UNIVERSITI PUTRA MALAYSIA

AN OPTIMIZED TEST CASE GENERATION TECHNIQUE FOR ENHANCING STATE-SENSITIVITY PARTITIONING

AMMAR MOHAMMED DAWOOD SULTAN

FSKTM 2018 9

AN OPTIMIZED TEST CASE GENERATION TECHNIQUE FOR ENHANCING STATE-SENSITIVITY PARTITIONING

By

AMMAR MOHAMMED DAWOOD SULTAN

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

November 2017

© COPYRIGHT UPM

All material contained within the thesis, including without limitation text, logos, icons, photographs and all other artwork, is copyright material of Universiti Putra Malaysia unless otherwise stated. Use may be made of any material contained within the thesis for non-commercial purposes from the copyright holder. Commercial use of material may only be made with the express, prior, written permission of Universiti Putra Malaysia.

Copyright © Universiti Putra Malaysia

© COPYRIGHT UPM

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

AN OPTIMIZED TEST CASE GENERATION TECHNIQUE FOR ENHANCING STATE-SENSITIVITY PARTITIONING

By

AMMAR MOHAMMED DAWOOD SULTAN

November 2017

Chair: Salmi binti Baharom, PhD
Faculty: Computer Science and Information Technology

Software testing is a vital phase in software development life cycle (SDLC) and its principal element is test case. Test case generation remains the most dominant the research area in software testing. One of the techniques that were proposed for generating test cases is State Sensitivity Partitioning (SSP). It aims to avoid the exhaustive testing of module's entire states. It partitions the entire data states based on their sensitivities towards events, conditions and actions. The test data for SSP is in the form of event sequences. As there is no limit on the number of events that any sequence can hold, lengthy test cases might be generated. Besides, no constrains were applied in order to avoid retesting a component that was already tested. Subsequently, a state explosion might be encountered.

The aim of this study was to address the problem of redundant states encountered within SSP test cases. An optimization technique was proposed, enSSP, featuring the generation of optimized test cases. The scope of this work is testing a module with memory where each module may consist of several programs. The essence of enSSP is to combine the features of Genetic Algorithm (GA) with a suite reduction technique to achieve optimization. GA removes redundant states from test cases while the reduction technique removes redundant sequences from the suite. Afterwards, a prioritization algorithm used for sorting the test cases so the first test case detects the highest number of mutants followed by the cases that kill its live mutants. Experiments were conducted using mutation analysis to compare the fault detection capabilities of enSSP and SSP. The main interest of the experiment is to demonstrate the capability of enSSP. With respect to both quality attributes, the effectiveness and the efficiency, the results indicate that enSSP is more effective and efficient than SSP.

© COPYRIGHT UPM

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia
sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

TEKNIK PENJANAAN UJIAN KES DIOPTIMUMKAN BAGI MENINGKATKAN PEMBAHAGIAN SENSITIVITI KEADAAN

Oleh

AMMAR MOHAMMED DAWOOD SULTAN

November 2017

Pengerusi: Salmi binti Baharom, PhD
Fakulti: Sains Komputer dan Teknologi Maklumat

Ujian perisian adalah sangat penting dalam Kitar Hidup Pembangunan Perisian (SDLC) dan bahan utamanya ialah ujian kes. Penjanaan ujian kes kekal menjadi paling dominan dalam penyelidikan ujian perisian. Salah satu teknik yang dicadangkan untuk penjanaan ujian kes ialah Fasa Sensitiviti Pembahagian (SSP). Ia bertujuan untuk mengelak ujian keseluruhan terhadap fasa model tersebut. Ia membahagikan keseluruhan keadaan data berdasarkan sensitivitinya terhadap situasi, keadaan dan tindakan. Ujian data untuk SSP adalah dalam bentuk urutan keadaan yang bersiri. Memandangkan tiada had maksimum untuk saiz urutan keadaan bersiri, jangka masa yang lama mungkin diperlukan untuk melaksanakan keseluruhannya. Tambahan pula, tiada kekangan dijalankan untuk mengelak daripada pengulangan ujian komponen yang telah diuji. Justeru itu, fasa ujian yang sangat besar mungkin berlaku.

Tujuan kajian ini adalah untuk menangani masalah fasa pengulangan yang berlaku dalam ujian kes SSP. Maka satu teknik yang optimum dicadangkan, enSSP adalah satu teknik yang bercirikan generasi ujian kes SSP yang telah dioptimumkan. Bidang kajian ini adalah untuk menguji modul yang mengandungi memori di mana setiap modul mempunyai beberapa program di dalamnya. Keistimewaan enSSP adalah ia menggabungkan ciri-ciri Genetik Algoritma (GA) dengan suatu set program teknik pengurangan untuk mencapai kesan optimum. GA menyingkirkan fasa pengulangan daripada ujian kes yang dijalankan manakala teknik pengurangan bertindak menghapuskan urutan pengulangan daripada set program tersebut. Seterusnya, satu algoritma digunakan untuk menyusun ujian kes tersebut supaya penyusunan akan bermula dengan ujian kes yang dapat mengesan jumlah mutan yang tertinggi dan diikuti oleh ujian kes yang dapat menghapuskan mutan hidup yang lain. Eksperimen telah dijalankan dengan menggunakan mutan analisis untuk membandingkan kebolehan mengesan mutan diantara enSSP dan SSP. Tujuan utama eksperimen untuk membuktikan keupayaan enSSP. Melalui perbandingan dari segi atribut kualiti

© COPYRIGHT UPM

antara enSSP dan SSP, keputusan menunjukkan enSSP lebih efektif dan efisien berbanding SSP.

© COPYRIGHT UPM

ACKNOWLEDGEMENTS

In the Name of Allah, Most Gracious, Most Merciful.

All praise and gratitude to Allah for His blessing, mercy, and guidance in complete this work and seen me in every moment of my life.

I would like to express my gratitude to Dr. Salmi binti Baharom for her constant encouragement, support, and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout the work.

I am deeply indebted to my committee members Prof. Abdul Azim Abd Ghani, Dr. Jamilah Din and Dr. Hazura Zulzalil for their encouragement, support, and invaluable suggestions.

© COPYRIGHT UPM

I certify that a Thesis Examination Committee has met on 14 November 2017 to conduct the final examination of Ammar Mohammed Dawood Sultan on his thesis entitled "An Optimized Test Case Generation Technique for Enhancing State-Sensitivity Partitioning" in accordance with the Universities and University Colleges Act 1971 and the Constitution of the Universiti Putra Malaysia [P.U.(A) 106] 15 March 1998. The Committee recommends that the student be awarded the Doctor of Philosophy.

