APPLICATIONS OF VALUE-CENTRIC REGRESSION TESTING FOR SOFTWARE APPLICATIONS

FARRUKH SHAHZAD AHMED
01-281162-001

A thesis submitted in fulfillment of the
requirements for the award of the degree of
Doctor of Philosophy (Software Engineering)

Department of Software Engineering

BAHRIA UNIVERSITY ISLAMABAD

OCTOBER 2023

APPROVAL FOR EXAMINATION

Scholar's Name: Farrukh Shahzad Ahmed Registration No.: 01-281162-001

Program of Study: Ph.D. (Software Engineering)

Thesis Title: Applications of Value-Centric Regression Testing for Software Applications

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for evaluation. I have also conducted a plagiarism test of this thesis using HEC-prescribed software and found a similarity index of 16% which is within the permissible limit set by the HEC for the Ph.D. degree thesis. I have also found the thesis in a format recognized by the BU for the Ph.D. thesis.

Principal Supervisor's Signature: _____

Name: Dr. Tamim Ahmed Khan_____

Date: _____

AUTHOR'S DECLARATION

I, Farrukh Shahzad Ahmed hereby state that my Ph.D. thesis titled "Applications of Value-Centric Regression Testing for Software Applications" is my work and has not been submitted previously by me for taking any degree from this university Bahria University Islamabad, or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw/cancel my Ph.D. degree.

Name of the scholar: Farrukh Shahzad Ahmed

Date: _____

PLAGIARISM UNDERTAKING

I, solemnly declare that the research work presented in the thesis titled "Applications of Value-Centric Regression Testing for Software Applications" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero-tolerance policy of the HEC and Bahria University towards plagiarism. Therefore, I as an Author of the above-titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred to/cited.

I undertake that if I am found guilty of any formal plagiarism in the above-titled thesis even after the award of my Ph.D. degree, the university reserves the right to withdraw/revoke my Ph.D. degree and that HEC and the University have the right to publish my name on the HEC / University website on which names of scholars are placed who submitted plagiarized thesis.

Scholar / Author's Sign: _____

Name of the Scholar: Farrukh Shahzad Ahmed

DEDICATION

To my beloved mother (Late)

ACKNOWLEDGMENT

In the name of Allah, the compassionate, the most merciful and beneficent, who made human being and their minds in such a way that they can create unbelievable ideas and implement those ideas by the ability gifted by Him. I think without the help of the Almighty, I would never have been able to complete this task.

Before I get into the details, I would like to add a few heartfelt words for the people who were a part of this work in different ways. I would like to express my deep respect and gratitude to my supervisor Dr. Tamim Ahmed Khan and co-supervisor, Dr. Awais Majeed for all the support and encouragement they gave me during my research work, and for their faith in this idea and the guidance; they provided me throughout the completion of my Ph.D. research work. Without their constant feedback and guidance, this Ph.D. would not have been achievable.

Furthermore, a very special note of thanks goes to my beloved father and my mother (late), whose prayers, appreciation, motivation, encouragement, and support have always been there for me and a great source of inspiration and devotion for this work. Their prayers have always been with me throughout my life. I would also like to pay cordial gratitude to my teachers, Dr. Shahid Nazir Bhatti, Dr. Muhammad Muzammil, Dr. Arif Khan, Dr. Raja Muhammad Suleman, Dr. Amina Jamil, and Dr. Sumaira Kausar, who have helped me in every aspect, encouraged and kept my morale up at the early stage of my Ph.D. study.

Finally, a very special thanks to Dr. Zahoor Sarwar who helped me a lot during my research proposal phase. In the end, I would like to thank all those who helped me in my Ph.D. May ALLAH bless all of them. Ameen.

Farrukh Shahzad Ahmed

ABSTRACT

In the current era, businesses are Information Technology (IT) reliant, but most companies are deteriorating to maximize the value of their IT initiatives to their businesses. IT professionals do not know the value of distinct software features to the business. Likewise, they do not know the business value of diverse software quality attributes to the business. Therefore, they prioritize their project tasks based on their perceptions without considering formally measured business value. Ignoring value in software processes, practices, and artifacts is a value-neutral approach. In regression testing, software testers cannot re-execute all the test cases to find out the ripple effects of the changes due to time and budget constraints. No company can afford exhaustive regression testing in rapidly growing applications. Therefore, software testing professionals need a way through which they can prioritize their test cases for regression testing to uncover maximum bugs and side effects by utilizing minimum time and cost. Test Case Prioritization (TCP) is one of the processes to address this challenge. TCP is a smart way for regression testing to handle testing resource constraints. The main advantage of TCP is to save time through the prioritization of critical tests earlier. Current TCP techniques can be categorized as Value-Neutral (VN) and Value-Based (VB) approaches. In a VB approach, the cost of test cases and severity of faults are considered while, in a VN approach these are not considered. The VN approach is dominant over VB approach, and it assumes that all test cases have identical cost and that all software faults have same severity. But this notion seldom holds in practice. Therefore, VN TCP techniques are likely to deliver unreliable results. To fill this gap, focus should be shifted from VN to VB test prioritization. Presently, limited research work is done in a VB approach. To address this issue, a Systematic Literature Review (SLR) of VB TCP techniques is performed, and its results are presented in this thesis. Its purpose is to combine the overall knowledge related to VB TCP techniques and to highlight some open research issues in this domain. The literature review yields that value-orientation is vital in the TCP process to achieve its targeted goals and this is potential area for further research. Many TCP techniques are available, and their performance is usually measured through a metric Average Percentage of Fault Detection (APFD). This metric is value-neutral because it only works well when all test cases have the identical cost, and all faults

have the equal severity. Using APFD for performance evaluation of test case orders where test cases cost or faults severity varies is prone to produce false results. Therefore, using the right metric for performance evaluation of TCP techniques is very important to get reliable and correct results. To the best of the author's knowledge, there is no formal technique available to quantify business value based on which test cases can be prioritized. To overcome this problem, a business value quantification model has been proposed in this work to estimate faults severities and test cases cost. The proposed model supports the business value measurement of software requirements. We use the term software features as functional requirements and software quality attributes as non-functional requirements. The business value calculation of software features and quality attributes is based on three factors client priority, feature complexity, and feature usage. To compute the value of client priority, the proposed model utilizes five business success factors including profitability, productivity, operational efficiency, customer satisfaction, and time to market. Software fault severity and test case cost are estimated through the business value of requirements because different test cases and faults are directly associated with some requirements. Business value has been incorporated into the TCP process through the proposed model. The model is validated through two working examples. Based on the proposed model, two value-based TCP techniques have been introduced in this thesis using Genetic Algorithms (GA). These techniques are Value-Cognizant Fault Detection-Based TCP (VCFDB-TCP) and Value-Cognizant Requirements Coverage-Based TCP (VCRCB-TCP). Two novel value-based performance evaluation metrics are also introduced for value-based TCP including the $APFD_v$ and Average Percentage of Requirements Coverage per value ($APRC_v$). Two case studies are performed to validate proposed techniques and performance evaluation metrics quantitatively. A statistical analysis of the results is performed by a statistical test. The statistical results reveal that the proposed approaches provide significantly better results than traditional value-neutral TCP techniques.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS AND ACRONYMS

TCP:	Test Case Prioritization
APFD:	Average Percentage of Fault Detection
$APFD_c$:	Average Percentage of Fault Detection per cost
$APFD_v$:	Average Percentage of Fault Detection per value
IT:	Information Technology
SE:	Software Engineering
ROI:	Return on Investment
VBSE:	Value-based Software Engineering
APRC:	Average Percentage of Requirement Coverage
APFC:	Average Percentage of Function Coverage
APLC:	Average Percentage of Loop Coverage
APCC:	Average Percentage of Condition Coverage
APSC:	Average Percentage of Statement Coverage
ACO:	Accountable Care Organization
GA:	Genetic Algorithm
APBIE:	Average Percentage of Business Importance Earned
SLR:	Systematic Literature Review
VCFDB-TCP	Value Cognizant Fault Detection-Based Test Case Prioritization
VCRCB-TCP	Value Cognizant Requirement Coverage-Based Test Case Prioritization
$APRC_v$:	Average Percentage of Requirement Coverage per value
$APFC_v$:	Average Percentage of Function Coverage per value
$APLC_v$:	Average Percentage of Loop Coverage per value
$APCC_v$:	Average Percentage of Condition Coverage per value
$APSC_v$:	Average Percentage of Statement Coverage per value
NSGA-II	Non-dominated Sorting Genetic Algorithm II

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1. Background

Making great decisions in Software Engineering (SE) requires a thorough understanding of the business consequences of those decisions [1]. SE research is primarily based on VN settings in which all software artifacts have equal importance and there are many limitations of this fashion [2]. Value-based SE (VBSE) has catered to these limitations by considering value in software development principles and practices [2]. According to Barry Boehm, the definition of VBSE is "the explicit concern with value concerns in the application of science and mathematics by which the properties of computer software are made useful to people" [3]. The early research in the currently popular approach of agile software development was focused on extreme programming. Now, there is a transition in this trend and the focus is on the value of developed features and continuous value delivery [4]. Current trends are focused on value delivery and good coordination between business teams and technical teams [4]. According to [5] there is a value-based view of software product quality. Software customers usually take the value-based view, and they are concerned about the value added by software products to their organization. They perform a cost-benefit analysis. This requires a good definition of customer expectations regarding software quality in terms of some value. To meet customers' software quality expectations in terms of value, software testing becomes more critical. Software testing consumes 40 to 50% budget of any software project [6]. Software testing is critical due to its complexity [7]. Software testing research is also based on a VN approach like other software activities [8]. In VN software testing, resources are allocated to the activities that are inefficient in the context of Return on Investment (ROI). VN testing is not directly linked to the business

objectives of the product and is considered agnostic to value considerations [9]. To address these challenges, VB testing was introduced [8].

VB testing involves testing software systems that can better align testing resources to meet the value objectives of the project [8]. The major thing in value-based testing is to integrate internal testing objectives with the client's business objectives and expectations [8]. The focus is on value delivery instead of verifying code against a set of requirements. According to [2], the VN testing generated a higher ROI of 1.22 with 100% test execution, and the value-based testing produced a higher ROI of 1.74 with the execution of about 40% of the most important cases. This indicates that testing resources should be utilized in such a way that they can prevent losses due to non-functional software. The test cases should be aligned with the written requirements as well as with the client's expectations. Therefore, testing activities should adopt a business value-based perspective. The value-based verification and validation are also included in the agenda of VBSE [10].

According to Boehm software cost is estimated at $1 trillion per year and testing activities cost half of this total cost investment [3]. Regression testing consumes a large amount of time as well as effort and mostly accounts for around half of the software maintenance costs [11]. Regression testing is normally associated with re-testing of the system after any code change, it can be performed at the system level, integration level, or unit level [11]. Complete test coverage is normally not possible during regression testing due to limited time and resources.

Then the question arises: how much regression testing is enough, which is always a challenge for the testing teams? There are four approaches of regression testing for test case coverage including retest all, test case reduction, test case selection, and TCP [13]. Figure 1.1 illustrates the different types of regression testing approaches.

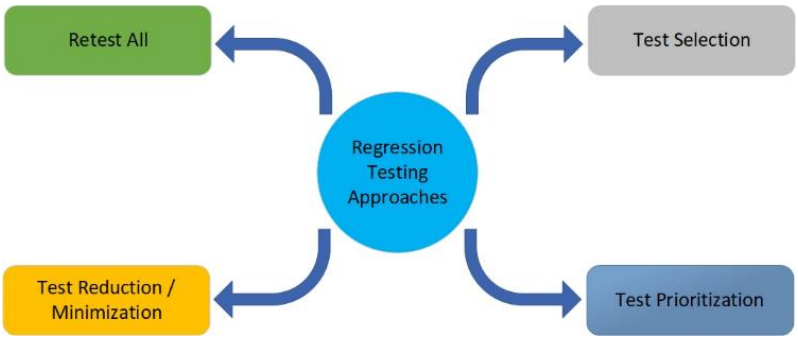


Figure 1.1: Regression testing approaches

The test selection approach is widely applied in the industry, but it is not risk-free because it is based on selection. Similarly, the test case reduction cannot guarantee that only unrelated test cases are eliminated from the test case pool. Whereas, TCP does not remove or reduce test cases from the test suite. That is why it is more secure, reliable, and popular in practice and a lot of research work is being done in this field.

TCP is an approach to optimize regression testing [14]. According to Rothermel *et al.* the TCP problem is defined as follows [15]. *Suppose T is a test suite, PT is a set of possible permutations of T, and f is a function from PT to real numbers, f: PT→R.*

Prioritization Goal: To search a $T^I \in PT$ *that maximizes the value of f.*

The TCP techniques consider number of defects detected, time, repetitiveness and size of test cases [16]. Various TCP techniques have been proposed and the primary purpose is to to increase the APFD rate, and save cost and time by prioritizing the test cases. There are two different fashions in which TCP techniques have been proposed. These fashions include value-based and value-neutral [17]. In a VB approach, test cases cost and bugs severity are considered while prioritizing test cases for regression testing [17]. Whereas the VN approach considers that all faults have same severity, and all test cases have identical cost. But in practice, this assumption seldom holds [18]. VN approach is dominant over VB approach.

The existing TCP techniques and their performance evaluation metrics are dominantly following a VN approach and are prone to produce unrealistic results [19]. Therefore, a paradigm shift is required from a VN to a VB approach. This thesis is focused on value-based cost cognizant TCP for regression testing.

1.2. Overview

In today's technological world, people are utilizing IT solutions in different domains like health care, education, sports, agriculture, and business [20]. Most businesses have been transformed into digitized forms. They are utilizing software applications to run their operations to deal with different products and services. Due to this tight reliance, the quality of systems has become more imperative. To attain the quality of software products, companies are spending a lot of money. Even then businesses go on stack due to software errors. There is a huge impact of software errors on the operations of any business. IT professionals seldom know what economic value their software application is going to add to the business. Now business owners and business units are focusing on the business value of the projects more than earlier [21]. The literature says that business value is taken from the client's perspective but in agile software development organizations, management defines it from the perspective of the portfolio of each project as a software supplier [22]. There would be a direct relationship between software project efforts and the associated business value delivered to the clients by that project [23]. Now in many organizations, the focus is being shifted from a cost-centric view to a value-centric view [22]. Predicting the business value of software is a challenging job [4]. There are two fashions of software engineering research one is value-neutral, and the other is value-based.

The objectives of coverage-based TCP are to increase requirements coverage, functions coverage, loop coverage, conditions coverage, or statements coverage. The traditional metrics Average APFD, APRC, Average Percentage of Function Coverage (APFC), Average Percentage of Loop Coverage (APLC), the APCC, or Average Percentage of Statement Coverage (APSC) are value-neutral metrics because they do not consider cost or execution time of test cases and value of items covered by the test cases.

The VB TCP techniques considers severity value and test case cost value. A cost-cognizant metric $APFD_c$ is available for fault-severity detection-based TCP [19]. However, there is no cost-cognizant metric available for coverage-based TCP techniques considering the value of requirements, functions, loops, conditions, statements, etc. The performance of a VB TCP technique depends on the cost of test cases as well as the value of covered items but there is no formal technique available to quantify the business value of requirements and to estimate fault severity and test case cost. TCP techniques are liable

to produce less satisfactory results due to these gaps. Another limitation of existing TCP techniques is that they only consider functional aspects of the application. Non-functional aspects of quality attributes are ignored in the prioritization process.

To fill the above-specified research gaps, four different dimensions pave the way to perform this research concurrently. The first goal of study is to analyze the research work done in value-based TCP. The second goal is to propose a mechanism for the estimation of fault severities and test case costs based on the business value of requirements for value-based TCP. The third goal is to propose VB TCP techniques for regression testing. The fourth goal is to propose novel performance evaluation metrics for value-based TCP techniques.

To achieve the first goal of the study, an SLR is performed on value-based TCP techniques to analyze the current state of work done in this area. To achieve the second goal, a business value quantification model is proposed in this thesis. The basic purpose of this model is to provide a way to determine the business value of software requirements and to provide a method to estimate faults severities and test case costs. to quantify the business value of software requirements. The purpose is to facilitate the value-based TCP process. The proposed model is applied in value-based TCP and empirically evaluated. The results demonstrate that value-based TCP produces more reliable and satisfactory results than value-neutral TCP techniques. Each item of software is associated with some requirement so its business value can be estimated through the business value of its respective requirement. There are two types of software requirements, one is functional, and the other is non-functional. We explicitly considered non-functional requirements as the quality attributes in the prioritization process because most of the time businesses suffer from non-functional aspects of the application like performance, security, privacy, etc. In the proposed model, the business value of software features and quality attributes is based on three factors client priority, complexity, and usage. The client priority is based on five business success factors including profitability, productivity, operational efficiency, client satisfaction, and time-to-market. The accumulative business value of a software feature is calculated through client priority, complexity, and usage of that feature. Similarly, the accumulative business value of a software quality attribute is estimated through the client priority of that quality attribute.

