International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 03 | Mar-2018 www.irjet.net p-ISSN: 2395-0072

# Approaching Highlights and Security issues in Software Engineering under Mobile Application Development

**Mohan Sai[1], Muppa Sai Karthik[2], Prof. Prabu S[3], B V Pavan[4], Mahesh Reddy[5]**

[1,2,4,5] *UG Student, Dept. of Computer Science and Engineering, VIT, Tamil Nadu, India*
[3]*Professor, Dept. of Computer Science and Engineering, VIT, Tamil Nadu, India*

---

**Abstract -** *There has been huge development in the utilization of mobile devices throughout the most recent couple of years. This growth has led to millions of the software application in the mobile applications. As per the current estimate there are hundreds of thousands of mobile app developers. In this paper discuss about present and future research trends with various stages in the software development life-cycle: requirements like functional and non-functional, design and development, testing, and maintenance and the evolution of the software engineering in the mobile applications. And the various life cycle models are used for various method based on the risk, security issues and resources. While there are several non-functional requirements based on the mobile application in this paper we focus on resources and security. In this paper we discuss the security issues of the software engineering for the mobile application. And for each phase of the life cycle we discuss about recent advances and then challenges in the current work and the chances for the advances in the future in the Mobile applications.*

***Key Words*: SDLC, Mobile App, Software Process, requirements, security issues, functional, non-functional**

## 1. INTRODUCTION

In this paper, a mobile application is characterized as the application created for the present age of mobile devices prominently known as smart phones. These applications are regularly dispersed through a stage particular, and unified application showcase. In this paper, we in some cases allude to versatile applications just as applications. In the previous couple of years, we are watching a blast in the prominence of cell phones and mobile applications [11]. Actually, late market considers demonstrate that the brought together application advertise for Apple's stage (IOS) and Google's stage (Android), each have in excess of 1.5 million applications [21]. These mobile application markets are to a great degree famous among engineers due to the adaptability and income potential. In the meantime, mobile applications bring an entire slew of new difficulties to programming experts - for example, challenges due to the very associated nature of these devices, the one of a kind dissemination channels accessible for versatile applications like Google Play and novel income models like Premium and membership applications. Software security issues showed up in the most recent century when software began having more dynamic parts in many applications. Be that as it may, the procedure of software advancement did not go through a develop designing procedure and did not contain an unequivocal stage for security [29]. Over the time, the quantity of the security issues has expanded particularly with across the board of new applications, for example, interpersonal organization stages despite the fact that the software improvement process has adjusted an orderly building procedure such as Software Development Life Cycle (SDLC). Software Engineering research for mobile applications and to draw a vision for its future. Note that we confine to only the software designing themes for mobile applications in this paper, and even that not thoroughly because of space limitations (we skip themes like ease of use or operational efficiency since a whole paper can be composed on every one of these points). We don't examine the headways in different territories of research for mobile applications such as cloud based arrangements, or systems administration in mobile applications.

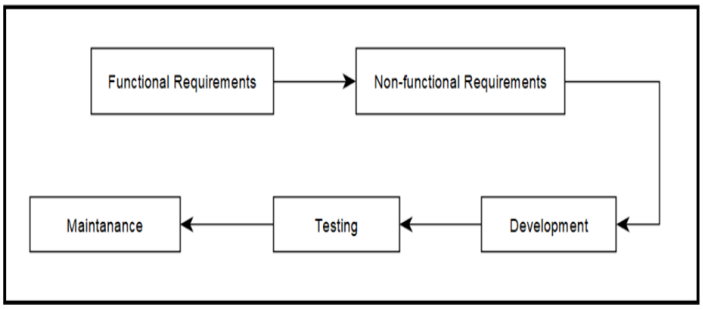


**Figure 1**: The system for exhibiting the software engineering in the mobile applications

The reason for this vision paper is to fill in as a kind of perspective point for mobile app work. We begin by giving a few foundation data on mobile apps. At that point, we examine the present best in class in the field, relating it to each of the software improvement stages, i.e., prerequisites, advancement, testing, and support as appeared in Figure 1. We likewise talk around two non-practical prerequisites: vitality utilize and security of mobile apps. At last, despite the fact that it isn't one of the software advancement stages, we discuss the software engineering difficulties and suggestions for adapting mobile apps. Alongside an exchange of the cutting edge, we additionally present the difficulties as of now looked by the analysts/designers of mobile apps. At that point we talk about our vision for the eventual fate of software engineering examination for mobile apps and the

dangers included, in light of our encounters. Our expectation is that our vision paper will help newcomers to rapidly pick up a foundation in the zone of mobile apps. In addition, we trust that our exchange of the vision for the territory will rouse and control future work and construct a group of scientists with shared objectives in regards to software building challenges for mobile apps. An expression of alert however - the dialog of the present cutting edge isn't intended to be an orderly writing overview and the future directions of research depend on our assessments that have been affected by our insight into the research in this community.

