes, it is not necessarily the case with usability bugs. Since none of the participants addressed the UI bug, this requires further experiments with different bugs.

Throughout the experiment participants were confused about why they had to give a summary and then split it into expected and observed behavior

The most interesting observation about this part was that the participants were confused about the summary followed by description of the behavior. They reported that they would not understand why they would have to write a summary, which could explain the whole problem by itself, and then split this summary into observed and expected behavior. Together with the results we discussed before, these provide support for a discussion of the requirement and meaning of a summary field, in addition to the behavior fields.

To help with the steps to reproduce, we discussed with developers about a recording tool (cf. 4.1.3.3 "Improving feedback"). All but two participants agreed that in case it would be easy to use, they would prefer using such method. Those who disagreed noted that they would have concerns regarding privacy.

Chapter 5

Summary and future work

In our work one of the primary aims was to better understand how the everyday user behaves in terms of frustration when dealing with software. The purpose of our approach was to determine if the lack of technical knowledge is an actual hindrance, holding them back from providing feedback. This approach consisted of a general evaluation of public and already existing feedback and two user studies. The first study obtained information from different developers regarding what information in bug reports is essential for them in order to be able to fix those issues. We then used this information to conduct and design a study with end users (with no technical background) and assess their ability to deliver this information, using our self developed application in a practical experiment. In addition to this we also used questionnaires in both studies to gain deeper insight into what developers think could be changed to improve the rate of helpful feedback and what users think would encourage them to become active and submit feedback to those developers.

5.1 Summary and contributions

Our results in the public feedback analysis showed, that feedback itself, indeed, is an active phenomenon. Users, regardless what kind, seem make heavy use of the app markets and also send a lot of feedback to developers. However, the quality and content of the given feedback was varying. Approaching developers of desktop applications and mobile applications revealed interesting insights into what they consider essential information when handling user feedback. Our most important finding was that both groups put emphasis on nearly the same criteria. This provides evidence, that feedback forms can be designed and structured regardless of the platform.

We saw that essential information for both platforms is nearly the same

Finally, our user study provided information about the ability of everyday users to deliver these important criteria when reporting feedback. Although we had a small sample size, our experiment showed that the subjects were capable of providing all necessary information. However, most were only capable of doing that after explaining some of the terminology used in the feedback form. Nevertheless, they showed enough potential, allowing future investigations to simplify the form which may then establish further results.

With some explanations of terminology participants were indeed able to fill out this information

Returning to the questions posed at the beginning of our work, it is not yet possible to find complete answers to both questions, but give directions for further iterations and some suggestions. However, for the first question regarding differences in the required information for developers, we provided evidence that concerning mobile and desktop platforms, there requirements are nearly the same.

We were able to answer our first question in the beginning but the second one requires further research in our provided directions

Our results have shown that both parties are interested in improving the communication. We also saw that using rewards or incentives might be a lead in this matter, for both sides. Furthermore such methods are also feasible in the technical context as we have seen in our analysis.

5.2 Future work

Our literature review suggested that as of right now, research has rather focused on improving the quality of feedback, which is reported by users who area already technically skilled, work in companies and are supposed to assess the quality of their internal software, or should receive training for reporting. Therefore our work sets a novel approach, taking in account the everyday users, who could be able to provide feedback just using the feedback form and no further guidance. Considering this, our findings can motivate further research and extend our studies to different directions in the same scope. Our approach consisted of studies with both parties who participate in feedback, the users and the developers. Both contain different components, with each offering spaces for opportunities. In this section we provide such possible opportunities in order gain more and different insight with further iterations.

Different components in our work allow further extensions in research to gain further insight

5.2.1 Feedback Form

Our observation of users showed, that the language or vocabulary in our feedback form was still too technical and caused confusion. As discussed, we are very certain, that this is mainly a problem of the choice of words and phrasing. Meaning that further studies could most probably analyze different forms and check if they improve the understanding of users. The main aim regard should be to make users completely capable of filling out the forms without receiving instructions and guidance.