The severity of a bug is dependent on the business value of a feature with which it is associated. If a feature has a higher business value, all its associated bugs will have higher

severity accordingly. Similarly, test cases cost depends upon the business value of the requirements it covers in the case of coverage-based TCP. The cost of a test case depends upon the severity of faults it detects in fault-based TCP. The proposed model can help estimate the business value of functions, loops, conditions, statements, etc. from the business value of software requirements.

The core objectives of TCP include early fault detection, quick product maturity, efficient utilization of testing resources, and an increased APFD rate. There are several TCP techniques available to achieve these objectives. Some of the categories of TCP techniques include risk-based [24], history-based, coverage-based, fault-based, search-based, and requirements-based [25]. Most of the existing TCP techniques are designed in a value-neutral fashion. In VN TCP, all test cases have the same execution time, and all bugs have the same severity. The performance evaluation of these techniques is also done in a VN approach.

The existing TCP techniques are mostly coverage based. 100% coverage does not guarantee 100% bug detection [26]. All requirements and code statements are not of identical worth because they have their unique values. Therefore, the traditional coverage metrics are not the best adequacy criterion for TCP. Similarly, there is a notion that all faults are of equal severity [27]. The traditional APFD metric is completely based on this notion. The existing TCP approaches have been dominantly validated through APFD without considering the severity and criticality of the faults. The APFD rate is not an appropriate criterion for the performance evaluation of TCP techniques where test cases vary in terms of cost and faults vary in terms of severity. A cost-cognizant metric $APFD_c$ is available for fault-severity detection based on TCP [19]. But to the best of the author's knowledge, there is no value-cognizant metric available for coverage-based TCP techniques considering the value of requirements, functions, loops, conditions, or statements. Due to these limitations, TCP techniques and their performance evaluation metrics are likely to produce unsatisfactory and unreliable results. Therefore, a mechanism is required to overcome these limitations by taking value considerations into account.

To achieve the third goal, two value-cognizant TCP approaches are proposed in this thesis including Value-Cognizant Faults Detection-Based TCP (VCFDB-TCP) and Value-Cognizant Requirements Coverage-Based Test Case Prioritization (VCRCB-TCP). To achieve the fourth goal, two value-cognizant metrics are proposed in this thesis including

the $APFD_v$ and the $APRC_v$. The purpose of these metrics is to gage the performance of proposed value-cognizant TCP techniques.

The proposed techniques have been implemented using Genetic Algorithms (GA). GA is a search-based evolutionary algorithm. It is used to find approximate or true solutions for search-based problems. In this study, we consider the $APFD_v$ metric as a appropriate fitness function to guide VB TCP. We used GA for the proposed technique because it is widely used to solve optimization problems. Value-based regression test prioritization is also an optimization problem. GA works on a search space and begins with a random population of permutations. It is based on natural genetics and provides the best results. It is highly parallelizable as compared to other search-based algorithms [28]. The proposed techniques have been evaluated through two case studies on multiple versions of different health care applications developed using .Net technologies. These are developed by a US-based software company to support care management of Accountable Care Organization (ACO)-based population. The proposed techniques are evaluated by using the proposed metrics $APFD_v$ and $APRC_v$. The results are compared with the state-of-the-art TCP techniques. The proposed techniques produced better results than other techniques.

The main contribution of this research includes (i) An SLR on value-based TCP ii) A business value quantification model iii) Two value-based TCP techniques proposed. iv) Two novel value-cognizant performance evaluation metrics for value-based TCP are proposed.

1.3 Research Motivation

In the current era, software quality is a major concern for clients. Poor-quality software applications are costing US organizations over two trillion dollars annually [29]. Client businesses are highly dependent on software applications. A single error or a glitch in the software can stop the business operations of the client. Software errors damage the business in many ways like data loss, downtime, and loss of transactions. There can be many other virtual effects like losing credibility, damage to reputation, losing market, loss of customers' trust, agreement termination, or even closure of the whole business. The list of the few biggest software disasters reported in [30] is given below to realize the business value of software errors.

I. The Mariner 1 Spacecraft, 1962: It was a space mission, and failed due to the omission of a hyphen in the code statement. This error caused an incorrect signal sent to the spacecraft. The overall cost of this error was more than $18 million.

II. Bitcoin Hack, Mt. Gox, 2011: There was a software glitch in the exchange creating transactions that were not fully redeemed. It was costing up to $ 1.5 million in lost bitcoins.

III. Heathrow Terminal 5 Opening, 2008: Thousands of items of passengers were not received at the destination due to the malfunctioning baggage handling system. Over 10 days, 42000 bags were lost, and more than 500 flights were canceled costing over £16 million.

To address the business sensitivity of software applications, different testing techniques come into play. It is very interesting to know that software testing is taking more than 40% of the overall software project cost. According to the National Institute of Standards and Technology (NIST), software errors cost around $59.5 billion to the US economy and this figure can be reduced by $22.2 billion with improvements in software testing [8]. Similarly, one minor privacy breach issue can lead to business closure. A severe performance issue can decrease the productivity of the resources and result in increased costs.

Due to software bugs, business operations are halted, and businesses largely suffer from financial losses. Businesses go on stack due to software errors. There is a huge impact of software errors on the return on investment (ROI) of any business. The continuous changes in software applications, make it riskier for the business as every new change in the system requires additional testing to check its ripple effects. However, due to limited time and budget for regression testing, this is not always possible to test every functionality of the system whenever a new change is incorporated into it. The existing techniques didn't address the business success factors. There is no TCP technique available to support critical business workflows. If a problem occurs in the system, the client's business operations stop. Consequently, they bear the financial loss in terms of cost, time, and reduced productivity.

This is the motivation behind this work how we can protect clients' businesses from losses through better software testing techniques. The intention here is to incorporate business value in the test case prioritization process. We want to propose a business value based

TCP technique for software regression testing for functional/non-functional aspects of the application to protect the core business functionalities of the applications.

1.4 Research Gaps

VN TCP techniques are likely to provide less satisfactory results [31]. To address this research issue, a paradigm shift is required from VN to VB TCP techniques. Currently, limited research work is available in VB approach, and no comprehensive review of VB cost-cognizant TCP techniques is available in the existing literature.

In the current era, businesses are IT-reliant, but most companies are deteriorating to maximize the value of their IT initiatives to their businesses. IT vendors do not know the value of distinct software features to the business. Likewise, they do not know the business value of diverse software quality attributes to the business. Therefore, they prioritize their project tasks based on their perceptions without considering formally measured business value. Ignoring value in software processes, practices, and artifacts is a value-neutral approach. The software Test Case Prioritization (TCP) process has the same problem. Most of the existing TCP techniques are developed in a value-neutral fashion and assume that all faults have the same severity and that all test cases have the same cost. But this assumption rarely holds. Therefore, existing techniques are prone to produce unreliable results. Value orientation in the TCP process is missing.

The focus is on the number of faults detected instead of their impact on the client's business. Ignoring value in the TCP process is prone to produce unsatisfactory results. Another problem with this fashion is that the use of value-neutral metrics for the performance evaluation of TCP techniques will produce unreliable results. To address these problems, value-based cost-cognizant TCP techniques have been introduced.

1.5 Problem Statement

Existing TCP techniques are dominantly based on the frequency of elements detected or covered by test cases assuming that all elements are equally important, and all test cases have equal costs. The value-neutral approaches are prone to produce less reliable results

because of the base assumptions paving the way for the value-oriented TCP process, and it is important to evaluate its viability, impact, and usefulness.

1.6 Research Questions

This research is initiated to solve the problem of value-based cost-cognizant test case prioritization. The primary research question is articulated as follows.

"*How the value orientation can be introduced in TCP techniques and their evaluation metrics?*"

This primary research question is split into four secondary research questions. The secondary questions are listed below.

RQ1: What are the existing VB-TCP techniques, their algorithms, validation methods, performance evaluation metrics, and open research problems?

- ✓ What are existing TCP techniques and their performance metrics?
- ✓ What are the limitations and open research problems related to TCP techniques and performance metrics?

RQ2: How severity of faults and cost of test cases can be estimated through the business value of software requirements for VB-TCP?

- ✓ How the business value of requirements can be measured?
- ✓ How severity of faults and cost of test cases can be estimated through the business value of software requirements for VB-TCP?

RQ3: How business value can be incorporated in VB cost-cognizant TCP?

- ✓ How test cases can be prioritized by incorporating VB fault severity and test case cost using a Genetic Algorithm (GA)?
- ✓ How varying business value of requirements can be incorporated in value-cognizant requirements coverage-based TCP?

RQ4: How performance of VB-TCP techniques can be measured using novel VB performance evaluation metrics?

1.7 Research Objectives

This research is focused on value-based TCP for regression testing. The value orientation is added to fill the gaps in a value-neutral fashion. The objectives of this research are as follows:

1. To explore the current state of the research in VB-TCP for regression testing and to highlight open research issues in this domain.
2. To propose a business-oriented quantification model to incorporate business value in fault severities and test cases' cost for VB- TCP.
3. To propose VB-TCP Techniques
   - To propose a value-cognizant fault detection-based TCP technique for VB regression testing.
   - To propose a value cognizant requirement coverage-based TCP for VB regression testing.
4. To propose a novel VB cost-cognizant metric for performance evaluation of proposed VB-TCP techniques.

1.8 Contribution of the Study

This section describes the significant contribution of this research, and it includes the following items:

i) An SLR on value-based TCP techniques

ii) Value orientation in the TCP Process through business value quantification model

iii) Value-based TCP techniques using a genetic algorithm

iv) Novel value-cognizant performance evaluation metrics for value-based TCP

A literature review of VB TCP techniques is given in this thesis. The objective is to gether the knowledge related to VB TCP techniques and to highlight open research issues in this domain. This literature review was well needed because there is limited work done on value-based TCP. Most of the existing work is dominantly following a value-neutral fashion. This literature review yields that value-orientation is important in the TCP

process to achieve its goals and this potential area for further research.

There was no significant work on value orientation in the TCP process. To overcome this problem, a business value quantification model has been proposed in this research to estimate faults severity and the test cases cost. It provides support to measure the business value of software requirements. To incorporate a value-oriented TCP process, fault severity and test cases cost are estimated based on the business value of software requirements. The software requirements includes both functional and non-functional. The existing TCP work is dominantly focused on functional aspects of the application and ignores non-functions aspects. In this research, both functional requirements (as features) and non-functional requirements (as quality attributes) are considered. The proposed model is applied in business value-based TCP and is empirically evaluated. The results indicate that using the proposed model provides better TCP results than traditional VN TCP techniques. It also supports the better alignment of IT units with business units to maximize the value of software initiatives to the business. The value orientation in the TCP process provides support to bridge gaps between IT units and business units.

Two value-based cost-cognizant TCP techniques have been proposed using a genetic algorithm. One is Value-Value Cognizant Fault Detection Based TCP (VCFDB-TCP) and the other is Value Cognizant Requirements Coverage Based TCP(VCRCB-TCP). Two novel value-based performance evaluation metrics have been proposed. One is the $APFD_{v,}$ and it is used for the performance evaluation of VCFDB-TCP. The other is the $APRC_{v,}$ and it is used for the performance evaluation of VCRCB-TCP. Two case studies are performed to evaluate the performance of proposed value-based TCP techniques using the proposed performance evaluation metrics. The results demonstrated that the proposed techniques are outperforming the existing state-of-the-art techniques.

1.9 Thesis Organization

The rest of the thesis is comprised of four chapters. The description of the chapters is given below.

Chapter 2: This presents a literature review of the domain under study. It includes a chapter introduction, VN vs VB software engineering, VN vs VB testing, regression testing, and test case prioritization techniques. The focus of the study is on VB TCP techniques, their research trends, and their performance evaluation metrics. This chapter includes a description of algorithms and validation methods used in the TCP process. Some common steps of the TCP process, common objectives, and taxonomy of TCP techniques have been presented. Some open research problems and recommendations to fill the research gaps have also been given in this chapter. The core objective of performing a literature review is to gether knowledge related to VB TCP techniques and their performance evaluation metrics. This chapter also presents software quality attributes, business success factors from a client's perspective, and the value of software quality to the business. Existing techniques for software business value measurement and value-based TCP have also been discussed. At the end of the chapter, gap analysis is done with some open research questions and future research directions. Hence the chapter is concluded.

Chapter 3: Research methodology describes all the details of the research methodology adopted to solve the research problem. It includes the research process, and research framework to carry out the research. All the steps followed to perform the research have been included in this chapter. It includes an enhanced taxonomy of TCP techniques and the proposed business value estimation model for estimating faults severities and test cases cost. It describes VB TCP presenting how test cases can be prioritized based on business value-based fault severity and test case cost.

A value-cognizant fault detection-based TCP technique and a novel metric $APFD_v$ for its performance evaluation are presented. This chapter also presents the proposed value-cognizant requirements coverage-based TCP and novel metric $APRC_v$ for its performance evaluation. A description of a genetic algorithm to implement the proposed techniques is given too.

Chapter 4: This chapter presents data analysis, results, and findings. Two case studies have been presented. It includes the context of the study, testing criteria, evaluation algorithm, evaluation metrics, and results and discussion of the study.

Chapter 5: This is the last chapter of the thesis that concludes the whole work done in this study. It includes a summary of the work done, the implication of the study, research contribution, threats to validity, and future research directions.

CHAPTER 2

LITERATURE REVIEW / THEORETICAL FRAMEWORK

2.1. Introduction

This chapter presents the literature review of the domain under study. It describes software testing, regression testing, and its three types including test case selection, test case reduction, and TCP. The focus of this work is on test case prioritization techniques. A comprehensive literature review of existing TCP techniques for regression testing is given in this chapter. It includes the difference between VN and VB TCP techniques. An analysis of the literature on VB TCP is done. The algorithms and validation methods used in VB TCP are described. VB cost-cognizant TCP process and its objectives are described. The performance evaluation metrics used in cost-cognizant TCP are also described. The open research problems and recommendations to fill the research gaps. The chapter conclusion is given at the end.

2.2. Software Testing

Software testing is the process of evaluating a software system to ensure its quality, functionality, and performance [32]. It involves checking the software to find bugs, errors, or deviations from expected behavior. Two common approaches to software testing are black-box and white-box. Both black-box and white-box testing are important for ensuring the overall quality and reliability of a software system. The choice of the testing approach depends on the specific requirements and objectives of the testing effort.

Black-Box Testing:

Black-box testing is a testing approach where the tester does not know the implementation details of the system being tested[32]. Testers view the system as a "black box" and focus on the inputs and outputs without considering the internal code or structure. The goal is to evaluate the system's functionality, usability, and adherence to

specifications, without any knowledge of how the system achieves its results. Black-box testing is typically performed from a user's perspective, simulating real-world usage scenarios. Testers design test cases based on functional requirements, specifications, or user stories. They verify if the system behaves as expected by providing various inputs and observing the outputs or responses. The focus is on the external behavior and functionality of the software without knowledge of the internal workings.

White-Box Testing:

White-box testing, also known as clear-box or structural testing, involves testing the internal structure, code, and implementation details of the system being tested [32]. Testers have full knowledge and access to the system's internal workings, including the source code, algorithms, and design. The tests are designed based on this knowledge to verify the correctness of the internal logic, code coverage, and performance of the system. White box testing techniques include unit testing, code review, code coverage analysis, and debugging. Testers can create test cases that target specific code paths, branches, or conditions. They evaluate the internal behavior of the software, validate the internal data structures, and check if the code adheres to coding standards and best practices.

The choice between black-box testing and white-box testing depends on various factors, such as the testing objectives, available resources, and the level of access to the system's internals. Both approaches have their strengths and weaknesses, and often a combination of both techniques is employed to achieve comprehensive test coverage. Black-box testing is valuable for validating functional requirements, ensuring usability, and testing the system as a whole. It can be conducted by individuals without programming knowledge and provides an objective assessment of the system's behavior. White-box testing, on the other hand, is effective in assessing the internal correctness of the software, checking code quality, and ensuring adequate code coverage.

2.3. Software Regression Testing

Software regression testing is a type of testing performed to ensure that changes to a software product have not caused existing functionality to regress. Regression testing is crucial to maintain the stability and reliability of the system as it evolves. Regression testing is among the most expensive testing activities and is a big challenge in rapidly growing and changing systems [11]. There are limited time and costs available for

regression testing. There is a chance to stop, or halt testing earlier due to these resource constraints and leave it incomplete. Incomplete regression testing is always a threat to the application, and it can harm business operations. The intelligent utilization of testing resources and smart execution of test cases are key to testing success.

On the other hand, TCP is a process through which test cases are prioritized for the early detection of bugs. The test cases detecting a higher number of bugs are prioritized first for execution.

