REGRESSION TESTING FRAMEWORK FOR TEST CASES GENERATION AND PRIORITIZATION

EGLAL MOHAMMED KHALIFA OSMAN

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

JUNE 2022

DEDICATION

I dedicate this thesis to my beloved father, who taught me that the education is the weapon for progress and development. I also dedicate to my mother who taught me that even the largest task can be accomplished if it is done one step at a time. Finally, I dedicate to my sisters and brother for their support, encouragement, and endless affection.

ACKNOWLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, **Prof. Dayang Norhayati Abang Jawawi**, for encouragement, guidance, critics and friendship. I am also very thankful to my co-supervisor Prof Safa'ai bin Deris for their guidance, advices and motivation. Without their continued support and interest, this thesis would not have been the same as presented here. My fellow postgraduate student should also be recognised for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family member

ABSTRACT

A regression test is a significant part of software testing. It is used to find the maximum number of faults in software applications. Test Case Prioritization (TCP) is an approach to prioritize and schedule test cases. It is used to detect faults in the earlier stage of testing environment. Code coverage is one of the features of a Regression Test (RT) that detects more number of faults from a software application. However, code coverage and fault detection are reducing the performance of existing test case prioritization by consuming a lot of time for scanning an entire code. The process of generating test cases plays an important role in the prioritization of test cases. The existing automated generation and prioritization techniques produces insufficient test cases that cause less fault detection rate or consumes more computation time to detect more faults. Unified Modelling Language (UML) based test case generation techniques can extract test cases from UML diagrams by covering maximum part of a module of an application. Therefore, a UML based test case generation can support a test case prioritization technique to find a greater number of faults with shorter execution time. A multi-objective optimization technique able to handle multiple objectives that supports RT to generate more number of test cases as well as increase fault detection rate and produce a better result. The aim of this research is to develop a framework to detect maximum number of faults with less execution time for improving the RT. The performance of the RT can be improved by an efficient test case generation and prioritization method based on a multi-objective optimization technique by handling both test cases and rate of fault detection. This framework consists of two important models: Test Case Generation (TCG) and TCP. The TCG model requires an UML use case diagram to extract test cases. A meta heuristic approach is employed that uses tokens for generating test cases. And, TCP receives the extracted test cases with faults as input to produce the prioritized set of test cases. The proposed research has modified the existing Hill Climbing based TCP by altering its test case swapping feature and detect faults in a reasonable execution time. The proposed framework intends to improve the performance of regression testing by generating and prioritizing test cases in order to find a greater number of faults in an application. Two case studies are conducted in the research in order to gather Test Case (TC) and faults for multiple modules. The proposed framework yielded a 92.2% of Average Percentage Fault Detection with less amount of testing time comparing to the other artificial intelligence-based TCP. The findings were proved that the proposed framework produced a sufficient amount of TC and found the maximum number of faults in less amount of time.

ABSTRAK

Ujian regresi merupakan bahagian penting dalam ujian perisian. Ia digunapakai untuk mencari bilangan kesalahan maksimum dalam aplikasi perisian. Keutamaan Kes Ujian (TCP) adalah suatu pendekatan yang memberi keutamaan dan penjadualan dalam kes ujian. Ia digunakan untuk mengesan kesalahan pada tahap awal persekitaran pengujian. Liputan kod merupakan salah satu ciri bagi ujian regresi yang dapat mengesan lebih banyak kesalahan dari aplikasi perisian. Walau bagaimanapun, liputan kod dan pengesanan kesalahan mengurangkan prestasi sedia ada keutamaan kes ujian kerana ia mengambil banyak masa untuk mengimbas keseluruhan kod. Proses penjanaan kes ujian memainkan peranan penting dalam keutamaan kes ujian. Teknik penjanaan dan keutamaan automatik sedia ada menghasilkan kes ujian yang tidak mencukupi menyebabkan kadar pengesanan kesalahan lebih sedikit atau mengambil masa pengiraan lebih banyak untuk mengesan lebih banyak kesalahan. Bahasa Pemodelan Bersatu (UML) berasaskan teknik penjanaan kes ujian boleh mengekstrak kes ujian dari gambarajah UML yang merangkumi bahagian maksimum modul aplikasi. Oleh itu, UML berasaskan penjanaan kes ujian dapat menyokong teknik keutamaan kes ujian untuk mencari bilangan kesalahan yang lebih besar dengan masa pelaksanaan yang lebih pendek. Teknik pengoptimuman pelbagai objektif dapat mengendalikan pelbagai objektif yang menyokong ujian regresi untuk menghasilkan lebih banyak kes ujian serta peningkatan kadar pengesanan kesalahan dan menghasilkan keputusan yang lebih baik. Tujuan penyelidikan ini adalah untuk mengembangkan kerangka bagi mengesan bilangan kesalahan maksimum dengan masa pelaksanaan yang lebih sedikit bagi menambahbaik ujian regresi. Prestasi ujian regresi dapat dipertingkatkan dengan kaedah penjanaan kes ujian dan keutamaan yang efisien berdasarkan teknik pengoptimuman pelbagai objektif dengan pengendalian kedua-dua kes ujian dan kadar pengesanan kesalahan. Kerangka ini terdiri daripada dua model penting: Penjanaan Kes Ujian (TCG) dan Keutamaan TCP. Model TCG memerlukan gambarajah kes kegunaan UML untuk mengekstrak kes ujian. Pendekatan meta heuristik digunakan bagi menghasilkan token untuk penjanaan kes ujian. Dan, TCP menerima kes ujian yang telah diekstrak sebagai input untuk menghasilkan set pengutamaan kes ujian. Penyelidikan yang dicadangkan telah mengubah suai Hill Climbing sedia ada yang berdasarkan TCP dengan mengubah ciri pertukaran kes ujiannya dan mengesan kesalahan dalam masa pelaksanaan yang sewajarnya. Kaedah kerangka yang telah dicadangkan ini bertujuan untuk meningkatkan prestasi pengujian regresi dengan penjanaan dan keutamaan kepada kes ujian untuk mencari lebih banyak jumlah kesalahan dalam aplikasi. Dua kajian kes telah dijalankan dalam penyelidikan ini untuk mengumpulkan Kes Ujian (TC) dan kesalahan bagi pelbagai modul. Kerangka kerja yang telah dicadangkan menghasilkan Purata Peratusan Pengesanan Kesalahan sebanyak 92.2% dengan jumlah masa yang lebih sedikit berbanding dengan TCP berasaskan kecerdasan buatan yang lain. Hasil kajian membuktikan bahawa kerangka kerja yang telah dicadangkan dapat menghasilkan keputusan TC yang mencukupi dan menjumpai bilangan kesalahan maksimum dalam jangka masa yang lebih sedikit.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
ABC	-	Artificial Bee Colony
APFD	-	Average Percentage Faults Detected
AP	-	Affinity Propagation
BCO	-	Bee Colony Optimization
C1	-	Category 1
C2	-	Category 2
C3	-	Category 3
D1	-	Dataset 1
D2	-	Dataset 2
DSP	-	Dependency Structure Prioritization
DOM	-	Document Object Model
EHCTCP	-	Enhanced Hill Climbing Test Case Prioritization
FV	-	Fitness Value / Function Traverse Value
GA	-	Genetic Algorithm
GUI	-	Graphical User Interface
HC	-	Hill Climbing
HPSO	-	Hybrid Particle Swarm Optimization
HYRTS	-	Hybrid Regression Test Selection
HTML	-	Hyper Text Markup Language
ICD	-	Inter Case Dependency
MH	-	MetaHeuristics
ML	-	Machine Learning
MOM	-	Multi-objective Optimization Method
PMI	-	Proposed Method 1
PFWK	-	Proposed FrameWork
RSS	-	Requirement Severity Score
RT	-	Regression Test
RTS	-	Regression Test Selection
SDLC	-	Software Development Life Cycle

SVM	-	Support Vector Machine
TC	-	Test Case
TCP	-	Test Case Prioritization
TCG	-	Test Case Generation
TSP	-	Test Suite Minimization
UML	-	Unified Modelling Language
UCD	-	Use Case Diagram
XML	-	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Overview

Software testing (ST) is playing an important role in the software development industry. It is used to reduce the cost of maintenance and ensure the quality of a software application (Alkawaz & Silvarajoo, 2019; Biswas, Mall, Satpathy, & Sukumaran, 2009). Testing intends to find bugs and make sure that the software is free from errors. ST is an expensive task that requires 50% of software development resources (Biswas et al., 2009; Khanna, Chaudhary, Toofani, & Pawar, 2019). There are different types of testing techniques available in the field of software testing. Testing tools are introduced in Software Development Life Cycle (SDLC) to carry out the process of testing either manually or automatically. Testing is comprising of activities such as planning, preparation, and evaluation, to ensure the software application is built according to the user requirement (AdiSrikanth, Kulkarni, Naveen, Singh, & Srivastava, 2011; Ali et al., 2019). Automatic testing tools are used in the software industry to reduce the utilization of software resources.