As discussed, the used feedback form causes confusion, but further iterations might improve it

5.2.2 Experiment

Our practical part in the user study involved our own custom calculator application. Since this is a very synthetic test, further and different types would need to be conducted. So far, we have tested a UI bug and a crash to see if users were able to report those. As we have mentioned in the beginning, Hartson and Castillo [1998] saw in their

Test the ability to report incidents of different severities results that with brief training users were also able to rate the severity correctly. Further iterations of our experiment could show if users with no training can also rate different severities and to what degree of complexity or severity they are able to. To solve the synthetic aspect of our application, one could conduct a study with applications the users also use in their everyday life. Although this would require finding possible issues which can be addressed in a report.

Possible variants of the experiment may include already existing software instead of custom

5.2.3 Technically-skilled users

In this study we mainly focused on users, whose technical skills are on or below average. But there are is also the other group of users, those for who we assumed that they indeed are capable of writing technical reports and most likely only lack the motivation for being active in regard to feedback. Additionally this could also show if there are problems with current feedback options and if so, how can those be improved. Therefore it might be interesting to see, if similar incentives or rewarding systems would also apply to those or if different approaches are necessary.

Analogue studies could analyze the behavior of technically savvy users

5.2.4 Incentivization

Concerning those, we only asked about a rough amount of possibilities for how to encourage users. In addition to this, we have seen in our results that there was no clear tendency of what would actually encourage most of them. Despite this, there were different ideas given by the participants what would actually encourage them. Further, have mentioned in the beginning, that there are studies on gamification and bug hunting is also a successful way, but those are not necessarily attractive to everyday users. This raises the need for further studies in this matter.

Further insight into rewarding systems and other incentives should be done

5.2.5 Further developers' perspective

Our first study asked the developers about criteria of bad or not helpful feedback. But only giving options to choose from, we were unable to see the reasons. Further insight into this might give more information about developers' needs concerning information. This is especially interesting for the hardware/software information which, as we discussed, can be acquired automatically. This also holds for our question regarding if bad feedback is better than no feedback, as this could also help designing better feedback forms.

We lack the reasons for why the criteria for bad feedback are actually not helping

We have also asked developers if they are offering free or paid software. The same question can be directed towards users, since similarly, they might also have different expectations of software, being free or with costs. Given a deeper analysis, this might give a better understanding of motivation.

5.2.5.1 Differentiation in feedback

Our work also focused on feedback in general. But further work would need to require a more distinctive differentiation between the public feedback (e.g. that in App Markets) and critical reports which is only directed towards the developers. For example we asked about possible benefits or rewards for providing feedback. Asking participants in detail about their submission revealed that most of them were thinking of both variants. Still, separated questions might give different results. This might also hold for questions about the specific rewarding systems and might further also be interesting to ask developers.

More studies are necessary about specifically feedback as in reviews and explicit reports

Appendix A

Questionnaire for Developers

Developer Survey

1. How old are you?

____

2. For which platform do you develop mainly?

Please choose only one

- ☐ Windows
- ☐ Linux
- ☐ Mac
- ☐ Android
- ☐ iOS
- ☐ Windows Phone
- ☐ Other: ______________________________

3. How many years have you been developing software?

____

4. Are you developing as a hobby or professionally?

Please choose only one

- ☐ Hobby
- ☐ Professionally

5. In case you develop professionally, how many employees do you have?

Please choose only one

- ☐ Less than 10
- ☐ Between 10-49
- ☐ Between 50-249
- ☐ Between 250 and 499
- ☐ At least 500

6. What is the target market of your software?

- ☐ Corporate customer
- ☐ Private customer

7. What is your job role in this project?
Software Developer, Software Engineer, UX/UI, etc.

____

8. Is your software free?

Check both in case there are paid and free versions

- ☐ Yes
- ☐ No

9. In what ways can users give you feedback in your app or program?
For example in the app itself (and if so, how) or mail, etc.