Members of the Thesis Examination Committee were as follows:

Rusli bin Hj. Abdullah, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

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

Rodziah binti Atan, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

John Collis Grundy, PhD
Professor
Deakin University
Australia
(External Examiner)

NOR AINI AB. SHUKOR, PhD
Professor and Deputy Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 28 December 2017

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

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

Abdul Azim Abd 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)

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

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

Date:

© COPYRIGHT UPM

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 property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before 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.: Ammar Mohammed Dawood Sultan (GS26764)

© COPYRIGHT UPM

Declaration by Members of 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 Chairman of Supervisory Committee: _____

Signature: _____
Name of Member of Supervisory Committee: _____

Signature: _____
Name of Member of Supervisory Committee: _____

Signature: _____
Name of Member of Supervisory Committee: _____

© COPYRIGHT UPM

TABLE OF CONTENTS

© COPYRIGHT UPM

© COPYRIGHT UPM

© COPYRIGHT UPM

LIST OF TABLES

© COPYRIGHT UPM

© COPYRIGHT UPM

LIST OF FIGURES

© COPYRIGHT UPM

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AI	Artificial Intelligence
APFD	Average Percentage of Fault Detection
BVA	Boundary Value Analysis
BVR	Case Based Reasoning
df	Degree of Freedom
DU	Define Use
EDS	Event Driven Systems
enSSP	Enhanced State Sensitivity Partitioning
FSM	Finite State Machine
GA	Genetic Algorithm
GDP	Gross Domestic Product
HC	Hill Climbing
LM	Live Mutant
LR	Literature Review
MC/DC	Modified Condition/Decision Coverage
MIDD	Module Internal Design Document
MIS	Module Interface Specification
NIST	National Institute of Standards and Technology
OO	Object Oriented
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SBSE	Search Based Software Engineering
SBST	Search Based Software Testing
SDLC	Software Development Life Cycle
SSP	State Sensitivity Partitioning
SUT	Software Under Test
TS	Tabu Search

© COPYRIGHT UPM

© COPYRIGHT UPM

CHAPTER 1

INTRODUCTION

1.1 Overview

The undertaken research presents a technique for generating optimal test cases in the form of sequences of events. It aims to overcome the problem of redundant states encountered in the suites generated by state-sensitivity partitioning (SSP) technique. Thus, the proposed technique minimizes the test suite while preserving its state coverage. The minimization was performed at two levels: test case and test suite. The former level targets the redundant states per sequence while the latter one focuses on the redundant states within the suite. Also, this study considers detecting all bugs within the fewest number of test cases rather than waiting until the end of the suite. This is done through prioritizing the minimized suite. Hence, this chapter presents a background on the investigated topic, an overview of the problem statement, research objectives, contributions and thesis organization.

1.2 Background

Software testing is one of the most crucial phases in the software development life cycle (SDLC). It stands for the process of executing a system or program with the intention of finding errors (Myers, Sandler, & Badgett, 2011). However, the software testing is a labor-intensive process that requires an extensive effort and consumes time, accounting about 50% of the total cost of software development (Pressman & Maxim, 2014). This includes detecting errors as much as possible within the specified time and validates that the software was built as intended by the users. Test case is the principal element for testing. It is a group of tests that are performed sequentially with the goal of meeting a test objective. Every test case produces a number of tests that comprise specific input values, observed output, expected output, and any further information that might be needed for running the test, such as environmental prerequisites (Fewster & Graham, 1999; Jorgensen, 2013; Mala & Mohan, 2007). The quality of test cases is evaluated through two factors, which are effectiveness and efficiency. The effectiveness stands for the coverage of the generated test cases with the goal of covering as much as possible of the software under test (SUT) with the minimum number of test cases. On the other hand, efficiency stands for the capability of detecting all mutants with the fewest number of test cases rather than waiting till the last test case in the suite to detect all mutants.

However, the classical way of generating test cases is exhaustive testing which focuses on examining all possible combinations of inputs and preconditions with the goal of finding errors. Although it may be appropriate for small systems, exhaustive testing is infeasible, especially for non-trivial systems such as health

© COPYRIGHT UPM

care systems. This adds more complexity for errors detection (Black, Graham, & Veenendaal, 2012).

Relatively, several techniques were proposed for generating test cases non-exhaustively. One of these techniques is State Sensitivity Partitioning (SSP) technique. It was proposed with the goal of avoiding the exhaustive testing of module's entire states. It was invented by Baharom and Shukur based on Parnas formal specifications (Baharom & Shukur, 2008, 2010, 2011). SSP focuses on a module that consists of one or more access programs, which share the same data structure. As a result, the test data for a module might include a sequence of events rather than a single event. The overall mechanism of SSP includes partitioning the entire data states of the SUT based on their sensitivities towards events, conditions, and actions. It is based on all-transition coverage criterion where the test cases are manually selected. As there is no limit on the number of events that any sequence can hold, lengthy test cases might be generated. Besides, no constraints were applied in order to avoid retesting a component that was already tested. Consequently, a state explosion might be encountered as a result of an infinite or lengthy sequence of events that modify the internal data states. This will result in wasting testing efforts and times. Hence, there is a need to optimize the test cases so the states of SUT are covered with the minimal number of events. Accordingly, test suites have to be optimized, too, with the goal of including the minimal number of test cases that cover the SUT.

As a result, several enhancements were proposed such as selecting a representative set of test cases based on heuristics to represent the test suite (Jeffrey & Gupta, 2005). Other researchers introduced a greedy algorithm for selecting test cases with the minimal redundancy whilst satisfying the maximum testing requirements (Parsa & Khalilian, 2010). Sapaat and Baharom (Sapaat & Baharom, 2011) proposed matrices to be used for analyzing the data flow of the SUT and eliminate its redundancies. Others adopted search techniques for the sake of generating optimized test cases (Alsmadi, Alkhateeb, Maghayreh, Samarah, & Doush, 2010; Kulkarni, Naveen, Singh, & Srivastava, 2011).

Search techniques, also known as optimization techniques, were proposed for generating optimal test cases and prioritizing test cases such as genetic algorithm and tabu search. Each technique has its own strengths and weaknesses. However, among search techniques, genetic algorithm (GA) is the most common technique employed for generating test cases (Ali, Briand, Hemmati, & Panesar-Walawege, 2010). This is inspired by its simplicity and quality of results it provides, compared with other search techniques.

The aforementioned issues in SSP provide a fundamental motivation for this research to look into this problem of what is an effective and efficient strategy to generate optimal test cases that cover all the SUT states and provides a faults detection with the minimum number of cases. Subsequently, a prioritization technique was applied on the minimized suite so that all bugs can be detected

with the fewest number of test cases. The tradeoff between effectiveness and efficiency has to be considered in order to achieve such a goal.

1.3 Problem Statement

SSP technique generates test cases for modules that consist of one or more access programs and share the same data structure. An access program is a program that is part of a module that can be used from outside the module. The module testing provides better coverage and higher detection rates of errors compared with other testing levels (McDonald, Murray, & Strooper, 1997). As it is impossible to exhaustively test all elements and their combinations, SSP partitions the entire data states based on the state's sensitivity towards events, conditions (pre-conditions) and actions (post-conditions).

