EVOLUTIONARY COST-COGNIZANT REGRESSION TEST CASE PRIORITIZATION FOR OBJECT-ORIENTED PROGRAMS

ABDULKARIM BELLO

FSKTM 2019 6

EVOLUTIONARY COST-COGNIZANT REGRESSION TEST CASE PRIORITIZATION FOR OBJECT-ORIENTED PROGRAMS

By

ABDULKARIM BELLO

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirement for the Degree of Doctor of Philosophy

April 2019

DEDICATION

To my parents.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the Degree of Doctor of Philosophy

EVOLUTIONARY COST-COGNIZANT REGRESSION TEST CASE PRIORITIZATION FOR OBJECT-ORIENTED PROGRAMS

By

ABDULKARIM BELLO

April 2019

Chairman: Professor Abu Bakar Md Sultan, PhD
Faculty: Computer Science and Information Technology

Regression testing is conducted to ensure that changes made to a software satisfy the requirements and do not adversely introduce bugs to its existing functionalities. It involves the process of re-testing software after modifications. Ideally, to perform regression testing is to re-execute all the test cases on the modified software. Re-execution of all test cases can be expensive as there might be wasting resources, could be costly and time consuming. The three regression testing techniques are test cases selection (TCS), test suite minimization (TSM) and test cases prioritization (TCP). TCS attempts to identify test cases that have the same relevance to some set of changes. This technique has the problem of selecting a significant number of test cases even for small changes made to a software. TSM removes obsolete test cases from the test suite. The drawback in minimizing the test suite is it could reduce the quality of test suite.

To overcome the limitations of TCS and TSM, researchers proposed TCP to avoid test case discarding. TCP deals with the problem of test discarding and attempts to order test cases in an optimized order such that those with highest priority are executed earlier. One such criterion, is the rate of fault detection to measures how fast test cases revealed faults. Improved rate of faults detection can give developers chance to debug the faulty software earlier. To improve the rate of fault detection during several TCP approaches are proposed for regression testing. Reports from literature show that these approaches are associated with some limitations. Most of the approaches usually considered test costs and fault severity to be uniform. In practice test case cost and fault severity can vary, and in such cases the previous metric and approaches designed to improve fault detection of a prioritized test cases can produced an unsatisfactory result.

The recent trend of software development uses OO paradigm. Therefore, this study proposed a cost-cognizant TCP approach for object-oriented software that uses path-based integration testing to identify the possible execution path extracted from the Java System Dependence Graph (JSDG) model of the source code using forward slicing

technique. Afterward evolutionary algorithm (EA) was employed to prioritize test cases based on the rate severity of fault detection per unit test cost. The proposed technique is named Evolutionary Cost-Cognizant Regression Test Case Prioritization (ECRTP).

The experiment conducted on the proposed approach and the result obtained was empirically evaluated and compared with some existing approaches to determine its efficiency and effectiveness. The average percentage of fault detection per cost (APFDc) metric was employed to measure the average cost per severity detection. The analysis showed significant differences for both the effectiveness, efficiency and APFDc of the ECRTP over existing approaches such as RanPrio, RevPrio, NonPrio, JaNaMa and EvolRTP, which make ECRTP a promising approach to use for regression testing.

In the future, there is a need to extend the scope of this work by incorporating information from the latest regression testing, consider addition object-oriented metrics such as coupling and cohesion, and incorporate multi-objective evolutionary processes. There is also a need to consider implementing this strategy for dynamic object-oriented languages such as Python, Lisp, and Smalltalk.

Abstrak tesis yang dekemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk Ijazah Doktor Falsafah

KEUTAMAAN KES UJIAN REGRESI KESAN KOS EVOLUSIONER UNTUK PROGRAM BERORIENTASIKAN OBJEK

Oleh

ABDULKARIM BELLO

April 2019

Pengerusi : Professor Abu Bakar Md Sultan, PhD
Fakulti : Sains Komputer dan Teknologi Maklumat

Ujian regresi dijalankan untuk memastikan bahawa perubahan kepada perisian menjadikannya menepati keperluan dan mengelak peranti pepijat dari menjejaskan kefungsiannya yang sedia ada. Ia melibatkan proses menguji semula perisian selepas ia diubahsuai. Secara ideal, menjalankan ujian regresi bermakna menjalankan semula semua kes ujian ke atas perisian yang diubahsuai. Pengendalian semula semua kes ujian boleh menelan belanja yang besar oleh kerana pembaziran sumber boleh berlaku, dan ia juga memakan masa. Ketiga-tiga teknik ujian regresi adalah pilihan kes ujian (TCS), minimisasi suit ujian (TSM) dan keutamaan kes ujian (TCP). TCS cuba untuk mengenalpasti kes ujian yang sama kerelevanannya dengan beberapa set perubahan. Teknik ini mempunyai masalah memilih beberapa kes ujian walaupun untuk perubahan yang kecil kepada sesuatu perisian. TSM menyahkan kes ujian yang sudah luput dari suit ujian. Kelemahan dalam meminima suit ujian ialah ia dapat mengurangkan kualiti 0suit ujian.

Untuk mengatasi kekangan TCS dan TSM, para pengkaji mencadangkan TCP untuk mengelak kes ujian dari dibuang. TCP mengendalikan masalah pembuangan ujian dan cuba untuk menyusun kes ujian dalam susunan yang optima dalam keadaan di mana ujian yang mempunyai keutamaan tertinggi telah dilaksanakan lebih awal. Satu kriterion, adalah kadar pengesanan ralat untuk mengukur sejauh mana kes ujian pantas mendedahkan ralat. Kadar meningkat pengesanan ralat boleh memberi peluang kepada para pembangun untuk menyah-peranti pepijat perisian yang rosak.

Trend pembangunan perisian baru-baru ini menggunakan paradigma OO. Laporan dari literatur menunjukkan bahawa pendekatan-pendekatan ini dikaitkan dengan beberapa kekangan. Kebanyakan pendekatan mempertimbangkan kos ujian dan keseriusan kerosakan agar ia diseragamkan, tetapi hakikatnya ia berlainan di antara satu sama lain. Tambahan pula, dapat diperhatikan bahawa ada kerosakan yang berlaku sebagai hasil daripada kerosakan yang lain. Mengenalpasti dan mengatasi kerosakan yang mempunyai

keseriusan yang lebih tinggi pada peringkat awal memberi peluang kepada pembangun untuk menyah-peranti pepijat perisian dengan lebih cepat, dan dengan itu meningkatkan lagi masa penyampaian.

Tren terkini pembangunan perisian menggunakan paradigma OO. Maka itu, kajian ini mencadangkan satu pendekatan TCP yang celik-kos untuk perisian berorientasikan objek yang menggunakan ujian integrasi berasaskan laluan. Ujian integrasi ini akan mengenalpasti laluan pelaksanaan yang berkemungkinan dan mengestrak laluan-laluan ini dari model Java System Dependence Graph (JSDG) kod sumber menggunakan teknik potongan ke depan *forward slicing technique*. Algoritma berevolusi atau EA kemudiannya digunakan untuk mengutamakan kes ujian berdasarkan keseriusan pengesanan per kos unit untuk setiap satu kerosakan terlibat. Teknik yang disarankan dikenali sebagai *Evolutionary Cost-Cognizant Test Case Prioritization* (ECRTP) dan ia digunakan sebagai alat ujian regresi untuk eksperimen.

Eksperimen yang dijalankan ke atas pendekatan yang disarankan dan keputusan yang diperolehi telah dinilai secara empirikal dan dibandingkan dengan beberapa pendekatan sedia ada untuk menentukan keberkesanan dan kecekapannya. Purata peratusan pengesanan metrik ralat per kos (APFDc) telah digunakan untuk menyukat kos purata untuk setiap pengesanan keseriusannya. Analisis menunjukkan kelainan yang signifikan untuk keberkesanan, kecekapan dan APFDc untuk ECRTP berbanding dengan pendekatan-pendekatan sedia ada seperti rawak, terbalik, tidak tersusun dan JaNaMa, yang menjadikan ECRTP satu alat yang baik untuk pengujian regresi.

Untuk masa yang akan datang, terdapat keperluan untuk mengembangkan skop kajian ini dengan menggunakan maklumat dari ujian regresi yang terkini, dengan memberi pertimbangan kepada metric-metrik tambahan yang berorientasikan objek, seperti *coupling* dan *cohesion*, dan turut menggunakan proses-proses evolusi objektif. Terdapat juga keperluan untuk melaksanakan strategi ini untuk bahasa-bahasa berorientasikan objek seperti Python, Lisp, dan Smalltalk.

AKNOWLEDGEMENT

First of all, I wish to thank Allah Subhanahu Wa Ta'ala for endowing me with the courage, patience, and guidance to complete this study. I will also like to thank several people who have contributed in one way or the other towards the successful completion of this PhD research work.

I would like to sincerely acknowledge and appreciate the consistent support, guidance, encouragements, contributions and fatherly advice I received from my Supervisor, Professor Dr Abu Bakar Md Sultan, who despite his tight schedules, always been very committed to the smooth progress of this research work. I greatly appreciate the contributions accorded to me by other members of the supervisory team, Professor Dr Abdul Azim Abdul Ghani and Associate Professor Hazura Zulzalil. They have been incredibly supportive, constructive, and remarkably contributed to the success of this study.