Test Case (TC) is a group of data, expected results, and post conditions for the verification of specific requirements. Test case ID, Test scenario, and Test parameters are some of the typical TC parameters. TC can be generated either manually or automatically. Automatic generation of TC is becoming familiar in SDLC due its efficiency. TC can be extracted from Unified Modelling Language (UML) diagram (Sharma, Sabharwal, & Sibal, 2014). An automated generation of TCs from UML Use Case Diagram (UCD) can support development to make useful decisions and reduce the testing cost (Augusto Diniz Teixeira, Orientadora, & Braga Silva, 2017).

Regression test (RT) is used to verify the existing functionality of a software application. It will ensure that new changes in code should not have negative impact on functionalities of an application(Ismail, Ibrahim, & ibrahim, 2007). Any new change in an existing code may adversely affect the ability of the software application. RT is an inevitable test in order to make sure that system is working fine and did not affect from recent code modifications (Augusto Diniz Teixeira et al., 2017). It is an expensive maintenance test in continuous integration development of software (Singh & Sumit Sharma, 2015). The recent studies have proved that RT, which are based on information sources and Artificial Intelligence (AI) have more efficiency than traditional RT (Ghai & Kaur, 2016). The process of identification of change impact will support management to find necessary steps with respect to change cost and avoid unnecessary resources (Bajaj & Sangwan, 2019).

Test Case Prioritization (TCP) is a technique in RT to sort TC in an order to find maximum faults in a software application. It can cover maximum possible changes that are made in existing application. TCP are based on the information related to coverage of code elements (Li, Harman, & Hierons, 2007). It is a better option for a software testing environment. A successful TCP can avoid unnecessary TCs. Prioritizing a set of effective TCs can reduce the testing cost.

1.2 Research Background

Software testing is a process of comparing the actual outcome with the expected outcome. Testing of the software will be done in order to check the correct functionality of the system or the project (Sharma et al., 2014). The improper testing may lead to catastrophic or improper results in the field. It is better to check or test the system at the initial stage to improve its performance of software.

A TC should be more effective to find faults in a system. A test suite is a collection of TC, a larger test suite needs more execution time. Automated tools for the generation of TCs from UML Use Case Diagram (UCD) can support a testing team to take decision at earlier stage. A good TC can help to find a critical fault in software (Enoiu & Frasheri, 2019; Ismail et al., 2007; Kamath, 2018; Pang, Xue, & Namin, 2017). An AI based RT is required for better efficiency and reduction of testing cost.

Ideally, RT is a testing that refers to the section of the test cycle in which programs are tested to make sure that changes do not affect features of software. It is a process of verifying the customized software in the maintenance phase. Time and budget constraints are the major disadvantages due to its complex process. It will re – execute several subsets of test that were conducted in the previous phases. The purpose of the RT is to find accidental errors in the newly built software (Azizi & Do, 2018; Rothermel, Harrold, Ostrin, & Hong, 1998). The introduction of automation tools in the field of RT is partially reduced the stress levels of testing team. Research has shown that at least 50% of the total software cost is consumed for testing activities. Companies are often experiencing lack of time and resources, which limits their ability to effectively complete the testing process (R. R. Sahoo & Ray, 2018). TCP is used to prioritize TCs to cover maximum faults in limited amount of time.

RT is used to retest the component of a system and ensures that the modifications like patches and code enhancements does not affect the functionality of the software. Automated tools are required for these types of testing (Alkawaz & Silvarajoo, 2019). The process of verifying the modified software in the maintenance phase is known as RT. Time and utilization of larger number of computer resources are its major disadvantage due to the complex process nature of RT. RT will ensure that changes are made to software, such as adding new features or modifying existing features, which may not adversely affect the features of software. It is usually performed by running some, or all, of the TCs to test the modifications in previous versions of the software.

Many techniques have been reported on how to select regression tests so that the number of TCs does not grow too large as the software evolves. The hybrid technique will combine modification, minimization, and prioritization-based selection using a list of modified source code and the execution traces from TCs run on previous versions (Mahadik, Thakore, & Professor, 2016; Mittal & Sangwan, 2018).

The TCP is one of the techniques of RT in which, TCs are prioritized, in order to find maximum faults from the newly modified part of an application. The TC that has highest priority are executed first and so on. The priorities of the TCs are defined according to the changes made in the project. An AI based prioritization technique can detect maximum number of faults from an application, in which some changes are done, for the new version release (Li et al., 2007). Each TC has a functional importance or Function Traverse Value (FV) or Fitness Value (FV) (Kerani & Sharmila, 2018; Marchetto, Islam, Asghar, Susi, & Scanniello, 2016). The slicing is the process of extracting TC with relevant functional importance from a TC. In the existing technique, the slicing technique will be applied to detect the individual functionality of a TC. The TC that has maximum importance will be executed first, and so on. The single -objective prioritization method are focussing to increase the fault detection not on reducing TC and computation time. To increase the fault detection rate with minimum number of TC with limited computation time, an automated technique has to be applied by using multi-objective optimization method to prioritize TC according to the FV in an automated (Singh & Sumit Sharma, 2015). The multi-objective technique is based on the initial population value. The mutation value will be calculated from the best fitness value.

The research studies proposed by(Augusto Diniz Teixeira et al., 2017; Heumann, 2001; Ismail et al., 2007; Singh & Sumit Sharma, 2015; C. Wang, Pastore, Goknil, & Briand, 2020) are used to generate TC from UML diagram. These methods were extracted TCs from UML diagram. TCG is used to extract TC from UML diagram and reduce the time for the generation or selection of TCs. It will provide useful TCs for prioritization methods.

The purpose of using TC is to check successful and acceptable development of the product requirement (Heumann, 2001). The primary source for TCG is UML diagram. Generally, the system requirements are represented as a UML UCD. A scenario will be created for each UCD. The flow of events is the important part of

UCD. Basic and alternate are the two types of flow of events(Biswas, Mall, Satpathy, & Sukumaran, 2011; Hemmati, Arcuri, & Briand, 2010). TCs can be extracted from both flow of events. The UML UCD will be processed and converted into token relation to fetch TCs according to the relevant activities. A search algorithm can be used to derive TC from graphs (Augusto Diniz Teixeira et al., 2017). After parsing initial TCs from UCD, strings from a library can be used to search for specific cluster of TCs in the set of primary TCs. It will be helpful to form a TC to find a fault in a modified software(Ismail et al., 2007). The token relation and graph can be parsed by a traversing technique and generate functional TCs (Singh & Sumit Sharma, 2015). The automation of generation of TC is used to increase testing productivity and minimize labour hours. The meaningful representation of flows and branch conditions can be made from generated TCs. The automated tool can minimize the testing time with limited amount of data from UCD. A meta – heuristic technique can be used to reduce the computation time for extracting TC with relevant Value (FV) or Fitness Value FV from UCD. Therefore, the efficiency of TCG will be better than traditional methods(Augusto Diniz Teixeira et al., 2017).