However, SSP does not apply constraints on the generated sequences of events. Thus, the sequences might be lengthy with a number of redundant states (Sapaat & Baharom, 2011). A prior work towards minimizing SSP test suites by (Sapaat & Baharom, 2011) focused on reducing states redundancies which result from having a sequence that is subset from other sequences within the same suite. Nonetheless, another type of redundancies may exist in the sequence itself. This type results from having events that do not affect the SUT state. For example, adding to a full queue will repeat the full queue state and will not cause a new state to be generated.

So, SSP states redundancies can exist at two levels: test case and test suite. Redundant states in the test case level results from having sensitive events that generate redundant states (i.e. will not affect the states of SUT in the same sequence of events). On the other hand, having test cases that present subset (part of) from others contribute to the redundancy in test suite as they produce identical states. Consequently, there is a need to optimize the sequences of events considering the SUT states rather than the events only.

Unlike redundancies in the test suite level, some test cases include redundancies for the purpose of testing and these cases have to be preserved. For example, adding an item to a full queue results in a redundant state, but this has to be kept as part of the test. As a result, there is a need to feature the redundancies in the test case level along with the test suite level considering which redundancies are needed and which have to be eliminated.

Besides, SSP does not provide a specific order of the generated suite which means that there is a need to wait until the end of the suite in order to identify all system bugs. Consequently, there is a need to detect all bugs with the first few test cases rather than waiting till completing the suite.

© COPYRIGHT UPM

1.4 Research Objectives

The objectives of this research are as follows:

1. To propose an optimization technique for generating minimized and prioritized test suite to test a component that consists of several functions sharing the same data structure.
2. To develop a prototype tool for the optimization technique.
3. To empirically evaluate the effectiveness and efficiency of the proposed technique in detecting faults.

1.5 Scope of Research

A test case is defined by IEEE standards (IEEE, 2010) as a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. In this research, the focus is on test cases that are composed of sequences of events. Other forms of test cases will not be considered.

Besides, this research focuses on SSP with the purpose of enhancement. SSP is module-based test case generation technique targeting modules that have memory. More specifically, it focuses on modules that store values as a result of specific triggers (i.e. sensitive events). It is based on Parnas specification documentations, specifically, Module Internal Design Document (MIDD). MIDD specifies the relations between SUT aspects. SSP combines the advantages of both Black-box and White-box. Similar to SSP, this research considers testing modules that have memories. Every module has a private data structure and one or more access programs. Hence, this research targets components that consist of several functions sharing a private data structure with the goal of avoiding exhaustive testing.

1.6 Contributions of the Thesis

This thesis has made the following contributions:

i. It defined a technique for generating optimized test cases which are both effective and efficient.
ii. It implemented a tool support for the test generation process suggested by the technique. Furthermore, the implemented tool, not only can be used to generate tests for SSP but also can be applied in any related experimental work.
iii. It provides empirical evidence that the proposed technique can be effective in testing and test cases generation compared with the original SSP.
iv. The comparison and evaluation results obtained from an empirical study made the advantages and disadvantages of the current SSP obvious for potential users. This contributes to a direction that can provide the basis of guidelines for testing practitioners for choosing techniques suiting their purposes and constraints.

© COPYRIGHT UPM

1.7 Organization of the Thesis

This thesis is divided into seven chapters. The first chapter is the introduction of the thesis. It describes the problem background and statement, research objectives, scope of the research and contributions of the thesis.

The second chapter is the literature review. It presents a detailed study of the existing test case generation techniques alongside SSP, which are the key areas to lay the foundation of this work. This chapter also highlights gaps in the related literature.

The third chapter is the methodology. It outlines a general overview of the research methods and materials used to define an optimized test generation technique, to implement the prototype tool support, and finally to conduct the empirical evaluation and analysis of the newly proposed technique. More specific details of how each objective was accomplished are presented in the respective chapters.

The fourth chapter proposes an optimized test generation technique, specifically targeted towards enhancing SSP by generating test data to exercise the advice and pointcuts. An illustrative case study on how the proposed technique can be used to generate tests are provided.

The fifth chapter is the implementation of the technique tool. The purpose of the tool is to automate the process of test generation and execution.

The sixth chapter presents a comprehensive experiment whose aim was to provide empirical evidence to evaluate the proposed technique, in terms of effectiveness and efficiency.

The seventh chapter is the conclusion and future work. It gives a general conclusion of the research presented in this thesis and also proposes some research directions that can be investigated as future work.

© COPYRIGHT UPM

REFERENCES

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

Ahmad, J., & Baharom, S. (2017). A Systematic Literature Review of the Test Case Prioritization Technique for Sequence of Events. *International Journal of Applied Engineering Research*, *12*(7), 1389–1395.

Ahmed, B. S. (2016). Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, *19*(2), 737–753. https://doi.org/10.1016/j.jestch.2015.11.006

Ali, S., Briand, L. C., Hemmati, H., & Panesar-Walawege, R. K. (2010). A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. *IEEE Transactions on Software Engineering*, *36*(6), 742–762. https://doi.org/10.1109/tse.2009.52

Alsmadi, I., Alkhateeb, F., Maghayreh, E. A., Samarah, S., & Doush, I. A. (2010). Effective Generation of Test Cases Using Genetic Algorithms and Optimization Theory. *Journal of Communication and Computer*, *7*(11), 72–82.

Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., … Mcminn, P. (2013). An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software*, *86*(8), 1978–2001. https://doi.org/10.1016/j.jss.2013.02.061

Andrews, J. H., Briand, L. C., & Labiche, Y. (2005). Is mutation an appropriate tool for testing experiments? (pp. 402–411). Presented at the Proceedings of the 27th International Conference on Software Engineering (ICSE '05), Saint Louis, MO, USA: IEEE. https://doi.org/10.1109/ICSE.2005.1553583

Andrews, J. H., Briand, L. C., Labiche, Y., & Namin, A. S. (2006). Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria. *IEEE Transactions on Software Engineering*, *32*(8), 608–624. https://doi.org/10.1109/tse.2006.83

Babbie, E. R. (2016). *The Practice of Social Research* (14th ed.). Boston, MA, USA: Cengage Learning.

Bagnall, A. J., Rayward-Smith, V. J., & Whittley, I. M. (2001). The next release problem. *Information and Software Technology*, *43*(14), 883–890. https://doi.org/10.1016/s0950-5849(01)00194-x

Baharom, S., & Shukur, Z. (2008). Module documentation based testing using Grey-Box approach (Vol. 2, pp. 1–6). Presented at the 3rd International Symposium on Information Technology 2008, ITSim, Kuala Lumpur, Malaysia: IEEE. https://doi.org/10.1109/ITSIM.2008.4631651

Baharom, S., & Shukur, Z. (2010). State-Sensitivity Partitioning Technique for Module Documentation-based testing. In K. S. Soliman (Ed.) (Vol. 1, pp. 472–483). Presented at the Business Transformation through Innovation and Knowledge Management: An Academic Perspective - Proceedings of the 14th International Business Information Management Association

© COPYRIGHT UPM

Conference, IBIMA 2010, Istanbul: International Business Information Management Association, IBIMA.

