IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

An Empirical Study on Software Engineering in Mobile Applications and Future Research Directions

R.Balamurugan[1*], M.Ravichandran[2]

[1] Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore, India
[2] Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore, India

[*]*Corresponding Author: bala2marudur@gmail.com, Tel.: 8610082911*

Available online at: www.ijcseonline.org

Accepted: 10/Jan/2019, Published: 31/Jan/2019

Abstract— Nowadays, mobile devices have reached its popularity in greater heights, specifically the usage of smart phones has extended its features in communication technology with rapid evolution. With regards to this, the developers are always passionate about providing the smart ways and approaches through the Mobile App for the common users so that they have smart lifestyle. To provide the smart apps which works on smart devices, the diversity is there in the usages of tools and technologies. In addition to hardware rapid evolution, mobile applications are also increasing in their complexity and performance to cover most the needs of their users. Both software and hardware design focused on increasing performance and the working hours of a mobile device. Different mobile operating systems are being used today with different platforms and different market shares. Like all information systems, mobile systems are vulnerable to several issues. In this paper survey on software engineering paradigm in mobile applications are discussed by analyzing various existing approaches in the field of mobile software testing, mobile software quality assurance and mobile application security threats.

Keywords—: Mobile Application Software, Malware detection, Code Metric, Maintenance, Quality assurance

I. Introduction

Mobile applications become strategic business tools and have changed the business landscape. They have also expanded to meet the communication and information sharing needs of social networks. Nowadays, there are variety of new scenarios and applications for mobile devices. Indeed, users and developers expect reliability, ease of use, performance and security. Unfortunately, the quality of mobile applications are lower than expected. While mobile applications are becoming so extraordinarily adopted, it is still unclear if they deserve any specific testing approach for their verification and validation. This paper wants to investigate new research directions on mobile applications testing automation and its strategy, mobile applications software quality, mobile application security threats

Software testing of mobile applications is not a trivial task due to several factors such as the variety of inputs that a mobile application normally requires and the heterogeneity of the technologies used to implement them. The variety of inputs that a mobile application normally requires (such as user input, input context and environment) makes it very difficult to design appropriate test cases to detect faults. Several studies state that the importance in the context which mobile applications are executed to ensure quality [7, 8, 9]. Performing testing on multiple platforms is needed in order to ensure quality.

This paper provides an overview of important software engineering research issues related to the development of applications that run on mobile devices. Among the topics are development processes, tools, user interface design, application portability, quality, and security. While application development for mobile devices get back at least 10 years, there has been exponential growth in mobile application development since the iPhone AppStore opened in July, 2008. Since then, device makers have created outlets for other mobile devices, including Android, BlackBerry, Nokia Ovi, Windows Phone, and more. Industry analysts estimate that there are more than 250,000 applications are available through the various stores and marketplaces, some of them are available for multiple types of devices. The work have recently conducted a small survey of mobile developers [1], using available mobile developer forums to solicit respondents. A key goal of the survey was to gain a better understanding of development practices for mobile applications. The conclusions are included the following points:1) Most of the applications were relatively small, averaging several thousand lines of source code, with one or two developers responsible for conceiving, designing, and implementing the application.

2) There was a sharp divide between “native” applications, those that run entirely on the mobile device, and web applications, which have a small device-based client with execution occurring on a remote server.

3) Developers adhered quite well to recommended sets of “best practices” but rarely used any formal development processes.

4) Developers did very little organized tracking of their development efforts and gathered few metrics.

II. LITERATURE SURVEY

MobiGUITAR (Mobile GUI Testing Framework) [1] provided automated GUI-driven testing of Android apps. It's based on observation, extraction, and abstraction of GUI widgets' run-time state. The abstraction is a scalable state machine model that, together with test coverage criteria, provides a way to automatically generate test cases.

Uskov [2] focused on main MSE topics in Mobile Computing curriculum. Research of MSE-focused programming methodologies, analysis models in MSE, design and development models in MSE, including architectural models, information models, functional models, interaction models, navigation models, graphic user interface (GUI) hierarchical models, analysis of integrated development environments (IDEs) for various mobile platforms (Android, Windows Phone, etc.), testing strategies and techniques for mobile software systems, mobile software quality management, security issues of mobile software systems, and MSE-focused implementation methods.