10. Which one is the used the most?

11. Which one is the most useful to you as a developer?

12. What is the ratio between positive and negative feedback you receive?

Please choose only one

- ☐ More negative than positive
- ☐ More positive than negative
- ☐ Almost equally

13. How helpful is the overall feedback you receive?

Useless						**Helpful**
	☐	☐	☐	☐	☐	
	1	2	3	4	5	

14. What type of feedback do you get?

- ☐ Usability/UI
- ☐ Feature request/removal
- ☐ Bug report (manually)
- ☐ Bug report (automatically)
- ☐ Others: ______________________________

15. What kind of information in feedback is the most important?

- ☐ Product/App name
- ☐ Version
- ☐ Severity
- ☐ Hardware, OS and version, App version
- ☐ Summary
- ☐ Observed behavior
- ☐ Expected behavior
- ☐ Steps to reproduce
- ☐ Stack traces
- ☐ Screenshots
- ☐ Error reports
- ☐ Test cases
- ☐ Others: ______________________________

16. What are some characteristics of bad feedback?

- ☐ Wrong/Missing Product/App name
- ☐ Wrong/Missing Hardware, OS and version, App version
- ☐ Wrong/Missing observed behavior
- ☐ Wrong/Missing expected behavior
- ☐ Errors in step to reproduce
- ☐ Errors in test cases
- ☐ Bad Grammar/No spell check
- ☐ Unstructured text
- ☐ Non-technical language
- ☐ Too long text
- ☐ Spam
- ☐ Incomplete information
- ☐ Others: ______________________________

17. What do you think are the reasons for that?

- ☐ Hard to reach the feedback option in the application
- ☐ Not enough motivation to write everything down
- ☐ Feedback system for developers does not offer enough options
- ☐ Others: ______________________________

18. How do you think users can be encouraged to give (helpful) feedback?

____

____

____

____

____

19. Is bad feedback better than no feedback at all?
Think of one-liner feedback such as "Crashes alot", "Very slow", etc. without further context

Please choose only one

☐ Yes
☐ No

20. If you could change anything in the whole system (Play Store, App Store, or on Desktop Operating Systems,...), what would it be?
For example how developers receive the feedback, are notified about it, etc.

____

____

____

21. What do you think about rewarding good bug reports with methods like gamification (for example levels, ranks,...), giving beta access to future builds/versions of the application, or privileges in the app store?

____

____

____

22. What do you think of the following methods?

	Bad				Good
Different feedback interfaces for novice and experts	☐	☐	☐	☐	☐
Give reporters a profile and assign certain reputation	☐	☐	☐	☐	☐
Give reporters feedback on the working status of a bug	☐	☐	☐	☐	☐
Provide tool assistance to record problems/bugs	☐	☐	☐	☐	☐
Provide possibility to reproduce bugs (e.g. "sandbox")	☐	☐	☐	☐	☐

23. Comments about feedback experiences
What has not been covered by questions or you just want to add

Appendix B

Pre-session Questionnaire for Users

Persönliche Angaben

1. Ihr Alter:

____

2. Ihr Geschlecht:

☐ Weiblich
☐ Männlich
☐ k.A.

3. Ihr aktueller Beruf:

____

4. Falls Sie Student sind, welcher Studiengang?

____

Feedback Erfahrung

5. Kaufen Sie schon einmal bei Online Versandhäusern ein, wie bspw. Amazon?

☐ Ja
☐ Nein

6. Lesen Sie sich (falls vorhanden) auch Bewertungen und Rezensionen durch?

☐ Ja
☐ Nein

7. Haben Sie diesbezüglich auch folgende Aktionen durchgeführt?

	Einmal	Mehrmals	Noch nie
Bewertung gegeben (Sterne, Daumen, ...)	☐	☐	☐
Rezension (Textlich)	☐	☐	☐

IT-Erfahrung

8. Wie technisch versiert sind Sie?

Weniger				**Sehr**
☐	☐	☐	☐	☐
1	2	3	4	5

9. Welche elektronischen Geräte nutzen Sie im Alltag?

Wahl mehrerer Antworten ist möglich

- ☐ PC/Laptop
- ☐ Smartphone
- ☐ Handy/Dumbphone/Feature phone
- ☐ Smart TV
- ☐ Ordinärer Fernseher
- ☐ Smartwatch