The success of this study would not have been a reality without the everlasting love, support, understanding, encouragements, and prayers I received from my parents whom I love and cherish dearly. I would also wish to acknowledge the encouragements and support I received from brothers, sisters, and friends. I wish to acknowledge the care and guidance I received from my mentor, in person of Dr. Abubakar Roko, who always listen to me whenever I have issues regarding my study. To all you, I am highly pleased and grateful.

I wish to acknowledge the management of Usmanu Danfodiyo University Sokoto, Nigeria for giving me this opportunity as well as the encouragements I received from colleagues in the University
.

This thesis was submitted to the senate of the Universiti Putra Malaysia and has been accepted as fulfilment for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Abu Bakar Md Sultan, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Abdul Azim Abdul Ghani, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Hazura Zulzalil, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ROBIAH BINTI YUNUS, PhD
Professor and Dean
School of Graduate Study
Universiti Putra Malaysia

Date:

Declaration by Graduate Student

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any other institutions;
- intellectual properties from the thesis and copyright of the thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from the supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before this thesis is published (in the form of written, printed or in electronic form), including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/ fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature:_______________________ Date: _______________

Name and Matric No.: Abdulkarim Bello (GS45175)

Declaration by Members of the Supervisory Committee

This is to confirm that:

- the research conducted, and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) are adhered to.

Signature: ______________________________

Name of the Chairman of Supervisory Committee: Professor Dr. Abu Bakar Md Sultan

Signature: ______________________________

Name of the Member of Supervisory Committee: Professor Dr. Abdul Azim Abdul Ghani

Signature: ______________________________

Name of the Member of Supervisory Committee: Associate Professor Dr. Hazura Zulzalil

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Ak	Test case award value
ACO	Ant Colony Optimization
ANOVA	Analysis of variance
APFD	Average Percentage of rate of Fault Detection
APFDc	Average Percentage of rate of Fault Detection per Cost
ATRS	Airline Ticket Reservation System
BAC	Bank Account
BCO	Bea Colony Optimization
BST	Binary Search Tree
CCNA	Cisco Certified Network Associate
CFG	Control Flow Graph
ClDG	Class Dependence Graph
CPU	Central Processing Unit
DU	Definition-Used
ECS	Elevator Control System
EFFpa	Effectiveness of Fault Detection3
EP	Evolutionary Programming
ESDG	Extended System Dependence Graph
GA	Genetic Algorithm
HCA	Hill Climbing Algorithm
HCS	Highly Configurable System
HDD	Hard Disk
HIR	Historical Information Repository
IDE	Integrated Development Environment
InDG	Interface Dependence Graph
JSDG	Java System Dependence Graph
LS	Local Search
mACO	Modified Ant Colony Optimization
MCSE	Microsoft Certified System Engineer
MDG	Method Dependence Graph
MEP	Module Execution Path
MuJava	Mutation System for Java
NonPrio	No Prioritization
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NYSC	National Youth Service Corps
OOM	Object-Oriented Modelling
OOPs	Object-Oriented Programs
PaDG	Package Dependence Graph
PDG	Program Dependence Graph
PSA	Particle Swarm Algorithm
RanPrio	Random Prioritization
RCBD	Randomized Complete Block Design
RevPrio	Reversed Prioritization
RQ1	First Research Question
RQ2	Second Research Question
RQ3	Third Research Question
RS	Random Search

RTP	Regression Test Prioritization
SA	Simulated Annealing
SBO	Search Based Optimization
SBST	Search Based Software Testing
SC	Single-point Crossover
SCIA	Software Change Impact Analysis
SDG	System Dependence Graph
SIR	Software-artefact Infrastructure Repository
SLL	Singly Linked List
SuT	System under Test
TCP	Test Case Prioritization
TCR	Test Case Repository
TEEpa	Test Effort Efficiency
Tri	Triangle
TS	Tabu Search
UcVs	Uniform Costs Varying Severities
UML	Unified Modeling Language
VcUs	Varying Costs Uniform Severities
VcVs	Varying Cost Varying Severity

CHAPTER 1

INTRODUCTION

1.1 Background

Software testing is an activity aimed at raising the quality and reliability of software product. It is the process of executing software with the aim of finding bugs. Testing plays important role in software quality assurance. It demonstrates that software work as expected. During the software development process, a software product is tested to validate the changes introduced into the already well-functioned software system. The process of revalidating software product during maintenance phase is called regression testing.

Regression testing is performed between two different versions of software to provide confidence that the newly introduced features of the System under Test (SuT) do not interfere with the existing features. It verifies that the software still performs correctly after it was changed. Changes may include software enhancements, patches, configuration changes, etc. during regression testing, new software bugs or regression may be revealed. Therefore, regression testing ensures that modifications to the software have not introduced new faults and fulfil their intended purpose by correctly updating the software functionality.

During the software development process, regression testing is performed as part of the software maintenance, before the software is released. Being performed multiple times, regression testing can have profound effect on the software budget (Malishevsky et al., 2006). In that instance, regression testing accounts for a large percent of software development cost (Elbaum et al., 2001; Huang et al., 2012; Jiang and Chan, 2015; Schwartz and Do, 2016; Tulasiraman and Kalimuthu, 2018; Wu et al., 2014), which means even small reduction in regression testing cost can have a significant effect on the software development cost. In addition to the reduction of the cost of software development, reducing regression testing time can speed the process of producing new software version earlier than could be possible.

Software engineers frequently develop test suite for regression testing and reuse it across different regression testing session (Harman et al., 2015). However, to test new software features, new test cases are added to the existing test suite. As a result, the test suite increases in size and consequently the cost of executing the test suite increases. For example, it was reported that, to test a software product of about 20,000 line of code, the whole test suite required seven weeks to run.

To reduce the cost of regression testing, several techniques have been proposed (Yoo and Harman, 2010). Khan et al. (2014), Miranda and Bertolino (2016), Sethi et al. (2014), Velmurugan and Mahapatra (2016), Zhang et al. (2014), Zhang et al. (2015) employed test suite minimization by reducing the test suite during regression testing. Researchers such as Chen and Lau (2001), Grindal et al. (2006), Kazmi et al. (2017), Musa (2014), Beena and Sarala (2013), Suppriya and Ilavarasi (2015), Yoo and Harman (2007) proposed a regression testing techniques that select a subset of the test suite to test a particular software.

Test case prioritization seeks to find the ideal ordering of test cases for regression testing, so that the tester obtain maximum benefit, even if the testing is prematurely halted at some arbitrary point (Indumathi and Selvamani, 2015; Kavitha and Sureshkumar, 2010; Kayes, 2011; Musa, 2014; Park et al., 2008; Patil et al., 2016; Shameem and Kanagavalli, 2013; Tulasiraman and Kalimuthu, 2018; Z. H. Zhang et al., 2012). The approach was first studied by (Wong et al., 1998). Later, Sinha et al. (1999) proposed the approach in a more general context which was evaluated by (Rothermel et al., 1999).

1.2 Problem Statement

Regression testing is performed between two different versions of software to provide confidence that the newly introduced features of the SuT do not interfere with the existing functionalities. Basically, test case prioritization is performed to increase the rate of faults detection for regression testing during software maintenance. An improved rate of fault detection can provide faster feedback on the SuT, enabling debugging to start earlier and increase the likelihood that, if testing is abruptly stopped, those test cases with the greatest fault detection in the test suite would have been executed.

Ideally, to perform regression testing, tester should re-execute all the test cases in the test suite on the affected program (Fang et al., 2014). Re-executing all the test cases can be pervasive, tedious and expensive especially when the test suite size is big (Do et al., 2006). Test cases can be chosen randomly to reduce the cost of executing the whole test suite (Zhou et al., 2011). However, chosen test cases at random might result in only executing small portion of the modified component of the software (Srikanth et al., 2016).

Over the years, several test case prioritization approaches have been developed (Rava and Wan-Kadir, 2016). These approaches have been explored and their efficacy is evaluated in achieving certain criteria. However, most of these approaches focus on procedural languages with only few on object-oriented programs (J. Chen et al., 2018), lots of features differentiate object-oriented and procedure oriented programming concepts (Stefik and Bobrow, 1985; Wiedenbeck et al., 1999). Authors such as Panda et al. (2016) and Sultan et al. (2014) addressed object-oriented programs but have the assumption that test case costs and fault severities are uniform. While in real sense, test case costs and fault severities vary (Malishevsky et al., 2006).

Although some of these approaches Tulasiraman and Kalimuthu (2018), Wang et al. (2016) used varying costs of test cases and severities of faults, but focused only on procedural programs. Moreover, most of these approaches adopted local search strategies to search for an optimized order of test cases for regression testing, meanwhile, these strategies mostly terminate at local optima (Sanchez et al., 2014) and (Srivastava and Kim, 2009). Consequently, an evolutionary optimization technique based on genetic algorithm (GA) Mitchell (1998) and Whitley (2001) has been reported to produce an astonishingly better result when applied for propitiating test cases.

Previous test case prioritization assumed that all test cases are equally expensive, and all fault are equally severe (Bello et al, 2018). While this is appropriate in some cases, in other cases is an oversimplification. Some test cases can simply detect an error in an input and terminate almost immediately, while other test cases can involve computations that requires hour to complete. Similarly, some test cases require resource usage such as equipment, expandable materials, or human labor (Malishevsky et al., 2006), while different test cases may utilize little or no equipment or human labor (Elbaum et al., 2001). Under these circumstances, when evaluating the relative worth of test cases, we need to account for these differences in costs. Similarly, in many situations, faults differ in severity. One fault can be a simple error in an interface which many users would tolerate. While another fault can result into inaccurate parameter supply to a device which can result in program failure, or even catastrophes such as aircraft control or radiation overdose (Huang et al., 2012). Fault severity, too, may be an important component to consider.