Kai Qian et al [3] presented mobile device-based hands-on lab ware (series of lab modules) for computer network and security learning, which is guided by authentic learning principles to immerse students in a real-world relevant learning environment. By using this lab ware in teaching computer network, mobile security, and wireless network classes. The student feedback shows that students can learn more effectively when they have hands-on authentic learning experiences.

Li et al [4] developed a feature vector is extracted from the Android Manifest file, which combines the permission information and the component information of the Android application. Combined with the Naive Bayes classification algorithm, this approach proposes a malicious application detection method based on the Android Manifest file information. This approach is a static method of malware detection which means that applications are not executed or analyzed at run time for behavioral analysis. So it cannot detect any new malware which are capable of repackaging and obfuscation to bypass their inner mechanisms.

Fereidooni et al [5] introduced a system to detect malicious Android applications through statically analyzing applications’ behaviors. ANASTASIA provides a complete coverage of security behaviors. It utilizes large number of statically extracted features from various security behavioral characteristics of an application. This approach is a static method of malware detection so it cannot protect the device from Zero Day attacks and malwares capable of modifying themselves.

Akhuseyinoglu et al [6] uses an automated feature-based static analysis method to detect malicious mobile applications on Android devices. This method uses metadata of applications and Naïve Bayes algorithm for malware detection. This approach is a static method of malware detection so it cannot protect the device from malwares that can transform themselves based on the ability to translate, edit and rewriting their own code. The work focuses on observing the behavior of the malicious software while it is actually running on a host system. The dynamic behaviors of an application are conducted by the system call sequences at the end. Hence, they observe the system call log of each application, use the same for the construction of this dataset, and finally use this dataset to classify an unknown application as malicious or benign.

Asmau Usman et al [10] proposed an approach for testing mobile apps considering the two sets of events: GUI events which identified through static analysis of bytecode and context events obtained from analysis of manifest.xml file. Results from the experimental evaluation indicated that our approach is effective in identifying and testing context events.

Lianfa Li et al [11] the authors proposed two methods of data mining static code metrics to enhance defect-proneness prediction. Given little non-metric or qualitative information extracted from software codes, they first suggest to use a robust unsupervised learning method, Shared Nearest Neighbors (SNN) to extract the similarity patterns of the code metrics. These patterns indicate similar characteristics of the components of the same cluster that may result in introduction of similar defects. By using the similarity patterns with code metrics as predictors, defect-proneness prediction may be improved. The second method uses the Occam's windows and Bayesian model averaging to deal with model uncertainty: first, the datasets are used to train and cross-validate multiple learners and then highly qualified models are selected and integrated into a robust prediction.

Claire Le Goues et al [12] the authors proposed to augment a temporal-property miner by incorporating code quality metrics. They measure code quality by extracting additional information from the software engineering process and using information from code that is more likely to be correct, as well as code that is less likely to be correct. While used as a preprocessing step for an existing specification miner, their technique identifies which input is most indicative of correct program behavior, which allows off-the-shelf techniques to learn the same number of specifications using only forty five percent of their original input.

Hamand et al [13] attempt malware detection by inspecting other application run-time parameters, such as CPU usage, network transmission, and process and memory information. Mas'ud et al. [14] also included Android system calls in the detection strategy. Furthermore, Elish et al. [15] proposed a single-feature classification system based on user behavior profiling. DroidChain authors [16] proposed a novel model which analyze static and dynamic features of applications assuming different malware models.

Rahman et al. [17] investigated how effectively static code can be used to predict security risk of Android applications. Based on twenty one static code metrics of 1,407 Android applications, and using radial-based support vector machine (r-SVM). Syer et al. [18] has examined the relationship between files defect-proneness and platform dependence in Android apps. They found that source code files that are defect-prone have a higher dependence on the platform than defect-free files. The previous studies also investigated that the undesirable effects of Android apps low-quality source code.

Belal Amro [19], studied and analyzed different malware detection techniques used for mobile operating systems. He focused on two competing mobile operating systems – Android and iOS. He assets each technique by summarizing its advantages and disadvantages. The aim of the work is to establish a basis for developing a mobile malware detection tool based on user profiling.