### 1.1 ORGANIZATION OF THE PAPER:

In this paper, in introduction (1) we have started with a basic idea of what the topic is about. Then, accordingly we have gone through the existing related works and among them, few are mentioned in the literature survey section (2). Continued with functional and non-functional (3,4) which play a major role in any development. Then, by following the traditional method, we have given a basic idea on the development (5) and the testing (6) phase of a development. Followed by the maintenance (7) of a project and the conclusion (8) part.

## 2. LITERATURE SURVEY:

The most present day emphasis of the mobile apps began in 2007, at the point when Apple reported the original of the iPhones. At a similar times Apple additionally reported the incorporated market for mobile apps called the 'App Store', through which, the end clients needed to download all their apps. Not long after in 2008, Google conveyed their own particular stage (Android) and their own app showcase the 'Android Market' (which was later renamed as 'Google Play'). Comparable app markets were discharged for the mobile telephone stages created by Microsoft, and BlackBerry also. With these other app markets, now the mobile app designers have a much bigger client base to pitch to. It is assessed that there are right now 2.6 Billion mobile telephone clients, who generally possess advanced cells [9]. An outline of the different partners in the realm of mobile apps is appeared in Figure 2. Mobile apps have been around for quite a while now. Back in the 1990s they were generally made by gadget makers like Nokia and Motorola. These apps played out certain fundamental undertakings.

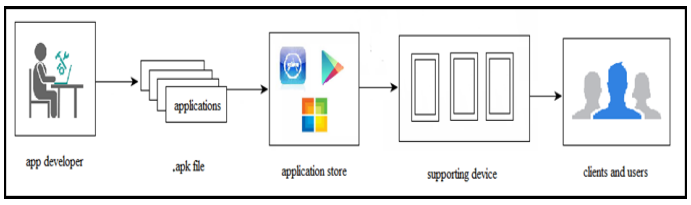


**Figure 2**: overview of the stakeholders in the modern mobile applications

Later on, remote specialist organizations began making apps to separate the devices sold on their system to others. At a similar time, outsider organizations began making apps for the mobile stages like the Windows mobile OS and the Symbian OS. These included recreations for the devices and other utility apps. In any case, there was no incorporated place where end clients could secure these apps. With the introduction of app markets for each platform, now developers have the ability to manage the distribution of their software through one centralized market for each platform. All designers of all shapes and sizes have the same app advertise, along these lines making it a notwithstanding playing field for anybody to succeed. Too, the app markets made it simple for the designers to transfer their apps, oversee updates to them, and push the most recent rendition flawlessly to the end clients. Subsequently a blend of market potential, convenience, and democratized stage, made it exceptionally lucrative for engineers to construct mobile apps.

With the expanded utilization of smartphones and mobile apps by end clients, and improvement of these mobile apps by software engineers, mobile apps turned into a conspicuous territory for software building specialists to analyze. One of the most punctual software building papers on such mobile apps was the investigation of miniaturized scale apps on the Android and BlackBerry stages by Syer et al. [13], and one of the most punctual examinations on the app markets was by Harman et al. [33]. From that point forward, there have been bounty of concentrates on a wide range of information that can be mined from the app markets, with the app themselves being only one kind of information. We think the expansion in such software designing investigations on mobile apps are a result of two reasons - (1) since the app markets are freely accessible, it is presently conceivable to mine the information generally effectively (albeit later in this segment we investigate where scientists confronted inconvenience in getting this open information), (2) a variety of new sorts of data that were past not accessible are currently accessible and dependably very much connected together. Some of these new types of data are discussed and the photo of play store is shown in figure 3. The app market also allows for the developer to post release notes on each of the app's versions. Analysts can mine this data to decide how the apps are advancing. Another snippet of data accessible in the app store for each app is the contact data for the designer. In this manner, presently analysts can contact app designers with anything intriguing that they find about the app. We are additionally ready to mine apps that are like the current app, and in this manner look at how comparative or diverse an app is from different apps. Knowing the similitude between apps is additionally encouraged in the app advertises by the class arrangement. Each app in the app store must be ordered in one of numerous predefined classes. Hence, now as analysts we approach apps that have been self-answered to be in a similar area. This gives scientist's huge potential to direct research that can be controlled for the area of the app. Frequently we see that a software building research