10. Wie häufig nutzen Sie diese Geräte in der Woche?

- ☐ Extrem oft
- ☐ Sehr oft
- ☐ Ab und zu
- ☐ Kaum
- ☐ Sehr selten

11. Bei welchen Geräten nutzen Sie auch die Internet-Funktionen?

Wahl mehrerer Antworten ist möglich

- ☐ PC/Laptop
- ☐ Smartphone
- ☐ Handy/Dumbphone/Feature phone
- ☐ Smart TV
- ☐ Ordinärer Fernseher
- ☐ Smartwatch

12. Welche Betriebssysteme nutzen Sie auf Ihren Geräten?

Wahl mehrerer Antworten (bei mehreren Geräten) ist möglich

- ☐ Windows (PC/Laptop)
- ☐ MacOS
- ☐ Linux (PC/Laptop)
- ☐ Android (Smartphone)
- ☐ iOS (Smartphone)
- ☐ Windows Phone (Smartphone)
- ☐ BlackBerry OS (Smartphone)

13. Sind Sie sich bewusst, dass Sie für Ihr Smartphone Apps herunterladen können?

- ☐ Ja
- ☐ Nein

14. Falls ja, haben Sie schon einmal Apps heruntergeladen?

- ☐ Ja, selbstständig
- ☐ Ja, aber mit Hilfe von jemandem
- ☐ Nein

15. Wie oft laden Sie Apps herunter?

- ☐ Sehr oft
- ☐ Ab und zu
- ☐ Kaum

16. Über welche Wege kommen Sie dazu, neue Apps zu finden oder herunterzuladen?

Wahl mehrerer Antworten ist möglich

- ☐ Trends/Charts im jeweiligen App Store
- ☐ In der Werbung gesehen
- ☐ Im Internet gelesen
- ☐ Von anderen empfohlen bekommen
- ☐ Sonstiges: ____________________

17. Gucken Sie manchmal einfach nach Apps, ohne eine App mit einer gewissen Funktion zu wollen?
"Schaufensterbummeln"

- ☐ Ja
- ☐ Nein

18. Falls Sie nach Apps im Store gesucht haben, gucken Sie sich auch die Bewertungen und Rezensionen an?

Bitte nur eins auswählen

- ☐ Ja, auch wenn ich weiß, dass ich die App sowieso herunterladen werde
- ☐ Ja, wenn ich mir unsicher bin
- ☐ Ja, aber nur manchmal (Abhängig von Lust, Zeit, etc.)
- ☐ Nein

19. Haben Sie auch mal die Möglichkeit genutzt um eine Rezension zu bewerten?
Tragen Sie bitte, soweit bekannt, die Gründe/Motivation hierfür bei "Sonstiges:" ein

- ☐ Ja
- ☐ Nein
- ☐ Sonstiges: ____________________

20. Falls ja, was gucken Sie sich diesbezüglich an?

Wahl mehrerer Antworten ist möglich

- ☐ Nur die Rezensionen
- ☐ Nur die Bewertungen (Sterne o.ä.)
- ☐ Sowohl Bewertungen als auch Rezensionen

21. Kommentare zu diesem Abschnitt

____

Feedback

22. Wie zufrieden sind Sie mit Ihren Geräten und den Programmen/Anwendungen?

Garnicht					**Ziemlich**
	☐	☐	☐	☐	☐
	1	2	3	4	5

23. Waren Sie schonmal frustriert oder verärgert beim Bedienen eines Programms?

- ☐ Ja
- ☐ Nein

24. Falls ja, wie oft passiert das?

- ☐ Extrem oft
- ☐ Sehr oft
- ☐ Ab und zu
- ☐ Kaum
- ☐ Sehr selten

25. Was frustriert oder verärgert Sie dann im konkreten Fall?
Das Programm...

Wahl mehrerer Antworten ist möglich

- ☐ ... stürzt ab
- ☐ ... tut nicht, was ich will
- ☐ ... ist schwer zu bedienen
- ☐ ... ist schwer zu verstehen
- ☐ Sonstiges: ______________________________

