ENHANCED Q-LEARNING ALGORITHM FOR POTENTIAL ACTIONS SELECTION IN AUTOMATED GRAPHICAL USER INTERFACE TESTING

By

GOH KWANG YI

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Master of Science

July 2023

FSKTM 2023 7

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Master of Science

ENHANCED Q-LEARNING ALGORITHM FOR POTENTIAL ACTIONS SELECTION IN AUTOMATED GRAPHICAL USER INTERFACE TESTING

By

GOH KWANG YI

July 2023

Chairman : Associate Professor Salmi bt Baharom, PhD
Faculty : Computer Science and Information Technology

Researchers have proposed automated testing tools to minimise the effort and resources spent on testing GUIs. A relatively simple strategy employed by the proposed tools thus far is the observe-select-execute approach, where all of a GUI's actions on its current state are observed, one action is selected, and the selected action is executed on the software. The strategy's key function is to select an action that may achieve new and desirable GUI states. Due to difficulties in comparing actions, most existing test generators ignore this step and randomly select an action. However, a randomly selected action has limitations. It does not test most parts of a GUI within a reasonable amount of time and there is a high probability that the same actions are re-selected. This reduces code coverage, thereby resulting in undetected failures. To overcome this limitation, the Q-Learning algorithm was proposed by several researchers to minimise randomness. The idea was to change the probability distribution over the sequence space. Instead of making purely random selections, the least frequently executed action is selected so that the GUI can be further explored. Q-Learning improve the overall exploration strategy, but it also presented a weakness. Q-Learning's reward function assigns the highest value to the least frequently executed action without taking into consideration its potential ability in detecting failures. Furthermore, the proposed techniques based on the Q-Learning algorithm do not consider context-based actions. Thus, these techniques are unable to detect failures that occur due to the improper use of context data, which is becoming an increasingly common issue in mobile applications nowadays. We enhanced the Q-Learning algorithm for action selection based on potential action abilities and proposed a tool, namely CrashDroid, that allows the automation of testing context-aware Android applications. We utilized the enhanced Q-Learning algorithm to compare actions, including context-based actions, to effectively achieve higher code coverage. An experiment was carried out, and the results collected were analyzed using a non-parametric statistical test, the Mann-Whitney U test, which indicates the level of significance between the distributions of data collection. The experimental results showed that CrashDroid, which employs the technique considering the code complexity of the action and the use of context data during the test, was more effective than the other

automated Android testing tool, AutoDroid, which did not consider the mentioned metrics.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains

ALGORITMA PEMBELAJARAN Q DIPERTINGKATKAN UNTUK PEMILIHAN TINDAKAN BERPOTENSI DALAM PENGUJIAN ANTARA MUKA PENGGUNA GRAFIK AUTOMATIK

Oleh

GOH KWANG YI

Julai 2023

Pengerusi : Profesor Madya Salmi bt Baharom., PhD
Fakulti : Sains Komputer dan Teknologi Maklumat

Para penyelidik telah mencadangkan alat ujian automatik untuk mengurangkan usaha dan sumber yang dihabiskan dalam ujian antaramuka pengguna grafik. Strategi yang digunakan oleh alat-alat yang dicadangkan sejauh ini agak mudah iaitu pemerhatian-pilih-laksana, di mana semua tindakan antaramuka pengguna grafik pada keadaan semasa diambil perhatian, satu tindakan dipilih dan tindakan yang dipilih dilaksanakan pada perisian. Fungsi utama strategi ini adalah untuk memilih tindakan yang boleh mencapai keadaan antaramuka pengguna grafik yang baru dan diingini. Oleh kerana kesukaran dalam membandingkan tindakan, kebanyakan penghasil ujian sedia ada mengabaikan langkah ini dan secara rawak memilih satu tindakan. Walau bagaimanapun, tindakan yang dipilih secara rawak mempunyai kekurangan. Ia tidak menguji sebahagian besar bahagian antaramuka pengguna grafik dalam masa yang munasabah dan terdapat kemungkinan yang tinggi bahawa tindakan yang sama dipilih semula. Ini mengurangkan liputan kod, dengan itu mengakibatkan kegagalan yang tidak dikesan. Untuk mengatasi kekurangan ini, algoritma Q-Learning dicadangkan oleh beberapa penyelidik untuk mengurangkan kebarangkalian rawak. Idea adalah untuk mengubah taburan kebarangkalian atas ruang urutan. Sebaliknya daripada membuat pilihan secara rawak, tindakan yang jarang dilaksanakan dipilih agar antaramuka pengguna grafik dapat dijelajahi lebih lanjut. Q-Learning meningkatkan strategi penerokaan keseluruhan, tetapi juga mempunyai kelemahan. Fungsi ganjaran Q-Learning memberikan nilai tertinggi kepada tindakan yang jarang dilaksanakan tanpa mengambil kira kemampuannya yang berpotensi dalam mengesan kegagalan. Selain itu, teknik yang dicadangkan berdasarkan algoritma Q-Learning tidak mempertimbangkan tindakan berdasarkan konteks. Oleh itu, teknik ini tidak dapat mengesan kegagalan yang berlaku kerana penggunaan data konteks yang tidak betul, yang menjadi isu yang semakin biasa dalam aplikasi mudah alih pada masa kini. Kami meningkatkan algoritma Q-Learning untuk pemilihan tindakan berdasarkan kebolehan tindakan yang berpotensi dan mencadangkan satu alat, yang dinamakan CrashDroid, yang membolehkan automatik ujian aplikasi Android yang peka konteks dan menggunakan algoritma Q-Learning yang diperkukuhkan untuk