A novel technique for the prioritization of TC from UML diagram was proposed by (Kerani & Sharmila, 2018). The criteria can cover whole software code in a minimal amount of time. Most of TCP techniques are based on the structural coverage and some prioritization techniques were presented with different criteria(Yan, Wu, Peng, & Nie, 2019; Shin Yoo, Harman, Tonella, & Susi, 2009). The existing TCP (Alkawaz & Silvarajoo, 2019; Azizi & Do, 2018; Butool, Nadeem, Sindhu, & Zaman, 2019; Gary & Jamie, 2010; Mahali & Acharya, 2013; S. Wang et al., 2014) were based on a machine learning (ML) technique for the prioritization of TCs. A weight is assigned for each TC and prioritized according to it. The execution of TC is based on the prioritization. The existing researches (Heumann, 2001; Li et al., 2007; Singh & Sumit Sharma, 2015) were found a TCG to generate TCs from UCD. The UCD are processed and necessary TC with relevant FV is extracted and prioritized for finding a critical fault from an application.

1.3 Research Problem

The TCP techniques are designed to execute RT effectively with less resources. The automated TCP greedily select a TC in an assumption that it finds a critical fault and covers a critical area of a modified part of an application. Some of the TCP has considered a TC based on the history of executing fault prone functions (Weixiang et al., 2019). The problem of this type of TC is the computation time. The investigation of history might take more time to find and execute a TC. The existing prioritization technique lacks in fault detection rate due to the inability of test cases. The process of generating test cases plays a vital role in finding critical faults from a software application. Many researches were developed to improve the process of prioritizing TCs for applications. However, there is a lack of TCP to attain better performance (Elbaum, Rothermel, & Penix, 2014; Miao, Qian, & Song, 2008).

RT is performed as a supportive testing task during the maintenance phase of a web-based application to ensure that the software evolution process, which is the primary characteristic of a web-based application, does not introduce new problems into the system(Nooraei Abadeh, 2021). While re-generation and re-execution of a new test suite may be unfeasible in terms of cost, time, and resource consumption, RT should ensure new test cases are progressively produced and finished the test suite to test system changes. In the web – based RT, making test cases and test data is a time-consuming and expensive process due to the nature of web application, which are constantly evolving(Zarrad, 2015). When a model contains a significant number of scalable sub-models, this method could result in a massive number of test cases, rendering these approaches ineffective. Existing technologies, such as random generators, can generate a large number of test data/test cases because of the random functions they use. The test suite may fail to identify test data to meet the requirement when using these ways because not enough information about the altered items in the test generating process(Mittal & Sangwan, 2015).

A web application's constant evolution makes it nearly hard to keep up with all of the changed paths and nodes. Uncovering hidden pathways in large, complicated Web applications takes time and effort. According to (Nooraei Abadeh, 2021), a challenge has been issued to find an unseen path based on session data. Automated RT techniques are beneficial and timeless, but any error in the development of test cases could result in the absence of some paths owing to dynamism, which would be problematic. In HTML DOM tree generation, additional paths are introduced when there are multiple pages in a Web application. In addition, fixing HTML errors might be a challenge. Another drawback is the inability to generate test cases. Many methods (Khanna, Chauhan, Sharma, & Toofani, 2017; Zarrad, 2015) necessitate the use of human intervention in the test set selection process. As a result, during the RT, not all problems could be identified.

The process of generating TCs from UML UCD will support the testing team to complete the test in small amount of time. An automated tool is required to generate TCs to improve the efficiency of TCs. The existing TCG are based on the coverage criterion and focussed to cover the complete code. They are not investigating the quality of TC that can cover maximum faults (Singh & Sumit Sharma, 2015). UML – UCD based test case generation is a model-based approach, which can cover maximum part of a newly modified software module. UCD have the capability to produce a better insight of a software application and produce more number of TC rather than other UML diagrams (Singh & Sumit Sharma, 2015), and (Augusto Diniz Teixeira et al., 2017). The existing generation technique extracts TC from UML diagrams without finding a feature of TC. The feature of TC can support TCP to understand its importance. A domain specific library can be used in the automated TCG to produce TC related to a specific problem in an application. UML based test case generation is familiar due to its ability to produce a greater number of test cases and cover maximum area of a module in large scale applications (Prasanna & Chandran, 2009).

A proper mechanism is required to arrange the generated TCs in an appropriate order, to increase the effectiveness of a system, to reach better performance and the rate of fault detection. TCP will execute the high prioritized TCs than the lower one to minimize time, cost, and effort. The performance of a testing system will be improved by this faster fault detection process. Therefore, efficiency of TCP will be increased by minimizing computation cost and time with small amount of information. The faster feedback will allow the software tester to correct the faults at the earliest time (Dalai, Abhinna, & Prasad, 2012; Gary & Jamie, 2010).

Prioritization of TC is one of the approaches to enhance the RT and retest the software after modification. RT is a process, of retesting the modified software and it ensures that there is no error in the previously tested source code due to the modifications. It is a very expensive testing process. In order to decrease the cost of RT, the software tester, may prioritize the test case, so that the test case which are more important, are run earlier during the RT process (Shin Yoo et al., 2009). In this context, prioritization techniques can take advantage of historical information of TC to achieve a superior results (AdiSrikanth et al., 2011; Kim & Porter, 2002; Padmnav, Pahwa, Singh, & Bansal, 2019; Panda, Acharya, Bhuyan, & Mohapatra, 2017; Spieker, Gotlieb, Marijan, & Mossige, 2017).

TCP is a method to prioritize and schedule TCs in appropriate order. The important test case may be prioritized and run earlier to decrease the cost of testing. RT will use clustering technique for TCP. The technique will cluster the TCs, having common properties and the similar fault detection ability will be grouped together as a single cluster. TCP is used to improve the cost effectiveness of RT. On the one hand, the existing TCP is focussing on high fault detection rate. On the other hand, consumes more time to achieve a better fault detection rate. Existing TCP is based on single – objective function, which focus on fault detection and consumes a greater processing time and vice versa.

The TCP technique is widely used to reduce the execution cost of the TCs for fault detection in software. In the previous research work, the TCP is done based on the functional importance. The integration of TCG and TCP improves the detection of more faults from the software in the least amount of time. If TCP could not choose a TC, then fault detection and code coverage is not possible. An ineffective TCP leads to failure in the detection of a critical fault in an application (Azizi & Do, 2018; Biswas, Mall, & Satpathy, 2013; Gupta, Sharma, & Pachariya, 2019).

The existing approaches in RT requires more time to generate and prioritize TC in order to find faults. RT is used to test the recently modified module of a software or web applications. Web applications are interconnected with multiple applications. A small bug can cause more damage to a web application. The performance of existing methods is not sufficient to save time and cover maximum amount of code. Model – based TCG produces more number of TCs rather than other TCG (Augusto Diniz Teixeira, Orientadora, & Braga Silva, 2017). The emergence of Artificial Intelligence (AI) leads to automate the process of generating and prioritizing TC with respect to find critical faults in a limited amount of time (Bajaj & Sangwan, 2019; Ashima, Shaheamlung, & Rote, 2020). AI based TCG and TCP consumes less testing time comparing to the traditional TCG and TCP. However, the capability of fault detection is not effective for a web-based application. On the other hand, RT demands model based TCG to generate a greater number of TCs. An automated TCG based on the model – based approach and TCP to find bugs from the software application. The automated generation of TC supports TCP to cover the maximum part of code in a newly modified application. AI based TCP can detect a greater number of faults rather than the traditional prioritization techniques. In addition, AI techniques are employed to overcome the issues such as more testing time and less fault detection.

1.4 Research Questions

The aim of this research is to solve the above discussed problems by using an AI based framework. The main research question is formulated as follows:

How to improve the regression testing by Artificial Intelligence based framework for generating and prioritizing test cases?

Research Question 1(RQ1): How to automate a process of generating TC from a UML use-case diagram?

Research Question 2(RQ2): How to find solutions for existing problems in existing TCG?

Research Question 3(RQ3): How generation technique supports prioritization process to find a fault in an application?

Research Question 4(RQ4): How can we improve regression test using an effective TCP technique?

Research Question 5(RQ5): How can we implement and evaluate the proposed technique to ensure its fault detection rate and relevant execution time?

1.5 Research Goal and Objectives

The available techniques for the prioritization of TCs are limited and not able to cover the complete code and failed to find maximum fault detection. The reason for the limitation is that TCPs are following a method to randomly order TCs in the test suite. The random ordering could not find fault severity in an application (Huang et al., 2019). The existing process of generation of TC requires the complete code for producing Tc. It consumes a lot of time and not able to produce effective TCs. This study proposes a framework to generate TCs from UML UCD and prioritize TCs in order to find critical faults from a newly changed module of an application. The framework is based on Meta Heuristics (MH) demands metadata to generate an optimal output by using less number of inputs. It is widely used in AI based approaches. The efficiency of a method is improved by reducing the computation time. The computation time can be reduced by providing correct information to a method for the production of better results. The overall aim of this research is to improve the performance of prioritization of TCs and increase the rate of fault detection.