Tulasiraman and Kalimuthu (2018) proposed a cost-cognizant history-based test case prioritization approach that uses historical information of test case such as test case costs, faults, and severities of fault for prioritization. The approach manually seeds faults to the original programs, and there is no clear representation of the internal structure of the programs considered during the experimentation. Furthermore, the approach cannot guarantee that the affected components, by the changed information, are those detected by the test cases. Moreover, the approach considers only procedural programs. Program features such as encapsulation, inheritance, polymorphism and dynamic binding are not available in procedure-oriented programs, as such approach developed to prioritize procedural programs may not be suitable for object-oriented programs. Musa et al., (2016) proposed a regression test case prioritization for object-oriented programs. The approach developed to prioritize test case for regression testing of object-oriented programs using reduced severity of faults used Genetic Algorithm (GA) for computing the fitness value of test cases. While developed with Java programming language, the approach uses ESDG for representing the internal structure of the program under test (PuT). ESDG was developed for C++, which does not support static member functions and static member variables, as such ESDG model developed to represent the internal structure of Java program may not capture the exact structure of the program intended to capture. Moreover, the authors used APFD for measuring the average percentage of fault detection of the approach. While the approach uses different fault severities for fitness value computation, and APFD was developed on the assumption that all faults across the PuT are uniform. The measure computed by APFD might not to be the exact measure intended to measure.

Furthermore, Lou et al. (2015) proposed test case prioritization approach for software evolution. The approach uses mutation faults on the difference between the early and later versions of a software. The approach uses statistical-based and probability-based models to measure the fault detection capability of the approach. Panigrahi and Mall (2014) proposed a heuristic-based TCP based on the analysis of dependence model of OOP. Their technique builds an intermediate dependence model of a program from the source code of the programs. The model is updated to reflect the corresponding changes whenever the program is modified. The approach identifies affected nodes by constructing the union of forward slices corresponding to each changed element. Test case that covers one or more affected nodes are selected for regression testing. The weights of test cases are computed by assigning a value that corresponds to the weight of the affected nodes. This approach assumed both test costs and faults severities to be uniform.

Velmurugan and Mahapatra (2016) proposed a GA-based regression TCP approach that considers branch coverage DU (Definition-Used) pair coverage for effective prioritization of test cases. However, the experimental procedure, experimental objects, and procedure used by the approach for representing the program were not clearly mentioned. Furthermore, there is no enough analysis to prove the validity of the results obtained from the experiment.

Consequently, this study proposes an evolutionary cost-cognizant regression TCP approach for OOP based on the use of the previous test case execution record and a GA. Tests costs, faults severities, and faults detected by each test case from the latest regression testing are gathered and then use a GA to find an order with the greatest rate of units of fault severity detected per unit test cost.

1.3 Research Question

This section presents the research questions for the study. The questions serve as the focal point of the investigation that will be addressed by the empirical study for this research. The questions addressed are as followed.

RQ1- How efficient is the GA-based Evolutionary Cost-cognizant Regression Test Prioritization (ECRTP) approach for OOP in terms of faults detection when compared with other approaches? In other words, does GA-based ECRTP approach for OOP increases the efficiency of the prioritize test cases for fault detection as compare to other approaches?

RQ2- How effective is GA-based Evolutionary Cost-cognizant Regression Test Prioritization (ECRTP) approach for OOP in terms of fault detection as compared with other approaches? In other words, does GA-based ECRTP approach for OOP increases the effectiveness of prioritized test cases to detect faults as compared with other approaches?

RQ3- Does GA-based Evolutionary Cost-cognizant Regression Test Prioritization (ECRTP) for OOP increases the average percentage of faults detection per cost (APFDc) as compared with other approaches? In other words, to what extent GA-based Evolutionary Cost-cognizant Regression Test Prioritization (ECRTP) performs in terms of APFDc as compared with other approaches?

1.4 Objectives of the Study

Regression test case prioritization and Object-oriented Programming (OOP) are active fields of research, integration of both concepts is an important activity in software maintenance, as it can improve software quality in general. Thus, the main objective of this research is to combine evolutionary algorithms, Genetic Algorithm (GA) specific, with OOP to develop automated test case prioritization for regression testing. In order to achieve the main objective, the list below outline the specific objectives of the research:

- To propose a GA-based evolutionary cost-cognizant regression testing approach for OOP that considers varying tests costs and faults severities.

- To develop a prototype tool that uses GA to implement a cost-cognizant regression test case prioritization for object-oriented programs.

- To empirically evaluate the efficiency of testing effort, effectiveness of fault detection and average percentage of fault detection per cost (APFDc) of the proposed approach.

1.5 Scope of the Study

This research has the following scopes:

1. This study is limited to test case prioritization of object-oriented programs and the coverage information generated from the source code using path-based integration testing. The mutants considered are generated at both class and method levels of the source code. The initial source codes were assumed to be tested and worked as designed. JUnit framework is the testing framework from which test cases would be developed.

2. This study focuses on object-oriented programs written in Java programming language that is the widely used programming language in implementing the OO technology. Therefore, this study does not consider programs created with other languages, like C++, C#.

3. The study did not use APFD metric for measuring the percentage of rate of fault detection for the prioritization approaches as a result of the limitations identified by the literature review that are associated to the metric.

1.6 Contributions of the Study

This is study is expected to make the following contributions:

- Contributes to the software testing community by providing an effective and efficient technique for faults detection with GA-based evolutionary cost-cognizant regression test case prioritization technique.
- Provide an efficient and effective test case prioritization technique to software development community which will make their development work faster at meeting the time scheduled time for the project.
- Increase into the body of knowledge of software testing and software engineering in general by adding another finding available literature.

1.7 Organization of the Thesis

This thesis is reported in seven chapters organized in chronological order from the introduction to the conclusion and future work. The first chapter gives an introduction of the thesis. It presents the background, problem statement, research question, research objectives, scope of the study and research contributions of the thesis. Chapter two presents the literature review of the thesis. It presents the detail review of the key areas that lay foundation for this research work and highlights gaps in the related literature. It also presents existing techniques for regression test case prioritization. Chapter three presents the general overview of the research methodology and the materials used for the research objectives to be achieved, and to implement the prototype support for the proposed regression testing technique. Chapter four presents the new regression testing technique, which an evolutionary regression testing approach for object-oriented programs. In chapter five, experiments were presented and trials to answer the research question were also demonstrated. Experimental results, analysis, and discussion were presented in chapter six. While seven covers the conclusion and feature work.

REFERENCES

Abhinandan, H. P., Goveas, N., & Rangarajan, K. (2016). Regression Test Suite Prioritization using Residual Test Coverage Algorithm and Statistical Techniques. International Journal of Education and Management Engineering, 6(5), 32–39. https://doi.org/10.5815/ijeme.2016.05.04

Acharya, A. A., Mahali, P., & Mohapatra, D. P. (2015). Computational Intelligence in Data Mining - Volume 3. Computational Intelligence in Data Mining Smart Innovation, Systems and Technologies, 33, 429–440. https://doi.org/10.1007/978-81-322-2202-6

Afzal, W., Torkar, R., & Feldt, R. (2009). A systematic review of search-based testing for non-functional system properties. Information and Software Technology, 51(6), 957–976. https://doi.org/10.1016/j.infsof.2008.12.005

Ahmed, A. A., Shaheen, M., & Kosba, E. (2012). Software Testing Suite Prioritization Using Multi-Criteria Fitness Function. *Iccta*, (c), 160–166.

Alakeel, A. M. (2012). A Fuzzy Test Cases Prioritization Technique for Regression Testing Programs with Assertions. In *ADVCOMP 2012 : The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences* (pp. 78–82).

Ansari, A., Khan, A., Khan, A., & Mukadam, K. (2016). Optimized Regression Test Using Test Case Prioritization. In *7th International Conference on Communication, Computing and Virtualization* (pp. 152–160). Elsevier Masson SAS.

Arafeen, M. J., & Do, H. (2013). Test case prioritization using requirements-based clustering. In *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013* (pp. 312–321). https://doi.org/10.1109/ICST.2013.12

Arora, Vinay, Rajesh Kumar Bhatia, M. S. (2012). Evaluation of Flow Graph and Dependence Graphs for Program Representation. *International Journal of Computer Applications (0975 – 8887)*, *56*(14), 18–23. https://doi.org/10.5120/8959-3161

Ashraf, E., Rauf, A., & Mahmood, K. (2012). Value based Regression Test Case Prioritization. In *World Congress on Engineering and Computer Science 2012 Vol I WCECS 2012, October 24-26, 2012, San Francisco, USA* (Vol. I, pp. 978–988).

Avritzer, A., & Weyuker, E. J. (1995). The Automatic Generation of Load Test Suites and the Assessment of the Resulting Software. *IEEE Transactions on Software Engineering*, *21*(9), 705–716. https://doi.org/10.1109/32.464549

Bannon, T. J., Ford, S. J., Joseph, V. J., Perez, E. R., Peterson, R. W., Sparacin, D. M., … others. (1994). System and method for database management supporting object-oriented programming. Google Patents.

Barillari, F., Gorga, I., & Piccinini, S. (2018). Collaborative maintenance of software programs. Google Patents.