membandingkan tindakan, termasuk tindakan berasaskan konteks, untuk mencapai liputan kod yang lebih tinggi secara efektif. Satu eksperimen dijalankan, dan hasil yang dikumpulkan dianalisis menggunakan ujian statistik bukan parametrik, ujian Mann-Whitney U, yang menunjukkan tahap kepentingan di antara distribusi pengumpulan data. Hasil eksperimen menunjukkan bahawa CrashDroid, yang menggunakan teknik yang mempertimbangkan kompleksiti kod tindakan dan penggunaan data konteks semasa ujian, lebih efektif daripada alat ujian Android automatik lain, AutoDroid, yang tidak mempertimbangkan metrik yang disebutkan.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my three Supervisory Committee members, **ASSOCIATE PROFESSOR DR. SALMI BINTI BAHAROM**, **DR. JAMILAH BINTI DIN**, and **ASSOCIATE PROFESSOR TS. DR. NURFADHLINA BINTI MOHD SHAREF**, for their invaluable guidance, encouragement, and support throughout the entire research process. Their feedback, expertise, and commitment were instrumental to the completion of this thesis.

I would also like to thank the Faculty of Computer Science & Information Technology at University of Putra Malaysia for providing me with access to the necessary resources, facilities, and equipment. Their support was critical to the success of this research.

Furthermore, I am grateful to my colleagues and classmates for their valuable insights, helpful discussions, and support throughout my academic journey. Their contributions were invaluable to the completion of this work.

I would also like to express my appreciation to my family and friends for their unwavering love, encouragement, and support throughout my academic career. Their belief in me sustained me through the challenges of this thesis.

Finally, I would like to acknowledge the countless authors, researchers, and scholars whose work provided the foundation for this thesis. Their dedication to advancing knowledge in their respective fields is an inspiration to me and countless others.
Thank you all for your unwavering support and contributions to this thesis.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the Master of Science. The members of the Supervisory Committee were as follows:

Salmi binti Baharom, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Jamilah binti Din, PhD
Senior Letutrer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Nurfadhlina binti Mohd Sharef, PhD
Associate Professor Ts.
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ZALILAH MOHD SHARIFF, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 14 March 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GUI	Graphical User Interface
AUT	Application Under Test
XML	Extensible Markup Language
GA	Genetic Algorithm
REGEX	Regular Expression
OS	Operating System
App	Application
CPU	Central Processing Unit
UI	User Interface
VB	Visual Basic
APK	Android Package Kit
MDP	Markov Decision Processes
ADB	Android Debug Bridge

CHAPTER 1

INTRODUCTION

1.1 Overview

Smartphones have become a crucial part of our lifestyles. Mobile applications have transformed the way we perform daily activities, whether ordering food, booking a flight, paying bills, or chatting with friends. Considering the fact that 3.2 billion smartphones were sold, 8.3 billion mobile subscriptions were registered, more than 3.14 million applications were developed, and 204 billion applications were downloaded worldwide in 2019 (Statista.com, 2021b, 2021c), the significance of testing should not be neglected for quality assurance purposes.