26. Sind die Probleme Ihrer Einschätzung nach eher größerer oder kleinerer Natur?

Gering					**Größer**
	☐	☐	☐	☐	☐
	1	2	3	4	5

27. Was machen Sie in dem Fall?

Wahl mehrerer Antworten ist möglich

- ☐ Ich versuch selber, gegebenenfalls das Problem zu lösen
- ☐ Ich suche nach Hilfe (Internet, Handbuch, etc.)
- ☐ Jemanden fragen
- ☐ Mich beschweren
- ☐ Mich ärgern und weiter machen wie gewohnt
- ☐ Sonstiges: ________________________________

28. Helfen diese Lösungsansätze oder sind Sie zufrieden mit diesen?

- ☐ Ja
- ☐ Nein

29. Haben Sie schon einmal Feedback zu einem Programm auf Ihrem PC gegeben?

- ☐ Ja, positives Feedback
- ☐ Ja, negatives Feedback
- ☐ Nein

30. Haben Sie schon einmal Feedback zu einer App auf Ihrem mobilen Gerät gegeben?

	Einmal	Mehrmals	Nie
Bewertung	☐	☐	☐
Rezension	☐	☐	☐

31. Falls Sie sich beschwert haben, haben Sie sich da auch beim Entwickler selbst beschwert bzw. ihm Ihre Probleme mitgeteilt?

Bitte nur eins auswählen

- ☐ Ja
- ☐ Nein, ich wusste nicht, dass das geht
- ☐ Nein, ich weiß, dass das geht aber habe das nicht gemacht

32. Wenn Sie nun wissen, dass das geht, wie wahrscheinlich ist es, dass Sie das in Zukunft machen werden?
Falls Sie zuvor ausgewählt hatten, Sie hätten es nicht gewusst

Sehr unwahrscheinlich						Sehr wahrscheinlich
	☐	☐	☐	☐	☐	
	1	2	3	4	5	

33. Falls Sie nur Bewertungen geben oder keine Bewertungen und keine Rezensionen, warum?

____

34. Wie wahrscheinlich ist es, dass Sie bewerten und Rezensionen schreiben, wenn es sehr einfach gestaltet ist und wenig Aufwand erfordert?

Sehr unwahrscheinlich						Sehr wahrscheinlich
	☐	☐	☐	☐	☐	
	1	2	3	4	5	

35. Wie wahrscheinlich ist es, dass Sie das machen, wenn Sie dafür "belohnt" werden?

Sehr unwahrscheinlich						Sehr wahrscheinlich
	☐	☐	☐	☐	☐	
	1	2	3	4	5	

36. Was für Belohnungen würden Sie eher dazu bewegen, Bewertungen und/oder Rezensionen zu geben bzw. zu schreiben?

____

37. Was sind die Gründe dafür, dass Sie es bisher nicht gemacht haben?

Wahl mehrerer Antworten ist möglich

- ☐ Nie dazu gekommen
- ☐ Die jeweilige Option zum Mitteilen war zu kompliziert
- ☐ Die Anwendung bot dazu keine Option
- ☐ Ich glaube nicht, dass das hilft
- ☐ Der Aufwand war zu hoch
- ☐ Sonstiges: ________________________________

38. **Haben Sie Apps/Anwendungen bewertet, weil Sie dazu aufgefordert wurden, oder auch von sich heraus?**

    ☐ Wurde in der App/Anwendung aufgefordert
    ☐ Habe von mir aus eine Bewertung abgegeben
    ☐ Sonstiges: ______________________________

39. **Wären Sie bereit, vom Entwickler kontaktiert zu werden und ggf. weitere benötigte Informationen zu vermitteln um die entsprechenden Fehler/Probleme zu beheben und somit das Produkt zu verbessern?**

    ☐ Ja
    ☐ Nein

40. **Wie ist Ihre derzeitige Laune?**

Eher negativ					**Eher positiv**
	☐	☐	☐	☐	☐
	1	2	3	4	5

41. **Würden Sie sich über Anerkennung wegen Ihrem Feedback freuen?**
    Ähnlich zu eBay Bewertungen oder Top Rezensenten bei Amazon