Beena, R., & Sarala, S. (2013). Code Coverage Based Test Case Selection. *International Journal of Software Engineering & Applications (IJSEA)*, *4*(6), 39–49. https://doi.org/10.5121/ijsea.2013.4604

Bello, A., Sultan, A., Abdul Ghani, A. A., & Zulzalil, H. (2018). Evolutionary Cost Cognizant Regression Test Prioritization for Object-Oriented Programs Based on Fault Dependency. *International Journal of Engineering and Technology*, *7*(4.1), 28–32.

Bennett, K. H. (1997). Software maintenance: A tutorial. *Software Engineering, IEEE Computer Society*, 289–303.

Bergeretti, J.-F., & Carré, B. A. (1985). Information-flow and data-flow analysis of while-programs. *ACM Transactions on Programming Languages and Systems*, *7*(1), 37–61. https://doi.org/10.1145/2363.2366

Bhojasia, M. K. (2011a). Java Program to Implement a Binary Search Tree using Linked Lists. Retrieved from https://www.sanfoundry.com/java-program-implement-binary-search-tree-using-linked-list/

Bhojasia, M. K. (2011b). Java Program to Implement Singly Linked List. Retrieved from https://www.sanfoundry.com/java-program-implement-singly-linked-list/

Bhojasia, M. K., & Sanfoundry. (2011). Java Program to Implement Binary Search Tree. Retrieved July 11, 2017, from https://www.sanfoundry.com/java-program-implement-binary-search-tree/

Biswal, B. N., Nanda, P., & Mohapatra, D. P. (2008). A novel approach for scenario-based test case generation. *Proceedings - 11th International Conference on Information Technology, ICIT 2008*, 244–247. https://doi.org/10.1109/ICIT.2008.43

Booch, G. (1982). Object-oriented design. *ACM SIGAda Ada Letters*, *1*(3), 64–76.

Brucker, A. D., & Julliand, J. (2014). Testing the Untestable. *Software Testing Verification and Reliability*, *24*(8), 591–592. https://doi.org/10.1002/stvr

Candea, G., Bucur, S., & Zamfir, C. (2010). Automated software testing as a service. *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10*, 155–160.

Cao, Y., Zhou, Z. Q., & Chen, T. Y. (2013). On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions. In *2013 13th International Conference on Quality Software* (pp. 153–162).

Chandra, A., Singhal, A., & Bansal, A. (2015). A Study of Program Slicing Techniques for Software Development Approaches. In *1st International Conference on Next Generation Computing Technologies (NGCT-2015)* (pp. 4–5).

Chantarangsi, W., Liu, W., Bretz, F., Kiatsupaibul, S., & Hayter, A. J. (2016). Normal probability plots with confidence for the residuals in linear regression. *Communications in Statistics: Simulation and Computation*, *47*(2), 367–379. https://doi.org/10.1080/03610918.2016.1165840

Chauhan, N. (2019). A Multi-factored Cost-and Code Coverage-Based Test Case Prioritization Technique for Object-Oriented Software. In *Software Engineering* (pp. 27–36). Springer.

Chen, J., Zhu, L., Chen, T. Y., Towey, D., Kuo, F. C., Huang, R., & Guo, Y. (2018). Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering. *Journal of Systems and Software*, *135*, 107–125. https://doi.org/10.1016/j.jss.2017.09.031

Chen, T. Y., & Lau, M. F. (2001). Test case selection strategies based on Boolean specifications. *Software Testing Verification and Reliability*, *11*(3), 165–180. https://doi.org/10.1002/stvr.221

Chen, Z., Chen, L., Zhou, Y., Xu, Z., Chu, W. C., & Xu, B. (2014). Dynamic slicing of python programs. *Proceedings - International Computer Software and Applications Conference*, 219–228. https://doi.org/10.1109/COMPSAC.2014.30

De Lucia, A. (2001). Program slicing: methods and applications. *Proceedings First IEEE International Workshop on Source Code Analysis and Manipulation*, 142–149. https://doi.org/10.1109/SCAM.2001.972675

De Lucia, Andrea, Pompella, E., & Stefanucci, S. (2002). Effort estimation for corrective software maintenance. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering* (pp. 409–416).

de S. Campos Junior, H., Araújo, M. A. P., David, J. M. N., Braga, R., Campos, F., & Ströele, V. (2017). Test case prioritization: a systematic review and mapping of the literature. In *Proceedings of the 31st Brazilian Symposium on Software Engineering - SBES'17* (pp. 34–43). https://doi.org/10.1145/3131151.3131170

Di Nardo, D., Alshahwan, N., Briand, L., & Labiche, Y. (2015). Coverage-based regression test case selection, minimization and prioritization: a case study on an industrial system. *Software Testing, Verification and Reliability*, *25*, 371–396.

Do, H., Rothermel, G., & Kinneer, A. (2006). Prioritizing JUnit test cases: An empirical assessment and cost-benefits analysis. *Empirical Software Engineering*, *11*(1), 33–70. https://doi.org/10.1007/s10664-006-5965-8

Donglin, L., & Harrold, M. J. (1998). Slicing objects using system dependence graphs. *Software Maintenance, 1998. Proceedings. International Conference On*, (November), 358–367. https://doi.org/10.1109/ICSM.1998.738527

Duran, J. W., & Ntafos, S. C. (1984). An Evaluation of Random Testing. *IEEE Transactions on Software Engineering*, *SE-10*(4), 438–444. https://doi.org/10.1109/TSE.1984.5010257

Elbaum, S., Kallakuri, P., Malishevsky, A., Rothermel, G., & Kanduri, S. (2003). Understanding the effects of changes on the cost-effectiveness of regression testing techniques. *Software Testing, Verification and Reliability*, *13*(2), 65–83. https://doi.org/10.1002/stvr.263

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002). Test case prioritization: a family of empirical studies. *IEEE Transactions on Software Engineering*, *28*(2), 159–182.

Elbaum, S., Malishevsky, A., & Rothermel, G. (2001). Incorporating varying test costs and fault severities into test case prioritization. In *Proceedings - International Conference on Software Engineering* (pp. 329–338). https://doi.org/10.1109/ICSE.2001.919106

Elodie, V. (2011). White Box Coverage and Control Flow Graphs.

Ermakov, S., Schmidt, B. J., Musante, C. J., & Thalhauser, C. J. (2019). A survey of software tool utilization and capabilities for quantitative systems pharmacology: what we have and what we need. *CPT: Pharmacometrics & Systems Pharmacology*, *8*(2), 62–76.

Fairley, R. (1978). Tutorial: Static Analysis and Dynamic Testing of Computer Software. *Computer*, *11*(4), 14–23.

Fang, C., Chen, Z., Wu, K., & Zhao, Z. (2014). Similarity-based test case prioritization using ordered sequences of program entities. *Software Quality Journal*, *22*(2), 335–361. https://doi.org/10.1007/s11219-013-9224-0

Fazlalizadeh, Y., Khalilian, A., & Azgomi, A. (n.d.). M., & Parsa, S. (2009). Incorporating historical test case performance data and resource constraints into test case prioritization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5668*, 43–57. Retrieved from https://doi.org/10.1007/978-3-642-02949-3_5

Feldt, R., Poulding, S., Clark, D., & Yoo, S. (2016). Test Set Diameter: Quantifying the Diversity of Sets of Test Cases. *Proceedings - 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016*, 223–233. https://doi.org/10.1109/ICST.2016.33

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution. Willey-IEEE Press. John Wiley.

Fraser, G., & Science, C. (2014). A Large Scale Evaluation of Automated Unit Test Generation Using EvoSuite. In *ACM Transactions on Software Engineering and Methodology (TOSEM)* (Vol. 24, p. 8).

Garg, D., & Datta, A. (2012). Test case prioritization due to database changes in web applications. In *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012* (pp. 726–730). https://doi.org/10.1109/ICST.2012.163

Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, *10*(2), 486–489. https://doi.org/10.5812/ijem.3505

GITHub. (2018). Java Program to Implement Coffea Vending Machine. Retrieved from https://github.com/8thlight/CoffeeMaker/tree/master/src/main/java

Goel, N., & Sharma, M. (2015). Prioritization of Test Cases and Its Techniques. *International Journal of Computer Application*, *5*(4), 85–89.

Grindal, M., Lindström, B., Offutt, J., & Andler, S. F. (2006). An evaluation of combination strategies for test case selection. *Empirical Software Engineering*, *11*(4), 583–611. https://doi.org/10.1007/s10664-006-9024-2

Gupta, N. K., & Rohil, M. K. (2008). Using Genetic Algorithm for Unit Testing of Object Oriented Software. In *First International Conference on Emerging Trends in Engineering and Technology* (Vol. 10, pp. 308–313).

Gupta, N., Sharma, A., & Pachariya, M. K. (2019). An Insight Into Test Case Optimization: Ideas and Trends With Future Perspectives. *IEEE Access*, *7*, 22310–22327.

Haidry, S.-Z., & Miller, T. (2013). Using Dependency Structures for Prioritisation of Functional Test Suites. *IEEE Transactions on Software Engineering*, *39*(2), 258–275. https://doi.org/10.1109/TSE.2012.26

Hao, D., Zhang, L., & Mei, H. (2016). Test-case prioritization: achievements and challenges. *Frontiers of Computer Science*, 1–9. https://doi.org/10.1007/s11704-016-6112-3

Hao, D., Zhang, L., Zang, L., Wang, Y., Wu, X., Xie, T., & Member, S. (2016). To Be Optimal or Not in Test-Case Prioritization, *42*(5), 490–504.