Graphic user interface also known as GUI allows users to operate the application functions easily. The better the GUI, the easier the user interacts with the mobile application. Hence, mobile applications GUI is one of the important factors of mobile application success. Because of that, GUI testing often replaces system testing. Testing GUIs involves creating sequences of GUI events that exercise GUI widgets (i.e., test cases), executing those events (i.e., test execution) and monitoring resulting changes to the software state (i.e., test oracle) (Memon et al., 2003; Nguyen et al., 2014). Even though the creation of test cases is associated with GUI widgets, research has shown that GUI testing is efficient at finding both non-GUI and GUI errors (Robinson & Brooks, 2009). This is because the test cases not only execute GUI codes but also non-GUI codes. GUI testing can be used to identify security flaws, crashes and exceptions that occur while using mobile applications. All of these necessitate simulating user behaviors within the software and therefore automatic GUI testing needs to mimic human interaction with the GUI widgets. However, GUI testing is costly and time consuming, thus it will cause the overall development cost and development man days to increase. Therefore, researchers have proposed automated testing tools to minimize the effort and resources spent on testing GUIs. A lot of research had been done by the researchers which implemented various types of techniques like random technique, model based and capture & replay in their proposed automated testing tools. However, techniques like model based and capture & replay still require human intervention, so there are still many researchers who prefer random techniques (Ardito et al., 2020). Most of these proposed automated testing tools that implement random techniques use a relatively simple strategy which is the observe-select-execute approach. The strategy starts by launching the application under test (AUT) and then proceeds by observing the GUI actions on the AUT's current state, selecting an action from those observed actions, and executing the selected action. The strategy's key function is to select an action that may achieve new and desirable GUI states.

Due to difficulties in comparing actions, most existing tools ignore this strategy's key functions and randomly select an action. However, a randomly selected action has limitations due to most GUIs having numerous and deeply nested actions. It does not test most parts of a GUI within a reasonable amount of time and re-selection of the same actions is quite likely to occur. Several researchers (Bauersfeld & Vos, 2012; Buzdalov

& Buzdalova, 2013; Carino & Andrews, 2016; Koroglu et al., 2018; Mariani et al., 2012) have adopted Q-Learning algorithm in their automated testing tools to overcome these limitations. The behavior of action reward in Q-Learning further explores the GUI by selecting the least frequently executed action instead of making purely random selections. The prospect of discovery in such an approach is considered more "interesting" to a tester. However, these techniques select an action based solely on its execution frequency without considering its potential ability to detect and reveal failures. For example, let's compare tapping a button to submit data to a database and tapping a button to reset data within the interface. If both these tapping actions have never been executed, the probability of each action being selected would be equal if the selection is based solely on the execution frequency. However, the former button executes a complex code that might involve data transmission over the network and multiple servers. Hence, from a tester's point of view, the action has a more significant potential for bringing more interesting results than the latter.

Furthermore, these techniques do not consider context-aware applications, therefore they may not detect defects that occur due to the improper use of context data. This is ongoing research that aims to propose a testing tool that can automatically GUI test Android applications. The Android platform is selected as it is the most popular mobile operating system in the world. As of July 2017, the number of available applications available on Google Play Store is 2.95 billion (Nguyen et al., 2014). Its popularity among developers is owing to the accessible development environment that is based on the familiar Java programming language as well as the availability of open-source libraries implementing diverse functionalities that accelerate the development process.

1.2 Problem Statement

Due to the fact that, the increase of the popularity of mobile application (Statista.com, 2021b, 2021c), the GUI testing of mobile application getting important to maintain the application quality (Robinson & Brooks, 2009). However, GUI testing requires huge resources and cost, hence building a tool to automate the testing process is becoming the trend.

There are a lot of past research which implemented automated testing tools with various types of techniques such as random, model based and capture & replay. Among these three techniques, many researchers prefer random technique, because random technique requires lower user intervention compared to the other two techniques (Muangsiri & Takada, 2017). Other than that, the biasness of the tester when creating the model or recording the test script, which resulting in low significant defect-finding power (Ardito et al., 2020) is one of the reasons that researchers prefer random technique. But, due to the nature of random technique and most GUIs have numerous and deeply nested actions, automated testing tools that implement random technique do not test most parts of a GUI within a reasonable amount of time. Besides that, there is a high probability for re-selection of same actions to be occured with the automated testing tools which implement random technique.