Baharom, S., & Shukur, Z. (2011). An experimental assessment of module documentation-based testing. *Information and Software Technology*, *53*(7), 747–760. https://doi.org/10.1016/j.infsof.2011.01.005

Basili, V. R., & Selby, R. W. (1987). Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, *13*(12), 1278–1296. https://doi.org/10.1109/TSE.1987.232881

Baskarada, S. (2009). *IQM-CMM: Information Quality Management Capability Maturity Model* (2010 edition). Weisbaden, Germany: Vieweg+Teubner Verlag.

Black, J., Melachrinoudis, E., & Kaeli, D. (2004). Bi-criteria models for all-uses test suite reduction (pp. 106–115). Presented at the Proceedings of the 26th International Conference on Software Engineering (ICSE '04), Edinburgh, UK: IEEE Computer Society. https://doi.org/10.1109/ICSE.2004.1317433

Black, R., Graham, D., & Veenendaal, E. V. (2012). *Foundations of Software Testing: ISTQB Certification* (3rd ed.). Cengage Learning.

Bogacki, B., & Walter, B. (2006). Evaluation of Test Code Quality with Aspect-Oriented Mutations. In P. Abrahamsson, M. Marchesi, & G. Succi (Eds.), *Extreme Programming and Agile Processes in Software Engineering: 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006. Proceedings* (pp. 202–204). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/11774129_26

Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011). A proposed test case generation technique based on activity diagrams. *International Journal of Engineering & Technology IJET-IJENS*, *11*(03).

Bradbury, J. S. (2007). *Using program mutation for the empirical assessment of fault detection techniques: a comparison of concurrency testing and model checking* (Doctoral Dissertation). Queen's University, Kingston, Ont., Canada.

Bradbury, J. S., Cordy, J. R., & Dingel, J. (2005). An Empirical Framework for Comparing Effectiveness of Testing and Property-based Formal Analysis. *ACM SIGSOFT Software Engineering Notes*, *31*(1), 2–5. https://doi.org/10.1145/1108768.1108795

Bryce, R. C., & Colbourn, C. J. (2006). Prioritized interaction testing for pair-wise coverage with seeding and constraints. *Information and Software Technology*, *48*(10), 960–970. https://doi.org/10.1016/j.infsof.2006.03.004

Bryce, R. C., Colbourn, C. J., & Cohen, M. B. (2005). A Framework of Greedy Methods for Constructing Interaction Test Suites (pp. 146–155). Presented at the Proceedings of the 27th International Conference on Software Engineering (ICSE '05), St. Louis, MO, USA: ACM. https://doi.org/10.1145/1062455.1062495

Bryce, R. C., & Memon, A. M. (2007). Test Suite Prioritization by Interaction Coverage (pp. 1–7). Presented at the Workshop on Domain Specific Approaches to Software Test Automation: In Conjunction with the 6th ESEC/FSE Joint Meeting (DOSTA '07), Dubrovnik, Croatia: ACM. https://doi.org/10.1145/1294921.1294922

Bryman, A., & Bell, E. (2015). *Business Research Methods* (4th ed.). Oxford University Press.

© COPYRIGHT UPM

Catal, C., & Mishra, D. (2013). Test case prioritization: a systematic mapping study. *Software Quality Journal*, *21*(3), 445–478. https://doi.org/10.1007/s11219-012-9181-z

Chen, L., Wang, Z., Xu, L., Lu, H., & Xu, B. (2010). Test Case Prioritization for Web Service Regression Testing (pp. 173–178). Presented at the Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE '10), Nanjing, China: IEEE. https://doi.org/10.1109/SOSE.2010.27

Chilenski, J. J., & Miller, S. P. (1994). Applicability of modified condition/decision coverage to software testing. *Software Engineering Journal*, *9*(5), 193–193. https://doi.org/10.1049/sej.1994.0025

Clark, I. (2006). *Writing the successful thesis and dissertation: entering the conversation* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Cohen, M. B., Dwyer, M. B., & Jiangfan, S. (2008). Constructing Interaction Test Suites for Highly-Configurable Systems in the Presence of Constraints: A Greedy Approach. *IEEE Transactions on Software Engineering*, *34*(5), 633–650. https://doi.org/10.1109/TSE.2008.50

Conrad, A. P., Roos, R. S., & Kapfhammer, G. M. (2010). Empirically Studying the Role of Selection Operators During search-based Test Suite Prioritization (pp. 1373–1380). Presented at the Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10), Portland, Oregon, USA: ACM. https://doi.org/10.1145/1830483.1830735

Creswell, J. W. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (4th ed.). SAGE Publications.

Dandan, G., Tiantian, W., Xiaohong, S., & Peijun, M. (2013). A test-suite reduction approach to improving fault-localization effectiveness. *Computer Languages, Systems & Structures*, *39*(3), 95–108. https://doi.org/10.1016/j.cl.2013.04.001

Del Grosso, C., Antoniol, G., Merlo, E., & Galinier, P. (2008). Detecting buffer overflow via automatic test input data generation. *Computers & Operations Research*, *35*(10), 3125–3143. https://doi.org/10.1016/j.cor.2007.01.013

DeMillo, R. A., Lipton, R. J., & Sayward, F. G. (1978). Hints on Test Data Selection: Help for the Practicing Programmer. *Computer*, *11*(4), 34–41. https://doi.org/10.1109/c-m.1978.218136

Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems Analysis and Design: An Object-Oriented Approach with UML* (5th ed.). John Wiley & Sons.

Denzin, N. K., & Lincoln, Y. S. (2017). *The SAGE Handbook of Qualitative Research* (5th ed.). CA, USA: SAGE Publications.

Do, H., Rothermel, G., & Kinneer, A. (2004). Empirical studies of test case prioritization in a JUnit testing environment (pp. 113–124). Presented at the Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE '04), IEEE Computer Society. https://doi.org/10.1109/ISSRE.2004.18

Donghua, C., & Wenjie, Y. (2011). The research of test-suite reduction technique (pp. 4552–4554). Presented at the Proceedings of the International Conference on Consumer Electronics, Communications and Networks (CECNet), XianNing, China: IEEE. https://doi.org/10.1109/CECNET.2011.5768284

© COPYRIGHT UPM

Elbaum, S., Gable, D., & Rothermel, G. (2001). Understanding and Measuring the Sources of Variation in the Prioritization of Regression Test Suites (pp. 169–179). Presented at the Proceedings of the 7th International Symposium on Software Metrics (METRICS '01), London, UK: IEEE Computer Society. https://doi.org/10.1109/METRIC.2001.915525

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2000). Prioritizing test cases for regression testing. *ACM SIGSOFT Software Engineering Notes*, *25*(5), 102–112. https://doi.org/10.1145/347636.348910

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.