Hao, D., Zhao, X., & Zhang, L. (2013). Adaptive test-case prioritization guided by output inspection. In *Proceedings - International Computer Software and Applications Conference* (pp. 169–179). https://doi.org/10.1109/COMPSAC.2013.31

Harle, R. (1988). Object Oriented Programming. In *Computer Software and Applications Conference* (Vol. 4, pp. 51–57).

Harman, M., Jia, Y., & Zhang, Y. (2015). Achievements, Open Problems and Challenges for Search Based Software Testing. In *8th International Conference on Software Testing, Verification and Validation (ICST), 2015 IEEE* (pp. 1–12). https://doi.org/10.1109/ICST.2015.7102580

Harman, M., & Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, *43*(14), 833–839. https://doi.org/10.1016/S0950-5849(01)00189-6

Harman, M., & McMinn, P. (2010). A theoretical and empirical study of search-based testing: Local, global, and hybrid search. *IEEE Transactions on Software Engineering*, *36*(2), 226–247. https://doi.org/10.1109/TSE.2009.71

Henard, C., Papadakis, M., Harman, M., Jia, Y., Traon, Y. Le, & Le Traon, Y. (2016). Comparing White-box and Black-box Test Prioritization. In *ICSE '16 Proceedings of the 38th International Conference on Software Engineering* (pp. 523–534). https://doi.org/10.1145/2884781.2884791

Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Ann Arbor: University of Michigan Press.

Horwitz, S., Reps, T., & Binkley, D. (1988). Interprocedural slicing using dependence graphs. *Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation - PLDI '88*, *12*(1), 35–46. https://doi.org/10.1145/53990.53994

Huang, Y. C., Huang, C. Y., Chang, J. R., & Chen, T. Y. (2010). Design and analysis of cost-cognizant test case prioritization using genetic algorithm with test history. *Proceedings - International Computer Software and Applications Conference*, 413–418. https://doi.org/10.1109/COMPSAC.2010.66

Huang, Y. C., Peng, K. L., & Huang, C. Y. (2012). A history-based cost-cognizant test case prioritization technique in regression testing. *Journal of Systems and Software*, *85*(3), 626–637.

IEEE. (1998). *IEEE Std 1219-1998: IEEE Standard for Software Maintenance*. IEEE. Retrieved from https://books.google.com.my/books?id=TxRlAQAACAAJ

Iftikhar, A., Musa, S., Alam, M., Su'ud, M. M., & Ali, S. M. (2018). A survey of soft computing applications in global software development. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)* (pp. 1–4).

Indumathi, C. P. P., & Selvamani, K. (2015). Test Cases Prioritization Using Open Dependency Structure Algorithm. *Procedia Computer Science*, *48*(Iccc), 250–255.

Islam, M. M., Marchetto, A., Susi, A., & Scanniello, G. (2012). A multi-objective technique to prioritize test cases based on Latent Semantic Indexing. In *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR* (pp. 21–30). https://doi.org/10.1109/CSMR.2012.13

ISO/IEC. (2006). *ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998)*. IEEE. Retrieved from https://books.google.com.my/books?id=y0tizgAACAAJ

ISO/IEC. (2011). ISO/IEC 25010:2011- Systems and Software Engineering.

Jackson, K., & Hunter, A. (1998). CSC108 Tutorials Java Program to Implement to Implement Triangle Program. Retrieved July 20, 2018, from http://www.cs.toronto.edu/~andria/csc/108s98/programs/tut4/Triangle.java

Jafrin, S., Nandi, D., & Mahmood, S. (2016). Test Case Prioritization based on Fault Dependency. *International Journal of Modern Education and Computer Science*, *8*(4), 33–45. https://doi.org/10.5815/ijmecs.2016.04.05

Jefferey, D., & Gupta, N. (2006). Test Case Prioritization using Clustering. In 30th Annual International Computer Software and Applications Conference (COMPSAC'06).

Jiang, B., & Chan, W. K. (2013). Bypassing code coverage approximation limitations via effective input-based randomized test case prioritization. In *Proceedings - International Computer Software and Applications Conference* (pp. 190–199). https://doi.org/10.1109/COMPSAC.2013.33

Jiang, B., & Chan, W. K. (2015). Input-based adaptive randomized test case prioritization: A local beam search approach. *Journal of Systems and Software*, *105*, 91–106.

Jovanovic, I. (2009). Software Testing Methods and Techniques. *The IPSI BgD Transactions on Internet Research*, *5*(1), 30–41.

Kajko-Mattsson, M., Glassbrook, A. G., & Nordin, M. (2001). Evaluating the predelivery phase of ISO/IEC FDIS 14764 in the Swedish context. In *Software Maintenance, 2001. Proceedings. IEEE International Conference on* (pp. 431–440).

Kanewala, U., & Bieman, J. M. (2014). Testing Scientific Software: A Systematic Literature Review. *Information and Software Technology*, *56*(10), 1219–1232. https://doi.org/10.1016/j.infsof.2014.05.006

Kavitha, R., & Sureshkumar, N. (2010). Test Case Prioritization for Regression Testing based on Severity of Fault. *International Journal on Computer Science and Engineering*, *02*(05), 1462–1466.

Kayes, M. (2011). Test case prioritization for regression testing based on fault dependency. (ICECT). *(ICECT), 2011 3rd International Conference On*, *2011*(3), 48–52.

Kazmi, R., Jawawi, D. N. A., Mohamad, R., & Ghani, I. (2017). Effective Regression Test Case Selection. *ACM Computing Surveys*, *50*(2), 1–32. https://doi.org/10.1145/3057269

Khalilian, A., Abdollahi Azgomi, M., & Fazlalizadeh, Y. (2012). An improved method for test case prioritization by incorporating historical test case data. *Science of Computer Programming*, *78*(1), 93–116. https://doi.org/10.1016/j.scico.2012.01.006

Khan, S. U. R., Lee, S. P., Parizi, R. M., & Elahi, M. (2014). A code coverage-based test suite reduction and prioritization framework. *2014 4th World Congress on Information and Communication Technologies, WICT 2014*, 229–234. https://doi.org/10.1109/WICT.2014.7076910

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Konsaard, P., & Ramingwong, L. (2015). Total Coverage Based Regression Test Case Prioritization using Genetic Algorithm. In 12th International Conference on Electrical/Electronics Engineering, Computer, Telecommunications and Information Technology (ECTI-CON) (pp. 1–6).

Kovács, G., Magyar, F., & Gyimóthy, T. (1996). Static slicing of java programs. Research Group on Artificial Intelligence, Hungarian Academy of Science, Joseph Attila University. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.8438

Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. MA. MIT Press, Cambridge.

Krüger, J., Berger, T., & Leich, T. (2018). Features and How to Find Them: A Survey of Manual Feature Location. *Software Engineering for Variability Intensive Systems: Foundations and Applications*, *32*(6), 153–172.

Kushwah, J. S. (2014). Testing for object oriented software. *Indian Journal of Computer Science and Engineering (IJCSE)*, *2*(1), 90–93.

Kwon, J.-H., Ko, I.-Y., Rothermel, G., & Staats, M. (2014). Test Case Prioritization Based on Information Retrieval Concepts. In *2014 21st Asia-Pacific Software Engineering Conference* (pp. 19–26). https://doi.org/10.1109/APSEC.2014.12

Ledru, Y., Petrenko, A., Boroday, S., & Mandran, N. (2012). Prioritizing test cases with string distances. *Automated Software Engineering*, *19*(1), 65–95. https://doi.org/10.1007/s10515-011-0093-0

Lehman, M. M. (1996). Laws of software evolution revisited. In *European Workshop on Software Process Technology* (pp. 108–124).

Leitner, A., Ciupa, H., Meyer, B., & Howard, M. (2007). Reconciling manual and automated testing: The AutoTest experience. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 1–10.

Li, Z., Bian, Y., Zhao, R., & Cheng, J. (2013). A fine-grained parallel multi-objective test case prioritization on GPU. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8084 LNCS, 111–125. https://doi.org/10.1007/978-3-642-39742-4_10

Li, Z., Harman, M., & Hierons, R. M. (2007). Search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*, *33*(4), 225–237. https://doi.org/10.1109/TSE.2007.38

Lin, C. T., Chen, C. D., Tsai, C. S., & Kapfhammer, G. M. (2013). History-based test case prioritization with software version awareness. *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 171–172. https://doi.org/10.1109/ICECCS.2013.33

Link, D., Behnam, P., Moazeni, R., & Boehm, B. (2019). The Value of Software Architecture Recovery for Maintenance. *ArXiv Preprint ArXiv:1901.07700*.

Losavio, F., Matteo, A., & Camejo, I. P. (2014). Unified Process for Domain Analysis integrating Quality , Aspects and Goals. *Clei Electronic Journal*, *17*(2), 1–21.