2.4. Test Case Prioritization

2.4.1. Value-Neutral TCP

The value-neutral TCP techniques consider that all faults have identical severity, and all test cases have the same cost [31]. Similarly, the metrics used for performance evaluation of TCP techniques like APSC, APFD, the Total Percentage of Fault Detection (TPFS), the Average Percentage of Branch Coverage (APBC), the APFC, the Average Percentage of Condition Coverage (APCC), and the Average Percentage of X elements Coverage (APXC) are also proposed in a value-neutral fashion. All these metrics assume that all faults have the same severity, all requirements have the same worth, and all code statements have the same value but practically this is very rare. Different bugs have different severity, and different requirements have different values. Similarly, the value of different functions, statements, conditions, branches, and methods may differ from other functions, statements, conditions, branches, and methods, respectively. Most of the existing TCP techniques are coverage-based and are effective at unit-level testing, but are time-consuming and consider that all bugs are equally severe and all test cases have equal cost [5]. This assumption is not possible in practice [33].

In this section, the existing TCP techniques have been summarized. An algorithm is proposed for the automatic prioritization of test cases based on output diversity as a representation of fault revealing probability and test coverage information [34]. In this technique, a dynamic prioritization based on the greedy algorithm is taken because test coverage information was already there and the size of the test suite to be prioritized was small containing around 500 test cases. A Firefly Algorithm based TCP technique is

proposed using a similarity distance model as fitness function [35]. This performance of technique was better than Genetic Algorithm (GA), Local Beam Search (LBS), Particle Swarm Optimization (PSO) and Greedy algorithm. A tool sOrTES is introduced to evaluate the independence and ranking of test cases based requirements coverage and execution time [36]. In [37], a GA based approach is proposed for ordering of JUnit test cases using optimization heuristics, Other algorithms are also applied including ACO, Multi-Objective Genetic Algorithm (MOGA) and Simulated Annealing (SA). ACO-based techniques are used to solve coverage-based TCP problems and are better than Genetic Algorithm based techniques [38].

A dissimilarity clustering-based TCP technique has been proposed using historical data and is reported to be better than random and similarity-based techniques, therefore this adequacy criterion for test prioritization can be risky [39]. A structural coverage-based TCP technique has been proposed as an optimal process including branch coverage, decision coverage, or statement coverage [40]. An epistasis-based Ant colony optimization algorithm is proposed for TCP [41]. It provides better results than traditional ACO-based techniques and Non-dominated Sorting Genetic Algorithm II (NSGA-II) in terms of APSC, and execution time. A Gready Algorithm based tie-breaking prioritization technique is proposed using lexicographical ordering [42]. Alessandro Marchetto et al. have proposed a technique to detect both business faults and technical faults early [43]. This technique is implemented through NSGA-II and is a metric-based approach. It improves execution time and fault detection. It considers code coverage, requirements coverage, and execution time. A dissimilarity-based TCP technique has been proposed by using historical failure data analysis [44]. It generates clusters of similar test cases and prioritizes the test cases based on dissimilarity. The proposed technique has been validated with random ordering, untreated ordering, and similarity order. It provided better APFD values than other comparison techniques.

A TCP technique has been proposed for JUnit test cases without having coverage information [45]. It is based on static call graphs to estimate the code coverage ability of Junit test cases. The prioritization of test cases is then made based on estimated code coverage instead of real code coverage information. The performance of this technique is then evaluated and compared with untreated, random, and dynamic coverage-based techniques. It is found that the proposed technique is better than random, and untreated prioritization but almost near to dynamic TCP in terms of APFD. Time constraints can

play an important role in the cost-effectiveness of TCP techniques and prioritization can be more effective if faults are greater in number [46]. The authors said that the techniques employing feedback are more effective than the techniques that do not employ feedback.

Five algorithms for TCP have been described including Greedy, Genetic, Additional Greedy, 2-Optimal, and Hill Climbing [47]. Their analysis indicates that the Greedy Algorithm is worse than 2-Optimal, Additional Greedy, and Genetic algorithms. There is no significant difference between 2-optimal algorithms and additional Greedy in terms of their effectiveness. GA performs well in cases where it is essential to consider the entire ordering. The use of Hill Climbing indicates that the fitness landscape is multimodal. The performance of algorithms is not dependent on the coverage criterion in the TCP problem. A prioritization approach (MR-TCP) has been proposed based on method-level risk computation [24]. The risk value of test cases is calculated based on the risk values of correlating methods within a system under test. Then test cases are prioritized based on associated risk values. The reported empirical evaluation shows that MR-TCP produced good results in terms of APFD in comparison with ANN approach (ANN-TCP), random order (RO-TCP), original order (OO-TCP), reverse order (REO-TCP), and total method coverage approach (TMC-TCP).

A prioritization technique has been proposed that is based on clustering the test cases that are related to HTTP requests which are collected from the server-side database logs [12]. Faults are supposed to be seeded to verify this technique. There is a Puzzle-based Automatic Testing (PAT) environment that breaks complex restriction-solving problems and object mutations into various small puzzles for human beings to solve [48]. This technique enhanced branch coverage. The most critical bug in the software is a crash. Test cases designed to detect crashes in the application are the most important. The Crash Locator is a method that is used to identify faulty functions by utilizing crash stack information in crash reports [49]. Do and Gregg have applied mutation testing as a TCP by measuring how rapidly a collection of test cases detects a mutant during testing and test sequences are rescheduled based on the mutant killing rate [50]. According to the author, mutation testing is also used to reduce the number of test cases without losing test effectiveness and the fault-finding rate can be increased through the automated test prioritization process.

A risk-based testing technique Rite DAP has been introduced which generates test cases for system testing from the activity diagram and performs risk-based prioritization [51].

A clustering-based prioritization technique has been proposed that utilizes the execution time of test cases and different metrics to reorder them [52]. It is claimed that this approach is better than the existing clustering and coverage-based techniques. This technique has only been applied to Java Programs. A module coupling-based technique has been proposed that takes module coupling value to prioritize the critical modules which in turn will identify high-priority test cases [53]. It is well known that using more than one technique is always better and more effective than utilizing a single technique because individual techniques detect a specific type of bug [3]. A hybrid TCP technique is developed for a better fault detection rate [54]. An input-based adaptive TCP technique is proposed is validated through experience [55]. It provided better APFD than the other code coverage-based techniques including like GA, greedy and ART. This technique is more efficient than greedy and genetic but less efficient than ART.

A FAST family of prioritization techniques has been described [56]. The FAST techniques handle huge-size test suites by utilizing Big Data techniques to achieve scalability in TCP to meet current industrial demands. The coverage-based technique usually does not consider non-code-based software artifacts like configuration files [57]. An empirical study was performed, which is a clustering approach that combines fault prediction for TCP [58]. This claims an improvement in the effectiveness of TCP. A Total coverage-based TCP approach using a modified genetic algorithm has been proposed [59]. This approach aims to improve condition coverage and execution time. A similarity-based risk-driven TCP in combination with fault prediction has been proposed [60]. In this approach, the risk of a test case increases if it is similar to a failing test case. It is better than the conventional risk measure where the risk of a test case rises if it is the very same test case, that failed in the past. A history-based TCP approach has been proposed by using TITAN technology [61]. The objective of this approach was to maximize fault detection and test coverage. A machine learning-based TCP technique has been proposed for black-box testing [62]. This technique showed that the natural language description of test cases plays a very important role in TCP. Due to this feature, APFD value can be increased for all machine learning algorithms.

An optimized test prioritization technique has been proposed by using an ant colony optimization (ACO) algorithm [63]. The objective of this technique is to increase the fault detection rate and reduce regression testing costs and time. A bat-inspired algorithm BITCP is proposed for TCP providing a good complexity percentage of fault detection

correlation [64]. The cost of individual test cases is considered for prioritizing the test cases. A multi-perspective technique for TCP is proposed for a time constraint environment. It considers the technical perspective, business perspective, and performance perspective. The objective of this technique is faster fault detection with maximum test case execution with higher failure frequency and cross-functional coverage [65]. A quality-aware TCP (QTEP) technique is proposed considering the likely dispersal of the faults in the code [66]. Test cases are prioritized based on the fault proneness of the source code. The test cases covering fault-prone source code are awarded high priority. QTEP can improve coverage-based techniques by leveraging a static bug finder, and a fault prediction model. A technique is proposed for the prioritization of the combinatorial tests set by using data flow techniques [67]. It provides a higher fault detection rate than unordered test cases. The similarity and length of test cases, and number of tuples covered, are considered for combinatorial testing.

A VB PSO based TCP algorithm is proposed for efficient random prioritization [68]. A reinforcement learning-based TCP approach has been presented for regression testing in a continuous integration context [69]. A comparative study is conducted on the performance evaluation of TCP techniques using real faults and mutants [70]. To prioritize the test cases an approach is proposed using the Firefly algorithm with a similarity distance model and evident to produce good results in terms of APFD [35]. A summary of the existing value-neutral TCP techniques along with their core objectives is presented in Table 2.1. The category of each technique is taken from the source paper.

Table 2.1: Summary of TCP techniques

Year	**Author**	**TCP objectives**	**Category**	**Ref.**
2008	Heiko *et al.*	Test case generation from the activity diagram and do risk-based prioritization.	Risk-based	[51]
2010	Askarunisa *et al.*	Prioritizes tests by using sequences of XML messages. Effective for fault detection for composite web services.	History-based	[71]
2011	Wang *et al.*	Risk-based regression testing detects most potential bugs with the minimum test cases. Saves computational resources and time using the Genetic Algorithm (GA).	Risk-based	[72]
2012	H. Mei *et al.*	A static TCP approach is proposed for the prioritization of the unit test case.	Coverage-based	[45]
2013	Ning *et al.*	A puzzle-based technique that improves branch coverage by decomposing object	Coverage-based	[48]

		mutation and constraints solving problems into small puzzles.		
2013	Ti *et al.*	History-based technique for better fault detection through version awareness.	History-based	[73]
2014	Rongxin *et al.*	Crash locating-based TCP method to uncover crash scenarios in the application through crash reports.	History-based	[49]
2015	Geetanjali *et al.*	Clustering-based novel TCP technique for better coverage with enhanced APFD considering the execution time of test cases.	Coverage-based	[52]
2015	Harish *et al.*	Coupling effect-based TCP technique that considers the module coupling effect while prioritizing tests to achieve higher APFD.	Other	[53]
2015	Dusica *et al.*	Multi-perspective regression TCP for time-constraint environments for faster fault detection with maximum test case execution with higher failure frequency and cross-functional coverage.	Other	[74]
2015	Jiang *et al.*	Input-based adaptive randomized cost-efficient TCP techniques provide a higher APFD value than ART and GA.	Search-based	[55]
2015	Konsaard *et al.*	Total coverage-based TCP using a modified GA that improves condition coverage and execution time.	Coverage-based	[59]
2015	Noor *et al.*	Similarity-based risk-driven TCP enhances the risk of a test case even if it is not the same as a failed test case but is like a failing test case.	History-Based	[75]
2016	Busjaeger *et al.*	A framework integrates existing TCP techniques through machine learning.	History-based	[76]
2016	Eghbali *et al.*	A TCP approach to increase entity coverage by using the Greedy Algorithm.	Coverage-based	[42]
2016	Marchetto *et al.*	A requirements coverage, source code coverage, and execution time-based TCP technique utilizing non-dominated sorting genetic algorithm II (NSGA-II).	Coverage-based	[43]
2016	Ansari *et al.*	A TCP approach uses ACO Algorithm to increase the fault detection rate and reduce cost and time.	Other	[63]
2017	Chen *et al.*	Adoptive random sequence-based TCP provides early fault detection and more effectiveness than random TCP techniques.	Other	[55]
2017	Xiao *et al.*	The clustering approach combines fault prediction to enhance the effectiveness of	Other	[77]

		TCP.		
2017	Wang *et al.*	QTEP, is a quality-aware TCP that considers fault proneness of code and improves existing coverage-based techniques by utilizing static defect prediction, and static bug-finder.	Coverage-based	[66]
2017	Aggarwal *et al.*	Combinatorial test data set prioritization by using data flow techniques. It provides better fault detection than unordered t-way test cases.	Coverage-based	[67]
2017	Marijan *et al.*	TITAN prioritization is based on a higher fault detection rate, failures with a high impact on users, higher requirement coverage, and test case execution time.	Coverage-based	[61]
2017	Bian *et al.*	Coverage and execution time-based TCP technique using the ACO algorithm.	Coverage-based	[41]
2017	Hasan *et al.*	A dissimilarity clustering-based TCP technique using historical data.	History-based	[44]
2018	Miranda *et al.*	Scalable similarity-based TCP in both black box and white box fashion.	Similarity-based	[56]
2018	Ozturk *et al.*	A bat-inspired algorithm-based that considers the cost of individual test cases and gives the best complexity percentage of fault detection correlation.	Fault-based	[64]
2018	Abdur *et al.*	Test cases are prioritized based on dissimilarity among test cases.	History-based	[39]
2019	Matinnejad *et al.*	A TCP technique using test coverage of test suit, their output diversity as a representation of fault revealing probability implemented through Greedy Algorithm.	Coverage-based	[34]
2019	Khatibsyarbini *et al.*	A TCP method using similarity/ dissimilarity weights and uniqueness of test cases, test cases distance implemented through the Firefly Algorithm.	Similarity-based	[35]
2019	Tahvili *et al.*	sOrTES supportive tool as TCP approach using requirements coverage, execution time, and the functional dependency between test cases.	Requirements-based	[36]
2019	Mukherjee *et al.*	A technique using modified lines covered by a test case, execution time, and the maximum amount of time required for the execution of a prioritized test case. GA, ACO, Simulated Annealing, and Knapsack Problem are used.	Coverage-based	[37]
2019	Lu *et al.*	An ACO-based TCP method to increase code coverage.	Coverage-based	[38]

2020	Jahan *et al.*	A TCP technique based on system method risk values.	Risk-based	[24]
2020	Lima *et al.*	A Multi-Armed Bandit TCP approach for a continuous integration environment is proposed.	History-based	[78]
2020	Zhou *et al.*	A distance-based TCP approach to beat random test prioritization.	Other	[79]
2020	Mohd- *et al.*	A model-based TCP technique to boost fault detection.	Model-based	[80]
2020	Venugopal *et al.*	A modification-aware TCP technique is proposed.	Modification-based	[81]
2020	WANG *et al.*	A new TCP method for service-oriented web applications using modification information.	Modification-based	[82]
2021	Iqbal *et al.*	A TCP approach for regression testing of model transformations.	Model-based	[83]
2021	Cheng *et al.*	A TCP approach for configuration testing.	Coverage-based	[84]
2021	Bagherzadeh *et al*.	Re-enforcement learning-based TCP is proposed for continuous integration.	Other	[69]
2022	F. S. Ahmed *et al*.	VB cost-cognizant TCP for regression testing.	Other	[31]

A mapping study is performed on TCP techniques in the context of continuous integration [33]. An SLR is performed on value-based cost-cognizant TCP for regression testing [31].

2.4.2. Value-Based TCP

The VB TCP techniques deal with the severityand cost in the prioritization process [85]. The VB TCP takes the challenge of integrating value consideration into the prioritization process. The value orientation in TCP ensures that prioritization satisfies its value objectives. In practice, 80% of the value exists in a 20% portion of the software [3], [8]. This fact supports the need for value orientation in software testing. However, a limited number of VB TCP techniques are available in the literature.

In this study, comprehensive literature is performed on VB TCP techniques to know the current state of research in this domain. An enhanced taxonomy of TCP techniques has been proposed so that value considerations can be considered in the TCP process. A generic cost-cognizant TCP [89–91] process with its objectives and an analysis of the

proportional differences between value-neutral and value-based TCP techniques are also given. This study emphasized the need for value orientation in TCP and highlighted that a paradigm shift is required from a VN to VB TCP process. The subsequent sections of this chapter represent the protocol and findings of the SLR. The study results yield that there is very little work done in value-based TCP as compared to value-neutral TCP. The study also highlighted a few open research problems and concluded that there is great potential for further research in value-based TCP.

2.5. SLR Protocol on Value-Based TCP

This study has been undertaken as an SLR following the standard guidelines proposed by Kitchenham and Carter [89], [90]. An SLR is a great means to know the status of research related to a specific phenomenon or a particular domain. Hence, the goal of this SLR is, to sum up the knowledge related to VB TCP techniques. The review protocol for this study contains four phases, each with two steps. In the first phase, research motivation and research questions have been described. The second phase is related to the selection of search repositories and the search process. The third phase describes two study selection criteria, including inclusion/exclusion criteria and quality assessment criteria. The last phase is related to data synthesis and data extraction. An external reviewer performed an evaluation and validation of the review protocol and provided feedback. All the feedback suggestions are incorporated to refine and improve the overall quality of the protocol. The review protocol is shown in Figure 2.1.