Elghondakly, R., Moussa, S., & Badr, N. (2016). A Comprehensive Study for Software Testing and Test Cases Generation Paradigms (p. 50:1-50:7). Presented at the Proceedings of the International Conference on Internet of Things and Cloud Computing (ICC '16), Cambridge, United Kingdom: ACM. https://doi.org/10.1145/2896387.2896435

Fewster, M., & Graham, D. (1999). *Software test automation: effective use of test execution tools*. New York, NY, USA: Addison-Wesley.

Fraser, G., & Zeller, A. (2012). Mutation-driven generation of unit tests and oracles. *IEEE Transactions on Software Engineering*, *38*(2), 278–292. https://doi.org/10.1109/TSE.2011.93

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, *14*(5), 771–780.

Gendreau, M., & Potvin, J. Y. (2010). *Handbook of Metaheuristics* (2nd ed., Vol. 146). Springer US.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning* (1st ed.). Addison-Wesley Professional.

Gray, D. E. (2017). *Doing Research in the Real World* (4th ed.). SAGE Publications.

Haidry, S.-Z., & Miller, T. (2013). Using Dependency Structures for Prioritization 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., Zang, L., Wang, Y., Wu, X., & Xie, T. (2016). To Be Optimal or Not in Test-Case Prioritization. *IEEE Transactions on Software Engineering*, *42*(5), 490–505. https://doi.org/10.1109/TSE.2015.2496939

Hao, D., Zhang, L., Zhang, L., Rothermel, G., & Mei, H. (2014). A Unified Test Case Prioritization Approach. *ACM Transactions on Software Engineering and Methodology*, *24*(2), 1–31. https://doi.org/10.1145/2685614

Harder, M., Mellen, J., & Ernst, M. D. (2003). Improving test suites via operational abstraction (pp. 60–71). Presented at the Proceedings of the 25th International Conference on Software Engineering (ICSE '03), Portland, Oregon, USA: IEEE Computer Society. https://doi.org/10.1109/ICSE.2003.1201188

Harman, M., Mansouri, S. A., & Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, King's College London, Technical Report TR-09-03*.

Harman, M., Mansouri, S. A., & Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing*

Surveys (CSUR), *45*(1), 1–61. https://doi.org/10.1145/2379776.2379787

Harris, P., & Raju, N. (2015). Towards test suite reduction using maximal frequent data mining concept. *International Journal of Computer Applications in Technology*, *52*(1), 48–58. https://doi.org/10.1504/ijcat.2015.071419

Harrold, M. J. (1999). Testing evolving software. *Journal of Systems and Software*, *47*(2–3), 173–181. https://doi.org/10.1016/s0164-1212(99)00037-0

Harrold, M. J., Gupta, R., & Soffa, M. L. (1993). A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering and Methodology*, *2*(3), 270–285. https://doi.org/10.1145/152388.152391

Harrold, M. J., & Kolte, P. (1992). Combat: A compiler based data flow testing system (pp. 311–323). Presented at the Proceedings of the 10th Pacific Northwest Software Quality Conference.

Heimdahl, M. P. E., & George, D. (2004). Test-suite reduction for model based tests: effects on test quality and implications for testing (pp. 176–185). Presented at the Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE '04), Linz, Austria: IEEE Computer Society. https://doi.org/10.1109/ASE.2004.1342735

Hierons, R. M., Krause, P., Lüttgen, G., Simons, A. J. H., Vilkomir, S., Woodward, M. R., … Kapoor, K. (2009). Using formal specifications to support testing. *ACM Computing Surveys*, *41*(2), 1–76. https://doi.org/10.1145/1459352.1459354

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (1st ed.). University of Michigan Press.

Horgan, J. R., & London, S. (1992). A data flow coverage testing tool for C (pp. 2–10). Presented at the Proceedings of the Second Symposium on Assessment of Quality Software Development Tools, New Orleans, LA, USA: IEEE. https://doi.org/10.1109/AQSDT.1992.205829

Hsu, H.-Y., & Orso, A. (2009). MINTS: A general framework and tool for supporting test-suite minimization (pp. 419–429). Presented at the Proceedings of the 31st International Conference on Software Engineering (ICSE '09), Vancouver, BC, Canada: IEEE Computer Society. https://doi.org/10.1109/ICSE.2009.5070541

Hu, P., Zhang, Z., Chan, W. K., & Tse, T. H. (2006). An Empirical Comparison Between Direct and Indirect Test Result Checking Approaches (pp. 6–13). Presented at the Proceedings of the 3rd International Workshop on Software Quality Assurance (SOQUA '06), Portland, Oregon, USA: ACM. https://doi.org/10.1145/1188895.1188901

Huang, C.-Y., Chen, C.-S., & Lai, C.-E. (2016). Evaluation and analysis of incorporating Fuzzy Expert System approach into test suite reduction. *Information and Software Technology*, *79*, 79–105. https://doi.org/10.1016/j.infsof.2016.07.005

IEEE. (2010). Systems and software engineering -- Vocabulary, 1–418. https://doi.org/10.1109/IEEESTD.2010.5733835

Isabella, A., & Retna, E. (2012). Study paper on test case generation for GUI based testing. *International Journal of Software Engineering & Applications*, *3*(1), 139–147.

© COPYRIGHT UPM

Jeffrey, D., & Gupta, N. (2005). Test suite reduction with selective redundancy (pp. 549–558). Presented at the Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSḾ05), Budapest, Hungary, Hungary: IEEE Computer Society. https://doi.org/10.1109/ICSM.2005.88

Jeffrey, D., & Gupta, N. (2006). Test Case Prioritization Using Relevant Slices (pp. 411–420). Presented at the Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAĆ06), Chicago, IL, USA: IEEE. https://doi.org/10.1109/COMPSAC.2006.80

Jeffrey, D., & Gupta, N. (2007). Improving fault detection capability by selectively retaining test cases during test suite reduction. *IEEE Transactions on Software Engineering*, *33*(2), 108–123.

Jia, Y., & Harman, M. (2009). Higher Order Mutation Testing. *Information and Software Technology*, *51*(10), 1379–1393. https://doi.org/10.1016/j.infsof.2009.04.016

Jia, Y., & Harman, M. (2011). An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, *37*(5), 649–678. https://doi.org/10.1109/TSE.2010.62

Jones, J. A., & Harrold, M. J. (2003). Test-suite reduction and prioritization for modified condition/decision coverage. *IEEE Transactions on Software Engineering*, *29*(3), 195–209. https://doi.org/10.1109/tse.2003.1183927

Jonker, J., & Pennink, B. (2010). *The Essence of Research Methodology: A Concise Guide for Master and PhD Students in Management Science* (1st ed.). Springer-Verlag Berlin Heidelberg.

Jorgensen, P. C. (2013). *Software Testing: A Craftsman's Approach* (4th ed.). Auerbach Publications.

Juristo, N., & Moreno, A. M. (2010). *Basics of Software Engineering Experimentation* (2001st ed.). Springer US.