Lou, Y., Hao, D., & Zhang, L. (2015). Mutation-based test-case prioritization in software evolution. *2015 IEEE 26th International Symposium on Software Reliability Engineering, ISSRE 2015*, 46–57. https://doi.org/10.1109/ISSRE.2015.7381798

Ma, Y.-S., Offutt, J., & Kwon, Y.-R. (2006). MuJava: a mutation system for Java. *Proceeding of the 28th International Conference on Software Engineering - ICSE '06*, 827. https://doi.org/10.1145/1134285.1134425

Maheswari, R. U., & Jeya Mala, D. (2013). A novel approach for test case prioritization. In *2013 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1–5). https://doi.org/10.1109/ICCIC.2013.6724209

Malhotra, R. (2016). *Empirical Research in Software Engineering*. *CRC Press*. Newyork: Taylor & Francis Group, LLC. https://doi.org/10.1039/c2lc90006h

Malishevsky, A. G., Ruthruff, J. R., Rothermel, G., & Elbaum, S. (2006). *Cost-cognizant Test Case Prioritization. Department of Computer Science and Engineering University of NebraskaLincoln Techical Report. Department of Computer Science and Engineering University of NebraskaLincoln Techical Report*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.6542&rep=rep1&type=pdf

Malz, C., Jazdi, N., & Göhner, P. (2012). Prioritization of Test Cases using Software Agents and Fuzzy Logic. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation* (pp. 483–486). https://doi.org/10.1109/ICST.2012.63

Marchetto, A., Islam, M., Asghar, W., Susi, A., & Scanniello, G. (2015). A Multi-Objective Technique to Prioritize Test Cases. *IEEE Transactions on Software Engineering*, *5589*(c), 1–22. https://doi.org/10.1109/TSE.2015.2510633

Marchetto, A., Tonella, P., & Ricca, F. (2008). State-based testing of Ajax Web applications. *Proceedings of the 1st International Conference on Software Testing, Verification and Validation, ICST 2008*, (June 2014), 121–130. https://doi.org/10.1109/ICST.2008.22

Marijan, D., Gotlieb, A., & Sen, S. (2013). Test case prioritization for continuous regression testing: An industrial case study. In *IEEE International Conference on Software Maintenance, ICSM* (pp. 540–543). https://doi.org/10.1109/ICSM.2013.91

Mary, L., Harrold, M. J., Larsen, L., & Harrold, M. J. (1996). Slicing Object-Oriented Software *. In *18th International Conference on Software Engineering* (pp. 495–505). https://doi.org/10.1109/ICSE.1996.493444

Mei, H., Hao, D., Zhang, L., Zhang, L., Zhou, J., & Rothermel, G. (2012). A static approach to prioritizing JUnit test cases. In *IEEE Transactions on Software Engineering* (Vol. 38, pp. 1258–1275).

Mendenhall, M., Beaver, R. J., & Beaver, B. M. (2007). *INTRODUCTION TO PROBANILITY & STATISTICS* (13th ed.). Canada: Nelson Education, Ltd.

Mikhajlova, A., & Sekerinski, E. (1997). Class refinement and interface refinement in object-oriented programs. In *International Symposium of Formal Methods Europe* (pp. 82–101).

Mills, C., Escobar-Avila, J., & Haiduc, S. (2018). Automatic Traceability Maintenance via Machine Learning Classification. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 369–380).

Miranda, B., & Bertolino, A. (2016). Scope-aided Test Prioritization, Selection and Minimization for Software Reuse. *Journal of Systems and Software*, *0*, 1–22. https://doi.org/10.1016/j.jss.2016.06.058

Mitchell, M. (1998). *An Introduction to Genetic Algorithms (Complex Adaptive Systems). The MIT Press* (Fifth Edit). The MIT Press (Fifth Edit). Cambridge, Massachusetts • London, England: Bradford Book The MIT Press. https://doi.org/10.1016/S0898-1221(96)90227-8

Mohapatra, S. K., & Prasad, S. (2013). Evolutionary search algorithm for Test Case Prioritization. *2013 International Conference on Machine Intelligence and Research Advancement (Icmira 2013)*, 115–119. https://doi.org/10.1109/ICMIRA.2013.29

Moubayed, N. Al, & Windisch, A. (2009). Temporal White-Box Testing Using Evolutionary Algorithms. *International Conference on Software Testing, Verification and Validation Workshops*, 150–151.

Musa, S. (2014). A Regression Test Case Selection and Prioritization for Object-Oriented Programs using Dependency Graph and Genetic Algorithm. *International Journal of Engineering And Science*, *4*(7), 54–64.

Musa, S., Sultan, A. B. M., Abdul Ghani, A. A. Bin, & Baharom, S. (2014). Regression Test Case Selection &PrioritizationUsing Dependence Graph and Genetic Algorithm. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *IV*(3), 2278–2661.

Musa, S., Sultan, A. B. M., Abdul Ghani, A. A. Bin, & Baharom, S. (2016). Regression Test Cases Prioritization for Object-Oriented Programs Using Genetic Algorithm with Reduced Value of Fault Severity. *International Journal of Soft Computing*, *11*(4), 247–254.

Muthusamy, T., & K., S. (2013). A Test Case Prioritization Method with Weight Factors in Regression Testing Based on Measurement Metrics. *International Journal of Advanced Research in Computer Science and Software Engineering*, *03*(12), 390–396.

Muthusamy, T., & Seetharaman K. (2014). A New Effective Test Case Prioritization for Regression Testing based on Prioritization Algorithm. *International Journal of Applied Information Systems*, *6*(7), 21–26.

Myers, J. G., Badget, T., & Sandler, C. (2012). *The Art of Software Testing* (Third Edit). New Jessy, Canada: John Wiley & Sons, Inc.,.

Nah, F. F.-H., Faja, S., & Cata, T. (2001). Characteristics of ERP software maintenance: a multiple case study. *Journal of Software Maintenance and Evolution: Research and Practice*, *13*(6), 399–414.

Najumudheen, E., Mall, R., & Samanta, D. (2009). A dependence graph-based representation for test coverage analysis of object-oriented programs. *ACM SIGSOFT Software Engineering Notes*, *34*(2), 1. https://doi.org/10.1145/1507195.1507208

Narula, S., Prioritization, C., Minimisation, T. C., & Case, T. (2016). Review Paper on Test Case Selection, *6*(4), 126–128.

Nayak, S., Kumar, C., & Tripathi, S. (2016). Effectiveness of Prioritization of Test Cases Based on Faults. In *3rd Int'l Conf. on Recent Advances in Information Technology* (pp. 1–6). https://doi.org/10.1109/RAIT.2016.7507977

Nidhra, S., & Dondeti, J. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*, *2*(2), 29–50. https://doi.org/10.5121/ijesa.2012.2204

Nirpal, P. B., & Kale, K. V. (2011). Using Genetic Algorithm for Automated Efficient Software Test Case Generation for Path Testing. *International Journal of Advanced Networking and Applications*, *915*(6), 911–915.

Noble, J., & Grundy, J. (1995). Explicit Relationships in Object-Oriented Development. *Proceedings of the 18^{th} Conference on the Technology of Object-Oriented Languages and Systems*, 211–226. Retrieved from http://citeseer.nj.nec.com/noble95explicit.html

Nucci, D. Di, Panichella, A., Zaidman, A., & Lucia, A. De. (2015). Hypervolume-Based Search for Test Case Prioritization. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9275, 157–172. https://doi.org/10.1007/978-3-319-22183-0

Offutt, J., & Untch, R. H. (2000). Mutation 2000: Uniting the Orthogonal. *Proc. of the Mutation 2000: Mutation Testing in the Twentieth and the Twenty First Centuries*, 45–55. https://doi.org/10.1007/978-1-4757-5939-6_7

Ogasawara, T. (2014). Workload characterization of server-side JavaScript. In *IISWC 2014 - IEEE International Symposium on Workload Characterization* (pp. 13–21). https://doi.org/10.1109/IISWC.2014.6983035

Ottenstein, K. J., & Ottenstein, L. M. (1984). The program dependence graph in a software development environment. *ACM SIGPLAN Notices*, *19*(5), 177–184. https://doi.org/10.1145/390011.808263

Ouriques, J. F. S., Cartaxo, E. G., & Machado, P. D. L. (2013). On the Influence of Model Structure and Test Case Profile on the Prioritization of Test Cases in the Context of Model-Based Testing. In *2013 27th Brazilian Symposium on Software Engineering* (pp. 119–128). https://doi.org/10.1109/SBES.2013.4

Panda, S., Munjal, D., & Mohapatra, D. P. (2016). A Slice-Based Change Impact Analysis for Regression Test Case Prioritization of Object-Oriented Programs, *2016*, 1–25. https://doi.org/10.1155/2016/7132404

Panigrahi, C. R., & Mall, M. (2013). An approach to prioritize the regression test cases of object-oriented programs, *1*(June), 159–173.

Panigrahi, C. R., & Mall, R. (2014). A heuristic-based regression test case prioritization approach for object-oriented programs. *Innovations in Systems and Software Engineering*, *10*(3), 155–163. https://doi.org/10.1007/s11334-013-0221-z

Parashar, P., Kalia, A., & Bhatia, R. (2012). Pair-Wise Time-Aware Test Case Prioritization for Regression Testing. In *ICISTM* (pp. 176–186).

Parejo, J. A., Sánchez, A. B., Segura, S., Ruiz-Cortés, A., Lopez-Herrejon, R. E., & Egyed, A. (2016). Multi-Objective Test Case Prioritization in Highly Configurable Systems: A Case Study. *Journal of Systems and Software*, *Accepted f*. https://doi.org/10.1016/j.jss.2016.09.045

Park, H., Ryu, H., & Baik, J. (2008). Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing. *Proceedings - The 2nd IEEE International Conference on Secure System Integration and Reliability Improvement*, 39–46. https://doi.org/10.1109/SSIRI.2008.52

Pavithra, L., & Sandhya, S. (2019). Survey on Software Testing. Journal of Network Communications and Emerging Technologies (JNCET) Www. Jncet. Org, 9(3).