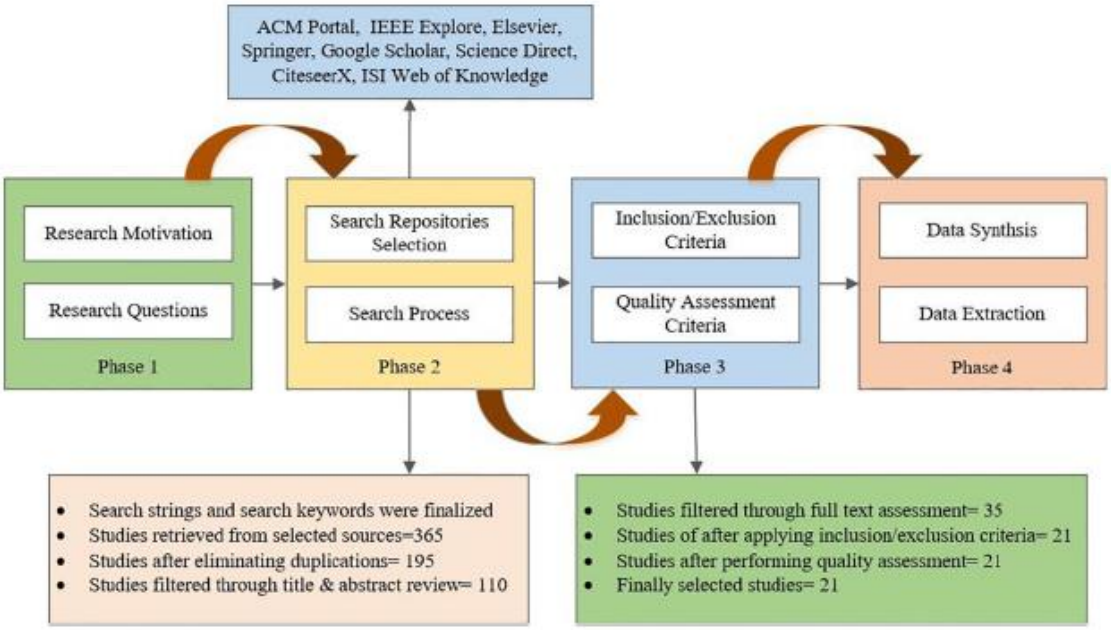


Figure 2.1: Review Protocol Phases

2.5.1. Motivation of SLR

Various SLRs are published on different dimensions of TCP. In [91], an SLR of TCP approaches is performed by Khatibsyarbinni *et al.* Its purpose is to comprehend the current research trends of TCP techniques. The taxonomic distribution of the TCP approaches is presented. It covers the pros and cons of TCP approaches in terms of their produced results. It covered the processes and artifacts involved in TCP and metrics used for the evaluation of TCP techniques. In [92], a survey is conducted on TCP techniques which contains a description of cost-cognizant TCP techniques. As per this study, Malishevsky *et al.* have suggested the cost cognitive metric APFDc for the performance evaluation of TCP techniques. $APFD_c$ considers varying test case costs and fault severity. This new metric is proposed to address the limitations of the existing metric APFD. It accounts for units of fault severity detected by units of test case cost. According to a study [91], APFD) 51%, coverage effectiveness (CE) at 10%, $APFD_c$ at 9%, Execution Time (ET) at 7%, and some other metrics are utilized at 23% for the performance measurement of TCP approaches. A study presented different categories of TCP approached including requirements-based, history-based, coverage-based, cost-aware-based, distribution-based, model-based, human-based, probabilistic-based, and others [85].

According to a study [93], the utilization of difference performance evaluation metrics is 8% (APFDc), 8% (APSC), 6% (NAPFD), 58% (APFD), and others 2%. In [33], a

mapping study is performed in a continuous integration environment for test prioritization approach. In [26], a review of GA-based TCP techniques is given. The review findings include methodologies, algorithms, performance evaluation metrics, adequacy criteria, dataset specifications, and validation criteria. According to this study, various metrics are utilized including $APFD_c$ 18%, APFD 24%, Execution Time (ET) 48%, Fault Detection 33%, Expense 15%, and NAPFD 9%. According to nother the utilization of APFD is highest, $APFD_c$ on second and then APSC is the least utilized performance metric [94].

The literature describes cost-aware/cost-cognizant as an explicit category of TCP techniques, but there is no SLR available on it. The main used TCP evaluation metrics are APFD and APFDc. For value-based cost-cognitive TCP techniques, the APFD metric is not appropriate because it has two limitations a) all test cases have identical cost and b) all faults have same severity [95]. The use of the APFD metric for performance evaluation of value-based TCP techniques is likely to produce unreliable results. These research gaps found in existing studies are the major motivation and inspiration that raised a need to publish a technically informative document in the domain of VB TCP techniques. To fill this research gap, an SLR of VB TCP techniques is performed in this study. Its purpose is, to combine the knowledge related to VB TCP techniques and to highlight the open research issues of this research domain.

To execute the SLR, a review protocol is developed to control the researcher's bias. It comprises of research questions, selection of the literature sources, search process, study selection procedure, quality assessment score, and data extraction and data synthesis. The review is assessed and validated by an external reviewer. A few suggestions are received and are incorporated to improve its quality. Each step of the review protocol is comprehensively described below.

2.5.2. Research Questions

Six research questions have been articulated that are required to be answered through this research. These questions are listed in Table 2.2. The motivation behind each research question is also presented.

Table 2.2: Research Questions

RQ	Question	Motivation/Purpose
RQ1	Which algorithms are used in value-based cost-cognizant TCP?	To know the state-of-the-art algorithms used for the implementation of the value-based TCP technique.
RQ2	Which methods (e.g., empirical study, case study, industrial case study, and experiment) are used for results validation of VB TCP?	To know the common methods of results validation for VB TCP techniques.
RQ3	What are the generic steps of the VB TCP process and its objectives?	To know the common procedure of VB TCP techniques.
RQ4	What is the enhanced taxonomy of TCP techniques considering value?	To know the current value-based categorization of TCP techniques in a taxonomic form.
RQ5	Which metrics are utilized for the performance evaluation of VB TCP techniques?	To distinguish VB metrics from value-neutral metrics.
RQ6	What are the open research problems related to TCP and what recommendations to fill the research gaps?	To highlight the limitations of current trends in TCP and to provide suggestions to fill the current research gaps.

To define the goals of SLR and scope of shortlisted studies we utilized the Population, Intervention, Comparison, Outcomes, and Context (PICOC) method [26] as mentioned below. The purpose is to address the risk of biasedness.

Population: *Literature on VB TCP.*

Intervention: *Taxonomic classification of TCP.*

Comparison: *Comparison among interventions to analyze current research of different methods.*

Outcomes: *Recommendations for further research on value-based TCP for a paradigm shift with evidence.*

Context: *An SLR combines the current body of knowledge.*

2.5.3. Literature Sources Selection

The selection of the literature sources is an important step for any SLR. We choosed prominant research databases that contains research publications related to TCP. Few existing review also utilized the same research repositories [26], [33], [91]. Below is the list of repositories that are utilized for this SLR.

- ACM Portal
- IEEE Explore
- Elsevier
- Springer
- Google Scholar
- Science Direct
- CiteseerX
- ISI Web of Knowledge
- IEEE Computer Society

2.5.4. Search Process

The search strings were formulated considering the research questions and study goals. The search strings were composed of the terms “Test Case Prioritization”, “Value-Based TCP”, “Cost-Cognizant TCP”, and “Evaluation Metrics for TCP”. The keywords used in the search process are listed below.

• Test case prioritization
• Value-based test case prioritization
• Cost-aware test case prioritization
• Cost-cognizant test case prioritization
• Test case prioritization reviews
• Test case prioritization for regression testing

• Evaluation metrics for test case prioritization

According to our search strategy, the above search strings are applied to the selected literature source databases. The literature is extracted from 2001 to August 2021. No paper was found before 2001. A total of 365 papers were retrieved. ACM Portal returned 45 papers, IEEE Explore 52, Elsevier 55, Springer 20, Google Scholar 34, Science Direct 32, CiteseerX 32, ISI Web of Knowledge 30, and IEEE Computer Society returned 65 papers.

2.5.5. Study Selection Procedure

The study selection procedure consisted of a set of steps, presented in Figure 2.2 following the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) statement [96]. A proper study selection procedure is adopted to select relevant studies and remove all irrelevant studies. Inclusion and exclusion criteria are defined to ensure that only relevant studies are selected for the study. Inclusion and exclusion criteria are presented in 2.3. The primary author selected primary studies. A test/retest approach has opted to verify the selection process. The co-author (Ph. D. Research Supervisor) performed a comparison of the results using random sampling.

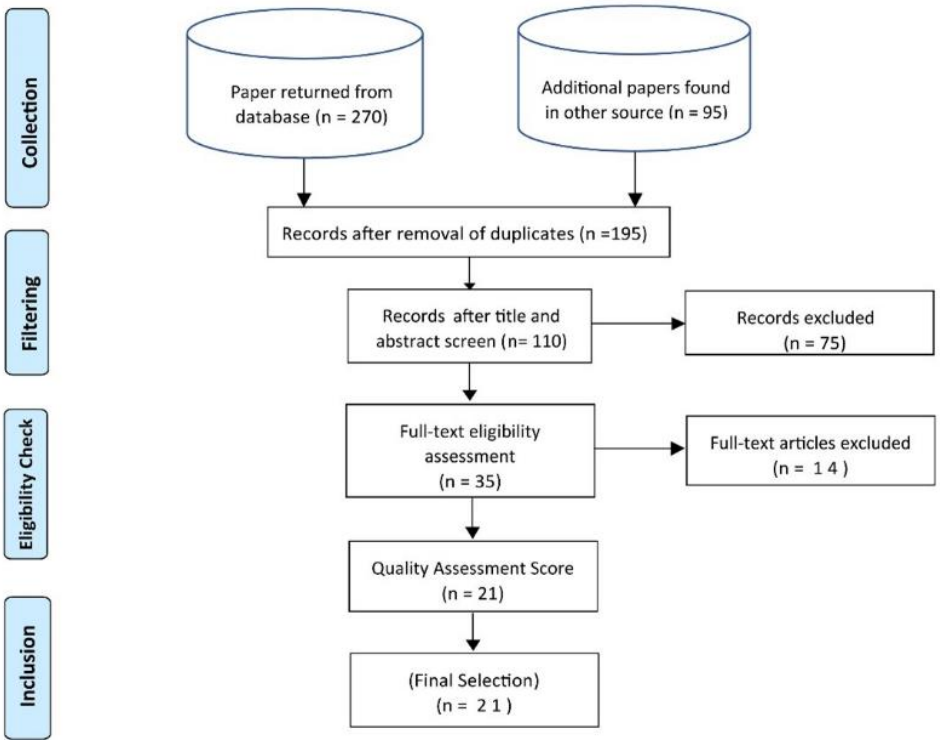


Figure 2.2: PRISMA flow diagram for the search process and selection procedure

2.5.6. Quality Assessment Score

The appropriateness of inclusion and exclusion criteria is tested and verified as per the guidelines of Brereton et al. [97]. The inclusion and exclusion criteria facilitate the selection of studies to be considered for SLR and it is utilized by existing reviews of TCP [26], [91]. The opted inclusion/exclusion criteria for this SLR are presented in Table 2.3.

Table 2.3: Inclusion and Exclusion Criteria

Inclusion Criteria	**Exclusion Criteria**
Papers on cost-aware/cost-cognizant TCP techniques	Papers not related to cost-aware/ cost-cognizant TCP techniques
Papers on value-based TCP techniques	Papers with no value-orientation
Papers on evaluation metrics/measures for cost-cognizant TCP	Papers without evaluation metrics and empirical study
The papers that are written in the English language only	Papers that are presented in a language other than English
The latest revised version of the papers is considered	Papers with duplicate revised versions have been removed

The Quality Assessment Score (QAS) provides help to evaluate the relevance and significance of the study [90]. A study is selected or rejected based on the QAS. For study selection, we formulated a three-point QAS checklist following the guidelines of Kitchenham et al. [90]. The checklist is given in Table 2.4.

Table 2.4: QAS Checklist

S. No.	QAS Question	Feedback	QAS
1.	Does the study propose a value-based cost-cognizant TCP technique?	(Yes=2) (Partial=1) (No=0)	
2.	Does the study contain empirical evaluation?	(Yes=2) (Partial=1) (No=0)	
3.	Does the study use cost-cognizant evaluation metrics?	(Yes=2) (Partial=1) (No=0)	

Based on the defined search process, we retrieved 365 papers from the selected literature sources. First, 170 duplicate papers were removed. On the remaining 195 papers, the title and abstract review were performed, and 75 papers were screened out. Through a detailed review of the full text and by applying inclusion and exclusion criteria, 14 more studies were removed. Afterward, we performed a quality assessment and as a result, 14 more studies were removed. Finally, 21 papers were selected for the study. The search process and selection procedure are depicted in the PRISMA flow diagram in Figure 2.2.

2.5.7. Data Extraction and Data Synthesis

In any review protocol, the process of extracting, and synthesizing data from the selected studies is a prominent reason that distinguishes an SLR from a traditional literature review. In the extraction process, data is extracted from the selected studies relevant to the SLR questions whereas the data synthesis process is the collective form of the results derived from those studies [89]. In the data extraction process, we collected bibliographic information (Unique ID, title, authors, year of publication, citations, paper type, publisher), common steps in the TCP process, the algorithm used, evaluation metrics used, the results validation method, the dataset availability, contribution, category of TCP technique, and open research problems. To collect data from the primary studies, a data extraction form is designed and is given in 2.6. In the data synthesis process, the extracted data is combined and organized in such a way that it can be useful to answer the defined research questions. Table 2.5 presents the outcome of the data synthesis process.

Table 2.5: Data Extraction Form

S. No.	Characteristic	Value
1.	Algorithm used	• TERMINATOR Algorithm • Particle swarm optimization (PSO) • Additional Greedy • Sorting Algorithm • Genetic Algorithm • Custom algorithm • Total Statement Coverage • Function coverage • PORT Algorithm
2.	Evaluation metric used	• APFDc • APFDc • APBIE • $APFD_a$ • MRP_TC • ASFD
3.	The results validation method used	• Empirical study • Case study • Industrial case study • Experiment
4.	Dataset availability	• Yes • No
5.	Category	• History-based • Search-based • Coverage-based • Fault-based • Requirements-based • Risk-based

Table 2.6 presents the data extraction results from the selected studies.

Table 2.6: Data Extraction Results

P.ID	Author	Prioritization Algorithm Used	Evaluation Metric Used	Results Validation	Dataset Availability	Category
P1	Yu *et al.* (2019), [98]	TERMINATOR Algorithm	$APFD_c$	Empirical study	Yes	History-based
P2	Ashraf *et al.* (2017), [68]	Particle swarm optimization PSO	APFD	Experiment	No	Search-based
P3	Miranda *et al.*	Additional Greedy	$APFD_c$	Experiment	No	Coverge based

	(2017), [99]					
P4	Wang *et al.* (2015), [100]	Sorting Algorithm	APFD	Experiment	No	Fault-based
P5	Epitropakis *et al.* (2015), [101]	-------	$APFD_c$	Empirical study	No	Coverage-based
P6	Rauf *et al.* (2015), [102]	Particle swarm optimization PSO	APFD	Experiment	No	Fault-based
P7	Hoq *et al.* (2015), [103]	Sorting Algorithm	$APFD_c$	Experiment	No	Fault-based
P8	Yc *et al.* (2012), [86]	Genetic Algorithm	$APFD_c$	Experiment	No	History-based
P9	Li *et al.* (2013), [104]	-------	APBIE	Industrial Case study	No	Coverage-based
P10	Marijan *et al.* (2013), [105]	-------	$APFD_c$	Industrial Case study	No	Fault-based
P11	Ashraf *et al.* (2012), [17]	Particle swarm optimization PSO	APFD	Experiment	No	Search-based
P12	Ramler *et al.* (2012), [9]	-------	-------	-----	No	Coverage-based
P13	Zhang *et al.* (2011), [106]	-------	$APFD_a$	Case study	No	Fault-based
P14	Bryce *et al.* (2011), [107]	Custom algorithm	$APFD_c$	Empirical study	No	Coverage-based
P15	Askarunisa *et al.* (2010), [71]	Total Statement Coverage	$APFD_c$	Experiment	No	Coverage-based

P16	Park *et al.* (2008), [88]	------	$APFD_c$	Experiment	No	History-based
P17	Zhang *et al.* (2007), [108]	------	MRP_TC	Case Study	No	Requirements-based
P18	Malishevsky *et al.* (2006), [87]	Function coverage	$APFD_c$	Case study	No	Fault-based
P19	Srikanth *et al.* (2005), [109]	PORT Algorithm	ASFD	Case study	No	Requirements-based
P20	Srikanth *et al.* (2005), [110]	------	ASFD	Experiment	No	Requirements-based
P21	Elbaum *et al.* (2001), [111]	------	$APFD_c$	Case study	No	Fault-based

Table 2.7 shows the studies published on value-based TCP. It contains paper ID as P.ID, author, the approach used, and Quality Assessment Score (QAS).