    ☐ Ja
    ☐ Nein

42. **Wenn es ein einfach zu bedienendes Mittel dafür gäbe, würden Sie dieses zum Aufzeichnen Ihrer Schritte verwenden?**
    Als unterstützendes Mittel

    ☐ Ja
    ☐ Nein

43. **Kommentare zu diesem Abschnitt**

____

Feedback am PC

44. Bei welchen Programmen haben Sie schon einmal Feedback gegeben?

____

45. Gab es hierbei Probleme oder würden Sie einiges verbessern?

____

46. Wurde das Problem behoben im Nachhinein?

- ☐ Ja
- ☐ Nein

Feedback auf mobilen Geräten

47. Bei welchen Apps haben Sie schon einmal Feedback gegeben?

____

48. Gab es hierbei Probleme oder würden Sie einiges verbessern?

____

49. Wurde das Problem behoben im Nachhinein?

- ☐ Ja
- ☐ Nein

50. Falls Sie schonmal eine App mit Sternen bewertet haben, war dies aus eigenem Interesse oder weil Sie dazu aufgefordert wurden?

- ☐ Eigenes Interesse
- ☐ Wurde aufgefordert

51. Falls Sie aufgefordert wurden/werden, wie bewerten Sie?

- ☐ Wie ich auch ohne Aufforderung bewertet hätte
- ☐ Willkürlich
- ☐ Schlechteste Bewertung, weil das Auffordern störend ist
- ☐ Höchste Bewertung, weil das Auffordern störend ist

52. Kommentare zu diesem Abschnitt

____

Appendix C

Post-session Questionnaire for Users

1. Ich konnte die mir gestellten Aufgaben problemlos lösen

Stimme eher zu					Stimme eher nicht zu
	☐	☐	☐	☐	☐
	1	2	3	4	5

2. Wie ist Ihre derzeitige Laune?

____

Eher negativ					Eher positiv
	☐	☐	☐	☐	☐
	1	2	3	4	5

3. Gab es der Benutzung des Programm mehr oder weniger Probleme als bei der Handhabung von Programmen/Apps in Ihrem Alltag?

Bitte wählen Sie nur eins

- ☐ Weniger
- ☐ Mehr
- ☐ Etwa genauso

4. Was hat Sie am meisten gestört?

____

____

5. Welche Probleme konnten Sie insgesamt beobachten?

____

____

____

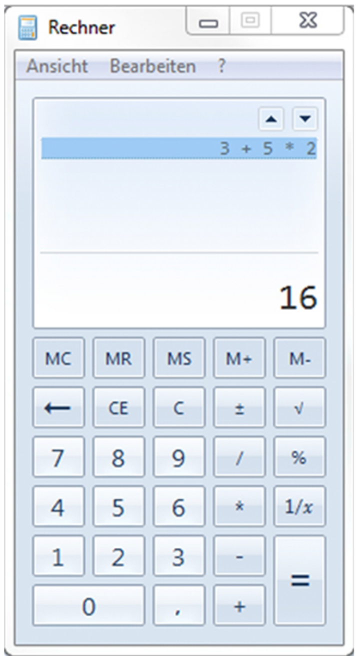
6. Gab es einige Funktionen, die nicht nötig waren oder haben Sie einige Funktionen vermisst?

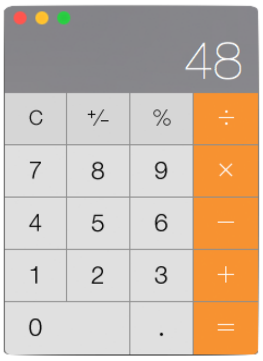
____

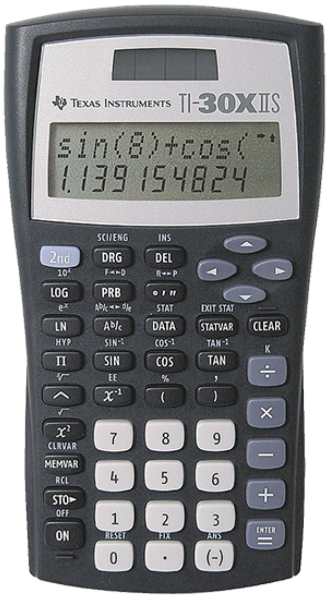
____

Beispiele für Taschenrechner

7. Hier sehen Sie nun einige Beispiele für andere Taschenrechner als Unterstützung. Falls Sie nun als Feedback weitere Funktionen wünschen würden (zusätzlich zu Ihrer vorhergehenden Antwort), tragen Sie diese bitte unten ein