contemplate is done on an IDE, as Obscuration and another OSS venture like the program Firefox [1]. In any case, we don't realize what areas of applications that these outcomes exchange to. In the realm of mobile apps, on the off chance that we lead our exploration on just amusement apps, at that point we can be more certain that our discoveries would apply to other diversion apps.

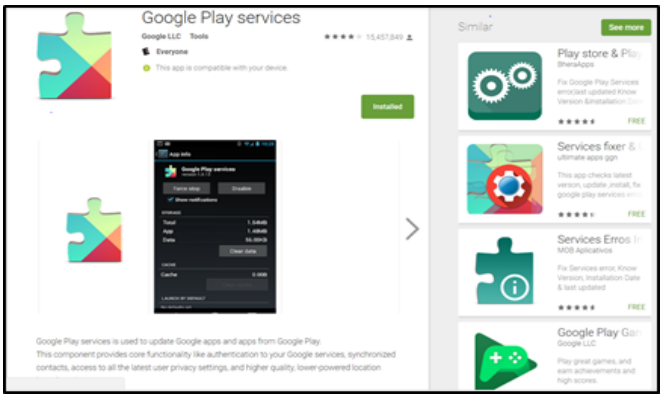


**Figure 3**: Snapshot of app in App Store

## 3. FUNCTIONAL REQUIREMENTS

### 3.1 Recent Advances:

Various examinations have concentrated on requirement extraction for mobile applications. In spite of customary work on software requirements, which mainly focused on investigation of the requirements and particulars archive, the lion's share of mobile app-related investigations utilized app audits posted by clients to extricate requirements. For instance, Iacob et al. [3] utilized semantic guidelines to identify highlight demands from client audits. At that point, they compress the element solicitations to produce more theoretical requirements. Galvis-Carreno and Winbladh [10] extricate subjects from client surveys to modify requirements. They demonstrate that their naturally separated requirements coordinate with physically removed requirements. Guzman and Maalej [4], [33] utilize common dialect preparing (NLP) strategies to distinguish app includes in the audits and utilize opinion examination to decide how clients feel about app highlights. They likewise contrast their extricated highlights with physically extricated highlights and find that the removed highlights are rational and applicable to requirements development tasks. Finally, numerous past examinations have taken a gander at the app surveys what's more, attempted to comprehend what protests that clients have about an app [23], [9]. In a past report, we physically investigated and labelled audits of iOS apps to distinguish the diverse issues that clients of IOS apps gripe about [15], [16]. We want to enable engineers to organize the issues that they should be testing.

### 3.2 Future Challenges:

The way that requirements are separated from app audits has its own particular difficulties. By and large and for some apps, there may not be sufficient client audits or the nature of the audits might be low. The greater part of the previously mentioned investigations requires a high amount and nature of client surveys. Chen et al. have done some underlying work in naturally distinguishing audits that are educational [13]. In any case, there is still more work left to be done here. For instance, regardless of whether there are top notch surveys accessible, we don't know whether we really got every one of the surveys from the app store [2]. Commonly app stores confine people in general to have the capacity to see just a subset of every one of the surveys. On account of Google Play, it is 500 audits. On account of the Windows Marketplace, they enable you to see the greatest number of as can be stacked in the page before the program crashes. Along these lines, we have an examining issue, which has been shown by Martin et al. [36]. One fascinating issue that has just been tended to by app markets like Google Play is the capacity for the designer to answer to client surveys when they have tended to a requirement. Another test is the applicability of the NLP methods used to extricate requirements from app surveys.

### 3.2 Security Issues:

One of the security issues associated with seeking after the above lines of look into is that we may have achieved the limit points of NLP when breaking down ineffectively composed client audits. Another hazard is that possibly clients incline toward the highlights that they are given previously they request it, and when the client whines about the highlights, at that point it is as of now past the point of no return. The main arrangement may be to fabricate a refreshed audit framework for the app stores that permits a superior instrument for include demands from the clients.