Ana Rosario Espada et al [20] proposed the use of model-based testing to describe the potential behaviors of users interacting with mobile applications. These behaviors are modeled by composing specially-designed state machines. These composed state machines can be exhaustively explored by using a model checking tool to automatically generate all possible user interactions. Each generated trace model checker can be interpreted as a test case to drive a runtime analysis of actual applications. They have implemented a tool that follows the proposed methodology to analyze ANDROID devices by using the model checker SPIN as the exhaustive generator of test cases.

Ahmad et al [21] focused on analyzing and measuring the maintainability of Android mobile applications by using Object Oriented Metrics and Android Metrics. The purpose of this paper is to assess the impact all of these metrics on the maintainability of Android applications and choose the metrics that has the highest impact.

Pardeep Kumar et al [22] developed a combination of UML class diagram, use cases and activity diagram are used to discover changes at both syntax and semantic level. Additionally, agents developed in java agent development environment are also used to collect these changes from different stake holders in the distributed environments.

Tingting Yu et al [23] proposed a set of novel static code metrics based on the unique properties of concurrent programs. They also leverage additional guidance from dynamic metrics constructed based on the mutation analysis. Their evaluation on four large open source projects shows that ConPredictor improved both within-project defect prediction and cross-project defect prediction compared to the traditional features.

III. PROBLEM DEFINITION

- There is a lack of knowledge in automation of test cases and optimal software testing strategies for mobile applications. This may greatly affect the performance of the mobile software's and its functionals. The approaches to generate automated test cases and optimal testing strategies have to be defined and modeled.
- The quality of mobile applications is major cause of reliability and flexibility in design and construction of applications. The code metrics are not well-defined in context of mobile application-based software engineering. The metrics have to be evaluated for proper analyses of the quality and maintenance of mobile software's.
- The security threats are high in mobile applications due to its wide and open usage by several community applications. Which has high degree of accessibility in archiving personal information stored on mobile phone? There is no proper security mechanism to handle the malware attacks in mobile based applications thus an intelligent security prevention approach has to be developed.

IV. CONCLUSION

The paper analyzed detailed study on impact of mobile software quality analysis, mobile application testing and mobile software malware detection. Further investigation on these areas provides motivation to develop an optimal approach for developing automated test cases, determining the quality of mobile applications to enhance its performance. And planning in future to develop a prevention mechanism to overcome security threats which attacks the mobile based applications.

REFERENCES

[1] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta and A. M. Memon, "MobiGUITAR: Automated Model-Based Testing of Mobile Apps," in IEEE Software, vol. 32, no. 5, pp. 53-59, Sept.-Oct. 2015.

[2] L. Uskov, "Mobile software engineering in mobile computing curriculum," 2013 3rd Interdisciplinary Engineering Design Education Conference, Santa Clara, CA, 2013, pp. 93-99.

[3] Kai Qian ; Yong Shi ; Lixin Tao ; Ying Qian, Hands-On Learning for Computer Network Security with Mobile Devices, 2017 26th International Conference on Computer Communication and Networks (ICCCN)

[4] X. Li, J. Liu, Y. Huo, R. Zhang, Y. Yao, 'An Android malware detection method based on Android Manifest file', International Conference on Cloud Computing and Intelligence Systems (CCIS), 2016, pp. 239-243.

[5] H. Fereidooni, M. Conti, D. Yao, A. Sperduti, 'ANASTASIA: ANdroidmAlware detection using STaticanalySIs of Applications', 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2016, pp. 1-5.

[6] N. B. Akhuseyinoglu, K. Akhuseyinoglu, 'AntiWare: An automated Android malware detection tool based on machine learning approach and official market metadata', IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2016, pp. 1-7

[7] Domenico Amalfitano, Anna Rita Fasolino, Portfirio Tramontana, “A GUI Crawling-based technique for Android mobile application Testing” – IEEE (2011)

[8] Pallavi Raut, Satyaveer Tomar, “Android Mobile Automation Framework” – IJECS (2014).

