What Do Mobile App Users Complain About?

Hammad Khalid, Emad Shihab, Meiyappan Nagappan, *Member, IEEE* and Ahmed E. Hassan *Member, IEEE*

***Abstract*—The quality of mobile apps is becoming an increasingly important issue. These apps are generally delivered through app stores that allow users to post reviews about apps. These user-reviews provide a rich data source that can be leveraged to understand user-reported issues. In this study, we qualitatively study 6,390 low rated user-reviews for 20 free iOS apps. Our study uncovers 12 types of user complaints. We find that functional errors, feature requests and app crashes are the most frequent complaints. Complaints about privacy and ethical issues, and hidden app costs have the most negative impact on the rating of an app. We also find that users attributed their complaint to a recent update of the app in 11% of the reviews. Our study provides developers insight into the user-reported issues of iOS apps, along with their frequency and impact, which can help developers better prioritize their limited Quality Assurance resources.**

***Index Terms*—Mobile applications, Software quality, User reviews, Quality assurance**

I. Introduction

Mobile applications (apps) continue to grow in popularity at a rapid pace. The Apple iOS (mobile operating system) App Store alone contained more than 900,000 apps (as of June 2013) and remains one of the most competitive app markets [1]. This competition, and the growth of apps as observed thus far in critical domains such as e-commerce, government and the health care industry has made the quality of apps an increasingly important issue.

The majority of recent work on the quality of apps has focused on quality issues from the perspective of developers (e.g., [2]). However, one of the first steps in understanding the issues that impact the quality of apps is to determine the challenges or issues that *users* face when using these apps.

iOS apps are distributed through the App Store which lets users review their downloaded apps. In addition to assigning a *star-rating* to iOS apps (all of which are aggregated and displayed at a version and app level basis), users can provide a *review-comment* to rationalize their star-rating. This data source captures a unique perspective about the perception of users regarding the apps. Such reviews, like product-reviews in online web stores, are highly correlated with download counts (i.e., purchases) and are a key measure of the app's success [3], [4]. A good understanding of these issues will help developers understand the concerns of users, avoid low-ratings, and better prioritize their Quality Assurance (QA) resources. Furthermore, such an understanding is crucial in guiding the Software Engineering community in tackling high-impact research problems in the fastest growing field of software development today.

In order to better understand the complaints of iOS users, we examine the low rating (1 and 2-star) reviews associated with 20 free[1] iOS apps. Through a manual analysis of a statistically representative sample of 6,390 iOS user-reviews, we arrive at the following findings:

- We identified 12 types of user complaints in iOS apps ranging from functional to privacy and ethical issues. Users attributed these complaints to a recent update of an app in 11% of the sampled reviews. This highlights the importance of regression testing in iOS apps.
- The most frequent complaints are about functional errors, feature requests and app crashes. Examining these complaints can help developers identify existing problems, and new features for their app.
- The most negatively-impacting complaints are related to privacy and ethical issues, hidden costs and disliked features that are degrading the end-user experience. Understanding these complaints is important as some complaints can be much more detrimental than others.

Based on our findings (i.e., frequency and impact of complaint types), developers can better anticipate possible complaints, and prioritize their limited QA resources on the complaints most important for them.

II. Background and Prior Work

User-reviews have become an important source of information about the perception of customers. In fact, Mudambi *et al.* [4] showed that user-ratings and reviews play a key role in the purchasing decision of products at the online retailer `Amazon`. Similarly, Kim *et al.* [5] interviewed 30 users who bought apps and found that ratings were a key determinant in the purchase of an app. Harman *et al.* [3] mined information from 30,000 BlackBerry apps and found that there is a strong correlation between an app's star-rating and its downloads.

Vasa *et al.* [6] and Hoon *et al.* [7] analyzed user-reviews of mobile apps and found that the depth of feedback, and the range of words is higher when the users give a low rating to an app – highlighting the usefulness of low star-reviews.