## 4. NON FUNCTIONAL REQUIREMENTS

### 4.1 Recent Advances:

One of the initially works identified with the estimation of resources of mobile applications is Green Miner by Hindle et al. [14] which is a committed equipment stage that empowers estimation of resources utilization of mobile devices. In other work, Hao et al. [17] propose a system that use program examination to give per-direction resources displaying. They demonstrate that their approach can evaluate resources utilization to inside 10% of the ground truth for Android apps. Liu et al. [23], introduce their apparatus Green Droid that will naturally distinguish the resources wastefulness bugs in Android apps. Thus Banerjee et al. [9] distinguish resources bugs in mobile apps. Different investigations performed experimental research on resources utilization to furnish engineers with methods for limiting it. For instance, Pathak et al. [20] proposed a scientific classification of resources bugs in light of in excess of 39,000 posts. They likewise propose a structure for the troubleshooting of resources bugs on smartphones. Li et al. [23] play out an exact investigation on 405 apps to better comprehend resources utilization.

### 4.2 Future Challenges:

A portion of the notable difficulties that face static analysis of software, apply to the security inquire about specified here as well. For instance, it is notable that most static analysis approaches experience the ill effects of a high rate of false positives. That issue in any case, might be less basic for mobile apps since they tend to be littler in measure. Other work relies upon information gave by the app engineers, for example, the app's portrayal. Such approaches can't ensure to perform well for applications that don't have all around reported portrayals. There are numerous bearings of future research that is conceivable around there. The most evident of this is to propel the state-of-the-workmanship in static analysis examine. Another more troublesome research issue is to get it why engineers are composing helpless code in the apps? How would we be able to enable them to avoid unexpectedly made security issues for end clients? This line of research expects us to get it step by step instructions to compose secure software first. At that point we should have the capacity to instruct the engineers. In the interim, would we be able to manufacture pointers to decide whether an app will probably have powerless code in it or not?

### 4.3 Security Issues:

A large portion of the work has been done on Android apps. This is for the most part because of the way that the Android stage is more open at that point different stages, e.g., iOS or BlackBerry. Additionally, the apps are written in Java for which there exists numerous decompiles and static analysis apparatuses. Playing out our examinations on Android causes a hazard regarding how applicable the proposed approaches would work for mobile apps from other platforms. Another hazard is in simply concentrating on receptive approaches to security with a specific end goal to tackle the present security issues and not concentrating on preventive arrangements. Concentrating on receptive approaches isn't only an issue with mobile apps yet with all software. Be that as it may, with mobile apps because of the speed at which they are developing, this issue could be significantly more intense - as we may never get up to speed.

## 5. DEVELOPMENT

### 5.1 Recent Advances:

One of the prior research papers in software engineering was by Syer et al. [17] who looked at the source code of Android and BlackBerry applications along three measurements, source code, code conditions and code beat. They find that BlackBerry apps are bigger and depend more on outsider libraries, though, Android apps have less records and depend intensely on the Android stage. Hecht et al. [38] proposed an instrument called Paprika to consider anti-patterns in mobile apps utilizing their byte code. Khalid et al. [26] analysed the relationship between notice from Find Bugs and app appraisals. They find that specific notices correspond with app evaluations. Cugola et al. [28] built up a revelatory dialect for a particular sort of mobile app. Around a similar time, Tillmann et al. created Touch Develop, a stage to construct mobile apps for the Windows Telephone [36]. This stage was worked to help amateur designers with next to zero involvement in either software engineering or software development to assemble apps. Moreover, Acerbis et al.built the Web Ratio Mobile Platform for display driven mobile app development [29].

### 5.2 Future Challenges:

With the popularity of all stages expanding previously barely any years, designers are enticed to build up the same app for numerous stages (cross-stage development). All together to empower this, there are a few structures that are accessible - Sencha, Phone Gap, and App celerator Titanium to name a maybe a couple (a portion of the cross-stage development structures like Cocos2d, Unity 3D, and Corona are particularly for recreations). The engineer needs to manufacture the app by just calling the APIs display in these structures, and at construct time, an app for each stage is produced by the structure. Be that as it may, all these systems, because of their plan have an unfavourable effect on both the execution of the app and its UI. Practically nothing investigate has been led to enable engineers to get it the expenses and advantages of the different approaches of creating cross-stage apps [99]. This issue in this way, furnishes specialists with a colossal chance to decidedly influence the designers. Concocting the following best cross-stage app development approach would be of high effect.