[9] Anuja Jain, Swarnalatha P, M R. Ghalib, S. Prabhu, “Web-Based Automation Testing Framework” – IJCA (2012)

[10] Asmau Usman, Noraini Ibrahim, Ibrahim Anka Salihu, Test Case Generation from Android Mobile Applications Focusing on Context Events, ICSCA 2018, 7th International Conference on Software and Computer Applications, pages 25-30, 2018

[11] Lianfa Li , Hareton Leung , Mining Static Code Metrics for a Robust Prediction of Software Defect-Proneness, International Symposium on Empirical Software Engineering and Measurement, Sept. 2011

[12] Claire Le Goues and Westley Weimer, Measuring Code Quality to Improve Specification Mining, IEEE Transactions on Software Engineering, Vol. 38, No. 1,2012.

[13] H.-S.Hamand M.-J.Choi, “Analysis of Android malware detection performance using machine learning classifiers,” in Proceedings of the 2013 International Conference on Information and Communication Technology Convergence, ICTC 2013, pp. 490–495, October 2013.

[14] M. Z. Mas’ud, S. Sahib, M. F. Abdollah, S. R. Selamat, and R. Yusof, “Analysis of features selection andmachine learning classifier in android malware detection,” in Proceedings of the 5th International Conference on Information Science and Applications, ICISA ’14, pp. 1–5, IEEE, May 2014.

[15] K. O. Elish, X. Shu, D. D. Yao, B. G. Ryder, and X. Jiang, “Profiling user-trigger dependence for android malware detection,” Computers & Security, vol. 49, pp. 255–273, 2015.

[16] Z. Wang, C. Li, Z. Yuan, Y. Guan, and Y. Xue, “DroidChain: A novel Android malware detection method based on behavior chains,” Pervasive and Mobile Computing, vol. 32, pp. 3–14, 2016.

[17] Rahman, A., Pradhan, P., Partho, A., Williams, L.: Predicting Android application security and privacy risk with static code metrics. In: Proceedings of the 4th International Conference on Mobile Software Engineering and Systems, pp. 149–153. IEEE Press (2017).

[18] Syer, M.D., Nagappan, M., Adams, B., Hassan, A.E.: Studying the relationship between source code quality and mobile platform dependence. Software Quality Journal, 23(3), 485–508 (2015)

[19] Belal Amro, Malware Detection Techniques for Mobile Devices, International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol.7, No.4/5/6, December 2017.

[20] Ana Rosario Espada, Marıa del Mar Gallardo, Alberto Salmeron, Pedro Merin, Using Model Checking to Generate Test Cases for Android Applications, Tenth Workshop on Model-Based Testing (MBT 2015, EPTCS 180, 2015, pp. 7–21, 2017 8th International Conference on Information Technology (ICIT), IEEE, 2017

[21] Ahmad A. Saifan, Areej Al-Rabadi, Evaluating Maintainability of Android Applications, 8th International Conference on Information Technology (ICIT), 2017

[22] Pardeep Kumar, Arora Rajesh Bhatia, Agent-Based Regression Test Case Generation using Class Diagram, Use cases and Activity Diagram, Procedia Computer Science, Volume 125, Pages 747-753, 2018.

[23] Tingting Yu , Wei Wen, Xue Han, Jane Huffman Hayes Member, ConPredictor: Concurrency Defect Prediction in Real-World Applications , IEEE Transactions on Software Engineering , 2018.

Authors Profile

Mr.R.Balamurugan pursed Bachelor of Computer Technology from Bharathiar University, Coimbatore in 2001 and Master of Computer Applications from Bharathiar University in year 2004. He is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Bharathiar University, Coimbatore since 2005. He has published more than 02 research papers in reputed conferences. His main research work focuses on Software Enginnering, Mobile Computing, Distributed Computing. He has 14 years of teaching experience and 5 years of Research Experience.

Dr.M.Ravichandran Bachelor of Science from Madras University and Master of Science from Bharathiar University in year 1986. He is currently working as Associate Professor in Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Bharathiar University , since 1989. He has published more than 8 research papers in reputed international it’s also available online. His main research work focuses on Software Engineering,Data Mining and Big Data Analytics. He has 29 years of teaching experience and 18 years of Research Experience.