Table 2.7: Summary of studies on value-based TCP

P. ID	Author	Approach Used	QAS
P1	Yu *et al.* (2019), [98]	An approach TERMINATOR for the prioritization of automated UI test cases. There is a computational overhead that recursively updates the Support Vector Machine (SVM) model to tweak the order of un-executed tests. Increased fault detection without the availability of source code. Made dataset available to reproduce results.	6
P2	Ashraf *et al.* (2017), [68]	A value-based TCP technique based on six prioritization factors using PSO to enhance the fault detection rate. An experimental method is used to validate the results.	4
P3	Miranda *et al.* (2016), [99]	A scope-aided TCP method for a better fault detection rate.	5
P4	Wang *et al.* (2015), [100]	A faults-severity-based TCP method to increase fault detection by accumulative severity detected by a test case.	5

P5	Epitropakis *et al.* (2015), [101]	An empirical evaluation of seven algorithms has been done on their fault detection capability and maximizing coverage. APFDc is used as an evaluation metric.	6
P6	Rauf *et al.* (2015), [102]	A value-based TCP method using PSO algorithm to enhance the fault detection rate. An experiment is done to prove the results.	4
P7	Hoq *et al.* (2015), [103]	A dependency-cognizant TCP technique to detect more severe faults earlier in the testing life cycle within minimum test case execution time.	5
P8	Yc *et al.* (2012), [86]	A history-based cost-cognizant TCP method by applying a GA to produce an effective test case order. A controlled experiment is performed to validate the results.	5
P9	Li *et al.* (2013), [104]	A value-based prioritization method that lets tests be ordered by how well the tests can decrease risk exposure. Combining this with the tests' relative costs aids them to be prioritized in terms of the Return on Investment (ROI) or risk reduction leverage (RRL). A novel metric Average Percentage of Business Importance Earned (APBIE) is proposed for performance evaluation.	5
P10	Marijan *et al.* (2013), [105]	ROCKET a TCP approach for continuous regression testing, was applied to an industrial case study to increase the fault detection rate with minimum execution time.	5
P11	Ashraf *et al.* (2012), [17]	A TCP algorithm that orders the system test cases based on six different factors: customer priority, changes in the requirement, requirement traceability, execution time, implementation complexity, and fault impact of the requirement.	4
P12	Ramler *et al.* (2012), [9]	A value-based coverage approach for requirement-based testing to enhance business value coverage.	4
P13	Zhang *et al.*(2011), [106]	A new cost-cognizant metric is proposed for the performance evaluation of TCP techniques.	4
P14	Bryce *et al.* (2011), [107]	A cost-based combinatorial interaction coverage TCP technique and a new metric for it are proposed. An improvement in fault detection is evident through an empirical study.	5
P15	Askarunisa *et al.* (2010), [71]	The cost and Coverage based TCP technique is proposed and cost and coverage-based metrics are used for performance evaluation.	5
P16	Park *et al.* (2008), [88]	A VB TCP approach based on historical value to estimate fault severity and test case cost to improve regression testing effectiveness.	5
P17	Zhang *et al.* (2007), [108]	A cost-cognizant TCP method based on varying requirements priority and test cases cost. A new evaluation metric MRP_TC has also been proposed.	4
P18	Malishevsky *et al.* (2006), [87]	The cost-cognizant TCP method and a new metric cost-cognizant evaluation metric APFDc consider fault severity and test case cost in the TCP process.	6
P19	Srikanth *et al.* (2005), [109]	A value-based TCP approach based on the Prioritization of Requirements for Tests (PORT) is presented. A case study was	5

		done to prove the results and an increase in severe fault detection is evident.	
P20	Srikanth *et al.* (2005), [110]	A requirements-based TCP technique to boost the rate of detection of severe faults.	5
P21	Elbaum *et al.* (2001), [111]	A new evaluation metric that incorporates varying test cases cost and fault severity for cost-cognizant TCP.	6

Table 2.8 shows the unique ID, title, authors, year of publication, citations, paper type, and publisher of the studies. Some common steps in the TCP process, the algorithm used, evaluation metrics used, the results validation method, the dataset availability, contribution, category of TCP technique, and open research problems are also highlighted.

Table 2.8: Data extraction results

P.ID	Author	Prioritization Algorithm Used	Evaluation Metric Used	Results Validation	Dataset Availability	Category
P1	Yu *et al.* (2019), [98]	TERMINATOR Algorithm	$APFD_c$	Empirical study	Yes	History-based
P2	Ashraf *et al.* (2017), [68]	Particle swarm optimization PSO	APFD	Experiment	No	Search-based
P3	Miranda *et al.* (2016), [99]	Additional Greedy	$APFD_c$	Experiment	No	Coverage based
P4	Wang *et al.* (2015), [100]	Sorting Algorithm	APFD	Experiment	No	Fault-based
P5	Epitropakis *et al.* (2015), [101]	-------	$APFD_c$	Empirical study	No	Coverage-based
P6	Rauf *et al.* (2015), [102]	Particle swarm optimization PSO	APFD	Experiment	No	Fault-based
P7	Hoq *et al.* (2015), [103]	Sorting Algorithm	$APFD_c$	Experiment	No	Fault-based
P8	Yc *et al.* (2012), [86]	Genetic Algorithm	$APFD_c$	Experiment	No	History-based
P9	Li *et al.* (2013), [104]	-------	APBIE	Industrial Case study	No	Coverage-based
P10	Marijan *et al.* (2013), [105]	-------	$APFD_c$	Industrial Case study	No	Fault-based
P11	Ashraf *et al.* (2012), [17]	Particle swarm optimization PSO	APFD	Experiment	No	Search-based
P12	Ramler *et al.* (2012), [9]	-------	-------	-----	No	Coverage-based

P13	Zhang *et al.* (2011), [106]	-------	$APFD_a$	Case study	No	Fault-based
P14	Bryce *et al.* (2011), [107]	Custom algorithm	$APFD_c$	Empirical study	No	Coverage-based
P15	Askarunisa *et al.* (2010), [71]	Total Statement Coverage	$APFD_c$	Experiment	No	Coverage- based
P16	Park *et al.* (2008), [88]	------	$APFD_c$	Experiment	No	History-based
P17	Zhang *et al.* (2007), [108]	------	MRP_TC	Case Study	No	Requirements-based
P18	Malishevsky *et al.* (2006), [87]	Function coverage	$APFD_c$	Case study	No	Fault-based
P19	Srikanth *et al.* (2005), [109]	PORT Algorithm	ASFD	Case study	No	Requirements-based
P20	Srikanth *et al.* (2005), [110]	------	ASFD	Experiment	No	Requirements-based
P21	Elbaum *et al.* (2001), [111]	------	$APFD_c$	Case study	No	Fault-based

2.5.8. Assessment and Findings

After a comprehensive analysis of the selected studies and synthesized data, the assessment is performed, and the findings are concluded. In this section, all the defined research questions have been answered.

The first paper related to value-based TCP was published in 2001 and proposed a cost-cognizant performance evaluation metric APFDc [111]. Later, a few other authors used this metric for the performance evaluation of their proposed TCP technique [87], [88], [71], [107], [105], [86], [103], [101], [99], [98]. We found that only a few papers were published in a value-based fashion that used test case cost and fault severity in the test case prioritization process. The current trend of VB TCP techniques shows that limited work is available in a VB approach. Therefore, more TCP studies are required in a value-based fashion to get better and more reliable results. There is great potential for further research to fill the gaps. The leading researchers in the domain of value-based cost-cognizant TCP techniques are Gregg Rothermel, Sebastian Elbaum, and Alexey Malishevsky [87], [111], [19].

Table 2.9: Research Trends of Value-based cost cognizant TCP techniques

P.ID	Author	Year	Reference	Publisher
P1	Yu *et al.*	2019	[98]	Journal
P2	Ashraf *et al.*	2017	[68]	Journal
P3	Miranda *et al.*	2017	[99]	Journal
P4	Wang *et al.*	2015	[100]	Journal
P5	Epitropakis *et al.*	2015	[101]	Journal
P6	Rauf *et al.*	2015	[102]	Journal
P7	Hoq *et al.*	2015	[103]	Conference
P8	Yc *et al.*	2012	[86]	Journal
P9	Li *et al.*	2013	[104]	Journal
P10	Marijan *et al.*	2013	[105]	Conference
P11	Ashraf *et al.*	2012	[17]	Journal
P12	Ramler *et al.*	2012	[9]	Journal
P13	Zhang *et al.*	2011	[106]	Conference
P14	Bryce *et al.*	2011	[107]	Journal
P15	Askarunisa *et al.*	2010	[71]	Journal
P16	Park *et al.*	2008	[88]	Conference
P17	Zhang *et al.*	2007	[108]	Journal
P18	Malishevsky *et al.*	2006	[87]	Journal
P19	Srikanth *et al.*	2005	[109]	Journal
P20	Srikanth *et al.*	2005	[110]	Journal
P21	Elbaum *et al.*	2001	[111]	Journal

2.5.8.1. Algorithms used in value-based cost-cognizant TCP (RQ1)

Different algorithms have been used in value-based cost-cognizant TCP techniques. The algorithms include TERMINATOR Algorithm, Particle swarm optimization (PSO), Additional Greedy, Sorting Algorithm, Genetic Algorithm, Custom algorithm, Total Statement Coverage, Function coverage, and PORT Algorithm. Nine studies did not use any algorithm because they sorted their test cases based on some prioritization criteria. PSO is a dominantly used algorithm in the studies P2, P6, and P11. Figure 2.3 shows the study distribution according to the algorithm used in the selected studies.

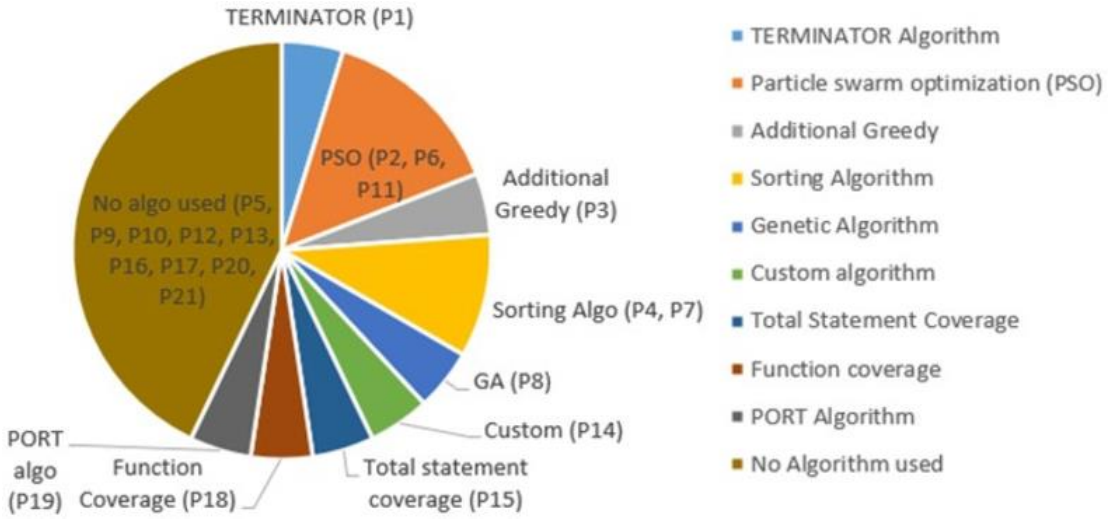


Figure 2.3: Distribution of studies according to the algorithm used.

2.5.8.2. Validation methods used in VB TCP (RQ2)

The literature indicates that four results validation methods have been used including, Empirical study, Case study, Industrial case study and Experiment. Papers P1, P5, and P14 used the empirical study method to validate their results. Empirical evaluation is usually based on the researcher's observations and investigation of the phenomenon. Papers P2, P3, P4, P6, P7, P8, P11, P15, P16, and P20 used the experiment method to compare results with the existing state-of-the-art techniques. The experiment method is usually applied for a small project. Most of the studies are done with a small scope. Papers P13, P17, P18, P19, and P21 used the case study method to validate their results. The case study is usually applied to a specific case to validate the results. Papers P9 and P10 used the industrial case study method to validate the results. Industrial case studies usually cover real industrial projects. Paper P12 did not use any validation method. Figure 2.4 shows the distribution of selected studies according to the validation method used.

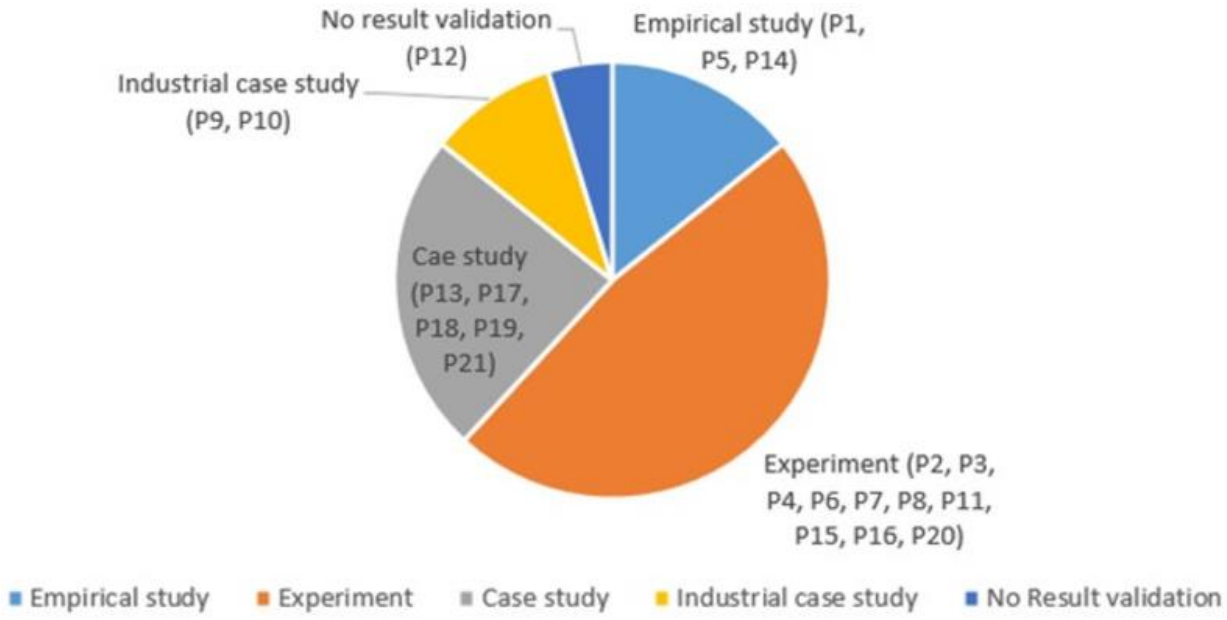


Figure 2.4: Distribution of studies according to results validation method used.

2.5.8.3. VB TCP process and its objectives (RQ3)

The use of standard processes and clarity of objectives are vital for the success of software projects. In the TCP process, test cases are not eliminated or removed, rather each test case is assigned a priority. The test cases in the test suite are sorted by their priority. Then the testing team starts executing the test cases with the highest priority and ends when regression time ends, or all test cases are covered. A variety of TCP techniques are available including search-based, requirements-based, coverage-based, history-based, fault-based, risk-based, and others [91]. This sub-section presents a TCP process that describes few common steps involved in VB TCP techniques. These steps are reported in different studies [17], [68], [88] and are shown in Figure 2.5.

1. *Prepare the list of test cases that are required to be prioritized.*
2. *Define the parameters to be used for prioritization assuming that different faults may have different severity and different test cases may have different cost.*
3. *Apply the formula to calculate the prioritization score for each test case.*
4. *Prepare test data set with prioritization score that needs to be prioritized.*
5. *Run the prioritization algorithm based on prioritization criteria/scores.*
6. *Prepare test dataset in prioritized order.*
7. *Execute the test cases as per the assigned priority in the above step using a value-based cost-cognizant metric and evaluate the results. Value-based cost cognizant*

metric is a metric in which varying severity of faults and test case costs are considered.