To achieve the research goal, the following objectives are considered:

1) To design a method based on Artificial Intelligence technique for generating test cases from UML use case diagram using Meta Heuristic (MH) technique to cover maximum amount of code.

2) To prioritize the test cases using a Multi objective Optimization Method (MOM) in order to find the critical faults from a newly modified software application.

3) To evaluate and compare the performance of test case generation and prioritization techniques in terms of accuracy and average percentage of fault detection with respective computation time, respectively.

1.6 Research Justification

The performance of TCG is limited for dynamic web application. A dynamic web application is interconnected with multiple independent applications. The existing TCG lacks the ability for identifying and allot the critical demands of domain in the process of generating TC. It fails to develop a minimum group of TC with maximal ability to find faults. The introduction of model based approach solves the code coverage problem (Pinkal & Niggemann, 2017; Pretschner & Philipps, 2005). This approach has overcome previous methodologies in recent years. One concern remains, however, that modelling of large and complex systems will take a tremendous amount of time and effort. A modularization of the model is one alternative to solve this issue (Pinkal & Niggemann, 2017; Pretschner & Philipps, 2005). The reuse of standard automation system component models and the possibility of maintaining the libraries of these standard components is the significant advantage of this approach. Nevertheless, the processing of models of an application is difficult. TCP is introduced in order to detect failures in the primary stage of RT. However, it is not evident that a particular TC finds a critical fault from an application. The popularity of Machine Learning (ML) and Artificial Intelligence (AI) leads to AI based TCP (Alkawaz & Silvarajoo, 2019; Ashima, Shaheamlung, & Rote, 2020). In recent years, AI based TCP is becoming familiar in the field of software testing. The vital parameters of TCP are fault detection rate, code coverage, and time (Bajaj & Sangwan, 2019). The existing AI based TCP lacks either fault detection rate or code coverage.

Implementing an AI based RT will reduce the testing time and achieve maximum fault detection. TCP is a technique, which is used in RT to prioritize the TCs according to the changes made in the developed project. The research work is based on automated and manual TCP techniques. The AI based TCP will prioritize TCs according to the faults that are extracted from the library. A TCP can prioritize TCs to cover maximum part of a software module with a greater number of faults. To support TCP, an automated TC generation is proposed with an evidence to prove its efficiency. The generation of TCs from UML diagram will be combined with prioritization technique to detect a critical fault in the earlier stage of testing process. A TCG, which is based on a MH technique, require a metadata and fetches TC with relevant fault (Pinkal & Niggemann, 2017) It will add features of TC as meta data so that TCP can use those features for the prioritization process. It will reduce the time to explore the huge number of TCs.

A MH method is an optimization technique and used to automate software testing task. A reliable and efficient TC can be produced by this type of techniques with effort and time (Roongruangsuwan & Daengdej, 2010b, 2010a). The efficiency means that the ability of an application to produce better results within less amount of data and time (Sarma, Kundu, & Mall, 2007). The web application is also called a time critical application that is used to solve complex and vague problems (Allworth & Zobel, 1987). It must be responsive in its environment. It will be modified frequently according to the business requirement (Stankovic & Ramamritham, 1990). Some of the online applications are mobile applications, web - based applications, and trading websites. The UML UCD of case studies can be utilized for the purpose of extraction and prioritization of TCs. TCP based on AI technique must be trained to understand the environment of online applications to produce a reliable result.

In the existing technique, TCP considers number of times function encountered and number of functions associated with the function for the prioritization of TCs. The calculations of FV to prioritize each function are performed based on parameters of test suite (Miao et al., 2008). The process involved in the existing TCP is complex and consumes more time. An Automated TCP is being implemented in the research to increase the rate of fault detection with less amount of time. An automated TCG can be used to extract TC from UML UCD. A clustering technique will be applied to cluster TCs with relevant faults. The cluster or fault matrix will be processed by an automated TCP to find a fault in a module of an application. The prioritization technique takes test case and faults as input and prioritize these test cases based on the faults. The proposed research applies AI based technique to carry out the prioritizing process of TC.

In this research work, a novel method is proposed to generate TC with its functional importance from UML diagram and integrate with a multi – objective optimization method-based prioritization technique to find maximum number of faults in a limited computation time.

1.7 Scope

In this section, the scope of the proposed research will be discussed in detail. This research is comprising of two important models: The first model is used to generate TC and integrated with a second model, an AI based TCP to improve the efficiency of RT and increase the rate of fault detection.

I. Generation of TC from UCD

The research is using a generating technique to produce TCs from UML UCD. The criterion such as accuracy and execution time are considered for the evaluation of proposed methods. A MH is a familiar AI technique that requires fewer data to produce a valuable result. It can be used for generating TC from UCD.

II. RT for Case – study applications

The research is using different metrics to measure the performance of RT. The better RT can find maximum errors in a newly modified application. The familiar metric Average Percentage of Faults Detected (APFD) is used to inform testers about the capability of RT. The research is based on AI. Therefore, execution time is also an important criterion.

III. Test case Prioritization

The proposed TCP is used to prioritize TCs in an order according to the functional importance. The functional importance will be used as a key by the prioritization technique to prioritize TCs.

IV. Soft Computing Techniques

The soft computing techniques can be used to improve the efficiency of RT by increasing the rate of APFD. The approaches of soft computing such as clustering, and optimization are employed in this study.

1.8 Thesis Outline

The thesis is organized in seven chapters. The chapters are structured as follows:

Chapter 1 – *Introduction*: This chapter provides an overview of the research. The details of the proposed research are discussed and explained. The goal and objective of the research are also presented in detail.

Chapter 2- *Review of literature*: This chapter will provide information about test cases and prioritization. It will discuss the advantages and disadvantages of literatures related to test case extraction, prioritizing test cases, regression testing and clustering techniques.

Chapter 3- *Research methodology*: This chapter presents idea of the research to find the solution for research questions. It will provide information on datasets of UML diagrams. The framework of the research will be discussed in this chapter.

Chapter 4 - *An effective method to generate test cases from UML Diagram (UML) from UML Diagram (UML)*: The chapter will discuss the proposed method for the generation of test cases. It will provide solution for the issues related to the process of extraction of test cases.

Chapter 5 – *Test case prioritization using optimization method*: This chapter will discuss the proposed method for the prioritization of test cases. It will present graphical representations of proposed methods.

Chapter 6 – *Results and discussions*: This chapter discuss the performance of the proposed framework in detail. It also discusses the threats to the validity of the research.

Chapter 7 – *Conclusion and future work*: This chapter will provide summary of whole thesis. It will present the achievements of proposed methods. The future of the research and its direction will be discussed in this chapter.

REFERENCES

AdiSrikanth, Kulkarni, N. J., Naveen, K. V., Singh, P., & Srivastava, P. R. (2011). Test case optimization using artificial bee colony algorithm. *Communications in Computer and Information Science*, *192 CCIS*(PART 3), 570–579. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22720-2_60

Ali, N. bin, Engström, E., Taromirad, M., Mousavi, M. R., Minhas, N. M., Helgesson, D., … Varshosaz, M. (2019). On the search for industry-relevant regression testing research. *Empirical Software Engineering*, *24*(4), 2020–2055. https://doi.org/10.1007/s10664-018-9670-1

Alkawaz, M. H., & Silvarajoo, A. (2019). A survey on test case prioritization and optimization techniques in software regression testing. *Proceeding - 2019 IEEE 7th Conference on Systems, Process and Control, ICSPC 2019*, 59–64. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSPC47137.2019.9068003

Allworth, S. T., & Zobel, R. N. (1987). Introduction to Real-time Software Design. In *Introduction to Real-time Software Design*. Macmillan Education UK. https://doi.org/10.1007/978-1-349-18821-5

Alrawashed, T. A., Almomani, A., Althunibat, A., & Tamimi, A. (2019). An automated approach to generate test cases from use case description model. *CMES - Computer Modeling in Engineering and Sciences*, *119*(3), 409–425. https://doi.org/10.32604/cmes.2019.04681

Ashima, Shaheamlung, G., & Rote, K. (2020). A comprehensive review for test case prioritization in Software Engineering. *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, 331–336. IEEE. https://doi.org/10.1109/ICIEM48762.2020.9160217

Asthana, S., Tripathi, S., & Singh, S. K. (2010). A novel approach to generate test cases using class and sequence diagrams. *Communications in Computer and Information Science*, *95 CCIS*(PART 2), 155–167. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14825-5_14

Augusto Diniz Teixeira, Orientadora, F., & Braga Silva, G. (2017). *EasyTest: An* *approach for automatic test cases generation from UML Activity Diagrams*. 411–417. Information Technology - New Generations.