Pigoski, T. M. (2001). Software Maintenance. In *Swebok* (pp. 1–16).

Pirlot, M. (1996). General local search methods. *European Journal of Operational Research*, *92*(3), 493–511. https://doi.org/10.1016/0377-2217(96)00007-0

Prakash, N., & Gomathi, K. (2014). Improving Test Efficiency through Multiple Criteria Coverage Based Test Case Prioritization. *International Journal of Scientific & Engineering Research*, *5*(4), 420–424.

Raheman, S. R., Rath, A. K., & Bindu, M. H. (2013). An Overview of Program Slicing and its Different Approaches. *Research Forum: International Journal of Social Sciences*, *1*(1), 1–8.

Raju, S., & Uma, G. V. (2012a). An efficient method to achieve effective test case prioritization in regression testing using prioritization factors. *Asian Journal of Information Technology*, *11*(5), 169–180. https://doi.org/10.3923/ajit.2012.169.180

Raju, S., & Uma, G. V. (2012b). Factors oriented test case prioritization technique in regression testing using genetic algorithm. *European Journal of Scientific Research*, *74*(3), 389–402. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84860316467&partnerID=40&md5=a43634427579913af48e406f72717c7c

Raman, B., & Subramani, S. (2015). An Efficient Specific Update Search Domain based Glowworm Swarm Optimization for Test Case Prioritization. *The International Arab Journal of Information Technology*, *12*(6), 748–754.

Rava, M., & Wan-Kadir, W. M. N. (2016). A Review on Automated Regression Testing. *International Journal of Software Engineering and Its Applications*, *10*(1), 221–232.

Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk , Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, *2*(1), 21–33. https://doi.org/doi:10.1515/bile-2015-0008

Rechenberg, I. (1973). Evolutionary Strategy: Optimization of Technical Syetems by means of Biological Evolution. *Fromman-Holzboog, Stuttgart*, *104*, 15–16.

Repository, S. I. (2018a). Java Program to Implement Airline Ticket Reservation System. Retrieved from https://sir.csc.ncsu.edu/content/bios/airline.php

Repository, S. I. (2018b). Java Program to Implement Elevator Control System. Retrieved from https://sir.csc.ncsu.edu/content/bios/elevator.php

Reps, T., & Bricker, T. (1989). Illustrating interference in interfering versions of programs. *{ACM} {SIGSOFT} Software Engineering Notes*, 46–55. https://doi.org/http://doi.acm.org.myaccess.library.utoronto.ca/10.1145/72910.73347

Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (1999). Test Case Prioritization: an Empirical Study. *Proceedings of the IEEE International Conference on Software Maintenance*, 179. https://doi.org/10.1109/ICSM.1999.792604

Rothermel, G., Untch, R. H., Chu, C., Harrold, M. J., & Society, I. C. (2001). Prioritizing Test Cases For Regression Testing Prioritizing Test Cases For Regression Testing. *IEEE Transactions on Software Engineering*, *27*(10), 929–948. https://doi.org/10.1145/347324.348910

Ryan, T. P., & Morgan, J. P. (2007). Modern experimental design. *Journal of Statistical Theory and Practice*, *1*(3–4), 31–51.

Sabharwal, S., Sibal, R., & Sharma, C. (2011). Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams. *International Journal of Computer Science Issues*, *8*(2), 433–444. https://doi.org/http://dx.doi.org/10.1109/ICCCT.2011.6075160

Saha, R. K., Zhang, L., Khurshid, S., & Perry, D. E. (2015). An information retrieval approach for regression test prioritization based on program changes. *Proceedings - International Conference on Software Engineering*, *1*, 268–279. https://doi.org/10.1109/ICSE.2015.47

Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *4*(2), 14.

Sanchez, A. B., Segura, S., & Ruiz-Cortes, A. (2014). A Comparison of Test Case Prioritization Criteria for Software Product Lines. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation* (pp. 41–50). https://doi.org/10.1109/ICST.2014.15

Sawant, A. a, Bari, P. H., & Chawan, P. M. (2012). Software Testing Techniques and Strategies. *Journal of Engineering Research & Applications*, *2*(3), 980–986.

Schwartz, A., & Do, H. (2016). Cost-effective regression testing through Adaptive Test Prioritization strategies. *Journal of Systems and Software, 115*, 61–81. https://doi.org/10.1016/j.jss.2016.01.018

Sethi, N., Rani, S., & Singh, P. (2014). Ants Optimization for Minimal Test Case Selection and Prioritization as to Reduce the Cost of Regression Testing. *International Journal of Computer Applications*, *100*(17), 48–54.

Shahid, Muhammad, & Ibrahim, S. (2014). A New Code Based Test Case Prioritization Technique. *International Journal of Software Engineering and Its Applications*, *8*(6), 31–38.

Shahid, Muhmmad, & Ibrahim, S. (2016). Change impact analysis with a software traceability approach to support software maintenance. In *13th International COnference on Applied Sciences and Technology* (pp. 391–396).

Shameem, A. M., & Kanagavalli, N. (2013). Dependency Detection for Regression Testing using Test Case Prioritization Techniques. *International Journal of Computer Applications*, *65*(14), 20–25.

Shamsiri, S., Rojas, J. M., Gazzola, L., Fraser, G., McMinn, P., Mariani, L., & Arcuri, A. (2017). Random or Evolutionary Search for Object-Oriented Test Suite Generation. *Software Testing, Verification and Reliability*, *22*(2), 1–29. https://doi.org/10.1002/stvr

Shapiro, S. S., Wilk, M. B., & Chen, H. J. (1968). A Comparative Study of Various Tests for Normality. *Journal of the American Statistical Association*, *63*(324), 1343–1372. https://doi.org/10.1080/01621459.1968.10480932

Sharma, C., Sabharwal, S., & Sibal, R. (2013). A Survey on Software Testing Techniques using Genetic Algorithm. *International Journal of Computer Science Issues*, *10*(1), 381–393.

Sharma, N., Sujata, & Purohit, G. N. (2015). Test case prioritization techniques "an empirical study." *2014 International Conference on High Performance Computing and Applications, ICHPCA 2014*. https://doi.org/10.1109/ICHPCA.2014.7045344

Sharma, P. (2014). Automated Software Testing Using Metahurestic Technique Based on an Ant Colony Optimization. *International Journal on Recent and Innovation Trends in Computing and Communication*, *2*(11), 3505–3510.

Sharma, S. (2012). A Genetic Algorithm for Regression Test Sequence Optimization. *International Journal of Advanced Research in Computer and Communication Engineering*, *1*(7), 478–481. Retrieved from www.ijarcce.com

Sharma, S., & Gera, P. (2014). Test Case Prioritization in Regression Testing using Various Metrics. *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, *4*(2), 166–173.

Sharma, T., & Spinellis, D. (2018). A survey on software smells. *Journal of Systems and Software*, *138*, 158–173.

Shu, G., Sun, B., Henderson, T. A. D., & Podgurski, A. (2013). JavaPDG: A new platform for program dependence analysis. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013*, 408–415. https://doi.org/10.1109/ICST.2013.57

Singh, Y., & Goel, B. (2007). A Step Towards Software Preventive Maintenance. *ACM SIGSOFT Software Engineering Notes*, *32*(4), 1–5.

Singh, Y., Kaur, A., & Malhotra, R. (2009). Empirical validation of object-oriented metrics for predicting fault proneness models. *Software Quality Journal*, *18*(1), 3–35. https://doi.org/10.1007/s11219-009-9079-6

Sinha, S., Harrold, M. J., & Rothermel, G. (1999). System-dependence-graph-based slicing of programs with arbitrary interprocedural control flow. *Software Engineering, 1999. Proceedings of the 1999 International Conference On*, *1999*(May 1999), 432–441. https://doi.org/10.1145/302405.302675

Solanki, K., Singh, Y. V, & Dalal, S. (2016). A comparative evaluation of “m-ACO” technique for test suite prioritization. *Indian Journal of Science and Technology*, *9*(30), 1–10. https://doi.org/10.17485/ijst/2016/v9i30/86423

Srikanth, H., & Banerjee, S. (2012). Controversy Corner Improving test efficiency through system test prioritization. *Journal of Systems and Software*, *85*(5), 1176–1187. https://doi.org/10.1016/j.jss.2012.01.007

Srikanth, H., Hettiarachchi, C., & Do, H. (2016). Requirements based test prioritization using risk factors: An industrial study. *Information and Software Technology*, *69*, 71–83. https://doi.org/10.1016/j.infsof.2015.09.002

Srivastava, P. R., & Kim, T. (2009). Application of Genetic Algorithm in Software Testing. *Intenational Journal of Software Engineering and Its Applications*, *3*(4), 87–96. https://doi.org/10.1007/978-3-642-14306-9_54

Srividhya, J., & Gunasundari, R. (2016). Gravitational Bee Search Algorithm with Fuzzy Logic for Effective Test Suite Minimization and Prioritization. *Indian Journal of Science and Technology*, *9*(40). https://doi.org/10.17485/ijst/2016/v9i40/99314

Staats, M., Loyola, P., & Rothermel, G. (2012). Oracle-centric test case prioritization. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE* (pp. 311–320). https://doi.org/10.1109/ISSRE.2012.13

Steel, T., & Dickson. (1996). *Principles and procedures of statistics a biometrical approach* (Third Edit). Department of Ststistics, the Ohio State University, USA: Jason C. Hsu Chapman Hall.