Appendix D

Feedback Form for Users

Feedback

1. Wie lautet der Produkt/App Name?

____

2. Was ist die benutzte Version?

____

3. Wie würden Sie den Schweregrad einschätzen?

- ☐ Niedrig
- ☐ Mittel
- ☐ Hoch

4. Geben Sie eine kurze Zusammenfassung des Fehlers an

____

____

____

5. Welches Verhalten wurde beobachtet?

____

____

6. Welches Verhalten hatten Sie erwartet?

____

____

7. Falls eine Fehlermeldung kam, was stand drin?

____

____

8. Geben Sie an, welche Schritte notwendig sind, damit der Fehler reproduziert werden kann:

Bibliography

Katie Bessiere, Irina Ceaparu, Jonathan Lazar, John Robinson, and Ben Shneiderman. Understanding computer user frustration: Measuring and modeling the disruption from poor designs. *Technical Reports from UMIACS*, 2003/01/21/ 2003. URL `http://drum.lib.umd.edu/handle/1903/1233`.

Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim. Extracting structural information from bug reports. In *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, MSR '08, pages 27–30, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-024-1. doi: 10.1145/1370750.1370757. URL `http://doi.acm.org/10.1145/1370750.1370757`.

Pamela Bhattacharya and Iulian Neamtiu. Bug-fix time prediction models: Can we do better? In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 207–210, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0574-7. doi: 10.1145/1985441.1985472. URL `http://doi.acm.org/10.1145/1985441.1985472`.

José C. Castillo, H. Rex Hartson, and Deborah Hix. Remote usability evaluation: Can users report their own critical incidents? In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, CHI '98, pages 253–254, New York, NY, USA, 1998. ACM. ISBN 1-58113-028-7. doi: 10.1145/286498.286736. URL `http://doi.acm.org/10.1145/286498.286736`.

K.K. Chaturvedi and V.B. Singh. Determining bug severity using machine learning techniques. In *Software En-*

gineering (CONSEG), 2012 CSI Sixth International Conference on, pages 1–6, 2012. doi: 10.1109/CONSEG.2012.6349519. URL `http://dx.doi.org/10.1109/CONSEG.2012.6349519`.

Parmit K. Chilana, Andrew J. Ko, Jacob O. Wobbrock, Tovi Grossman, and George Fitzmaurice. Post-deployment usability: A survey of current practices. pages 2243–2246, 2011. doi: 10.1145/1978942.1979270. URL `http://doi.acm.org/10.1145/1978942.1979270`.

Facebook White Hat. Facebook White Hat. `https://www.facebook.com/whitehat`. Accessed: 2016-01-16.

Gerhard Fischer. Beyond "couch potatoes": From consumers to designers and active contributors. *First Monday*, 7(12), 2002. ISSN 13960466. URL `http://firstmonday.org/ojs/index.php/fm/article/view/1010`.

G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication, November 1987. ISSN 0001-0782. URL `http://doi.acm.org/10.1145/32206.32212`.

Wilbert O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1997. ISBN 0-471-15755-4.

Gamasutra. Serious Play Conference 2011: Microsoft's 'Productivity Games'. `http://www.gamasutra.com/view/news/36824/Serious_Play_Conference_2011_Microsofts_Productivity_Games.php`. Accessed: 2016-01-16.

A. Ghose and P.G. Ipeirotis. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *Knowledge and Data Engineering, IEEE Transactions on*, 23(10):1498–1512, Oct 2011. ISSN 1041-4347. doi: 10.1109/TKDE.2010.188. URL `http://dx.doi.org/10.1109/TKDE.2010.188`.

Github Bounty. Github Security - Bug Bounty Program. `https://bounty.github.com/`. Accessed: 2016-01-16.