Several researchers (Bauersfeld & Vos, 2012; Buzdalov & Buzdalova, 2013; Carino & Andrews, 2016; Koroglu et al., 2018; Mariani et al., 2012) have adopted Q-Learning algorithm in their automated testing tools and it show better results to improve the random exploration strategy. The core of using Q-Learning is to intelligently guide the action selection with the purpose to favor exploration of the GUI which reduces the redundant execution of events and increases coverage.

However, a common limitation to these techniques is that the reward function assigns the highest reward when the action is executed for the first time to maximize coverage or locate crashes. The selection of action is only based on the least executed action without taking into consideration that some actions are more potential than others with respect to testing (Adamo et al., 2018). For example, once completed a form to add a new event in a personal agenda, the application usually displays the entire calendar, producing a major change in terms of displayed widgets, and enabling many new actions to be tested. On the contrary, actions like filling text areas or clicking on combo boxes cause small changes of the GUI state and are thus less potential to be tested. Other than that, the existing implementations also do not take context aware applications into consideration. Therefore, there is a need to further investigate the adoption of Q-Learning in GUI testing to improve coverage and crash detection by selecting potential actions and consider context of the actions.

1.3 Research Objective

This research objectives that are addressed in this thesis are as follows:

1. To propose an action selection algorithm based on actions potentials abilities.
2. To implement the proposed action selection algorithm into an automated GUI testing tool.
3. To evaluate the effectiveness of the proposed action selection algorithm.

1.4 Research Scope

Several researchers (Bauersfeld & Vos, 2012; Buzdalov & Buzdalova, 2013; Carino & Andrews, 2016; Koroglu et al., 2018; Mariani et al., 2012) have adopted Q-Learning algorithm in their automated testing tools to overcome the limitation of the random algorithm. However, their techniques select an action based solely on its execution frequency without considering the code complexity. Besides that, these techniques also do not consider context-aware action, therefore they may not detect defects that occur due to the improper use of context data. Our research focuses on:

1. Improving the exploration strategy of the Q-Learning algorithm by proposing a technique to calculate the action weight based on the code complexity.
2. Propose a technique to consider context data when calculating the action weight, but only focus on WIFI and GPS.

3. Development an automated testing tool for android application.
4. Determine which code complexity metric to use to measure the action weight.
5. Include Jaccard Distance in the enhancement of algorithm.

1.5 Research Contributions

This research aimed to improve the exploration strategy of the Q-Learning algorithm based on the gaps left by previous work in this field. Based on the limitations and gaps, this research has made the following contributions:

- It provides an algorithm for selecting potential actions (Adamo et al., 2018).
- It implemented automated Android testing tools which consider the code complexity of the action and the use of context data.
- It provides empirical evidence based on the comparisons made between previous study by Adamo et al. (Adamo et al., 2018) and our implemented automated Android testing tools. The measurement uses the same AUTs and same action selection algorithm which is Q-Learning, but different approaches in selecting an action.
- The proposed action selection algorithm, implemented in an automated GUI testing tool, offers substantial benefits to software developers, quality assurance teams, and stakeholders involved in the mobile application development lifecycle. By enhancing testing efficiency and effectiveness, the contribution ensures that developers can release more robust and reliable mobile applications, QA teams can streamline their testing processes, and stakeholders can have increased confidence in the overall quality of the deployed applications, ultimately fostering a more successful and user-friendly mobile app ecosystem.

1.6 Thesis Organization

The thesis comprises seven chapters. The first chapter consists of the introduction of this research. Hence, the research problem, the objective and scope, as well as the contributions of this research are also described in the first chapter.

The second chapter presents the detailed study of the existing automated testing technique. The research background, which consisted of the issues that have been left out by the existing works, are discussed in detail in this chapter. The knowledge gaps left by previous works are also highlighted in this chapter.

The third chapter presents the methodology involved in this research. The research methods, materials or resources, and the deliverables obtained throughout the phases are explained in this chapter. Generally, this chapter highlights the three phases that have been previously defined, namely, the definition of the analysis and problems, the design

and development of the tool to support the proposed approach, and finally, the evaluation phase involved in producing the empirical result.

The fourth chapter describes the implementation of the testing tool, which involves the four criteria in calculating the action weight. This chapter discusses in detail how each criterion was processed during the tool execution phase.

The fifth chapter presents the experimental procedure and the results of the experiments. It provides statistical analysis using the Mann-Whitney U test.

The final chapter is Chapter Six, which consists of the conclusion and future works for this research. Some suggestions for future work that can be investigated by future researchers are explained in this chapter.