H. Khalid is with the School of Computing, Queen's Univ., Kingston, Canada.

E. Shihab is with the Department of Software Engineering, Rochester Institute of Technology, Rochester, NY, USA.

M. Nagappan and A. E. Hassan are with the School of Computing, Queen's Univ., Kingston, Canada.

[1]free-to-download; while these apps are labeled 'free' in the App Store, some of them require a fee for premium features.

TABLE I
STATISTICS OF THE STUDIED IOS APPS

	App Name	Category	Rating	Total Low Reviews	Sampled Reviews
High star-rating iOS apps (rating above 3.5)	Adobe Photoshop Express	Photo & Video	3.5	1,030	280
	CNN app	News	3.5	1,748	315
	ESPN Score center	Sports	3.5	2,630	335
	EverNote	Productivity	3.5	1,760	315
	Facebook	Social Networking	4	171,618	383
	Four Square	Social Networking	4	1,990	322
	MetalStorm: Wingman	Games	4.5	1,666	312
	Mint.com Personal Finance	Finance	4	1,975	322
	Netflix	Entertainment	3.5	13,403	373
	Yelp	Travel	3.5	2,239	328
Low star-rating iOS apps (rating below 3.5)	Epicurious Recipes & Shopping List	Lifestyle	3	940	273
	FarmVille by Zynga	Games	3	10,576	371
	Find My iPhone	Utilities	3	846	264
	Gmail	Productivity	3	1,650	312
	Hulu Plus	Entertainment	2	4,122	351
	Kindle	Books	3	3,188	343
	Last.fm	Music	3	1,418	302
	Weight Watchers Mobile	Health & Fitness	3	1,437	303
	Wikipedia Mobile	Reference	3	1,538	308
	Word Lens	Travel	2.5	1,009	278

Other researchers have also performed manual analysis to highlight critical information for developers. Thung *et al.* [8] manually categorized the bugs that occur in Machine-learning systems. Similarly, Tian *et al.* [9] performed manual analysis on the content of Software Engineering microblogs to better understand what developers microblog about.

Our work complements the prior work since we manually analyze user-reviews to identify the most frequent and impactful complaints that lead to low ratings.

III. STUDY DESIGN

Users tend to write reviews when they are either extremely satisfied or extremely dissatisfied with a product [10]. The low star-reviews have a greater impact on the sales than high star-reviews since buyers are more likely to react to low ratings and complaints [11]. Therefore, in order to understand why users give low ratings to iOS apps, we focus our study on 1 and 2-star reviews. We explore two sources of information in these reviews: 1) the star-ratings and 2) the free-form review-comments associated with each review. We manually tag review-comments in order to uncover common complaints across iOS apps. Such manual tagging is time consuming, therefore we focus on a subset of apps (20) and tag a statistically representative sample of their reviews (6,390 reviews across the 20 apps). Figure 1 highlights the design of our study. In the following subsections, we describe each step in detail.

A. Selecting the Apps

We pick the 20 most popular iOS apps, as defined by the iOS Market during June 2012 (see Table I), which are free to download. We make sure that the selected apps have at least 750 reviews so that a few users do not skew the tagged reviews that we analyze. We also ensure that half of the selected apps have an overall high star-rating (3.5 stars or better) and that the other half of the apps have an overall low star-rating (below 3.5 stars), since we want to

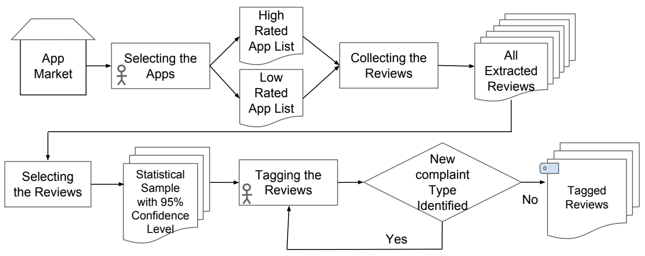


Fig. 1. Overview of our study process

identify the complaints in both high and low rated apps. We end up with 20 apps that cover 15 of the 23 categories (e.g., Productivity, Finance) in the iOS market, ensuring the breadth of the studied apps.