Azizi, M., & Do, H. (2018). Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization. *Proceedings - 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2018*, 245–251. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ISSREW.2018.00014

Bajaj, A., & Sangwan, O. P. (2019). A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms. *IEEE Access*, *7*, 126355–126375. https://doi.org/10.1109/ACCESS.2019.2938260

Batra, S., & Rishi, R. (2011). IMPROVING QUALITY USING TESTING STRATEGIES. *Journal of Global Research in Computer Science Journal of Global Research in Computer Science*, *2*(6), 113–117. Retrieved from www.jgrcs.info

Biswas, S., Mall, R., & Satpathy, M. (2013). A regression test selection technique for embedded software. *Transactions on Embedded Computing Systems*, *13*(3). https://doi.org/10.1145/2539036.2539043

Biswas, S., Mall, R., Satpathy, M., & Sukumaran, S. (2009). A model-based regression test selection approach for embedded applications. *ACM SIGSOFT Software Engineering Notes*, *34*(4), 1–9. https://doi.org/10.1145/1543405.1543413

Biswas, S., Mall, R., Satpathy, M., & Sukumaran, S. (2011). Regression Test Selection Techniques: A Survey. *Informatica*, *35*, 289–321.

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

Budha, G., Panda, N., & Acharya, A. A. (2011). Test case generation for use case dependency fault detection. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, *1*, 178–182. https://doi.org/10.1109/ICECTECH.2011.5941585

Butool, R., Nadeem, A., Sindhu, M., & Zaman, O. U. (2019). Improving requirements coverage in test case prioritization for regression testing.

Proceedings - 22nd International Multitopic Conference, INMIC 2019. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/INMIC48123.2019.9022761

Cai, G., Su, Q., & Hu, Z. (2021). Automated test case generation for path coverage by using grey prediction evolution algorithm with improved scatter search strategy. *Engineering Applications of Artificial Intelligence*, *106*, 104454. https://doi.org/10.1016/J.ENGAPPAI.2021.104454

Catal, C., & Diri, B. (2009, May). A systematic review of software fault prediction studies. *Expert Systems with Applications*, Vol. 36, pp. 7346–7354. https://doi.org/10.1016/j.eswa.2008.10.027

Chen, Y.-F., Chen, Y.-F., Rosenblum, D. S., & Vo, K.-P. (1997). TESTTUBE: a system for selective regression testing. *Proceedings of 16th International Conference on Software Engineering*, 211–220. Retrieved from https://www.academia.edu/10083578/TestTube_a_system_for_selective_regression_testing

Chouhan, C., Shrivastava, V., & S Sodhi, P. (2012). Test Case Generation based on Activity Diagram for Mobile Application. *International Journal of Computer Applications*, *57*(23), 4–9. https://doi.org/10.5120/9436-3563

Dalai, S., Abhinna, A., & Prasad, D. (2012). Test Case Generation For Concurrent Object-Oriented Systems Using Combinational Uml Models. *International Journal of Advanced Computer Science and Applications*, *3*(5), 97–102. https://doi.org/10.14569/ijacsa.2012.030515

De Nicola, G., di Tommaso, P., Rosaria, E., Francesco, F., Pietro, M., & Antonio, O. (2005). *A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems*. https://doi.org/10.1007/11408901_23

Devi Rajendran, V. K., Pradeepa Assistant Professor, R., & VimalaDevi, K. (2013). Effectiveness of Test Case Prioritization using APFD Metric: Survey Resource Provisioning in Cloud View project Wireless Cognitive network View project Effectiveness of Testcase Prioritization using APFD Metric: Survey. *International Conference on Research Trends in Computer Technologies*, 975–8887. International Journal of Computer Applications. Retrieved from https://www.researchgate.net/publication/268034515

Dhiman, R., & Chopra, V. (2019). Novel Approach for Test Case Prioritization Using ACO Algorithm. *2019 IEEE 2nd International Conference on Information and Computer Technologies, ICICT 2019*, 292–295. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/INFOCT.2019.8711039

Dirim, S., & Sozer, H. (2020). Prioritization of Test Cases with Varying Test Costs and Fault Severities for Certification Testing. *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 386–391. IEEE. https://doi.org/10.1109/ICSTW50294.2020.00069

Do, H., Mirarab, S., Tahvildari, L., & Rothermel, G. (2010). The effects of time constraints on test case prioritization: A series of controlled experiments. *IEEE Transactions on Software Engineering*, *36*(5), 593–617. https://doi.org/10.1109/TSE.2010.58

El Houda Dehimi, N., & Mokhati, F. (2019). A Novel Test Case Generation Approach based on AUML sequence diagram. *Proceedings - ICNAS 2019: 4th International Conference on Networking and Advanced Systems*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICNAS.2019.8807874

Elbaum, S., Rothermel, G., Kanduri, S., & Malishevsky, A. G. (2004). Selecting a cost-effective test case prioritization technique. *Software Quality Journal*, *12*(3), 185–210. https://doi.org/10.1023/B:SQJO.0000034708.84524.22

Elbaum, S., Rothermel, G., & Penix, J. (2014). Techniques for improving regression testing in continuous integration development environments. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *16-21-Nove*, 235–245. Association for Computing Machinery. https://doi.org/10.1145/2635868.2635910

Enoiu, E., & Frasheri, M. (2019). Test agents: The next generation of test cases. *Proceedings - 2019 IEEE 12th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2019*, 305–308. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSTW.2019.00070

Garg, D., Datta, A., & French, T. (2012). New test case prioritization strategies for regression testing of web applications. *International Journal of System Assurance Engineering and Management*, *3*(4), 300–309. https://doi.org/10.1007/s13198-012-0134-5

Gary, C., & Jamie, R. (2010). Arranging software test cases through an optimization method - IEEE Conference Publication. *PICMET 2010 TECHNOLOGY MANAGEMENT FOR GLOBAL ECONOMIC GROWTH*, 1–5. Retrieved from https://ieeexplore.ieee.org/abstract/document/5602131/authors#authors

Ghai, S., & Kaur, S. S. (2016). Functional Dependency based Test Case Prioritization for Regression Testing Using Hill-Climbing Approach. *International Journal of Innovative Research in Multidisciplinary Field*, *2*(9).

Grano, G., Laaber, C., Panichella, A., & Panichella, S. (2019). Testing with Fewer Resources: An Adaptive Approach to Performance-Aware Test Case Generation. *IEEE Transactions on Software Engineering*. https://doi.org/10.1109/TSE.2019.2946773

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. https://doi.org/10.1109/ACCESS.2019.2899471

Habtemariam, G. M., & Mohapatra, S. K. (2019). A Genetic Algorithm-Based Approach for Test Case Prioritization. *Communications in Computer and Information Science*, *1026*, 24–37. Springer Verlag. https://doi.org/10.1007/978-3-030-26630-1_3

Haidry, S. E. 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

Hemmati, H., Arcuri, A., & Briand, L. (2010). Reducing the cost of model-based testing through test case diversity. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6435 LNCS*, 63–78. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16573-3_6

Heumann, J. (2001). *Generating Test Cases From Use Cases*. Retrieved from http://www.therationaledge.com/content/jun_01/m_cases_jh.html

Huang, R., Zong, W., Chen, T. Y., Towey, D., Zhou, Y., & Chen, J. (2019).