REFERENCES

Adamo, D., Khan, M. K., Koppula, S., & Bryce, R. (2018). Reinforcement learning for android GUI testing. *A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-Located with FSE 2018*, 2–8. https://doi.org/10.1145/3278186.3278187

Ahmad, J. (2018). *Multifactor Approach to Prioritize Event Sequence Test Cases*.

Al-Ahmad, B. I., Altaharwa, I., Alkhawaldeh, R. S., Alazzam, I. M., & Ghatasheh, N. A. (2021). Jacoco-coverage based statistical approach for ranking and selecting key classes in object-oriented software. *Journal of Engineering Science and Technology*, *16*(4), 3358–3386.

Aljahdali, S. H., Ghiduk, A. S., & El-Telbany, M. (2010). The limitations of genetic algorithms in software testing. *2010 ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2010, June*. https://doi.org/10.1109/AICCSA.2010.5586984

Amalfitano, D., Amatucci, N., Fasolino, A. R., Tramontana, P., Kowalczyk, E., & Memon, A. M. (2015). Exploiting the Saturation Effect in Automatic Random Testing of Android Applications. *Proceedings - 2nd ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft 2015*, 33–43. https://doi.org/10.1109/MobileSoft.2015.11

Amalfitano, D., Riccio, V., Amatucci, N., Simone, V. De, & Fasolino, A. R. (2019). Combining Automated GUI Exploration of Android apps with Capture and Replay through Machine Learning. *Information and Software Technology*, *105*(November 2017), 95–116. https://doi.org/10.1016/j.infsof.2018.08.007

Arcuri, A., & Briand, L. (2011). A practical guide for using statistical tests to assess randomized algorithms in software engineering. *Proceedings - International Conference on Software Engineering*, 1–10. https://doi.org/10.1145/1985793.1985795

Ardito, L., Coppola, R., Leonardi, S., Morisio, M., & Buy, U. (2020). Automated Test Selection for Android Apps Based on APK and Activity Classification. *IEEE Access*, *8*, 187648–187670. https://doi.org/10.1109/ACCESS.2020.3029735

Baek, Y. M., & Bae, D. H. (2016). Automated model-based android GUI testing using multi-level GUI comparison criteria. *ASE 2016 - Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, *December*, 238–249. https://doi.org/10.1145/2970276.2970313

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A Survey on context-aware systems. *Information Systems*, *2*. https://doi.org/10.1504/IJAHUC.2007.014070

Bauersfeld, S., & Vos, T. (2012). A Reinforcement Learning Approach to Automated GUI Robustness Testing. *In 4th Symposium on Search Based- Software Engineering (SSBSE2012)*, 7–12.

Butler, C. W., & McCabe, T. J. (2021). Cyclomatic Complexity-Based Encapsulation, Data Hiding, and Separation of Concerns. *Journal of Software Engineering and Applications*, *14*(01), 44–66. https://doi.org/10.4236/jsea.2021.141004

Buzdalov, M., & Buzdalova, A. (2013). Adaptive selection of helper-objectives for test case generation. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, 2245–2250. https://doi.org/10.1109/CEC.2013.6557836

Carino, S., & Andrews, J. H. (2016). Dynamically testing GUIs using ant colony optimization. *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 138–148. https://doi.org/10.1109/ASE.2015.70

Chang, J.-F. (2009). Algorithms And Particle Swarm Optimization Applied. *International Journal of Innovative Computing, Information and Control*, *5*(12 (B)), 5069–5079.

Chen, T. Y., Kuo, F. C., Merkel, R. G., & Tse, T. H. (2010). Adaptive Random Testing: The ART of test case diversity. *Journal of Systems and Software*, *83*(1), 60–66. https://doi.org/10.1016/j.jss.2009.02.022

Chhillar, U., & Bhasin, S. (2011). A New Weighted Composite Complexity Measure for Object-Oriented Systems. *International Journal of Information and Communication Technology Research*, *1*(3), 101–108. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.208.2952&rep=rep1&type=pdf

Costa, P., Paiva, A. C. R., & Nabuco, M. (2014). Pattern based GUI testing for mobile applications. *Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014*, 66–74. https://doi.org/10.1109/QUATIC.2014.16

Dayan, C. J. W. and P. (1992). *Q-learning, Machine learning 8 (1992)* (pp. 3–4, 279–292).