Kandl, S. (2015). Cost Effectiveness of Coverage-Guided Test-Suite Reduction for Safety-Relevant Systems. In H. Selvaraj, D. Zydek, & G. Chmaj (Eds.), *Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering* (pp. 595–601). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-08422-0_84

Khalilian, A., Azgomi, M. A., & 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, F. A., Gupta, A. K., & Bora, D. J. (2015). An Efficient Technique to Test Suite Minimization using Hierarchical Clustering Approach. *International Journal of Emerging Science and Engineering (IJESE)*, *3*(11).

Khan, S. U. R., Lee, S. P., Ahmad, R. W., Akhunzada, A., & Chang, V. (2016). A survey on Test Suite Reduction frameworks and tools. *International Journal of Information Management*, *36*(6), 963–975. https://doi.org/10.1016/j.ijinfomgt.2016.05.025

Kim, J.-M., & Porter, A. (2002). A history-based test prioritization technique for regression testing in resource constrained environments (pp. 119–129). Presented at the Proceedings of the 24th International Conference on Software Engineering (ICSE '02), Orlando, FL, USA: IEEE. https://doi.org/10.1109/ICSE.2002.1007961

© COPYRIGHT UPM

Korel, B., Koutsogiannakis, G., & Tahat, L. H. (2007). Model-based Test Prioritization Heuristic Methods and Their Evaluation (pp. 34–43). Presented at the Proceedings of the 3rd International Workshop on Advances in Model-based Testing (A-MOST '07), London, UK: ACM. https://doi.org/10.1145/1291535.1291539

Korel, B., Koutsogiannakis, G., & Tahat, L. H. (2008). Application of system models in regression test suite prioritization (pp. 247–256). Presented at the Proceedings of the International Conference on Software Maintenance (ICSM '08), Beijing, China: IEEE. https://doi.org/10.1109/ICSM.2008.4658073

Krishnamoorthi, R., & Mary, S. A. (2009). Factor oriented requirement coverage based system test case prioritization of new and regression test cases. *Information and Software Technology*, *51*(4), 799–808. https://doi.org/10.1016/j.infsof.2008.08.007

Kulkarni, N. J., Naveen, K. V., Singh, P., & Srivastava, P. R. (2011). Test case optimization using artificial bee colony algorithm (pp. 570–579). Presented at the International Conference on Advances in Computing and Communications, Springer. https://doi.org/10.1007/978-3-642-22720-2_60

Kumar, S. V., & Kumar, M. (2010). Test case prioritization using fault severity. *International Journal of Computer Science and Technology*, *1*(1).

Lakhotia, Kiran, Harman, Mark, & Gross, Hamilton. (2010). AUSTIN: A Tool for Search Based Software Testing for the C Language and Its Evaluation on Deployed Automotive Systems. In *2nd International Symposium on Search Based Software Engineering* (pp. 101–110). https://doi.org/10.1109/SSBSE.2010.21

Leavens, G. T. (2009). The Java Modeling Language (JML) Home Page. Retrieved August 5, 2017, from http://www.eecs.ucf.edu/~leavens/JML//index.shtml

Leedy, P. D., & Ormrod, J. E. (2016). *Practical Research: Planning and Design, Global Edition* (11th ed.). Pearson Education Limited.

Leitner, A., Oriol, M., Zeller, A., Ciupa, I., & Meyer, B. (2007). Efficient Unit Test Case Minimization (pp. 417–420). Presented at the Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering (ASE '07), Atlanta, Georgia, USA: ACM. https://doi.org/10.1145/1321631.1321698

Leon, D., & Podgurski, A. (2003). A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases (pp. 442–453). Presented at the Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE '03), Denver, CO, USA: IEEE Computer Society. https://doi.org/10.1109/ISSRE.2003.1251065

Li, J., Dai, G., & Li, H. (2009). Mutation Analysis for Testing Finite State Machines (pp. 620–624). Presented at the Proceedings of the Second International Symposium on Electronic Commerce and Security (ISECS '09), Nanchang City, China: IEEE. https://doi.org/10.1109/ISECS.2009.158

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

© COPYRIGHT UPM

Liu, S., & Chen, Y. (2008). A relation-based method combining functional and structural testing for test case generation. *Journal of Systems and Software*, *81*(2), 234–248. https://doi.org/10.1016/j.jss.2007.05.036

LocMetrics. (2007, October). LocMetrics - Source Code Line Counting Tool. Retrieved August 5, 2017, from http://www.locmetrics.com/index.html

Ma, Y.-S., Offutt, J., & Kwon, Y.-R. (2006). MuJava: A Mutation System for Java (pp. 827–830). Presented at the Proceedings of the 28th International Conference on Software Engineering (ICSE '06), Shanghai, China: ACM. https://doi.org/10.1145/1134285.1134425

Mala, D. J., Kamalapriya, M., Shobana, R., & Mohan, V. (2009). A non-pheromone based intelligent swarm optimization technique in software test suite optimization (pp. 1–5). Presented at the Proceedings of the International Conference on Intelligent Agent & Multi-Agent Systems (IAMA '09), Chennai, India: IEEE. https://doi.org/10.1109/IAMA.2009.5228055

Mala, D. J., & Mohan, V. (2007). IntelligenTester-Software Test Sequence Optimization Using Graph Based Intelligent Search Agent (pp. 22–27). Presented at the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), Sivakasi, Tamil Nadu, India: IEEE. https://doi.org/10.1109/ICCIMA.2007.161

Mayer, J., & Schneckenburger, C. (2006). An Empirical Analysis and Comparison of Random Testing Techniques (pp. 105–114). Presented at the Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06), Rio de Janeiro, Brazil: ACM. https://doi.org/10.1145/1159733.1159751

McDonald, J., Murray, L., & Strooper, P. (1997). Translating Object-Z Specifications to Object-Oriented Test Oracles (pp. 414–423). Presented at the Proceedings of Joint 4th International Computer Science Conference and 4th Asia Pacific Software Engineering Conference, Hong Kong: IEEE Computer Society. https://doi.org/10.1109/APSEC.1997.640198

Mcmaster, S. D. (2008). *A context-sensitive coverage criterion for test suite reduction* (Doctoral Dissertation). University of Maryland, College Park.

McMinn, P. (2004). Search-based software test data generation: a survey: Research Articles. *Software Testing, Verification and Reliability*, *14*(2), 105–156. https://doi.org/10.1002/stvr.v14:2

McMinn, P., Harman, M., Lakhotia, K., Hassoun, Y., & Wegener, J. (2012). Input Domain Reduction through Irrelevant Variable Removal and Its Effect on Local, Global, and Hybrid Search-Based Structural Test Data Generation. *IEEE Transactions on Software Engineering*, *38*(2), 453–477. https://doi.org/10.1109/TSE.2011.18

Miao, H., Liu, P., Mei, J., & Zeng, H. (2009). A new approach to automated redundancy reduction for test sequences (pp. 93–98). Presented at the Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'09), Shanghai, China: IEEE. https://doi.org/10.1109/PRDC.2009.23

Mirarab, S., & Tahvildari, L. (2007). A Prioritization Approach for Software Test Cases Based on Bayesian Networks. In M. B. Dwyer & A. Lopes (Eds.), *Fundamental Approaches to Software Engineering: 10th International Conference, FASE 2007, Held as Part of the Joint European Conferences, on Theory and Practice of Software, ETAPS*

© COPYRIGHT UPM

2007, Braga, Portugal, March 24 - April 1, 2007. Proceedings (pp. 276–290). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71289-3_22

Momoh, J. A. (2015). *Adaptive Stochastic Optimization Techniques with Applications*. CRC Press.