B. Collecting the Reviews

The main data source for our study are the reviews posted by iOS app users on `iTunes`. However, iTunes does not provide a public API for automatically retrieving user-reviews. Instead, we obtain the reviews from a web-service called `Appcomments`, which collects reviews of all iOS apps [12]. We build a web crawler which visits each unique page with a specific iOS review and parses the user-reviews to extract data such as the app name, the review title, the review-comment and the numerical star-rating assigned by the user. We collected all the reviews for each of the 20 studied apps during the first week of June 2012.

C. Selecting the Reviews

As we want to examine the complaints of iOS users, we focus on 1 and 2-star reviews since they are more likely to have user complaints. The studied apps have over 250,000 1 and 2-star reviews. Since manually examining all of these reviews is extremely time-consuming, we study a statistically representative sample for these reviews [13]. This sample of reviews is randomly chosen to achieve a 95% confidence level and a 5% confidence interval. This means that we are 95% confident that each of the

TABLE II
IDENTIFIED COMPLAINT TYPES WITH A DESCRIPTION AND AN EXAMPLE

Complaint Type	**Description**	**Example Review**
App Crashing	The app is often crashing	“Crashes immediately after starting.”
Compatibility	App has problems on a specific device or a OS version	“I can’t even see half of the app on my ipod touch...”
Feature Removal	Complaint about a disliked feature that is degrading the experience	”This app would be great, but get rid of the ads!!!”
Feature Request	App needs additional feature(s) to get a better rating	“No way to customize alerts. ”
Functional Error	An app specific problem was mentioned	“Not getting notifications unless u actually open the app..”
Hidden Cost	Complaint about the hidden costs for full experience	“Great if you weren’t forced to buy coins for REAL money...”
Interface Design	Complaint about the design, controls or visuals	“The design isn’t sleek and not very intuitive”
Network Problem	The app has trouble with the network or is slow to respond	“New version can never connect to server!”
Privacy and Ethical	The app invades privacy or is unethical	“Yet another app that thinks your contacts are fair game.”
Resource Heavy	The app consumes too much battery or memory	“Makes GPS stay on all the time. Kills my battery.”
Uninteresting Content	The specific content is unappealing	“It looks great but actual gameplay is boring and weak.”
Unresponsive App	The app is slow to respond to input, or laggy overall	“Bring back the old version. Scrolling lags.”
Not specific	A review-comment that’s not useful or doesn’t point out a problem	“Honestly the worst app ever.”

results is within a margin of error of ±5%. For example, ‘Adobe Photoshop Express’ has a total of 1,030 1 and 2-star reviews. The statistically representative sample for 1,030 reviews, with a 95% confidence level and a 5% confidence interval, is 280 reviews. Thus, we randomly select 280 reviews from the 1,030 1 and 2-star reviews for manual examination.

In total, we manually examine 6,390 reviews. We perform our sampling on a per app basis since different apps have varying number of reviews and we want to capture the complaints across the different apps. The number of randomly sampled reviews for each app ranges from 264 to 383 and is shown in the sixth column of Table I.

D. *Tagging the Reviews*

Once we determine the number of reviews to examine, we follow an iterative process called *Coding* as suggested by Seaman *et al.* to identify the different complaint types [14], [15]. The coding is used to turn qualitative information into quantitative data. The first author of the article read each review to determine the type of complaint mentioned in the review. We follow the procedure below for tagging the reviews:

```
Inputs = All reviews (each with a review-title
and a review-comment), a list of complaint types
(which is initially empty)

For each review:
    Manually examine all of the text in the review.

    If review matches an existing complaint type:
        Tag review with a complaint type(s).

    Else:
        Add a new complaint type to the list of
        complaint types.
        Restart tagging with new list of complaint
        types.

Outputs = All reviews (tagged with appropriate
complaint types), and a list of complaint types
```