De Cleva Farto, G., & Endo, A. T. (2015). Evaluating the model-based testing approach in the context of mobile applications. *Electronic Notes in Theoretical Computer Science*, *314*, 3–21. https://doi.org/10.1016/j.entcs.2015.05.002

Dey, A. K., Abowd, G. D., & Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human–Computer Interaction*, *16*(2–4), 97–166. https://doi.org/10.1207/S15327051HCI16234_02

Esparcia-Alcázar, A. I., Almenar, F., Martínez, M., Rueda, U., & Vos, T. E. J. (n.d.). *Q-learning strategies for action selection in the TESTAR automated testing tool*. http://www.testar.org

Gomez, L., Neamtiu, I., Azim, T., & Millstein, T. (2013). RERAN: Timing- and touch-sensitive record and replay for Android. *Proceedings - International Conference on Software Engineering*, 72–81. https://doi.org/10.1109/ICSE.2013.6606553

Google. UI/Application Exerciser Monkey|Android Developers. (2021). https://developer.android.com/studio/test/ monkey

Google. (2021). *Android Overview*. https://developer.android.com/

Google Java Format. (2015). https://github.com/google/google-java-format

Gu, T., Cao, C., Liu, T., Sun, C., Deng, J., Ma, X., & Lü, J. (2017). AIMDROID: Activity-insulated multi-level automated testing for android applications. *Proceedings - 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017*, 103–114. https://doi.org/10.1109/ICSME.2017.72

Halpern, M., Zhu, Y., Peri, R., & Reddi, V. J. (2015). Mosaic: Cross-platform user-interaction record and replay for the fragmented android ecosystem. *ISPASS 2015 - IEEE International Symposium on Performance Analysis of Systems and Software*, 215–224. https://doi.org/10.1109/ISPASS.2015.7095807

Hu, C., & Neamtiu, I. (2011). Automating GUI testing for android applications. *Proceedings - International Conference on Software Engineering*, 77–83. https://doi.org/10.1145/1982595.1982612

Hu, Y., & Neamtiu, I. (2016). VALERA: An effective and efficient record-and-replay tool for android. *Proceedings - International Conference on Mobile Software Engineering and Systems, MOBILESoft 2016*, 285–286. https://doi.org/10.1145/2897073.2897712

Javaparser. (2008). https://javaparser.org/about.html

Joshi, S., & Orso, a. (2005). *Capture and Replay of User Executions to Improve Software Quality*. http://www.cc.gatech.edu/people/home/orso/papers/joshi.orso.TR06.pdf

Kaasila, J., Ferreira, D., Kostakos, V., & Ojala, T. (2012). *Testdroid*. 1. https://doi.org/10.1145/2406367.2406402

Kim, H. K. (2013). Hybrid model based testing for mobile applications. *International Journal of Software Engineering and Its Applications*, *7*(3), 223–238.

Koppula, & Sreedevi. (2017). *Automated GUI Tests Generation for Android Apps Using Q-Learning*. https://search-proquest-com.proxy.lib.uwaterloo.ca/docview/2008972348/?pq-origsite=primo

Koroglu, Y., & Sen, A. (2018). QBE : QLearning-Based Exploration of Android Applications. *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, 105–115. https://doi.org/10.1109/ICST.2018.00020

Koroglu, Y., Sen, A., Muslu, O., Mete, Y., Ulker, C., Tanriverdi, T., & Donmez, Y. (2018). QBE: QLearning-Based Exploration of Android Applications. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation, ICST 2018*, 105–115.

https://doi.org/10.1109/ICST.2018.00020

Lafi, M., Osman, M. S., & Wasmi, H. A. (2019). Improved Monkey Tool for Random Testing in Mobile Applications. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, 658–662. https://doi.org/10.1109/JEEIT.2019.8717506

Lam, W., Wu, Z., Li, D., Wang, W., Zheng, H., Luo, H., Yan, P., Deng, Y., & Xie, T. (2017). Record and replay for android: Are we there yet in industrial cases? *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *Part F1301*, 854–859. https://doi.org/10.1145/3106237.3117769

Lämsä, T., & Mäntylä, M. (2017). *Comparison of GUI testing tools for Android applications*. 105.