Prioritising abstract test cases: An empirical study. *IET Software*, *13*(4), 313–326. https://doi.org/10.1049/iet-sen.2018.5199

Ismail, N., Ibrahim, R., & ibrahim, N. (2007). *Automatic Generation of test Cases from Use-Case Diagram*. 699–704.

Jain, P., & Soni, D. (2020). A Survey on Generation of Test Cases using UML Diagrams. *International Conference on Emerging Trends in Information Technology and Engineering, Ic-ETITE 2020*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ic-ETITE47903.2020.395

Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2017, January 17). *Software Testing Techniques: A Literature Review*. 177–182. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ict4m.2016.045

Jha, A. K., Kim, D. Y., & Lee, W. J. (2019). A framework for testing android apps by reusing test cases. *Proceedings - 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2019*, 20–24. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/MOBILESoft.2019.00012

Jung, B., & Kruse, P. M. (2020). Runtime Prioritization with the Classification Tree Method for Test Automation. *Bar2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 376–379. IEEE. https://doi.org/10.1109/ICSTW50294.2020.00067

Kamath, P. B. (2018). Generation of Test Cases from Behavior Model in UML. In *International Journal of Applied Engineering Research* (Vol. 13). Retrieved from http://www.ripublication.com

Karaboga, D., & Basturk, B. (2007). Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4529 LNAI*, 789–798. Springer Verlag. https://doi.org/10.1007/978-3-540-72950-1_77

Karasneh, B., & Chaudron, M. R. V. (2013a). Extracting UML models from images. *2013 5th International Conference on Computer Science and Information Technology, CSIT 2013 - Proceedings*, 169–178.

https://doi.org/10.1109/CSIT.2013.6588776

Karasneh, B., & Chaudron, M. R. V. (2013b). Img2UML: A system for extracting UML models from images. *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*, 134–137. https://doi.org/10.1109/SEAA.2013.45

Kaur, A., & Goyal, S. (2011). A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization. *International Journal of Advanced Science and Technology*, *29*, 17–29.

Kayes, M. I. (2011). Test case prioritization for regression testing based on fault dependency. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, *5*, 48–52. https://doi.org/10.1109/ICECTECH.2011.5941954

Kerani, M., & Sharmila. (2018). Novel technique for the test case prioritization in regression testing. *Communications in Computer and Information Science*, *905*, 362–371. Springer Verlag. https://doi.org/10.1007/978-981-13-1810-8_36

Khalifa, E. M., & Jamil, H. A. (2019). An Efficient Method to Generate Test Cases From UML-USE CASE DIAGRAM. *International Journal of Engineering Research and Technology*, *12*(7), 1138–1145. Retrieved from http://www.irphouse.com

Khanna, M., Chaudhary, A., Toofani, A., & Pawar, A. (2019). Performance Comparison of Multi-objective Algorithms for Test Case Prioritization During Web Application Testing. *Arabian Journal for Science and Engineering*, *44*(11), 9599–9625. https://doi.org/10.1007/s13369-019-03817-7

Khanna, M., Chauhan, N., Sharma, D. K., & Toofani, A. (2017). Test case prioritisation during web application testing. *International Journal of Computer Applications in Technology*, *56*(3), 230–243. https://doi.org/10.1504/IJCAT.2017.088200

Khatibsyarbini, M., Isa, M. A., Jawawi, D. N. A., Hamed, H. N. A., & Mohamed Suffian, M. D. (2019). Test Case Prioritization Using Firefly Algorithm for Software Testing. *IEEE Access*, *7*, 132360–132373. https://doi.org/10.1109/ACCESS.2019.2940620

Khurana, N., & Chillar, R. S. (2015). Test Case Generation and Optimization using

UML Models and Genetic Algorithm. *Procedia Computer Science*, *57*, 996–1004. Elsevier. https://doi.org/10.1016/j.procs.2015.07.502

Kim, J. M., & Porter, A. (2002). A history-based test prioritization technique for regression testing in resource constrained environments. *Proceedings - International Conference on Software Engineering*, 119–129. New York, New York, USA: IEEE Computer Society. https://doi.org/10.1145/581339.581357

Kumar Rapolu, R. (2018). *Selection of UML Models for Test Case Generation: A Discussion on Techniques to Generate Test Cases* (Vol. 26). Retrieved from https://repository.stcloudstate.edu/csit_etds/26

Kumar, S., Rajkumar, & Rani, M. (2019). Collaborative Filtering-based Test Case Prioritization and Reduction for Software Product-Line Testing. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, *2019-Octob*, 498–503. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/TENCON.2019.8929705

Kumar Swain, S., & Prasad Mohapatra, D. (2010). Test Case Generation from Behavioral UML Models. In *International Journal of Computer Applications* (Vol. 6).

Kumar, V., & Kumar, M. (2010). Test Case Prioritization Using Fault Severity. *International Journal of Computer Science and Technology*, *1*(1), 67–71.

Kundu, D., & Samanta, D. (2009). A novel approach to generate test cases from UML activity diagrams. *Journal of Object Technology*, *8*(3), 65–83. https://doi.org/10.5381/jot.2009.8.3.a1

Leung, H. K. N., & White, L. (1990). A study of integration testing and software regression at the integration level. *Conference on Software Maintenance*, 290–301. Publ by IEEE. https://doi.org/10.1109/icsm.1990.131377

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

Lu, C., Zhong, J., Xue, Y., Feng, L., & Zhang, J. (2019). Ant Colony System With Sorting-Based Local Search for Coverage-Based Test Case Prioritization. *IEEE Transactions on Reliability*. https://doi.org/10.1109/TR.2019.2930358

Mahadik, P. P., Thakore, D. M., & Professor, I. (2016). Survey on Automatic Test

Data Generation Tools and Techniques for Object Oriented Code. *International Journal of Innovative Research in Computer and Communication Engineering*, *4*(1). https://doi.org/10.15680/IJIRCCE.2016

Mahali, P., & Acharya, A. A. (2013). *MODEL BASED TEST CASE PRIORITIZATION USING UML ACTIVITY DIAGRAM AND EVOLUTIONARY ALGORITHM*.

Mahmood, K., Khan, T., Ahmad, S., Ashraf, E., Khan, T. A., & Ahmed, S. (2017). Value based PSO Test Case Prioritization Algorithm. *Article in International Journal of Advanced Computer Science and Applications*, *8*(1). https://doi.org/10.14569/IJACSA.2017.080149

Malhotra, R., Kaur, A., & Singh, Y. (2010). A Regression Test Selection and Prioritization Technique. *Journal of Information Processing Systems*, *6*(2), 235–252. https://doi.org/10.3745/jips.2010.6.2.235

Marchetto, A., Islam, M. M., Asghar, W., Susi, A., & Scanniello, G. (2016). A Multi-Objective Technique to Prioritize Test Cases. *IEEE Transactions on Software Engineering*, *42*(10), 918–940. https://doi.org/10.1109/TSE.2015.2510633

McLaughin, B., Police, G., & West, D. (2006). *Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to OOA&D - Brett McLaughlin, Gary Pollice, David West - Google Books* (First). O‟Reily Media Inc. Retrieved from https://books.google.com.sa/books?hl=en&lr=&id=-QpmamSKl_EC&oi=fnd&pg=PR9&dq=%5B72.%5D%09McLaughlin,+B.,+Pollice,+G.,+West,+D.+(2006).+Head+First+Object+Oriented+Analysis+and+Design,+O%27Reilly+Media,+Inc.&ots=nfaSZS_6nS&sig=PSboybGcB97a4i2sfxKl03Yg4vU&re

Meiliana, Septian, I., Alianto, R. S., Daniel, & Gaol, F. L. (2017). Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm. *Procedia Computer Science*, *116*, 629–637. Elsevier B.V. https://doi.org/10.1016/j.procs.2017.10.029

Miao, H., Qian, Z., & Song, B. (2008). Towards automatically generating test paths for Web application testing. *Proceedings - 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, TASE 2008*, 211–

218. https://doi.org/10.1109/TASE.2008.26

Mishra, D., Mishra, R., Acharya, A. A., & Das, D. P. (2019). Test case optimization and prioritization based on multi-objective genetic algorithm. *Advances in Intelligent Systems and Computing*, *741*, 371–381. Springer Verlag. https://doi.org/10.1007/978-981-13-0761-4_36