This process is iterative such that each time a new complaint type is identified, we go through all the previously tagged reviews and see if they should be tagged using the new complaint type as well. This iterative process also helps us minimize the threat of human error while tagging the reviews. In total, we ended up having to restart the tagging process 3 times after discovering new complaint types. In certain cases, a user may provide no meaningful comment for their review (e.g., simply saying the app is bad). In such cases, we tag these types of reviews as being ‘Not Specific’. Some reviews may also contain multiple complaints; in these cases, we tag the review with multiple complaint types. For example, if a network problem is mentioned in a review that also contains a complaint about the app crashing, the review will be tagged with the ‘Network Problem’ and ‘App Crashing’ complaint types.

IV. RESULTS

Once we are done looking through all of the reviews, we end up with 12 different complaint types. Table II lists the different complaint types, provides a description for each type and gives an example review. We calculate the frequency and impact of each complaint type below.

A. *Frequency of each complaint type*

We calculate the frequency of the complaint types for each app. Once we have this frequency, we normalize it for each complaint type (i.e., number of complaints of a specific type divided by the total number of sampled reviews for an app), so that we can compare results across different apps with a varying number of reviews. Due to the high deviance of each complaint type between different apps, we use the median to summarize the frequency of each complaint type across all the studied apps.

Table III shows the rank and median percentage of the complaints in column two and three respectively. We see the ‘Functional Error’ complaints in 26.68% of the reviews, ‘Feature Request’ in 15.13%, and ‘App Crashing’ in 10.51%. Together, these three complaint types account for more than 50% of all complaints.

To better understand ‘Functional Error’, the most frequent complaint type, we examine the most frequently-used terms in these reviews. Then, we read through all the review-comments that use these most frequently-used terms. We find that 4.5% of functional errors are about *location* issues and 7.3% are about *authentication* problems. Below is an example of a functional error review where a user reported an authentication problem.

Don’t do the update!!! : when I try to login it just keeps refreshing the screen...

Examining ‘Feature Request’, the second most frequent complaint type, we find that most requests are very app specific. However, we do find that 6.12% of all feature requests by users are for better notification support in apps.

Overall, we find that ‘Network Problem’, ‘Interface Design’ and ‘Feature Removal’ complaints are also frequent. Another complaint that we identify is ‘Compatibility’ which is an important issue for iOS devices; this refers to a complaint where the app does not work correctly on a specific device or a version of the OS. Surprisingly, complaints about compatibility, resources and the responsiveness of an app are not as frequent – we expected more of such complaints.

In addition to measuring the frequency, we also examine whether the complaint types vary between high and low rated apps. To do so, we compare the frequency of each complaint type among the 10 high and the 10 low rated apps. We carry out this comparison using a two-tailed *Mann-Whitney U-Test* with $\alpha < 0.05$. We find that there is no statistically significant difference between high and low rated apps.

Our findings highlight the importance of software maintenance activities for iOS apps since many of the frequent complaints are directly related to developmental issues (e.g, ‘Functional Error’, ‘App Crashing’, ‘Network Problem’); many of the low star-reviews can be avoided by an increased focus on QA. In addition, our findings show that low star-reviews frequently contain information that can help developers identify the features which their users want, or really hate (e.g, ‘Feature Request’, ‘Feature removal’).

Functional Error, Feature Request and App Crashing are the most frequent complaints and account for more than 50% of the reported complaints. Many of the complaints in reviews can help app developers identify new features, and existing problems with their app.

B. Impact of each complaint type

Having identified the most common complaint types by analyzing 1 and 2-star reviews, we determine which of these complaints are the most negatively-perceived by users. We determine the most negatively-perceived complaints by looking at the ratio of 1 to 2-star ratings for each complaint type (across all apps). For example, a 1 to 2-star ratio of 5 for a complaint type indicates that this complaint type has 5 times as many 1-star ratings as 2-star ratings.