Stefik, M., & Bobrow, D. G. (1985). Object-Oriented Programming: Themes and Variations. *The AI MagazineAI Magazine*, *6*(4), 40–62. https://doi.org/10.1609/aimag.v6i4.508

Stevenson, J., & Wood, M. (2018). Recognising object-oriented software design quality: a practitioner-based questionnaire survey. *Software Quality Journal*, *26*(2), 321–365. 017-9364-8

Sultan, A. B. M., Abdul Ghani, A. A. Bin, Baharom, S., & Musa, S. (2014). An Evolutionary Regression Test Case Prioritization based on Dependence Graph and Genetic Algorithm for Object-Oriented Programs. In *International Conference on Emerging Trends in Engineering and Technology (ICETET, 2014)* (pp. 22–26). London, UK.

Suppriya, M., & Ilavarasi, A. K. (2015). Test Case Selection and Prioritization Using Multiple Criteria. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(10), 280–283.

Szabo, C. (2015). Novice Code Understanding Strategies during a Software Maintenance Assignment. *Proceedings - International Conference on Software Engineering*, *2*, 276–284. https://doi.org/10.1109/ICSE.2015.341

Tahat, L., Korel, B., Harman, M., & Ural, H. (2012). Regression test suite prioritization using system models. *Software Testing Verification and Reliability*, *24*(8), 591–592. https://doi.org/10.1002/stvr

Tahvili, S., Afzal, W., Saadatmand, M., Bohlin, M., Sundmark, D., & Larsson, S. (2016). Towards Earlier Fault Detection by Value-Driven Prioritization of Test Cases Using Fuzzy TOPSISs. *Advances in Intelligent Systems and Computing*, *448*, 1135–1144.

Tulasiraman, M., & Kalimuthu, V. (2018). Cost Cognizant History Based Prioritization of Test Case for Regression Testing Using Immune Algorithm. *Journal of Intelligent Engineering and Systems*, *11*(1), 221–228. https://doi.org/10.22266/ijies2018.0228.23

Tyagi, M., & Malhotra, S. (2015). An Approach for Test Case Prioritization Based on Three Factors. *International Journal of Information Technology and Computer Science*, *7*(4), 79–86. https://doi.org/10.5815/ijitcs.2015.04.09

Utting, M., Pretschner, A., & Legeard, B. (2006). A Taxonomy of Model-Based Testing. Department of Computer Science, the Univ. of Waikato Working paper series. https://doi.org/10.1.1.100.7032

Velmurugan, P., & Mahapatra, R. P. (2016). Effective Branch and DU Coverage Testing through Test Case Prioritization using Genetic Algorithm. *Journal of Theoretical and Applied Information Technology*, *90*(1).

Walcott, K. R., Soffa, M. Lou, Kapfhammer, G. M., & Roos, R. S. (2006). TimeAware test suite prioritization. *Proceedings of the 2006 International Symposium on Software Testing and Analysis - ISSTA'06*, 1. https://doi.org/10.1145/1146238.1146240

Walkinshaw, N., Roper, M., & Wood, M. (2003). The Java system dependence graph. *Proceedings - 3rd IEEE International Workshop on Source Code Analysis and Manipulation, SCAM 2003*, 55–64.

Wang, H., Xing, J., Yang, Q., Han, D., & Zhang, X. (2015). Modification Impact Analysis Based Test Case Prioritization for Regression Testing of Service-Oriented Workflow Applications. In *2015 IEEE 39th Annual Computer Software and Applications Conference* (pp. 288–297).

Wang, H., Xing, J., Yang, Q., Wang, P., Zhang, X., & Han, D. (2017). Optimal control based regression test selection for service-oriented workflow applications. *Journal of Systems and Software*, *124*, 274–288.

Wang, R., Lu, Y., Qu, B., & Zhang, Y. (2016). Empirical study of the effects of different profiles on regression test case reduction. *Mathematical Problems in Engineering*, *9*(2), 29–38. https://doi.org/10.1049/iet-sen.2014.0008

Wang, Shuai, Ali, S., Gotlieb, A., & Liaaen, M. (2016). A systematic test case selection methodology for product lines: results and insights from an industrial case study. *Empirical Software Engineering*, *21*(4), 1586–1622. https://doi.org/10.1007/s10664-014-9345-5

Wang, Song, Nam, J., & Tan, L. (2017). QTEP: quality-aware test case prioritization. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017* (pp. 523–534).

Weiser, M. (1984). Program Slicing. *IEEE Transactions on Software Engineering*, *SE-10*(4), 352–357.

Whitley, D. (2001). An overview of evolutionary algorithms: Practical issues and common pitfalls. *Information and Software Technology*, *43*(14), 817–831. https://doi.org/10.1016/S0950-5849(01)00188-4

Whitley, Darrell. (2001). An overview of evolutionary algorithm: practical Issues and common pitfalls. *Information and Software Technology*, *43*, 817–831. https://doi.org/10.1016/S0950-5849(01)00188-4

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. L. (1999). A Comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*(3), 255–282. https://doi.org/10.1016/S0953-5438(98)00029-0

Wohlin, C., Runeson, P., H¨ost, M., Ohlsson, M. C., Regnell, B., & Wessl´en, A. (2012). *Experimentation in SOftware Engineering*. London: Springer Heidelberg New York Dordrecht. https://doi.org/10.15713/ins.mmj.3

Wong, W. E., Horgan, J. R., London, S., & Mathur, A. P. (1998). Effect of Test Set Minimization on Fault Detection Effectiveness. *Software Practice and Experience*, *28*(July 1996), 347–369.

Wong, W. E., Mathur, A. P., & Maldonado, J. C. (2013). Mutation versus All-uses: An Empirical Evaluation of Cost, Strength and Effectiveness. *Software Quality and Productivity*, *3*(1), 258–265.

Wu, C. T. (2006). An Introduction to object-oriented programming with Java TM. McGraw-Hill Incorporated.

Wu, H., Nie, C., & Kuo, F. C. (2014). Test suite prioritization by switching cost. In *Proceedings - IEEE 7th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2014* (pp. 133–142). https://doi.org/10.1109/ICSTW.2014.15

Wu, K., Fang, C., Chen, Z., & Zhao, Z. (2012). Test Case Prioritization Incorporating Orederd Sequence of Program Elements. In *AST '12 Proceedings of the 7th International Workshop on Automation of Software Test* (pp. 124–130).

Xu, B., Qian, J., Zhang, X., Wu, Z., & Chen, L. (2005). A brief survey of program slicing. *SIGSOFT Softw. Eng. Notes*, *30*(2), 1–36. https://doi.org/http://doi.acm.org/10.1145/1050849.1050865

Yahaya, N. A., Al-Fawareh, H. J., & Abdul, A. A. (2001). Concepts for slicing object-oriented programs. *Malaysian Journal of Computer Science*, *14*(2), 106–115.

Yoo, S., & Harman, M. (2010). Regression testingminimization, selection and prioritization: asurvey. *Software Testing Verification and Reliability*, *22*, 67–120. https://doi.org/10.1002/stvr

Yoo, Shin, & Harman, M. (2007). Pareto efficient multi-objective test case selection. *Proceedings of the 2007 International Symposium on Software Testing and Analysis - ISSTA '07*, 140. https://doi.org/10.1145/1273463.1273483

Yu, Y. T., & Lau, M. F. (2012). Fault-based test suite prioritization for specification-based testing. *Information and Software Technology*, *54*(2), 179–202. https://doi.org/10.1016/j.infsof.2011.09.005

Zabinsky, Z. (2011). Random Search Algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 1–16. https://doi.org/10.1002/9780470400531.eorms0704

Zhang, C., Groce, A., & Alipour, M. A. (2014). Using test case reduction and prioritization to improve symbolic execution. *Proceedings of the 2014 International Symposium on Software Testing and Analysis - ISSTA 2014*, 160–170. https://doi.org/10.1145/2610384.2610392

Zhang, L., Hao, D., Zhang, L., Rothermel, G., & Mei, H. (2013). Bridging the gap between the total and additional test-case prioritization strategies. In *Proceedings - International Conference on Software Engineering* (pp. 192–201). https://doi.org/10.1109/ICSE.2013.6606565

Zhang, X, Xie, X., & Chen, T. Y. (2016). Test Case Prioritization Using Adaptive Random Sequence with Category-Partition-Based Distance. In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)* (pp. 374–385). https://doi.org/10.1109/QRS.2016.49

Zhang, Xiaohong, Wang, Z., Zhang, W., Ding, H., & Chen, L. (2015). Spectrum-Based Fault Localization Method with Test Case Reduction. *2015 IEEE 39th Annual Computer Software and Applications Conference*, 548–549. https://doi.org/10.1109/COMPSAC.2015.272

Zhang, Z. H., Mu, Y. M., & Tian, Y. A. (2012). Test case prioritization for regression testing based on function call path. In *Proceedings - 4th International Conference on Computational and Information Sciences, ICCIS 2012* (pp. 1372–1375). ICCIS 2012. https://doi.org/10.1109/ICCIS.2012.312

Zhou, Z. Q., Sinaga, A., & Susilo, W. (2012). On the fault-detection capabilities of adaptive random test case prioritization: Case studies with large test suites. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (pp. 5584–5593). https://doi.org/10.1109/HICSS.2012.454