8. *Repeat step 3.*

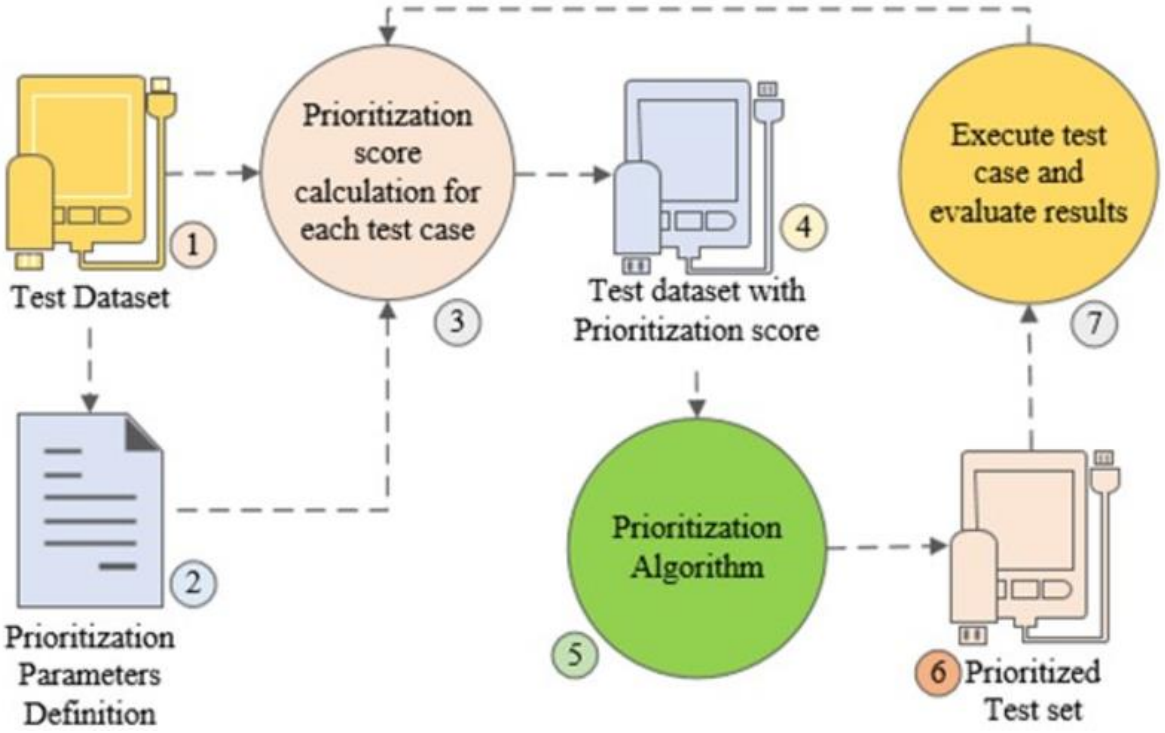


Figure 2.5: Common Test Case Prioritization Process

A few common objectives of VB TCP techniques given in different studies have also been presented and are depicted in Figure 2.6.

- Early fault severity detection is the major objective of value-based TCP. Late detection of bugs is more costly. Therefore, there is a direct relationship between the cost with this TCP objective.
- To provide quick product maturity through critical fault detection first. Maximum bugs are detected and fixed earlier, therefore, the product gets mature and ultimately builds confidence to meet the deadline.
- Efficient utilization of testing resources during regression testing.
- Early business value coverage
- Saving time and budget

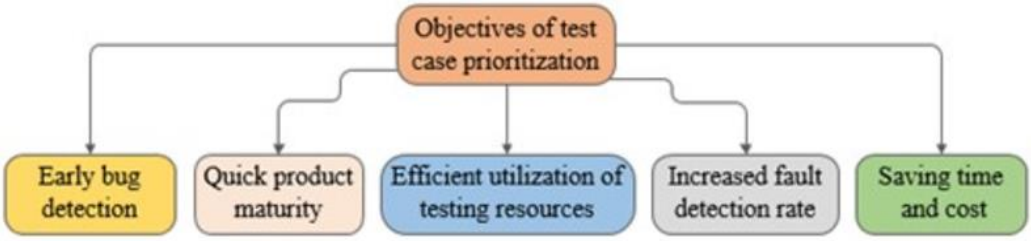


Figure 2.6: The objectives of Test Case Prioritization

2.5.8.4. An enhanced taxonomy of TCP techniques (RQ4)

RQ4 has been answered in section 3.3. Table 3.1 shows the categorization of value-based cost-cognizant TCP techniques selected for this study, and an enhanced taxonomy of TCP techniques has been presented in Figure 3.2.

2.5.8.5. Performance evaluation metrics used in cost-cognizant TCP (RQ5)

For the validation of any proposed technique, its performance is evaluated. Using the right metric for the performance evaluation of TCP is imperative to get reliable and correct results. Various metrics are there to evaluate the performance of different permutations. Here we have described a few well-known metrics. This section describes the performance evaluation metrics used for TCP techniques given in different studies. There are many metrics used for the performance evaluation of TCP techniques.

The Average Percentage of Fault Detection (APFD)

APFD is a classical metric used for the performance measurement of TCP techniques and was developed by Sebastian Elbaum et al. in 2000 [112]. It is dominantly used and popular among researchers who worked on TCP problem. APFD is presented by equation 1.

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{n} TF_i}{mn} + \frac{1}{2n} \qquad (1)$$

In this formula, m is the total number of faults detected and n is the total number of test cases, and TFi is the place of the first test case that reveals fault Fi. This is a value-neutral metric and is very likely to produce unsatisfactory results in cases where the severity of faults and cost of test cases vary.

The Average Percentage of Fault Detection Per Cost ($\mathbf{APFD_C}$)

$APFD_c$ metric was proposed by the researchers to overcome the shortcomings of the APFD metric [19]. $APFD_c$ considers varying test case costs and fault severity. This new cost-cognizant metric was proposed in a value-based fashion. It accounts for units of fault severity exposed by units of test case cost. The x-axis shows the total units of test case cost instead of simply showing the percentage of test cases executed and similarly y-axis implies the total units of fault severity instead of simply showing the percentage of faults detected. $APFD_c$ has been presented by equation 2.

$$\mathrm{APFD_C} = \frac{\sum_{i=1}^{m}(f_i \times (\sum_{j=TF_i}^{n} t_j - \frac{1}{2}t_{TF_i}))}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} f_i} \qquad (2)$$

In equation 2, *T* is the test suite and *n* is the number of test cases with costs *t1, t2, …, tn. F* is a set with *m* number of faults detected by *T*, and *f1, f2…, fm* is the severities of faults. *TFi* is the first test case in a test case order that detects fault *i*. APFDc is also widely used as a performance evaluation metric for TCP techniques.

The Average Percentage of Fault Detection ($\mathbf{APFD_a}$)

This $APFD_a$ is recommended by Zhang *et al* and is an improved form of $APFD_c$ [106]. It well presents the physical details of the testing process. This metric has been presented by equation 3.

$$\mathrm{APFD_a} = \left(1 - \sum_{i=1}^{m} \frac{\sum_{j=1}^{TF_i} C_j}{m \sum_{j=1}^{n} C_j}\right) \times 100\% \qquad (3)$$

Here, T is the test suite of n test cases, C_j is the cost of test cases and F is the set of faults containing m faults. TF_i is the first test case that detects i^{th} fault.

Metric Based on Varying Requirements' Priority and Test Cases' Cost (M_{RP_TC})

This metric was introduced by Zhang *et al.* and it is based on varying requirements' priority and test case cost [108]. It can be represented by equation 4. The value range of $M_{RP_TC}$is 0 to 100% and higher value implies the better performance.

$$\mathrm{M_{RP\_TC}} = \frac{\sum_{j=1}^{m}\left(rf_i \times (\sum_{i=TR_i}^{n} tc_i - \frac{1}{2} * tcTR_i)\right)}{\sum_{j=1}^{m} rp_i \times \sum_{j=1}^{n} tc_j} \qquad (4)$$

The Average Severity of Faults Detected (ASFD)

The ASFD metric was introduced by Hema Srikanth and Laurie Williams [110]. ASFD value is the ratio of the sum of severities detected by a specific requirement and the Total Severity of the Faults Detected (TSFD). It can be represented by equation 5.

$$\mathrm{ASFD_i} = [\frac{\sum_{j=1}^{m} \mathrm{SV_j}}{\mathrm{TSFD}}] \qquad (5)$$

The Average Percentage of Business Importance Earned (APBIE)

The APBIE was introduced by Qi Li, and Barry Boehm[104]. This metric is proposed to cover the business significance of the system under test. It can be represented by equation 6.

$$\mathrm{APBIE} = \sum_{i=1}^{n} \mathrm{PBIE_i}/n \qquad (6)$$

Modified APFDc: A location-based TCP technique has been proposed and its performance is evaluated by using APFD and a modified version of APFDc considering that test case cost is identical but fault severity may vary [113]. The modified form of APFDc is represented by the following equation 7.

$$\mathrm{APFD_c} = 1 - \frac{\sum_{i=1}^{m}(\mathrm{fs_i\ X\ TF_i})}{n\ x \sum_{i=1}^{m}(\mathrm{fS_i})} + \frac{1}{2n} \qquad (7)$$

The above-modified version of APFDc is partially cost-cognizant because it only considers the varying severity of faults and does not consider the varying cost of the test cases. Test cases' cost or execution time is an important factor and is among the primary reasons for TCP.

Average Percentage of Statement Coverage (APSC) is also a popular metric to evaluate of performance of the TCP mechanism in the context of meeting its statement coverage objective. This metric is designed on a similar pattern as APFD. It can be represented by equation 8.

$$\mathrm{APSC} = 1 - \frac{\mathrm{TS_1+TS_2+TS_3+\cdots+TS_m}}{n*m} + \frac{1}{2*n} \qquad (8)$$

In [19], two cost-cognizant prioritization techniques have been discussed including additional statement coverage prioritization (st-addtl) and additional function coverage prioritization (fn-addtl). In st-addtl, the author estimated the criticality of the statement by the severity of faults that occurred in that statement. Similarly, the criticality of a function is estimated by the severity of faults that occurred in that function. Test cases are then prioritized on the "statement criticality ratio cost of test case" for st-addtl prioritization. Similarly, test cases are prioritized on the "function criticality ratio cost of the test case" for fn-addtl prioritization. Estimating the criticality of statements, and functions through the severity of their associated faults is not an appropriate method used in prioritization. We believe that the term "value" is more appropriate instead of criticality. The value of statements and the value of functions should be estimated through the business value of the requirement with which they are associated instead of estimated through fault severities.

2.5.8.6. Open research problems and recommendations to fill the research gaps (RQ6)

After analysis of the existing literature on value-based TCP techniques some open research problems are highlighted here and a few directions on how to improve the reliability of value-based TCP techniques are also given. According to a study [114], cost-cognizant TCP techniques are used when assumptions associated with APFD do not hold. These assumptions are that all faults have similar severity, and all test cases have similar costs. In practice, these assumptions seldom hold and, in this scenario, the APFD metric does not remain appropriate to evaluate the performance of TCP techniques. Below are a few open research questions related to the evaluation metric selection.

- How often do APFD assumptions hold?
- If the above assumptions rarely hold, then why is the APFD measure highest in popularity in TCP research?
- Is $APFD_c$ a good alternative to APFD? If yes, then why $APFD_c$ is far behind in comparison with APFD?
- Does the research community need a new standard measure for the performance measurement of TCP techniques?

A limitation of the existing literature is that while using the APFD metric, the researchers

did not explicitly mention whether the basic assumptions of APFD hold or not. They did not mention the reason why they selected APFD as an evaluation metric. Most of the researchers expressed that APFD is the most popular measure which is why we are using it. This is not a strong and valid justification for the metric selection. Therefore, the results produced by using APFD are mostly unreliable in cases where the assumption "all faults have equal severity and all test cases have equal cost" does not hold [111]. The selection of performance evaluation metrics is still an open research problem. The coverage-based techniques are related to a specific element like a statement, condition, branch, method, or requirement. The metric used for the performance evaluation of these techniques is the APSC, APCC, APBC, Average Percentage of Method Coverage (APMC), and APRC. These metrics can be collectively described as the Average Percentage of Element Coverage (APEC) [26]. These metrics are like APFD and are based on the same assumption that all statements, conditions, branches, methods, or requirements are of the same worth, and the test cases used to cover these certain elements have the same cost. This is a major limitation of coverage-based techniques and their utilized metrics, and they might not produce the intended results. We suggest that value considerations should be considered in coverage-based techniques and traditional coverage-based metrics should be replaced with value-based coverage metrics. Introducing value in the TCP process can make TCP techniques more efficient and effective.

The existing coverage-based TCP techniques are not aligned with the client's priorities. The client may be bothered by some features and may not be with others. Some features may involve profitability and productivity for the client business, and some may not. There is a slogan in SE that "Client is always right" [115]. However, the client's perspective is missing in existing TCP techniques. Most of the existing TCP techniques have been proposed in a value-neutral fashion and do not consider the client's business value expectations. Business value orientation has great potential in the TCP process. Proposed techniques detect a huge number of faults even then releases become late because high-severity bugs are detected late in the regression cycle. Debugging and fixing such critical faults at the eleventh hour creates stress on development teams and fixes become prone to further errors. Therefore, detecting critical faults early in the regression testing life cycle is vital. The value-based TCP branch is still a gray area. There is great potential for further research related to value orientation in all categories of TCP. Value orientation should be considered in research methods, study contexts, prioritization

approaches, and performance evaluation metrics. This is a big surprise that TCP research is continuously coming in a value-neutral fashion despite knowing the fact that all software elements do not have equal worth. Here are a few recommendations to fill the research gap.

VB test prioritization should be more focused. VN TCP techniques are not likely to produce satisfactory and reliable results. Therefore, TCP remains unable to achieve its intended goals. To fill this major research gap, further research in the domain of TCP should be done in a value-based fashion, and performance evaluation should be done through value-based cost-cognizant metrics. The study results show that there is a limited application of machine learning techniques in value-based cost-cognizant TCP techniques. Further research should try to solve the TCP problem by applying machine learning techniques to achieve efficiency in the prioritization process. It is also evident that the dataset used for results validation is publicly available only for one study P1. The rest of the studies did not use the public data set for validation. Public datasets should be used so that future researchers can conduct empirical studies to reproduce the results and make further improvements. A limitation is reported in the recent literature that simple statements and traditional coverage cannot guarantee 100% fault detection [26]. Coverage-based techniques are producing less optimistic outcomes. Utilizing value coverage can overcome this limitation. According to the study, most of the work done covers only the functional aspects of the applications [14]. Security, usability, privacy, and performance are very important and perhaps these are not addressed in traditional code coverage metrics. There is a need for coverage of non-functional aspects in the TCP process as well.

2.6. Summary

The TCP is a popular approach for regression testing to meet time and budget constraints. There are two major classes of TCP techniques 1) Value-neutral TCP techniques and 2) Value-based TCP techniques. Both classes have many other categories like coverage-based, history-based, and risk-based. The value-neutral TCP techniques assume that all elements like statements, requirements, test cases, use cases, methods, and bugs are equally important. This assumption seldom holds therefore VN TCP techniques are likely to produce unreliable results. Due to this major limitation of the TCP process, VB TCP

techniques are gaining popularity. The objective of the comprehensive literature review on existing TCP techniques is to see the current state of research in this field.

- This literature review is evident that there is very limited work on value-based test prioritization. It is needed to realize that without value considerations in the TCP process, its intended results cannot be achieved.
- The right metric selection for the performance evaluation of TCP techniques is essential to get reliable results. Popularity-based metric selection is not a valid justification, and it cannot produce reliable results. This is a big area for further improvement. The efficiency and effectiveness of TCP approaches are strongly dependent on the correct evaluation metric because a researcher usually targets an improvement in a metric value while proposing a TCP technique.

This literature review yields that there is great potential in value-based cost-cognizant TCP and future research should cover this important dimension.

CHAPTER 3

RESEARCH METHODOLOGY

3.1. Introduction

In software engineering, VN fashion is dominant over VB fashion. The same goes for software testing, regression testing, and the TCP. There is limited work done in VB TCP. The focus of this work is value-based cost cost-cognizant TCP. This chapter contains different sections. Section 3.2 describes the research design very comprehensively with an overview of the research methodology to achieve the above-specified goals. It has three phases as depicted in Figure 3.1. Phase 1 is related to the literature review. An SLR is conducted in this phase and all details are mentioned in chapter 2. As an outcome of phase 1, TCP techniques are categorized into two major categories including value-neutral TCP and value-based TCP. These are part of phase 2 and are described as an Enhanced Taxonomy of TCP techniques mentioned in section 3.3. In phase 2 of the research methodology, a model is proposed to estimate the business value of requirements. The business value is estimated based on five business success factors. This estimated value of requirements is later utilized to estimate fault severity and test case cost. The purpose is to incorporate business value in the TCP process. Two novel value-based TCP techniques and two novel performance evaluation metrics have been proposed for it. The proposed techniques have been implemented using Genetic Algorithms (GA). Phase 3 described the validation process. The proposed quantification model, TCP techniques, and performance evaluation metrics have been validated with the help of two working examples and two case studies. Finally, statistical analysis is performed to conclude the results.

3.2. Research design

In this work, quantitative research methods are adopted to achieve the defined goals. There are three phases of this research work. The first phase consists of comprehensive literature on the domain under study. The first goal is achieved in this phase. The second goal of the study is achieved in the second phase of the study. In the second phase, an enhanced taxonomy of the TCP techniques has been proposed to segregate value-based TCP techniques from value-neutral TCP techniques. To introduce business value in the TCP process, a business value estimation model for requirements has been introduced. A mechanism is introduced to estimate the fault severity and test case cost based on the business value of requirements. The third goal was to propose value-based TCP techniques and performance evaluation metrics. To achieve this goal, a Value Cognizant Fault Detection Based Test Case Prioritization (VCFDB-TCP) technique and a Value Cognizant Requirements Coverage Based TCP (VCRCB-TCP) have been proposed. is proposed. The fourth goal was to propose novel metrics for performance evaluation of the value-based TCP techniques. To achieve this goal a novel metric $APFD_v$ has been introduced for the performance evaluation of VCFDB-TCP and a novel metric $APRC_v$ has also been introduced for the performance evaluation of VCRCB-TCP. The proposed business value estimation model, TCP techniques, and performance evaluation metrics have been presented in this chapter.