Moore, I. (2001). Jester a JUnit test tester. In M. Marchesi & G. Succi (Eds.). Presented at the Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering. Retrieved from http://cf.agilealliance.org/articles/system/article/file/881/file.pdf

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation*, *1*(1), 25–49. https://doi.org/10.1162/evco.1993.1.1.25

Mungara, M. B. (2003). *A Method for Systematically Generating Tests for Object-Oriented Class Interfaces* (Master of Science). Virginia Polytechnic Institute & State University, Blacksburg, Virginia, USA.

Myers, G., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). Wiley.

Nam, D. H., Mousset, E. C., & Levy, D. C. (2007). Automating the Testing of Object Behaviour: A Statechart-Driven Approach. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *1*, 3686–3691.

Nardo, D. 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*(4), 371–396. https://doi.org/10.1002/stvr.1572

Ng, P. (2007). A Concept Lattice Approach for Requirements Validation with UML State Machine Model (pp. 393–400). Presented at the Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007), Busan, South Korea. https://doi.org/10.1109/SERA.2007.8

Ng, P., & Fung, R. Y. K. (2009). An Incremental Approach for Model-Based Test Suite Reduction Using Formal Concept Analysis (pp. 1–6). Presented at the Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications (ICUT '09), Fukuoka, Japan: IEEE. https://doi.org/10.1109/ICUT.2009.5405725

Nie, C., & Leung, H. (2011). A survey of combinatorial testing. *ACM Computing Surveys (CSUR)*, *43*(2), 1–29. https://doi.org/10.1145/1883612.1883618

Offutt, A. J., Lee, A., Rothermel, G., Untch, R. H., & Zapf, C. (1996). An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *5*(2), 99–118. https://doi.org/10.1145/227607.227610

Offutt, J. (2011). A mutation carol: Past, present and future. *Information and Software Technology*, *53*(10), 1098–1107. https://doi.org/10.1016/j.infsof.2011.03.007

Offutt, J. A., Pan, J., & Voas, J. M. (1995). Procedures for reducing the size of coverage-based test sets (pp. 111–123). Presented at the Proceedings of the Twelfth International Conference on Testing Computer Software.

© COPYRIGHT UPM

O'Keeffe, M., & Cinneide, M. O. (2006). Search-Based Software Maintenance (pp. 249–260). Presented at the Proceedings of the Conference on Software Maintenance and Reengineering (CSMR '06), Bari, Italy: IEEE Computer Society. https://doi.org/10.1109/CSMR.2006.49

Orlov, M., & Sipper, M. (2011). Flight of the FINCH Through the Java Wilderness. *IEEE Transactions on Evolutionary Computation*, *15*(2), 166–182. https://doi.org/10.1109/TEVC.2010.2052622

Orso, A., & Rothermel, G. (2014). Software Testing: A Research Travelogue (2000--2014) (pp. 117–132). Presented at the Proceedings of the on Future of Software Engineering (FOSE 2014), Hyderabad, India: ACM. https://doi.org/10.1145/2593882.2593885

Parnas, D. L. (1992). *Tabular Representation of Relations* (http://citeseer.ist.psu.edu/parnas92tabular.html). McMaster University: Communications Research Laboratory.

Parnas, D. L. (1997). Precise Description and Specification of Software (pp. 1–14). Presented at the Proceedings of the Second International Conference on Mathematics of Dependable Systems II (MDS '95), Univ. of York, England: Oxford University Press, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=294012.294013

Parnas, D. L., & Madey, J. (1995). Functional documents for computer systems. *Science of Computer Programming*, *25*(1), 41–61.

Parsa, S., & Khalilian, A. (2010). On the optimization approach towards test suite minimization. *International Journal of Software Engineering and Its Applications*, *4*(1), 15–28.

Peters, J. F. (1998). *Computational intelligence in software engineering*. River Edge, NJ, USA: World Scientific Publishing.

Poggenpohl, S., & Sato, K. (2003). Models of Dissertation Research in Design (pp. 125–132). Presented at the Proceedings of the 3rd Doctoral Education in Design Conference, Tsukuba, Japan.

Prema, P., Ramadoss, B., & Balasundaram, S. R. (2013). Identification and deletion of duplicate subtrees in classification tree for test case reduction. *International Journal of Information Systems and Change Management*, *6*(4), 374–388. https://doi.org/10.1504/ijiscm.2013.060986

Pressman, R., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

Ray, M., & Mohapatra, D. P. (2014). Multi-objective test prioritization via a genetic algorithm. *Innovations in Systems and Software Engineering*, *10*(4), 261–270. https://doi.org/10.1007/s11334-014-0234-2

Rothermel, G., Harrold, M. J., Ostrin, J., & Hong, C. (1998). An Empirical Study of the Effects of Minimization on the Fault Detection capabilities of Test Suites (pp. 34–43). Presented at the Proceedings of the International Conference on Software Maintenance (ICSM '98), Bethesda, MD, USA: IEEE Computer Society. https://doi.org/10.1109/ICSM.1998.738487

Rothermel, G., Harrold, M. J., Von Ronne, J., & Hong, C. (2002). Empirical studies of test-suite reduction. *Software Testing, Verification and Reliability*, *12*(4), 219–249.

Rothermel, G., Untch, R. H., Chengyun, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, *27*(10), 929–948. https://doi.org/10.1109/32.962562

© COPYRIGHT UPM

Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (1999). Test Case Prioritization: An Empirical Study (pp. 179–188). Presented at the Proceedings of International Conference on Software Maintenance (ICSM '99), IEEE Computer Society Press. https://doi.org/10.1109/ICSM.1999.792604

Saifan, A. A. (2016). Test Case Reduction Using Data Mining Classifier Techniques. *Journal of Software*, *11*(7), 656–663.