Columns 4 and 5 of Table III show the rank and the 1:2 star ratio for each complaint type. The most negatively-perceived complaints are different from the most frequent complaints. Observing Table III, we see that ‘Privacy and Ethical’, ‘Hidden Cost’ and ‘Feature Removal’ are the top three most negatively-perceived complaints – meaning that users are most bothered by issues related to the invasion of their privacy and unethical actions of the app developer (e.g., unethical business practices or selling the user’s personal data). Developers should only access the data (e.g., contacts of the user, or a user’s location) that they specified in the app’s description.

‘Hidden Cost’ is the second most negatively-perceived complaint that indicates the dissatisfaction of users with the hidden costs needed for the full experience of an app. This complaint showed up in 15 out of the 20 studied apps. While the apps we studied are called free apps, the term ‘free’ only refers to downloading the apps for free – and not necessarily using them for free. We find that when an app is free to download but not free to use, end-users are disappointed and often end up giving low rating reviews. This suggests that the trust between the developers and users is extremely important. For example, the ‘Hulu Plus’ app is free to download, but has a monthly subscription cost and ads in streaming videos. Because of the need for a monthly subscription, over 55% of the low star-reviews for Hulu were about the hidden costs. On closer examination, we find that the problem is that of a poor description of the app by the developer and/or a misunderstanding by the user.

Developers should devote extra attention to the ‘App Crashing’, ‘Hidden Cost’ and ‘Feature Removal’ complaints as they occur frequently *and* they are negatively-perceived by iOS users (see Table III).

Privacy and Ethical, Hidden Cost and Feature Removal complaints are the most impactful complaints and are mostly found in 1-star reviews. For developers, this finding stresses the importance of establishing trust and expectations with the app users.

V. Discussion

While reading through the complaints, we notice that for many of the complaints, users also report that they recently updated their app. Hence, we want to study what appears to be a relationship between updates and complaints. This can help developers prioritize regression testing for iOS apps. Then, we discuss the relevance of different types of complaints to the various stakeholders of a software project (e.g., developer vs. project managers).

A. Complaints related to app updates

It is important to mention that we can only know if a complaint is post-update if the user mentions it in the

TABLE III
THE MOST FREQUENT AND IMPACTFUL COMPLAINT TYPES (ALL RESULTS ARE AT 95% CONFIDENCE LEVEL)

	Most frequent		Most impactful	
Complaint Type	**Rank**	**Median (%)**	**Rank**	**1:2 star**
Functional Error	1	26.68	7	2.1
Feature Request	2	15.13	12	1.28
App Crashing	3	10.51	4	2.85
Network Problem	4	7.39	6	2.25
Interface Design	5	3.44	10	1.5
Feature Removal	6	2.73	3	4.23
Hidden Cost	7	1.54	2	5.63
Compatibility	8	1.39	5	2.44
Privacy and Ethical	9	1.19	1	8.56
Unresponsive App	10	0.73	11	1.4
Uninteresting Content	11	0.29	9	1.5
Resource Heavy	12	0.28	8	2
Not specific	-	13.28	-	3.8

review; other complaints could be because of an update as well.

We find that ∼11% of the sampled reviews mentioned that the recent update impaired existing functionality. In 22% of these reviews, the users mentioned 'Functional Error' complaints after updating their app. Most of these complaints are app specific. For example, in the review below a user is facing a bug that affects the function key of the 'Adobe Photoshop Express' app:

Useless now: Was very useful till last update... function keys no longer appear during editing

We also find that 18.8% of reviews after a reported update include requests from users for a new or a previously removed *feature*. We also found that 18.2% of post-update reviews complained about the frequent *crashing* of the app.

Developers often release free apps in hopes of eventually monetizing them by transforming free content/features to paid ones. We find that 6.8% of post-update complaint reviews report complaints about this *hidden cost*. Another important complaint that users report with recent updates is related to changes in the *interface design*. We find that 6.2% of post-update complaint reviews report complaints about the user interface.