This third phase of the study is related to the validation of the proposed business value estimation model, proposed TCP techniques, and proposed performance evaluation metrics. It includes statistical analysis of the case studies. It is presented in Chapter 4.

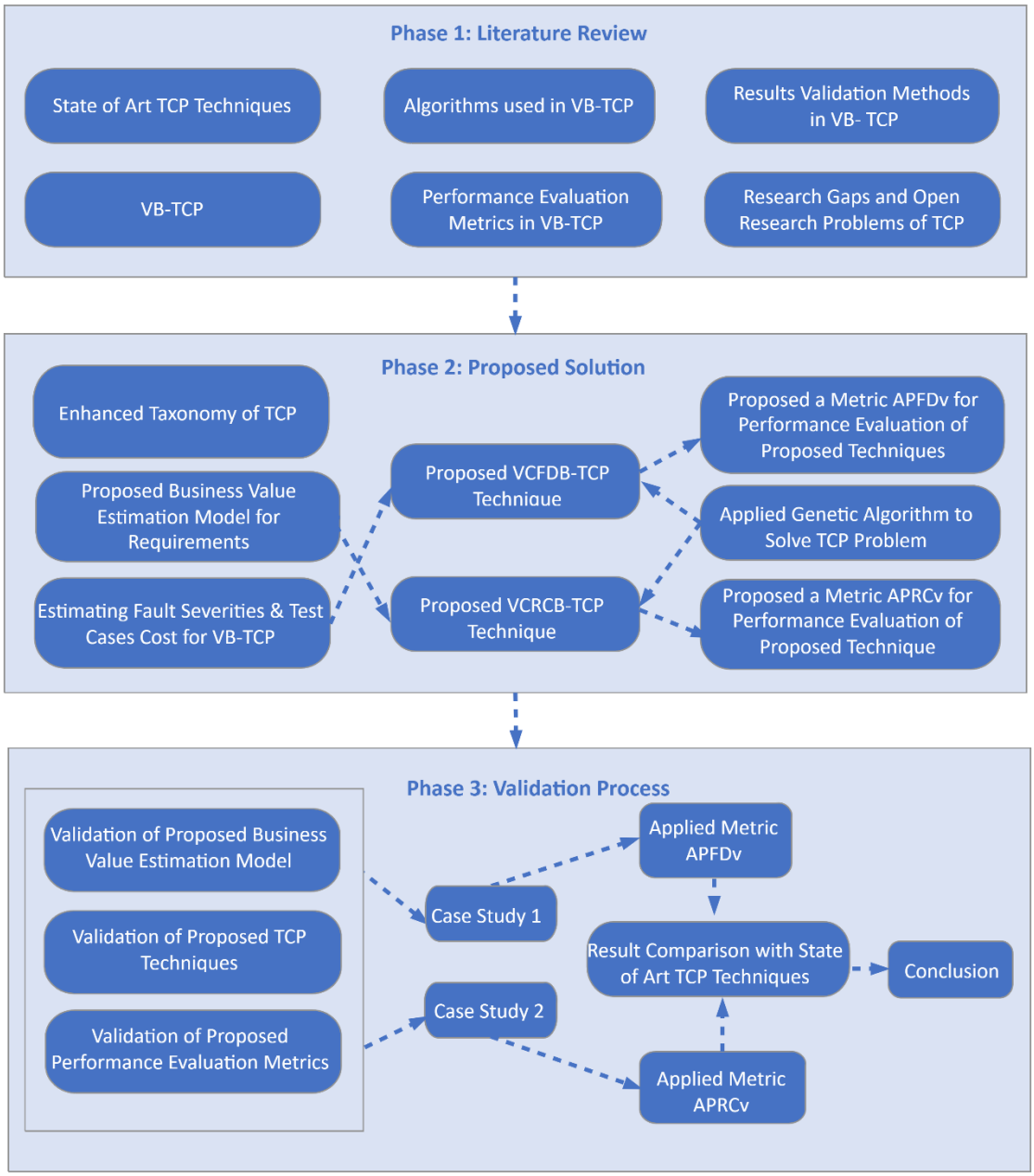


Figure 3.1: Overview of research methodology

3.3. An enhanced taxonomy of TCP techniques

This section presents the details about the categorization of TCP techniques given in the different studies. According to a study, there are seven categories of TCP techniques including search-based, coverage-based, requirements-based, fault-based, risk-based, history-based, and others [91]. The most widely used approach is Search-based, while other techniques used are coverage-based and fault-based. Another study presented a

categorization including probabilistic, cost-based, history-based, human-based, distribution-based, coverage-based, model-based, and others [33]. Table 3.1 shows the categorization of VB TCP techniques selected for this study.

Table 3.1: Classification of value-based cost-cognizant TCP techniques

Categories	**Paper ID**	**References**	**Year of Publication**
Search-based	P2, P11	[68], [17]	2017, 2012
Coverage-based	P5, P9, P12, P14, P15,	[101], [104], 65], [107], [71]	2015, 2013, 2012, 2011, 2010,
Requirement-based	P3, P17, P19, P20	[99], [108], [109], [110]	2016, 2007, 2005, 2005
Fault-based	P4, P6, P7, P10, P13, P18, P21	[100], [102], [103], [105], [106], [87], [111]	2015, 2015, 2015, 2013, 2011, 2006, 2001
History-based	P1, P8, P16	[98], [86], [88]	2019, 2012, 2008

The existing categorizations and taxonomies of TCP techniques have recognized “cost-based” as one category among other categories. But we believe that “cost-based” or “cost-cognition” is a value-based fashion. We must recognize cost-cognizant test prioritization to segregate it from value-neutral TCP techniques. To address this need we proposed two abstract classes of TCP techniques including value-neutral and value-based. For this, we have proposed an enhanced taxonomy of TCP techniques presented in Figure 3.2. Our enhanced taxonomy of TCP techniques is comprised of two major classes including VN test prioritization and VB test prioritization.

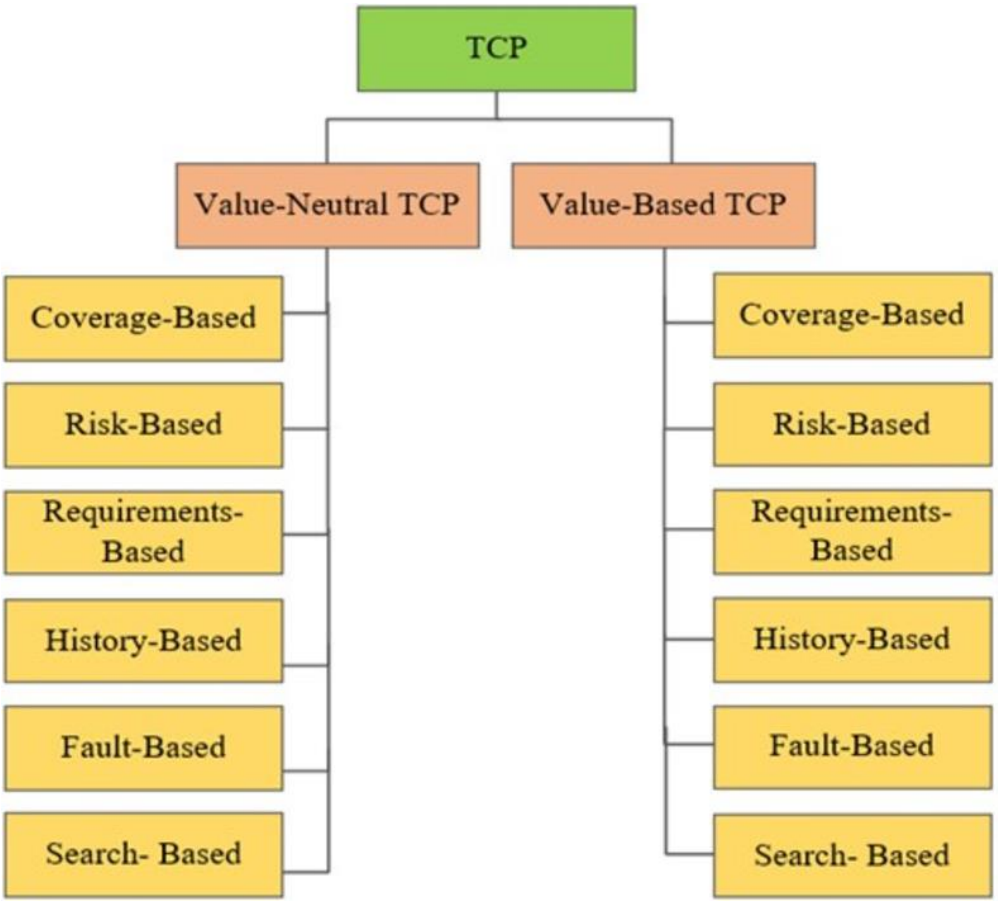


Figure 3.2: Enhanced Taxonomy of Test Case Prioritization

3.4. Proposed Business Value Estimation Model for TCP

In this section, a model is proposed to estimate fault severities and test case costs based on the business value of requirements for value-based TCP. The value-neutral TCP does not consider fault severity and test case cost while prioritizing test cases and assumes that all faults have the same severity, and all test cases have the same cost. Unlike this, the proposed model rejected this assumption and considers varying fault severities and test case costs in the TCP process. The following steps describe the proposed model.

- Estimating or quantifying the Business Value (BV) of software requirements (software features as functional requirements and software quality attributes as non-functional requirements).
- Deriving and estimating fault severities and test case costs through the business value of requirements.

- Prioritizing test cases considering faults severities and test case costs in Fault-Detection-Based TCP (FDB-TCP).
- Prioritizing test cases considering the business value of requirements and cost of test cases in Requirements-Coverage-Based TCP (RCB-TCP).

Figure 3.3 depicts an overview of the proposed model. Initially, we collected the feature set and quality attributes. The business value of software features is based on three factors including Feature Client Priority (FCP), Feature Complexity (FC), and Feature Usage (FU). The business value of quality attributes is based on Quality Attribute Client Priority (QACP). The value of FCP and QACP is determined with the help of the business analysis team by using five business success factors. The value of FC is determined with the help of the application development team and the value of FU is collected from the application database for each of the features in the feature set.

The business value of features is calculated through FCP, FC, and FU, and the business value of quality attributes is calculated by taking the direct value of QACP. Once the business value of requirements is calculated, the test cases and faults are mapped with these requirements. Then the severity of faults is derived from the business value of the features with which they are associated. Similarly, test case cost is derived from the business value of requirements with which they are associated. For instance, if the business value of a requirement is 4, then the business value of its associated test case will be 4. Similarly, the severity of its associated bug will be 4.

Two types of TCP techniques are presented in this model including (FDB-TCP), and (RCB-TCP). In (FDB-TCP), test cases are prioritized considering varying test case costs and fault severities. In (RCB-TCP), test cases are prioritized considering varying test case costs and requirements' business value.

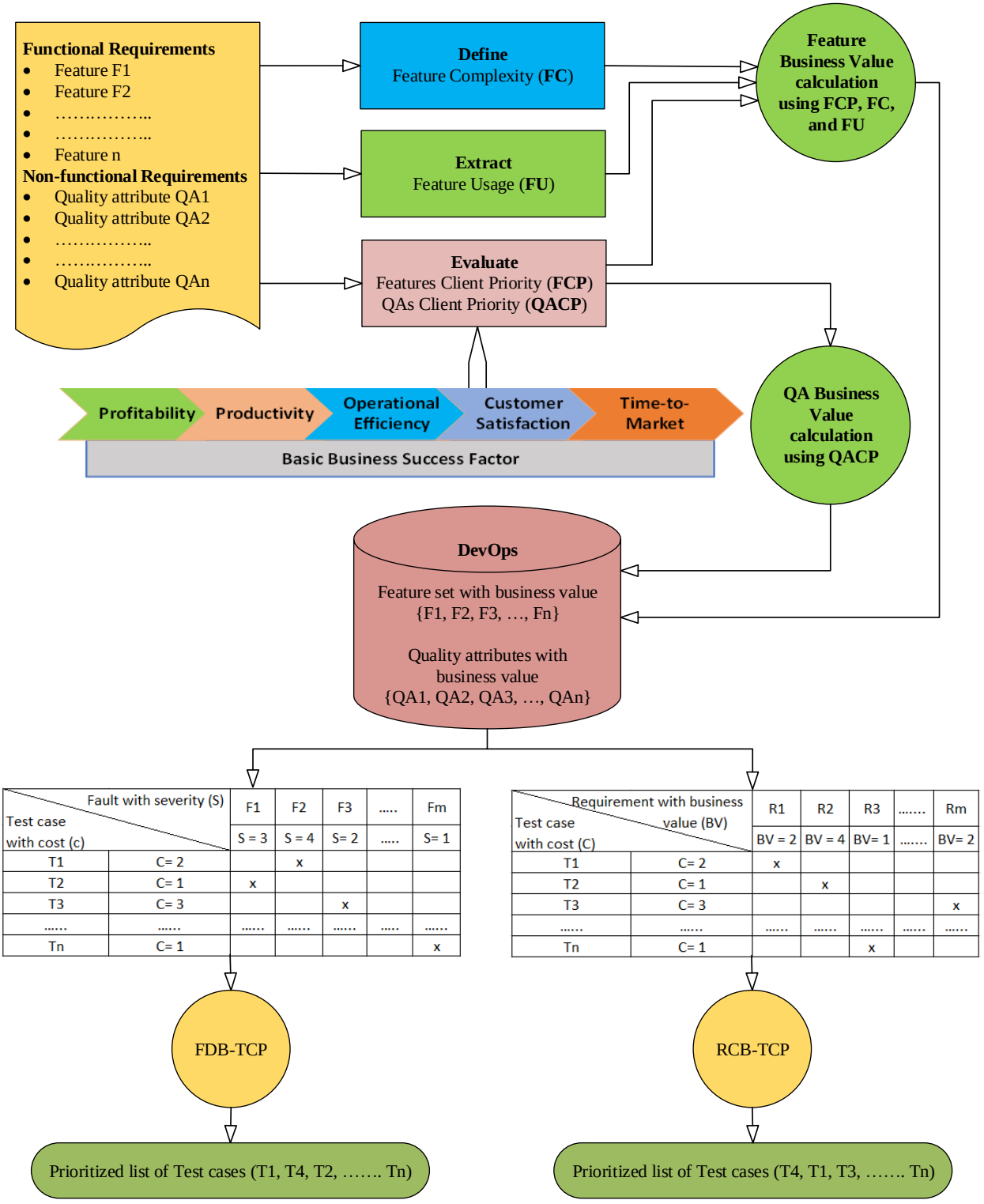


Figure 3.3: Proposed business value quantification model for value-based TCP

3.4.1. Business Value Factors

Knowing the business value of software requirements is imperative for various software

development activities. In this section, business value factors have been described. According to Barry Boehm, the critical success factors of a business lie in the value domains [116]. Traditionally the success of a business is related to the ROI, increased productivity, and saving time, and cost [117]. Here is a description of important business success factors found in the literature related to software quality.

3.4.1.1 Feature Client Priority (FCP)

The client's priority of any requirement is the most important factor while the prioritization of software activities [118]. To calculate client priority of different features and quality attributes, the proposed model uses five business success factors including profitability, productivity, operational efficiency, customer satisfaction, and time to market.

a) Profitability:

Putting more effort into improving the quality of software can lead to growth in revenue and increased profitability [119]. Companies get a return on investment only if their software applications satisfy customer needs and meet user expectations [120].

b) Operational Efficiency:

Operational efficiency is the level of performance for business tasks. There are multiple factors involved in operational efficiencies like cost, time, and people involved [121]. The users of the system expect a direct or instant response from the system when they perform a business operation on the system [122]. If an application takes too much time to complete the operation of the user, then it will be operationally inefficient. For instance, a healthcare coordinator has a target to make calls to fifty patients and schedule visits with their primary care physicians in an electronic health record (EHR) system. If the system is not efficient enough to execute the call schedule feature, then the care coordinator will miss the target. He/she will require more time to complete the calls and to be paid extra against extra time spending. Operational efficiency directly affects cost and time to perform the tasks [121].

c) Productivity

The usage of IT solutions is supposed to enhance the productivity of resources in a business firm. The quality of software solutions directly influences the productivity of its

users.

d) Customer Satisfaction

Software quality has a direct impact on customer satisfaction which directly affects the financials of any organization including its profit, and sales growth [123]. For client satisfaction, software quality plays a vital role. The factors that have a greater influence on client satisfaction include quality of execution, implementation, and relationship in a software-as-a-service (SaaS) business model [124].

e) Time to Market:

Time to market is also a key business driver [120]. This factor also has a great influence on business. But it is a little conflicting with other quality attributes. To achieve a good time to market, other quality attributes may suffer. Time to market is also a very important success factor. For instance, launching a new product, or service earlier can gain a bigger market share. Time to market for such initiatives is dependent on the readiness of IT support. Developing a product or service and developing software features for it goes hand in hand.