Mittal, S., & Sangwan, O. P. (2015). Metaheuristic Based Approach to Regression Testing. *International Journal of Computer Science and Information Technologies*, *6*(3). Retrieved from www.ijcsit.com

Mittal, S., & Sangwan, O. P. (2018). Prioritizing test cases for regression techniques using metaheuristic techniques. *Journal of Information and Optimization Sciences*, *39*(1), 39–51. https://doi.org/10.1080/02522667.2017.1372150

Mohapatra, S. K., Mishra, A. K., & Prasad, S. (2020). Intelligent Local Search for Test Case Minimization. *Journal of The Institution of Engineers (India): Series B*, *101*(5), 585–595. https://doi.org/10.1007/s40031-020-00480-7

Mor, A. (2014). Evaluate the Effectiveness of Test Suite Prioritization Techniques Using APFD Metric. *IOSR Journal of Computer Engineering*, *16*(4), 47–51. Retrieved from www.iosrjournals.orgwww.iosrjournals.org

Munialo, S. W., Muketha, G. M., & KOmieno, K. (2020). Automated Feature Extraction from UML Images to Measure SOA Size. *International Journal of Recent Technology and Engineering*, *9*(2), 1132–1137. https://doi.org/10.35940/ijrte.b4131.079220

Muthusamy, T., & K, S. (2014). Effectiveness of Test Case Prioritization Techniques Based on Regression Testing. *International Journal of Software Engineering & Applications*, *5*(6), 113–123. https://doi.org/10.5121/ijsea.2014.5608

Nejad, Mo. R. (2018). *Hybrid and dynamic criteria models for test case prioritization of web application regression testing*. University Technology Malaysia.

Nooraei Abadeh, M. (2021). Genetic-based web regression testing: an ontology-based multi-objective evolutionary framework to auto-regression testing of web applications. *Service Oriented Computing and Applications 2021 15:1*, *15*(1), 55–74. https://doi.org/10.1007/S11761-020-00312-Y

Nurmuradov, D., Bryce, R., Piparia, S., & Bryant, B. (2018). Clustering and

Combinatorial Methods for Test Suite Prioritization of GUI and Web Applications. *Advances in Intelligent Systems and Computing*, *738*, 459–466. Springer Verlag. https://doi.org/10.1007/978-3-319-77028-4_60

Padmnav, P., Pahwa, G., Singh, D., & Bansal, S. (2019). Test case prioritization based on historical failure patterns using ABC and GA. *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, 293–298. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CONFLUENCE.2019.8776936

Panda, N., Acharya, A. A., Bhuyan, P., & Mohapatra, D. P. (2017). Test case prioritization using UML state chart diagram and end-user priority. *Advances in Intelligent Systems and Computing*, *556*, 573–580. Springer Verlag. https://doi.org/10.1007/978-981-10-3874-7_54

Pang, Y., Xue, X., & Namin, A. S. (2017). A Clustering-Based Test Case Classification Technique for Enhancing Regression Testing. *Journal of Software*, *12*(3), 153–164. https://doi.org/10.17706/jsw.12.3.153-164

Parashar, P., Arvind, K., & Rajesh, B. (2012). How Time-Fault Ratio helps in Test Case Prioritization for Regression Testing. *International Journal of Software Engineering*, *5*(25–36).

Paterson, D., Campos, J., Abreu, R., Kapfhammer, G. M., Fraser, G., & McMinn, P. (2019). An empirical study on the use of defect prediction for test case prioritization. *Proceedings - 2019 IEEE 12th International Conference on Software Testing, Verification and Validation, ICST 2019*, 346–357. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICST.2019.00041

Pérez, C., & Marín, B. (2018). *Automatic Generation of Test Cases from UML Models* (Vol. 21).

Pinkal, K., & Niggemann, O. (2017). A new approach to model-based test case generation for industrial automation systems. *Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017*, 53–58. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/INDIN.2017.8104746

Pomeranz, I. (2018). Test compaction with dynamic updating of faults for coverage of undetected transition fault sites. *Proceedings of the Asian Test Symposium*, 30–35. IEEE Computer Society. https://doi.org/10.1109/ATS.2017.19

Pradhan, D., Wang, S., Ali, S., Yue, T., & Liaaen, M. (2019). Employing rule mining and multi-objective search for dynamic test case prioritization. *Journal of Systems and Software*, *153*, 86–104. https://doi.org/10.1016/j.jss.2019.03.064

Prasanna, M., & Chandran, K. R. (2009). Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm. In *Int. J. Advance. Soft Comput. Appl* (Vol. 1). Retrieved from www.i-csrs.org

Praveen Ranjan, S. (2008). *Test case prioritization*. *4*(3), 178–181. Retrieved from https://scholar.google.com/citations?user=TjVMlqYAAAAJ&hl=en#d=gs_md_cita-d&u=%2Fcitations%3Fview_op%3Dview_citation%26hl%3Den%26user%3DTjVMlqYAAAAJ%26citation_for_view%3DTjVMlqYAAAAJ%3Ad1gkVwhDpl0C%26tzom%3D-180

Pretschner, A., & Philipps, J. (2005). Methodological issues in model-based testing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3472 LNCS*, 281–291. Springer Verlag. https://doi.org/10.1007/11498490_13

Qu, X., Acharya, M., & Robinson, B. (2012). Configuration selection using code change impact analysis for regression testing. *IEEE International Conference on Software Maintenance, ICSM*, 129–138. https://doi.org/10.1109/ICSM.2012.6405263

Roongruangsuwan, S., & Daengdej, J. (2010a). A test case prioritization method with practical weight factors. *Journal of Software Engineering*, *4*(3), 193–214. https://doi.org/10.3923/jse.2010.193.214

Roongruangsuwan, S., & Daengdej, J. (2010b). TEST CASE PRIORITIZATION TECHNIQUES. *Journal of Theoretical and Applied Information Technology* , *18*(2), 45–60. Retrieved from www.jatit.org

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*.

S.GeethaDevasena, M., & L. Valarmathi, M. (2012). Meta Heuristic Search Technique for Dynamic Test Case Generation. *International Journal of*

Computer Applications, *39*(12), 1–5. https://doi.org/10.5120/4869-7294

Sahoo, R. K., Ojha, D., Mohapatra, D. P., & Patra, M. R. (2017). AUTOMATIC GENERATION AND OPTIMIZATION OF COURSE TIMETABLE USING A HYBRID APPRAOCH. *Journal of Theoretical and Applied Information Technology*, *15*(1), 68–77. Retrieved from www.jatit.org

Sahoo, R., Mohapatra, D. P., Kumar Sahoo, R., & Patra, M. R. (2016). A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases. *International Journal of Computer Sciences and Engineering International Journal of Computer Sciences and Engineering*. Retrieved from www.ijcseonline.org

Sahoo, R. R., & Ray, M. (2018). Metaheuristic Techniques for Test Case Generation: A ReviewMetaheuristic Techniques for Test Case Generation: A Review. *Journal of Information Technology Research*, *11*(1), 158–171.

Sampath, S., Bryce, R. C., Viswanath, G., Kandimalla, V., Günes, A., & Günes¸koru, G. (2011). Prioritizing User-session-based Test Cases for Web Applications Testing. *International Journal of System Assurance Engineering and Management*, *2*(2), 126–134.

Sarma, M., Kundu, D., & Mall, R. Automatic Test Case Generation from UML Sequence Diagram. , 15th International Conference on Advanced Computing and Communications (ADCOM 2007) § (2007). IEEE.