Based on these findings, we recommend that developers pay special attention (e.g., via regression testing and user focus groups) to features that they might consider removing, to adding additional fees, and to user interfaces changes that they might plan to introduce in an update, since these seem to be some of the more common complaints of iOS app updates. Thus, even if a user previously liked an app, a bad update could be irritating enough to make them give the app a low rating.

In ∼11% of the sampled reviews, users attributed their complaints to an app update – highlighting the importance of regression testing in mobile development.

B. Identifying stakeholders of complaints

Since users review apps as a whole, they often raise issues that are not directly the responsibility of the developers; some complaints are directed towards product managers or other team members. To identify these stakeholders, we divide these complaints into three different categories: developer, strategic and content issues.

Developer issues are complaints that are directly related to developmental issues. These issues include 'Apps Crashing', 'Functional Error', 'Network Problem', 'Resource Heavy', and 'Unresponsive App' complaints and accounted for 45.6% of all complaints. Hence, many of the complaints are directly related to problems that developers can address.

Strategic issues are complaints that primarily concern project managers, but could partially target developers as well. These issues include 'Feature Removal', 'Feature Request', 'Interface Design' and 'Compatibility' complaints and makeup 22.7% of all complaints. Strategic issues require a greater knowledge about the project and priorities, and usually do not have a straightforward solution.

Content issues encompass complaints about the content or value of the app itself – developers have little or no control over these issues. These issues include 'Privacy and Ethical', 'Hidden Cost' and 'Uninteresting Content' complaints. Addressing these issues requires rethinking the core strategy of the app (i.e., business model or the content offered). While these issues account for only 3.02% of all complaints, 'Privacy and Ethical' and 'Hidden Cost' issues are the most negatively-impactful complaints.

VI. THREATS TO VALIDITY

External Validity: Our study was performed on a sample of 20 iOS apps. Hence, our results about complaints may not generalize to all iOS apps. To mitigate this threat we maximized the coverage of complaints by studying apps which cover most of the categories in the App Store.

Internal Validity: The first author of this article manually tagged the reviews. During this process, human error or subjectivity may have lead to incorrect tagging. This threat was addressed by random inspection of the reviews and the corresponding tags by the second and third authors of this article.

VII. CONCLUSION

Individual developers and organizations that develop iOS apps are strongly impacted by user-reviews since low ratings negatively reflect on the quality of their apps, and thus affect the app's popularity and eventually their revenues. To compete in an increasingly competitive market, app developers must understand and address the concerns of their users. In this study we identify 12 types of complaints and calculate the frequency and impact of each complaint type. *Our findings can help developers better anticipate the complaints and prioritize their limited QA resources towards the most impactful complaints.* At the same time, our findings point to new Software Engineering research avenues, such as the effect of ethics, privacy and user-perceived quality on mobile apps. In the future, we plan to expand on this study by considering more apps and comparing our findings across other mobile platforms.

References

[1] L. Stangel, "Apple statistics from wwdc 2013," June 2013. [Online]. Available: http://www.bizjournals.com/sanjose/news/2013/06/11/14-eye-popping-apple-statistics-from.html

[2] S. Agarwal, R. Mahajan, A. Zheng, and V. Bahl, "Diagnosing mobile applications in the wild," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 22.

[3] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR '12)*, Zurich, Switzerland, 2-3 June 2012.

[4] S. M. Mudambi and D. Schuff, "What makes a helpful online review? a study of customer reviews on amazon.com," *MIS Quarterly*, vol. 34, no. 1, pp. 185–200, 2010.

[5] H.-W. Kim, H. L. Lee, and J. E. Son, "An exploratory study on the determinants of smartphone app purchase," in *Proceedings of the 11th International DSI and the 16th APDSI Joint Meeting*, Taipei, Taiwan, July 2011.

[6] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference*. ACM, 2012, pp. 241–244.