Salkind, N. J. (2010). *Encyclopedia of Research Design*. SAGE Publications. https://doi.org/10.4135/9781412961288.n222

Sampath, S., Mihaylov, V., Souter, A., & Pollock, L. (2004). A scalable approach to user-session based testing of web applications through concept analysis (pp. 132–141). Presented at the Proceedings of the 19th International Conference on Automated Software Engineering (ASE '04), Linz, Austria: IEEE Computer Society. https://doi.org/10.1109/ASE.2004.6

Sapaat, M. A., & Baharom, S. (2011). A preliminary investigation towards test suite optimization approach for enhanced State-Sensitivity Partitioning (pp. 40–45). Presented at the 2nd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME)., Bandung, Indonesia: IEEE. https://doi.org/10.1109/ICICI-BME.2011.6108592

Shahbazi, A., Tappenden, A. F., & Miller, J. (2013). Centroidal Voronoi Tessellations- A New Approach to Random Testing. *IEEE Transactions on Software Engineering*, *39*(2), 163–183. https://doi.org/10.1109/TSE.2012.18

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

Siami Namin, A., Andrews, J. H., & Murdoch, D. J. (2008). Sufficient Mutation Operators for Measuring Test Effectiveness (pp. 351–360). Presented at the Proceedings of the 30th International Conference on Software Engineering (ICSE '08), Leipzig, Germany: ACM. https://doi.org/10.1145/1368088.1368136

Sidi, F., Jabar, M. A., Selamat, M. H., Ghani, A. A. A., Sulaiman, M. N., & Baharom, S. (2011). Malay interrogative knowledge corpus. *American Journal of Economics and Business Administration*, *3*(1), 171–171.

Smith, A. M., Geiger, J., Kapfhammer, G. M., & Soffa, M. L. (2007). Test Suite Reduction and Prioritization with Call Trees (pp. 539–540). Presented at the Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering, ACM. https://doi.org/10.1145/1321631.1321733

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software Testing Foundations: A Study Guide for the Certified Tester Exam* (4th ed.). Rocky Nook.

Srikanth, H., Williams, L., & Osborne, J. (2005). System test case prioritization of new and regression test cases. Presented at the International Symposium on Empirical Software Engineering (ISESE '05), Noosa Heads, Qld., Australia: IEEE. https://doi.org/10.1109/ISESE.2005.1541815

© COPYRIGHT UPM

Srivastava, A., & Thiagarajan, J. (2002). Effectively prioritizing tests in development environment. *ACM SIGSOFT Software Engineering Notes*, *27*(4), 97–106. https://doi.org/10.1145/566171.566187

Tallam, S., & Gupta, N. (2006). A concept analysis inspired greedy algorithm for test suite minimization. *ACM SIGSOFT Software Engineering Notes*, *31*(1), 35–35. https://doi.org/10.1145/1108768.1108802

Tan, R. P., & Edwards, S. H. (2004). Experiences evaluating the effectiveness of JML-JUnit testing. *ACM SIGSOFT Software Engineering Notes*, *29*(5), 1–4. https://doi.org/10.1145/1022494.1022545

Tassey, G. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Diane Publishing Company.

Tonella, P., Avesani, P., & Susi, A. (2006). Using the Case-Based Ranking Methodology for Test Case Prioritization (pp. 123–133). Presented at the Proceedings of the 22Nd IEEE International Conference on Software Maintenance (ICSM '06), Philadelphia, PA, USA: IEEE Computer Society. https://doi.org/10.1109/ICSM.2006.74

Usaola, M. P., Mateo, P. R., & Lamancha, B. P. (2012). Reduction of Test Suites Using Mutation. In J. de Lara & A. Zisman (Eds.), *Fundamental Approaches to Software Engineering: 15th International Conference, FASE 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings* (pp. 425–438). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28872-2_29

Utting, M., Pretschner, A., & Legeard, B. (2012). A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability*, *22*(5), 297–312. https://doi.org/10.1002/stvr.456

Wagner, N. R. (2005). Queue in Java. Retrieved August 5, 2017, from http://www.cs.utsa.edu/~wagner/CS2213/queue/queue.html

Wang, R., Jiang, S., Chen, D., & Zhang, Y. (2016). Empirical Study of the Effects of Different Similarity Measures on Test Case Prioritization. *Mathematical Problems in Engineering*, *2016*, 19 pages. https://doi.org/10.1155/2016/8343910

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering* (1st ed.). Springer-Verlag Berlin Heidelberg.

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*(4), 347–369. https://doi.org/10.1002/(sici)1097-024x(19980410)28:4%3C347::aid-spe145%3E3.0.co;2-l

Wong, W. E., Horgan, J. R., Mathur, A. P., & Pasquini, A. (1999). Test set size minimization and fault detection effectiveness: A case study in a space application. *Journal of Systems and Software*, *48*(2), 79–89. https://doi.org/10.1016/s0164-1212(99)00048-5

Yang, X. S., & Koziel, S. (2011). *Computational Optimization and Applications in Engineering and Industry* (1st ed.). Springer-Verlag Berlin Heidelberg.

Yoo, S., & Harman, M. (2009). *TR-09-09: Regression Testing Minimisation, Selection and Prioritisation-A Survey*.

Yoo, S., & Harman, M. (2010). Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. *Journal of Systems and Software*, *83*(4), 689–701. https://doi.org/10.1016/j.jss.2009.11.706

Yoo, S., & Harman, M. (2012). Test data regeneration: generating new test data from existing test data. *Software Testing, Verification and Reliability*, *22*(3), 171–201. https://doi.org/10.1002/stvr.435

Yoo, S., Harman, M., Tonella, P., & Susi, A. (2009). Clustering Test Cases to Achieve Effective and Scalable Prioritisation Incorporating Expert Knowledge (pp. 201–212). Presented at the Proceedings of the Eighteenth International Symposium on Software Testing and Analysis (ISSTA '09), Chicago, IL, USA: ACM. https://doi.org/10.1145/1572272.1572296

Yuan, X., Cohen, M. B., & Memon, A. M. (2011). GUI Interaction Testing: Incorporating Event Context. *IEEE Transactions of Software Engineering*, *37*(4), 559–574. https://doi.org/10.1109/tse.2010.50

Zabinsky, Z. B. (2013). *Stochastic Adaptive Search for Global Optimization* (Vol. 72). Springer US.

Zhang, R., Jiang, J., Yin, J., Jin, A., Lou, J., & Wu, Y. (2008). A New Method for Test Suite Reduction (pp. 1211–1216). Presented at the Proceedings of the 9th International Conference for Young Computer Scientists (ICYCS '08), Hunan, China: IEEE. https://doi.org/10.1109/ICYCS.2008.501

Zhang, W., Wei, B., & Du, H. (2014). Test Case Prioritization Based on Genetic Algorithm and Test-Points Coverage. In X. Sun, W. Qu, I. Stojmenovic, W. Zhou, Z. Li, H. Guo, … L. Liu (Eds.), *Algorithms and Architectures for Parallel Processing: 14th International Conference, ICA3PP 2014, Dalian, China, August 24-27, 2014. Proceedings, Part I* (pp. 644–654). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-11197-1_50

Zheng, J. (2005). In Regression Testing Selection when Source Code is Not Available (pp. 752–755). Presented at the Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE '05), Long Beach, CA, USA: ACM. https://doi.org/10.1145/1101908.1101997

© COPYRIGHT UPM