### 5.3 Security issues:

With the platforms advancing as quick as they are to keep up with the opposition, it might be exceptionally hard to assemble a static answer for cross-stage development - the arrangements must advance similarly as quick. Also, there are equipment and app advertise approach confuses that must be dealt with. Indeed, the investigation of the issues in cross-stage development, might be troublesome on the grounds that it might be hard to connect the apps over the app markets. In conclusion, at times, mobile app designers may muddle their apps, making the investigation of their development a challenge since one would need to manage the confusion of the code before having the capacity to think about the app.

## 6. TESTING

### 6.1 Recent Advances:

An extensive variety of studies have created methods to help mobile app engineers enhance the testing of mobile applications, specifically by endeavoring to enhance UI and framework testing scope. Hu et al. propose the Monkey device, which robotizes the GUI testing of Android apps [32]. Monkey produces arbitrary occasions, instruments the apps and breaks down follows that are created from the apps to identify blunders. Another apparatus proposed by Machiry

et al. [14] is Dynodroid, which is an apparatus that progressively creates contributions to test Android apps. As opposed to Monkey, Dynodroid empowers the testing of UI and framework occasions. Because of this distinction, the creators demonstrated that Dynodroid can accomplish 52% higher test scope contrasted with Monkey. Mahmood et al. [35] introduced the Evo Droid device, which consolidates program analysis and transformative calculations to test Android apps. The creators demonstrate that Evo Droid can outflank Monkey and Dynordoid, accomplishing scope esteems in the scope of 70-80%. Linares-Vasquez et al. [36] propose Monkey Lab, which mines recorded executions to manage the testing of Android mobile apps. While all these approaches are universally useful test age approaches, Kim et al. [37] take a gander at execution testing of mobile apps at the unit test level.

### 6.2 Future Challenges:

One of the biggest challenges that analysts look in their ebb and flow line of research on mechanized tests for mobile apps is that they are not ready to accomplish high code scope [38]. This is halfway a result of the powerlessness to create a wide range and assortment of data sources and mostly due to apps that are intended for client input (like diversion apps or apps that require a login), which can't be naturally produced. Frequently the mechanized testing devices can't continue down a certain execution way because of the powerlessness to create inputs, and 26 in this way can't test anything further along that execution way. In this manner explore in producing a more extensive scope of information that can emulate a human could greatly affect robotized mobile app testing instruments. Another test is that regularly specialists construct apparatuses that will take a shot at the app parallel since that is the main thing to which they approach. The accessibility of more OSS apps could yield in more vigorous instruments. One store of OSS apps is the F-Droid vault [20]. Be that as it may, from past research we know that lone a little level of these apps are really effective apps in the app showcase [37]. An archive of OSS apps with the relating app pairs made accessible as a benchmark suite could enormously help analysts in progressing the best in class in app testing. We might likewise want to point out that accessibility of effective OSS apps would propel the best in class in every aspect of software engineering for mobile apps.

### 6.3 Security issues:

One of the biggest security issues in seeking after the above line of research is that researches might not approach all the different devices as well as stages. Moreover, there is no simple method to distinguish cross stage apps from the app stores. Up until now, there has been no push to manufacture such a database of cross stage apps that researches can examine.

## 7. MAINTENANCE

### 7.1 Recent Advances:

The area of software maintenance is a standout amongst the most inquired about regions in Software Engineering. Notwithstanding, due to the way that mobile apps is a youthful subarea inside SE, the upkeep of mobile applications stays to be to a great extent unfamiliar. In addition, since mobile apps are extraordinary, the examines identified with the upkeep of mobile apps tend to center around issues that have not been generally examined in past software support ponders. For instance, generally mobile apps show ads, and as has been appeared in earlier thinks about, these promotions require a huge sum of upkeep [7]. So, various earlier investigations explored the support of mobile apps from various points of view, e.g., code ruse and promotion related support Mojica-Ruiz et al. [22] analysed the degree of code reuse in the distinctive classes of Android applications. They find that approximately 24% of the classes acquire from a base class in the Android API and 28% of the classes acquire from a space particular base class. Besides, they locate that 215 mobile apps are totally reused by another mobile app. Syer et al. [14] looks at mobile apps to bigger "customary" software frameworks as far as size and time to settle deserts. They locate that mobile apps look like Unix utilities. A different profession inspected Android-related bug reports. Bhattacharya et al. [9] consider 24 mobile Android apps all together to comprehend the bug-settling process. They locate that mobile bug reports are of high calibre, particularly for security related bugs. Martie et al. [27] broke down points in the Android stage bugs so as to reveal the most faced off regarding points after some time. Thus, Liu et al. [9] recognized and portrayed execution bugs among Android apps.