[7] L. Hoon, R. Vasa, J.-G. Schneider, and K. Mouzakis, "A preliminary analysis of vocabulary in mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference*. ACM, 2012, pp. 245–248.

[8] F. Thung, S. Wang, D. Lo, and L. Jiang, "An empirical study of bugs in machine learning systems," in *Proceedings of the 23rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2012, pp. 271–280.

[9] Y. Tian, P. Achananuparp, I. N. Lubis, D. Lo, and E.-P. Lim, "What does software engineering community microblog about?" in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 247–250.

[10] N. Hu, P. A. Pavlou, and J. Zhang, "Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication," in *Proceedings of the 7th ACM conference on Electronic commerce*, ser. EC '06, 2006, pp. 324–330.

[11] J. A. Chevalier and D. Mayzlin, "The effect of word of mouth on sales: Online book reviews," *Journal of marketing research*, vol. 43, no. 3, pp. 345–354, 2006.

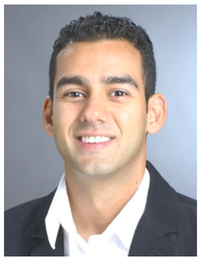
[12] appComments, "Read user reviews online and by rss," June 2012. [Online]. Available: http://appcomments.com/

[13] "Sample size calculator - creative research systems," February 2014. [Online]. Available: http://www.surveysystem.com/sscalc.htm

[14] C. B. Seaman, F. Shull, M. Regardie, D. Elbert, R. L. Feldmann, Y. Guo, and S. Godfrey, "Defect categorization: making use of a decade of widely varying historical data," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 2008, pp. 149–157.

[15] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *Software Engineering, IEEE Transactions on*, vol. 25, no. 4, pp. 557–572, 1999.

Hammad Khalid Hammad Khalid is a Masters student in the Software Analysis and Intelligence Lab (SAIL) at Queen's University, Canada. His research focuses on examining the link between user feedback, and the quality of software. Hammad is passionate about building better web and mobile applications. You can find out more about him at http://hammad.ca/.



Emad Shihab Dr. Emad Shihab is an Assistant Professor in the Department of Software Engineering at the Rochester Institute of Technology, New York, USA. His general research area is Software Engineering. He is particularly interested in Mining Software Repositories, Software Quality Assurance, Software Maintenance, Empirical Software Engineering and Software Architecture. He mines historical project data and applies Data Mining, Artificial Intelligence and Statistical Analysis techniques in order to build pragmatic solutions that practitioners can use to maximize their software quality with the least amount of resources. He worked as a software research intern at BlackBerry in Waterloo, Ontario and Microsoft Research in Redmond, Washington. He is the recipient of a number of prestigious awards including an NSERC Alexander Graham Bell Canada Graduate Scholarship (CGS D3). He served on numerous program committees of various software engineering conferences such as ICSM, MSR, ICPC, WCRE and served as a co-organizer of the MSR 2013 data challenge, the MSR 2014 data showcase and as program chair for the IWESEP 2013 workshop.

Meiyappan Nagappan Meiyappan Nagappan is a postdoctoral fellow in the Software Analysis and Intelligence Lab (SAIL) at Queen's University, Canada. His research is centered around the use of large-scale Software Engineering (SE) data to address the concerns of the various stakeholders (e.g., developers, operators, and managers). He received a PhD in computer science from North Carolina State University.

Dr. Nagappan has published in various top SE venues such as TSE, FSE, EMSE, and IEEE Software. He has also received a best paper award at the International Working Conference on Mining Software Repositories (MSR 12). He continues to collaborate with both industrial and academic researchers from the US, Canada, Japan, Germany, Chile, and India. You can find more about him at http://sailhome.cs.queensu.ca/ mei/.

Ahmed E. Hassan Ahmed E. Hassan is the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. Hassan also serves on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, and Springer Journal of Computing. Contact him at ahmed@cs.queensu.ca.