Leotta, M., Clerissi, D., Ricca, F., & Tonella, P. (2013). Capture-replay vs. programmable web testing: An empirical assessment during test case evolution. *Proceedings - Working Conference on Reverse Engineering, WCRE*, 272–281. https://doi.org/10.1109/WCRE.2013.6671302

Levenshtein, V. (1966). *Binary codes capable of correcting deletions, insertions, and reversals*. https://doi.org/10.1016/S0074-7742(08)60036-7

Li, Y., Yang, Z., Guo, Y., & Chen, X. (2017). DroidBot: A lightweight UI-guided test input generator for android. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, 23–26. https://doi.org/10.1109/ICSE-C.2017.8

Li, Y., Yang, Z., Guo, Y., & Chen, X. (2019). Humanoid: A deep learning-based approach to automated black-box android app testing. *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019*, 1070–1073. https://doi.org/10.1109/ASE.2019.00104

Lin, Y. D., Rojas, J. F., Chu, E. T. H., & Lai, Y. C. (2014). On the accuracy, efficiency, and reusability of automated test oracles for android devices. *IEEE Transactions on Software Engineering*, *40*(10), 957–970. https://doi.org/10.1109/TSE.2014.2331982

Liu, C. H., Lu, C. Y., Cheng, S. J., Chang, K. Y., Hsiao, Y. C., & Chu, W. M. (2014). Capture-replay testing for android applications. *Proceedings - 2014 International Symposium on Computer, Consumer and Control, IS3C 2014*, 1129–1132. https://doi.org/10.1109/IS3C.2014.293

Liu, Z., Gao, X., & Long, X. (2010). Adaptive random testing of mobile application. *ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings*, *2*, 297–301. https://doi.org/10.1109/ICCET.2010.5485442

Luna, J. M. (2021). Introduction to Data Mining. *Periodic Pattern Mining: Theory, Algorithms, and Applications*, 1–22. https://doi.org/10.1007/978-981-16-3964-7_1

Malik, Y. M. (2010). *Model Based Testing : An Evaluation*.

Mao, K., Harman, M., & Jia, Y. (2016). Sapienz: Multi-objective automated testing for android applications. *ISSTA 2016 - Proceedings of the 25th International Symposium on Software Testing and Analysis*, 94–105. https://doi.org/10.1145/2931037.2931054

Mariani, L., Pezzè, M., Riganelli, O., & Santoro, M. (2011). AutoBlackTest: A tool for automatic black-box testing. *Proceedings - International Conference on Software Engineering*, *January*, 1013–1015. https://doi.org/10.1145/1985793.1985979

Mariani, L., Pezzè, M., Riganelli, O., & Santoro, M. (2012). AutoBlackTest: Automatic black-box testing of interactive applications. *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, 81–90. https://doi.org/10.1109/ICST.2012.88

Martin L Puterman. (2014). Markov decision processes: discrete stochastic dynamic programming. In *John Wiley & Sons, 2014* (pp. 3–4, 279–292).

Memon, A., Banerjee, I., & Nagarajan, A. (2003). GUI ripping: Reverse engineering of graphical user interfaces for testing. *Proceedings - Working Conference on Reverse Engineering, WCRE*. https://doi.org/10.1109/WCRE.2003.1287256

Menninghaus, M., Wilke, F., Schleutker, J. P., & Pulvermüller, E. (2017). Search based GUI test generation in Java comparing code-based and EFG-based optimization goals. *ENASE 2017 - Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*, *Enase*, 179–186. https://doi.org/10.5220/0006277801790186

Muangsiri, W., & Takada, S. (2017). Random GUI testing of android application using behavioral model. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, 266–271. https://doi.org/10.18293/SEKE2017-099

Nguyen, B. N., Robbins, B., Banerjee, I., & Memon, A. (2014). GUITAR: An innovative tool for automated testing of GUI-driven software. *Automated Software Engineering*. https://doi.org/10.1007/s10515-013-0128-9

Nicola, A. (2016). *Automated GUI Testing Techniques For Android Applications*.

Prykhodko, S., & Prykhodko, N. (2022). A Technique for Detecting Software Quality Based on the Confidence and Prediction Intervals of Nonlinear Regression for RFC Metric. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, *2022-Novem*, 499–502. https://doi.org/10.1109/CSIT56902.2022.10000532

Prykhodko, S., Prykhodko, N., & Smykodub, T. (2022). A Joint Statistical Estimation of the RFC and CBO Metrics for Open-Source Applications Developed in Java. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, *2022-Novem*, 442–445. https://doi.org/10.1109/CSIT56902.2022.10000457

Rauf, A., Jaffar, A., & Shahid, A. A. (2011). Fully automated gui testing and coverage analysis using genetic algorithms. *International Journal of Innovative Computing, Information and Control*, *7*(6), 3281–3294.