### 7.2 Future Challenges:

A portion of the difficulties in support inquire about for mobile apps is that regularly there is an absence of recorded information. The software support explore group has significantly profited from straightforwardly accessible ancient rarities like source control and bug vaults of OSS projects. They now have a vast trove of information to assess their speculations on. Such a help has impelled an expanded level of research in software support as prove by the quantity of research productions on it. In any case, generally there are relatively few OSS mobile apps as talked about in the past area. The greater part of the current inquire about depends on the information accessible in the app markets. Hence, with constrained fine grained confer level data it is hard to direct support research. Finally, as specified in Section IX, there are a few organizations that gather operational information from mobile apps that have been introduced on a huge number of devices. The majority of these organizations furnish the app designers with the information and a few simple analyses on them. There is a wide assortment of dependability and execution issues that can be comprehended by building apparatuses and

approaches that mine such operational information (past work has scarcely touched the most superficial layer of such an issue by taking a gander at the server side of mobile applications than the customer side [25])

**7.3 Security issues:**

From past research we have seen that mobile apps are little, and have speedy discharge cycles. With such quick discharge, the reality of the situation may prove that the support exertion may cover a ton with the advancement exertion. Thus, it may not be anything but difficult to distinguish costs relating to upkeep. Furthermore, the assortment of apps is significantly more than the assortment of effective work area applications. For instance, a little formula app like All-the-Cooks [37], and a substantial application like Microsoft Exceed expectations [38] are similarly prevalent, yet they may have totally diverse upkeep endeavors. In this manner the issue of setting the brings about the correct setting winds up vital. In this manner, it is much prescribed to monitor the app area when leading support contextual analyses.

## 8. CONCLUSION

Finally, we conclude stating - In the present day life, smartphones are playing a major role in each and every moment in an individuals' life and still we have chance to develop a lot of new integrated applications, fascinating exploration openings still left to be explored, and a lively group being worked around it, software engineering research for mobile apps is an extraordinary place for youthful developers to start with. It can be concluded as a great platform for one to choose this development.

## REFERENCES

[1] Tse-Hsun Chen, Stephen W. Thomas, Meiyappan Nagappan, and Ahmed E. Hassan. Explaining software defects using topic models. In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories

[2] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. The app sampling problem for app store mining. In Proceedings of the 12th Working Conference on Mining Software Repositories.

[3] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In Proceedings of the 10th Working Conference on Mining Software Repositories, San Francisco, CA, USA, May 18-19, 2013, pages 41–44, 2013.

[4] Emitza Guzman and Walid Maalej. How do users like this feature? A fine grained sentiment analysis of app reviews. In IEEE 22nd International Requirements Engineering Conference, Karlskrona, Sweden, August 25- 29, 2014, pages 153–162, 2014.

[5] Berg Insight. The mobile application market. Online: http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf, Last accessed Oct 2013.

[6] Ingrid Lunden. 6.1b smartphone users globally by 2020, overtaking basic fixed phone subscriptions. Online: http://techcrunch.com/2015/06/02/6- 1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phonesubscriptions/, Last accessed Oct 2015.

[7] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empirical Software Engineering, page to Appear, 2016.

[8] Mark D. Syer, Bram Adams, Ying Zou, and Ahmed E. Hassan. Exploring the development of micro-apps: A case study on the blackberry and Android platforms. In Proceedings of the 2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation

[9] Laura V. Galvis Carreno and Kristina Winbladh. Analysis of user com-˜ments: an approach for software requirements evolution. In Proceedings of the 2013 International Conference on Software Engineering

[10] Dennis Pagano and Wiem Maalej. User feedback in the appstore: An empirical study. In 21st IEEE International Requirements Engineering Conference, pages 125–134, 2013

[11] Hammad Khalid. On identifying user complaints of ios apps. In Proceedings of the 2013 International Conference on Software Engineering.

[12] Mark D. Syer, Bram Adams, Ying Zou, and Ahmed E. Hassan. Exploring the development of micro-apps: A case study on the blackberry and 31 Android platforms. In Proceedings of the 2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation.