Shah, S. A. A., Bukhari, S. S. A., Humayun, M., Jhanjhi, N. Z., & Abbas, S. F. (2019). Test case generation using unified modeling language. *2019 International Conference on Computer and Information Sciences, ICCIS 2019*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCISci.2019.8716480

Shakya, S., & Smys, S. (2020). Reliable Automated Software Testing Through Hybrid Optimization Algorithm. *Journal of Ubiquitous Computing and Communication Technologies (UCCT*. https://doi.org/10.36548/jucct.2020.3.002

Sharma, C., Sabharwal, S., & Sibal, R. (2014). A Survey on Software Testing Techniques using Genetic Algorithm. *International Journal of Computer Science*, *10*(1), 381–393. Retrieved from http://arxiv.org/abs/1411.1154

Singh, A., & Sumit Sharma, E. (2015). Functional Test Cases Generation Based on Automated Generated Use Case Diagram. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, *8*, 2349–2163. Retrieved from www.ijirae.com

Sivaji, U., Shraban, A., Varalaxmi, V., Ashok, M., & Laxmi, L. (2019). Optimizing regression test suite reduction. *Advances in Intelligent Systems and Computing*, *815*, 187–192. Springer Verlag. https://doi.org/10.1007/978-981-13-1580-0_18

Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2017). Reinforcement learning for automatic test case prioritization and selection in continuous integration. *ISSTA 2017 - Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 12–22. Association for Computing Machinery, Inc. https://doi.org/10.1145/3092703.3092709

Stankovic, J. A., & Ramamritham, K. (1990). What is Predictability for Real-Time Systems?*. *Real - Time Systems*, *2*, 247–254.

Strandberg, P. E., Sundmark, D., Afzal, W., Ostrand, T. J., & Weyuker, E. J. (2016). Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 12–23. IEEE Computer Society. https://doi.org/10.1109/ISSRE.2016.23

Su, W., Li, Z., Wang, Z., & Yang, D. (2020). A Meta-heuristic Test Case Prioritization Method Based on Hybrid Model. *Proceedings - 2020 International Conference on Computer Engineering and Application, ICCEA 2020*, 430–435. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCEA50009.2020.00099

Suman, & Seema. (2012). A Genetic Algorithm for Regression Test Sequence ... - Ijarcce.com. *International Journal of Advanced Research in Computer and Communication Engineering*, *1*(7), 478–481. Retrieved from https://www.yumpu.com/en/document/read/32212562/a-genetic-algorithm-for-regression-test-sequence-ijarccecom

Suresh, Y., & Rath, S. K. (2014). Evolutionary algorithms for object-oriented test data generation. *ACM SIGSOFT Software Engineering Notes*, *39*(4), 1–6. https://doi.org/10.1145/2632434.2632446

Suri, B., Mangal, I., & Srivistava, V. (2011). Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm. *Special Issue of International Journal of Computer Science & Informatics* , *II*(1,2), 165–172. Retrieved from https://www.researchgate.net/publication/228460782_Regression_Test_Suite_Reduction_using_an_Hybrid_Technique_Based_on_BCO_And_Genetic_Algorithm/citation/download

Swain, R., Panthi, V., & Behera, P. (2013). GENERATION OF TEST CASES USING ACTIVITY DIAGRAM. *International Journal of Computer Science and Informatics*, *3*(2), 1–10. Retrieved from https://www.researchgate.net/publication/255964244_GENERATION_OF_TEST_CASES_USING_ACTIVITY_DIAGRAM/citation/download

Swain, S. K. (2010). *Test case generation and priortization of object oriented software using behavioral UML models*. KIIT University, India.

Tan, C., Behjati, R., & Arisholm, E. (2019). A model-based approach to generate dynamic synthetic test data: A conceptual model. *Proceedings - 2019 IEEE 12th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2019*, 11–14. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSTW.2019.00026

Thillaikarasi, M., & Seetharaman, K. (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*, *3*, 390–396.

Tonella, P., Avesani, P., & Susi, A. (2006). Using the case-based ranking methodology for test case prioritization. *IEEE International Conference on Software Maintenance, ICSM*, 123–132. https://doi.org/10.1109/ICSM.2006.74

Tripathy, A., & Mitra, A. (2013). Test case generation using activity diagram and sequence diagram. *Advances in Intelligent Systems and Computing*, *174 AISC*, 121–129. Springer Verlag. https://doi.org/10.1007/978-81-322-0740-5_16

Vanhecke, J., Devroey, X., & Perrouin, G. (2019). AbsCon: A test concretizer for model-based testing. *Proceedings - 2019 IEEE 12th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2019*, 15–

22. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICSTW.2019.00027

Wang, C., Pastore, F., Goknil, A., & Briand, L. (2020). Automatic Generation of Acceptance Test Cases from Use Case Specifications: an NLP-based Approach. *IEEE Transactions on Software Engineering*, 1–1. https://doi.org/10.1109/tse.2020.2998503

Wang, L., Yuan, J., Yu, X., Hu, J., Li, X., & Zheng, G. (2004). Generating test cases from UML activity diagram based on gray-box method. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 284–291. https://doi.org/10.1109/APSEC.2004.55

Wang, R., Sato, Y., & Liu, S. (2019). Specification-based Test Case Generation with Genetic Algorithm. *2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings*, 1382–1389. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CEC.2019.8790233

Wang, S., Ali, S., Yue, T., Bakkeli, O., & Liaaen, M. (2016). Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search. *Proceedings - International Conference on Software Engineering*, 182–191. IEEE Computer Society. https://doi.org/10.1145/2889160.2889240

Wang, S., Buchmann, D., Ali, S., Gotlieb, A., Pradhan, D., & Liaaen, M. (2014). Multi-objective test prioritization in software product line testing: An industrial case study. *ACM International Conference Proceeding Series*, *1*, 32–41. Association for Computing Machinery. https://doi.org/10.1145/2648511.2648515

Weixiang, Z., Yuhua, Q., Xuebo, Z., Bo, W., Min, Z., & Zhaohui, D. (2019). *On Test Case Prioritization Using Ant Colony Optimization Algorithm*. 2767–2773. Retrieved from https://ieeexplore.ieee.org/document/8855691/authors#authors

Wong, W. E., Wong, W. E., Horgan, J. R., London, S., & Agrawal, H. (1997). A Study of Effective Regression Testing in Practice. *IN PROCEEDINGS OF THE EIGHTH INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING*, 230--238. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.2665

Yadav, D. K., & Dutta, S. (2020). Regression test case selection and prioritization for object oriented software. *Microsystem Technologies*, *26*(5), 1463–1477. https://doi.org/10.1007/s00542-019-04679-7

Yan, Y., Wu, L., Peng, Y., & Nie, C. (2019). A Test Case Design Method Based on Path Depth Coverage. *Proceedings - Companion of the 19th IEEE International Conference on Software Quality, Reliability and Security, QRS-C 2019*, 89–96. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/QRS-C.2019.00030

Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: A survey. *Software Testing Verification and Reliability*, *22*(2), 67–120. https://doi.org/10.1002/stv.430

Yoo, Shin, Harman, M., Tonella, P., & Susi, A. (2009). Clustering test cases to achieve effective & scalable prioritisation incorporating expert knowledge. *Proceedings of the 18th International Symposium on Software Testing and Analysis, ISSTA 2009*, 201–211. New York, New York, USA: Association for Computing Machinery, Inc. https://doi.org/10.1145/1572272.1572296

Yu-Hsin Chen, G., & Wang, P.-Q. (2014). *Test Case Prioritization in a Specification-based Testing Environment*. *9*(8), 2056–2064. https://doi.org/10.4304/jsw.9.8.2056-2064

Zarrad, A. (2015). A Systematic Review on Regression Testing for Web-Based Applications. *Journal of Software*, *10*(8), 971–990. https://doi.org/10.17706/jsw.10.8.971-990

Zhang, L. (2018, May 27). *Hybrid regression test selection*. 199–209. Associationfor Computing Machinery (ACM). https://doi.org/10.1145/3180155.3180198

Zhang, T., Wang, X., Wei, D., & Fang, J. (2018). Test Case Prioritization Technique Based on Error Probability and Severity of UML Models. *International Journal of Software Engineering and Knowledge Engineering*, *28*(6), 831–844. https://doi.org/10.1142/S0218194018500249

Marinescu, P. D., & Candea, G. (2011). Efficient testing of recovery code using fault injection. ACM Transactions on Computer Systems (TOCS), 29(4), 1-38

.Banabic, R. (2015). Techniques for identifying elusive corner-case bugs in systems software (No. THESIS). EPFL.)

LIST OF PUBLICATIONS

Indexed Journal (SCOPUS)

1. **Eglal Mohamed Khalifa**, D. Jawawi, and H. A. Jamil, “An efficient method to generate test cases from UML-use case diagram,” International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 7 (2019), pp. 1138-1145.