Google Alpha/Beta Program. Google Play Developer Help - Use alpha/beta testing & staged rollouts. `https://support.google.com/googleplay/android-developer/answer/3131213?hl=en`. Accessed: 2015-12-28.

Google VRP. Google Vulnerability Reward Program (VRP) Rules. `https://www.google.com/about/appsecurity/reward-program/`. Accessed: 2016-01-16.

Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy. Characterizing and predicting which bugs get fixed: An empirical study of microsoft windows. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 495–504, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-719-6. doi: 10.1145/1806799.1806871. URL `http://doi.acm.org/10.1145/1806799.1806871`.

H. Rex Hartson and José C. Castillo. Remote evaluation for post-deployment usability improvement. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '98, pages 22–29, New York, NY, USA, 1998. ACM. doi: 10.1145/948496.948499. URL `http://doi.acm.org/10.1145/948496.948499`.

Israel Herraiz, Daniel M. German, Jesus M. Gonzalez-Barahona, and Gregorio Robles. Towards a simplification of the bug report form in eclipse. pages 145–148, 2008. doi: 10.1145/1370750.1370786. URL `http://doi.acm.org/10.1145/1370750.1370786`.

Andrew J. Ko, Brad A. Myers, and Duen Horng Chau. A linguistic analysis of how people describe software problems. In *Proceedings of the Visual Languages and Human-Centric Computing*, VLHCC '06, pages 127–134, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2586-5. doi: 10.1109/VLHCC.2006.3. URL `http://dx.doi.org/10.1109/VLHCC.2006.3`.

A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals. Predicting the severity of a reported bug. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 1–10, May 2010. doi: 10.1109/MSR.2010.

5463284. URL http://dx.doi.org/10.1109/MSR.2010.5463284.

Jonathan Lazar, Katie Bessiere, Irina Ceaparu, John Robinson, Ben Shneiderman, and User Frustration Lazar. Help! i'm lost: User frustration in web navigation. *IT & Society*, 1:18–26, 2003.

Rafael Lotufo, Leonardo Passos, and Krzysztof Czarnecki. Towards improving bug tracking systems with game mechanisms. pages 2–11, 2012. URL http://dl.acm.org/citation.cfm?id=2664446.2664448.

Mozilla. Firefox input, 2015 (accessed December 12, 2015). URL https://input.mozilla.org/de.

Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *Proceedings of the International Workshop on Principles of Software Evolution*, IWPSE '02, pages 76–85, New York, NY, USA, 2002. ACM. ISBN 1-58113-545-9. doi: 10.1145/512035.512055. URL http://doi.acm.org/10.1145/512035.512055.

NetMarketShare. Browser market share, 2015 (accessed December 14, 2015). URL http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0&qpsp=191&qpnp=13&qptimeframe=M.

David M. Nichols, Dana McKay, and Michael B. Twidale. Participatory usability: Supporting proactive users. pages 63–68, 2003. doi: 10.1145/2331829.2331841. URL http://doi.acm.org/10.1145/2331829.2331841.

Prashanth U. Nyer. An investigation into whether complaining can cause increased consumer satisfaction. *Journal of Consumer Marketing*, 17(1):9–19, 2000. doi: 10.1108/07363760010309500. URL http://dx.doi.org/10.1108/07363760010309500.

Smart Insights. Mobile Marketing Statistics compilation. http://www.smartinsights.com/mobile-marketing/

`mobile-marketing-analytics/`
`mobile-marketing-statistics/`. Accessed: 2016-02-23.

Hongyu Zhang, Liang Gong, and Steve Versteeg. Predicting bug-fixing time: An empirical study of commercial software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1042–1051, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3. URL `http://dl.acm.org/citation.cfm?id=2486788.2486931`.

Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schroter, and Cathrin Weiss. What makes a good bug report? *IEEE Trans. Softw. Eng.*, 36(5):618–643, September 2010. ISSN 0098-5589. doi: 10.1109/TSE.2010.63. URL `http://dx.doi.org/10.1109/TSE.2010.63`.

Index

Typeset July 14, 2016