[13] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E. Hassan. What do mobile app users complain about? a study on free ios apps. IEEE Software, 2014.

[14] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In Proceedings of the 36th International Conference on Software Engineering.

[15] Abram Hindle, Alex Wilson, Kent Rasmussen, E. Jed Barlow, Joshua Charles Campbell, and Stephen Romansky. GreenMiner: A hardware based mining software repositories software energy consumption framework. In Proceedings of the 11th Working Conference on Mining Software Repositories

[16] Number of apps available in leading app stores as of july 2015. Online: http://www.statista.com/statistics/276623/number-of-apps-availablein-leading-app-stores/, Last accessed Oct 2015.

[17] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, pages 5:1–5:6, 2011.

[18] Ding Li, Shuai Hao, Jiaping Gui, and William G.J. Halfond. An empirical study of the energy consumption of Android applications. In Proceedings of the 30th International Conference on Software Maintenance and Evolution, September 2014.

[19] Abhijeet Banerjee, Lee Kee Chong, Sudipta Chattopadhyay, and Abhik Roychoudhury. Detecting energy bugs and hotspots in mobile apps. In Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[20] Geoffrey Hecht, Benomar Omar, Romain Rouvoy, Naouel Moha, and Laurence Duchien. Tracking the Software Quality of Android Applications along their Evolution. In 30th IEEE/ACM International Conference on Automated Software Engineering, Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering, page 12. IEEE, November 2015.

[21] Hammad Khalid, Meiyappan Nagappan, and Ahmed E. Hassan. Examining the relationship between findbugs warnings and end user ratings: A case study on 10,000 Android apps. Software, IEEE, page to appear, 2015.

[22] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1276–1284, 2013.

[23] Gianpaolo Cugola, Carlo Ghezzi, Leandro Sales Pinto, and Giordano Tamburrelli. Selfmotion: a declarative language for adaptive serviceoriented mobile apps. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, page 7, 2012

[24] Stuart McIlroy, Nasir Ali, and Ahmed E. Hassan. Fresh apps: an empirical study of frequently-updated mobile apps in the google play store. Empirical Software Engineering, page to Appear, 2016.

[25] Nikolai Tillmann, Michal Moskal, Jonathan de Halleux, Manuel Fahndrich, and Sebastian Burckhardt. Touchdevelop: App development on mobile devices. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering.

[26] Aravind Machiry, Rohan Tahiliani, and Mayur Naik. Dynodroid: An input generation system for Android apps. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013, pages 224–234, 2013.

[27] App Annie, howpublished = Online: https://www.appannie.com/, year = Last accessed Oct 2015, source = https://www.appannie.com/.

[28] Riyadh Mahmood, Nariman Mirzaei, and Sam Malek. EvoDroid: Segmented evolutionary testing of Android apps. In Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[29] Heejin Kim, Byoungju Choi, and W. Eric Wong. Performance testing of mobile applications at the unit test level. In IEEE International Conference on Secure Software Integration and Reliability Improvement.

[30] Mark Harman, Yue Jia, and Yuanyuan Zhang Test. 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

[31] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In 23rd IEEE International Requirements Engineering Conference, pages 116–125, Aug 2015.

[32] F-Droid, howpublished = Online: https://f-droid.org/, year = Last accessed Oct 2015, source = https://f-droid.org/.

[33] Allthecooks recipies. Online: https://play.google.com/store/apps/details?id=com.mufumbo.android.recipe.search&hl=en, October Last accessed Oct 2015.

[34] Microsoft excel. Online: https://play.google.com/store/apps/details?id=com.microsoft.office.excel&hl=en, October Last accessed Oct 2015

[35] Abram Hindle. Green mining: A methodology of relating software change to power consumption. In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, pages 78–87, 2012.

[36] Mario Linares-Vasquez, Martin White, Carlos Bernal-C´ ardenas, Kevin ´ Moran, and Denys Poshyvanyk. Mining Android app usages for generating actionable GUI-based execution scenarios. In Proceedings of the 12th Working Conference on Mining Software Repositories, MSR '15, pages 111–122, 2015.

[37] Allthecooks recipies. Online: https://play.google.com/store/apps/details?id=com.mufumbo.android.recipe.search&hl=en, October Last accessed Oct 2015.

[38] Microsoft excel. Online: https://play.google.com/store/apps/details?id=com.microsoft.office.excel&hl=en, October Last accessed Oct 2015.