Reddivari, S., & Raman, J. (2019). Software quality prediction: An investigation based on machine learning. *Proceedings - 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science, IRI 2019*, 115–122. https://doi.org/10.1109/IRI.2019.00030

Regular Expression. (2020). https://www.datakwery.com/techniques/regular-expressions/.

Richard Hamming. (1950). Error Detecting and Error Correcting Codes. *Journal of the Franklin Institute*. https://doi.org/10.1016/s0016-0032(23)90506-5

Riley, G., & Henderson, T. (2010). The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation, ISBN 978-3-642-12330-6. Springer-Verlag Berlin Heidelberg, 2010, p. 15* (pp. 15–34). https://doi.org/10.1007/978-3-642-12331-3_2

Robinson, B., & Brooks, P. (2009). An initial study of customer-reported GUI defects. *IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2009*, 267–274. https://doi.org/10.1109/ICSTW.2009.22

Run Apps on the Android Emulator. (2021). https://developer.android.com/studio/run/emulator

Sahin, O., Aliyeva, A., Mathavan, H., Coskun, A., & Egele, M. (2019). RANDR: Record and replay for android applications via targeted runtime instrumentation. *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019*, 128–138. https://doi.org/10.1109/ASE.2019.00022

Statista.com. (2021a). *gps-required-apps-share-by-category-worldwide*. https://www.statista.com/statistics/906644/gps-required-apps-share-by-category-worldwide/

Statista.com. (2021b). *Number of apps available in leading app stores as of 4th quarter 2020*.

Statista.com. (2021c). *Number of mobile app downloads worldwide from 2016 to 2020(in billions)*.

Statista.com. (2022). *share of mobile apps bycategory that require internet access*. https://www.statista.com/statistics/649070/share-of-mobile-apps-by-category-that-require-internet-access/

Su, T. (2016). FSMdroid: Guided GUI testing of android apps. *Proceedings - International Conference on Software Engineering*, *May 2016*, 689–691. https://doi.org/10.1145/2889160.2891043

Su, T., Meng, G., Chen, Y., Wu, K., Yang, W., Yao, Y., Pu, G., Liu, Y., & Su, Z. (2017). Guided, stochastic model-based GUI testing of android apps. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *Part F1301*, 245–256. https://doi.org/10.1145/3106237.3106298

Suram, S. (2019). *Android App Code Coverage with Manual + Appium automation tests*. https://www.linkedin.com/pulse/android-app-code-coverage-manual-appium-automation-tests-suram/

Suresh, Y., Pati, J., & Rath, S. K. (2012). Effectiveness of Software Metrics for Object-oriented System. *Procedia Technology*, *6*, 420–427. https://doi.org/10.1016/j.protcy.2012.10.050

Vaattovaara, M. (2019). *Performance of Model-Based Testing for an Android Application*. *October*.

Vuong, T., & Takada, S. (2019). Semantic analysis for deep Q-network in android GUI testing. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, *2019-July*, 123–128. https://doi.org/10.18293/SEKE2019-080

Wang, Y. (2008). Fuzzy Clustering Analysis By Using Genetic Algorithm. *ICIC Express Letters*, *2*(4), 331—337. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.510.4905&rep

Wijendra, D. R., & Hewagamage, K. P. (2021). Analysis of Cognitive Complexity with Cyclomatic Complexity Metric of Software. *International Journal of Computer Applications*, *174*(19), 14–19. https://doi.org/10.5120/ijca2021921066

Witte. (2016). *Statistics*.

Yan, J., Yan, J., Wu, T., & Zhang, J. (2017). Widget-sensitive and back-stack-aware GUI exploration for testing android apps. *Proceedings - 2017 IEEE International Conference on Software Quality, Reliability and Security, QRS 2017*, 42–53. https://doi.org/10.1109/QRS.2017.14

Yasin, H. N., Hamid, S. H. A., & Yusof, R. J. R. (2021). Droidbotx: Test case generation tool for android applications using q-learning. *Symmetry*, *13*(2), 1–30. https://doi.org/10.3390/sym13020310

Zhang, M., & Baddoo, N. (2007). *Performance Comparison of Software Complexity Metrics in an Open Source Project*. https://doi.org/10.1007/978-3-540-75381-0_15