We selected these factors through a comprehensive literature review. The business team is involved in measuring how the success of the business is dependent on each feature and quality attribute. The business analysis team involves the client and assigns the priority value by utilizing the Delphi technique in two steps. First, a numeric value ranging from 1 to 5 is assigned to each success factor against all features and quality attributes defined in the requirements list. Then the average value of success factors is calculated for them. The average value of success factors against a feature is considered its client priority. Similarly, the average value of success factors against a quality attribute is considered its client priority.

Suppose ***F*** is a software feature, and ***S*** is a set of success factors including ***s1, s2, s3, …, sn***. If ***SVAL*** is the value of success factor ranges from 1 to 5 and ***FCP*** is the feature client priority, then it can be calculated as shown in equation 9.

$$\mathrm{FCP} = \sum_{i=1}^{i=n}(\mathrm{SVAL_i})/\mathrm{n} \qquad (9)$$

Similarly, suppose ***QA*** is a quality attribute, and ***S*** is a set of success factors including ***s1, s2, s3, …, sm***. If ***SVAL*** is the value of success factor ranges from 1 to 5 and ***QACP*** is the quality attribute client priority, then it can be calculated as shown in equation 10.

$$\mathrm{QACP} = \sum_{k=1}^{k=m}(\mathrm{SVAL_k})/\mathrm{m} \qquad (10)$$

3.4.1.2. Feature Complexity (FC)

The software features with higher complexity are prone to more errors and 20% of the features result in 80% of faults [110]. Some features are developed in such a way that they have an impact on some other features. The feature that has more impact on other features has greater complexity. Most of the software faults detected late in the testing life cycle are due to poor impact analysis. The comprehensive and complete impact analysis is vital for software feature development during the requirements analysis phase. It is the main responsibility of the business analysis team to compute feature complexities while finalizing requirements. In our proposed method we quantify complexity as the percentage ratio value. If we have a feature set ***F*** having ***n*** features, then suppose ***f*** is a feature of the feature set and ***i*** is the number of features for which ***f*** has an impact. The complexity of ***f*** is denoted by ***fc,*** and it can be computed by equation 11.

$$fc = \frac{i}{n} * 5 \qquad (11)$$

According to Equation 11, the minimum value of *fc* can be greater than 0 and the maximum can be less than or equal to 5. The value ***i*** cannot be 0 because the feature being developed has its own impact too. If a feature has an impact on one other feature, then the value of ***i*** will be considered as 2 because the feature itself is also included in the impact. Similarly, if a feature does not have an impact on any other feature, then the value of ***i*** will be considered as 1 because it has its own impact. In this case, if the feature itself is not considered then the value of ***i*** will be 0, and consequently, equation 11 will return the value of ***fc*** as 0. To address this scenario, we considered the feature itself in the value of ***i*** because the complexity of a feature cannot be 0.

3.4.1.3. Feature Usage (FU)

The different software features have different usage frequencies. According to a study, 45% of features are never used, 19% of features are rarely used and 36% of the software features are used regularly or most often [110]. This fact supports that the features most often used possess greater business value. Therefore, the usage factor is also vital in

estimating the business value of a software feature. The usage of a different features can be detected from the application database logs. Usually, the usage history of features is maintained to analyze the adaptability of the systems. If we have a feature set ***F*** and the total usage frequency of ***F*** is ***h*** then suppose ***f*** is a feature of the feature set and ***u*** is its usage frequency. The usage value of the feature is denoted by ***fu,*** and it can be computed by the following equation 12.

$$fu = \frac{u}{h} * 5 \qquad (12)$$

According to Equation 12, the value range of *fu* can be greater than 0 and less than or equal to 5. The value ***u*** cannot be 0 because every feature is used at least once. Sometimes features are used for one time only then they become obsolete like time-specific reports. The feature usage value can be 5 if and only if there is only a single feature in the application. In this case, the value of ***u*** and ***h*** will be equal and equation 12 will return the maximum value of ***fu*** which is 5.

3.4.2. Business Value Computation

The feature business value (FBV) is represented by the formula given in Equation 13. In this equation, *FCP* is feature client priority, *FC* is feature complexity, and *FU* represents feature usage.

$$\text{FBV} = (\frac{\text{FCP}}{10} * \alpha + \frac{\text{FC}}{10} * \beta + \frac{\text{FU}}{10} * \gamma)/10 \qquad (13)$$

The above formula is designed in such a way that the maximum business value of a software feature can be up to 5 and the minimum value can be greater than 0. We need values of α, β, and γ to operationalize equation 13. We provide them as follows. We use the weightage of the client priority factor α = 0.40, feature complexity $\beta = 0.30$, and feature usage $\gamma = 0.30$ respectively. It is pertinent to highlight that individual organizations may adjust them according to their local requirements. Table 3.2 shows six sample features and their business values calculated by using the equation (9), (11), (12), and (13).

Table 3.2: Features list with their business value

Feature	Business Success Factors					Feature Client Priority (FCP)	Feature Impact		Feature Complexity (FC)	Usage Frequencies		Feature Usage (FU)	Feature Business value (FBV)
	Profitability	Productivity	Operational Efficiency	Customer Satisfaction	Time-to-Market		Total Features	Impact Value		Total Frequency	Feature Frequency		
F1	4	3	4	3	3	3.40	5	1	1.00	76	4	0.26	1.74
F2	3	2	4	3	3	3.00	5	3	3.00	76	20	1.32	2.49
F3	3	4	2	3	4	3.20	5	3	3.00	76	21	1.38	2.59
F4	5	5	5	5	4	4.80	5	4	4.00	76	2	0.13	3.16
F5	1	1	1	1	1	1.00	5	2	2.00	76	1	0.07	1.02
F6	5	5	5	5	5	5.00	5	5	5.00	76	28	1.84	4.05

The Quality Attribute Business Value (QABV) is the direct value taken from the quality attribute client priority. Therefore, QABV can be represented by the same formula used for QACP and is shown in equation 10.

The business value of a quality attribute ranges from 1 to 5 and it is the direct value of its client priority. Table 3.3 shows six sample quality attributes and their business values calculated by using Equation 10.

Table 3.3: Quality attributes list with business value

Quality Attribute (QA)	Business Success Factors					QA Client Priority (QACP)	QA Business value (QABV)
	Profitability	Productivity	Operational Efficiency	Customer Satisfaction	Time-to-Market		
QA1	2	3	2	3	3	2.60	2.60
QA2	3	4	3	2	3	3.00	3.00
QA3	3	4	3	2	2	2.80	2.80
QA4	3	2	2	3	2	2.40	2.40
QA5	1	1	1	1	1	1.00	1.00
QA6	5	5	5	5	5	5.00	5.00

3.4.3. Business Value Management

The business value of software features and quality attributes can be managed in any requirement management tool like the Microsoft Team Foundation Server (TFS) tool. Microsoft TFS is a powerful tool used in the software industry. Software professionals use it for the management of software requirements, code, test cases, bugs, and other related artifacts. It provides full support to the software development life cycle and software testing life cycle. Since it is a flexible tool, therefore, it can be utilized to manage the business value of individual software features and quality attributes. Figure 3.4 shows its management. The "Business Value" attribute of a specific feature is highlighted in the rectangular box.

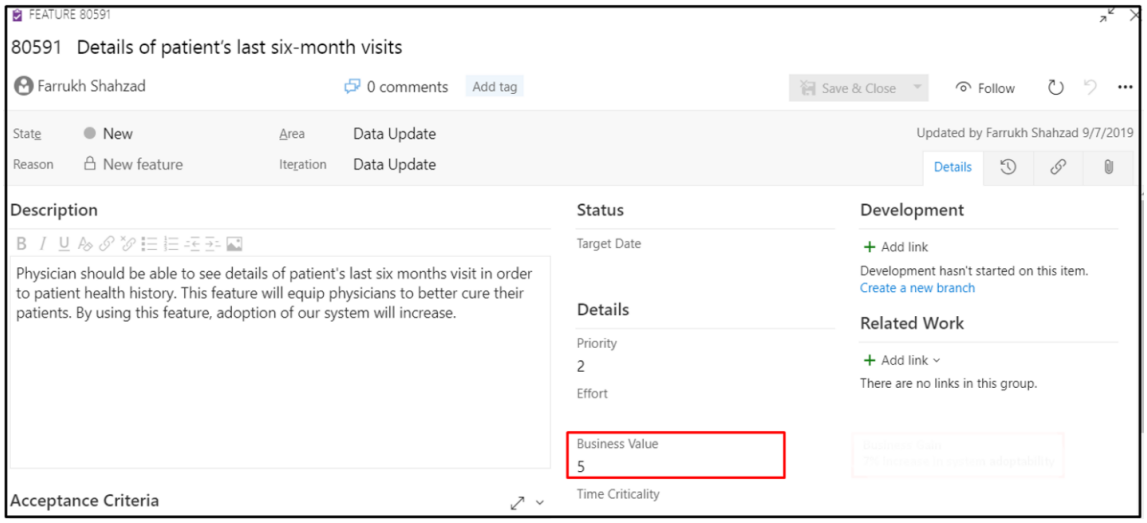


Figure 3.4: Business value management in TFS

3.4.4. Estimating Fault Severities

There are different techniques available in the literature to automatically assign a severity value to a newly filed bug/fault in the bug reporting system [125], [126], [127]. Assigning severity to a fault is a very subjective process and different testers are likely to assign different severity to the same bug [128]. This makes bug reports unreliable while prioritizing the bugs for their resolution. To deal with this situation, the following factors are recommended to estimate fault severities.

i) The business value of the associated feature

ii) Tester's expertise

The severity of a fault is closely dependent on the business value of the software feature with which it is associated. Therefore, it is proposed here that the software tester should define the severity of a fault based on the business value of the feature to which it belongs. There are some cases where faults have a minor impact like graphical user interface faults or some other non-functional faults like spell mistakes. In that case, the tester should use his/her expertise and common sense to define the severity of faults. Fault severity should be defined by using feature business value in conjunction with the tester's expertise following common guidelines. It is proposed here that units of severity should be mapped with units of business value. Suppose there is a feature *"Generate an automatic email for customer on successful order submission"*. If an email is not generated or this feature is not working as intended, then the tester should file a fault in the bug reporting system by defining its severity equivalent to the business value units of its associated feature. For instance, if a feature is not achieving its objectives or it is not working as intended and its business value is 6 then the tester should file a bug of severity 6. In case, the feature is working and there are minor issues in it then the tester should use his/her expertise or general guidelines to define the bug severity. The maximum severity of a fault can be 10 and the minimum can be 1.

3.4.5. Estimating Test Cases Cost

For cost-cognizant TCP, the cost of test cases can be estimated from their previous executions of the test suite [19]. Execution time is usually reported in test script outputs and this historical data can be referred to estimate the regression test case costs [129]. It is proposed here that the cost of a test case can be estimated from the business value of the features it covers. For instance, if a test case ***t*** covers requirements ***r1***, and ***r2*** having business values 5 and 7 respectively then the test case cost will be 12 twelve units of time. It is assumed that a test case requires 1 unit of time to cover 1 unit of the requirement's business value. Similarly, if a test case detects two faults ***f1*** and ***f2*** having severity values of 5, and 8 respectively then the test case cost will be 13 units of time. If ***t*** is a test case in test suite ***T*** and ***n*** is the number of requirements it covers and ***bv*** is the business value of a requirement ***i*** then the test case's cost ***tc*** can be calculated by equation 14.

$$t_c = \sum_{i=1}^{n} bv_i \qquad (14)$$

Similarly, if ***t*** is a test case in test suite ***T*** and ***n*** is the number of faults it detects, and ***s*** is the severity of a fault ***m*** then the test case's cost ***tc*** can be calculated by using equation 15.

$$t_c = \sum_{m=1}^{n} s_i \qquad (15)$$

3.5. Incorporating Value in TCP Process

In the value-neutral software engineering fashion, all software artifacts have the same worth. Similarly, in value-neutral TCP, all bugs have the same cost, and all test cases have equal costs. In requirements-based TCP all requirements are treated as equally important. But these assumptions rarely hold in practice. Unlike value-neutral fashion, in value-based fashion, different software artifacts have different worth like every software feature, and software quality attributes can have a different business value. The proposed method supports the value-based TCP process for regression testing.

3.5.1. Incorporating Value in Fault-Based TCP Process

In the value context, the test cases having greater business value will get higher priority and will come earlier in the prioritization order. The business value of a test case is defined as the ratio of total fault severity detected by the test case and the cost of that test case. In value-based TCP, Test Case Business Value (TCBV) serves as prioritization criteria or adequacy criteria. Suppose T is a test suite having test cases t1, t2, t3, …, tn with already calculated cost C1, C2, C3, …, Cn respectively, and F is the fault set having faults f1, f2, f3, … fm. Each fault has already defined severity values like S1, S2, S3, …, Sm respectively. Let TC-F be the test case vs fault detection traceability matrix. Let K be the list of faults detected by a specific test case that is not already detected by any other test case. If SK is the severity of any fault in K and C is the cost of a test case, then the Test Case Business Value (TCBV) can be calculated by using Equation 16.

$$\mathrm{TCBV} = \frac{\sum_{i=1}^{n} SK_i}{C} \qquad (16)$$

Suppose a test case T has cost 2 and it detects two faults F1 and F2 having severities 3, and 1 respectively, then the business value of T can be defined by putting the values in equation (16) and the resultant value of TCBV will be 2.

In the business value calculation of a test case, the severity of only those faults will be considered that are not already detected by any other test case. In Table 3.4, the fault detection scores of test cases T1, T2, and T3 are equal because each test case detects 2 faults. Therefore, all test cases have the same business value.

Table 3.4: Fault severity detection of test cases

Faults & severity / Test Cases & cost	F1	F2	F3
	Severity = 4	Severity = 2	Severity = 5
T1 (cost = 2)	√	√	
T2 (cost = 2)		√	√
T3 (cost = 3)	√		√

But when the business value of test cases is considered in the prioritization process, each test case will get a different priority. The business value of test cases presented in 8 is computed and depicted in Table 3.5. The sum of severities of faults detected by test case T1 is 6 because it detects faults F1 and F2 having severities of 4 and 2 respectively, and the cost of T1 is 2, therefore the business value of T1 is 3. The sum of severities of faults detected by test case T2 is 5 because it detects faults F2 and F3 having severities 2 and 5 respectively. In this case, the severity of fault F2 is not added to the sum because it is already detected by test case T1. The cost of test case T2 is 2 so the business value of T2 is calculated as 2.5. The sum of the severities of faults detected by test case T3 is 0 because the faults detected by T3 are already detected by T1 and T2. The cost of T3 is 3 and its business value is computed as 0.

Table 3.5: Test cases with business value

Test Case	$\frac{SK_1 + SK_2 + SK_3 + \cdots + SK_n}{C}$	TCBV
T1	$\frac{4+2}{2}$	3
T2	$\frac{5}{2}$	2.5
T3	$\frac{0}{3}$	0

Now the prioritized order of given test cases will be (T1, T2, and T3) because T1 has a higher business value than T2 and T3, and similarly, T2 has a higher business value than T3. In value-neutral TCP, all three cases have the same priority but when the value is considered, all three test cases get different priorities. Hence the business value makes the difference.

3.5.2. Incorporating Value in Requirements Coverage-Based TCP)

In the proposed requirements coverage-based TCP, the test cases are prioritized based on their business value. The test cases having greater business value will get higher priority and will come earlier in the prioritization order. The business value of a test case is defined as the ratio of the total business value of requirements covered by the test case and the cost of that test case.

In value-based TCP, Test Case Business Value (TCBV) serves as prioritization criteria or adequacy criteria. Suppose T is a test suite having test cases t1, t2, t3, …, tn with already calculated cost C1, C2, C3, …, Cn respectively, and R is the requirements set having requirements r1, r2, r3, … rm. Each requirement has already calculated business values V1, V2, V3, …, Vm respectively. Let TC-R be the test case vs requirement coverage traceability matrix. Let K be the list of requirements/features covered by a specific test case that is not already covered by any other test case. If BVK is the business value of any requirement in K and C is the cost of a test case, then TCBV can be calculated by using equation 17.

$$\mathrm{TCBV} = \frac{\sum_{i=1}^{n} \mathrm{BVK_i}}{C} \qquad (17)$$

Suppose a test case T has cost 2 and it covers two requirements R1 and R2 having business values of 5, and 4.49 respectively, then the business value of T can be calculated by putting the values in equation 17. The resultant value of TCBV is 4.745.

In the business value calculation of a test case, the business value of only those requirements will be considered that are not already covered by any other test case. In Table 3.6, the business value coverage of test cases T1, T2, and T3 are equal because each test case covers 2 requirements. Therefore, all test cases have the same priority.

Table 3.6: Business value coverage of test cases

Requirements & business value / Test cases & cost	R1	R2	R3
	Business Value = 5.00	Business Value = 4.49	Business = 4.61
T1 (cost = 2)	√	√	
T2 (cost = 1)		√	√
T3 (cost = 3)	√		√

When the business value of test cases is considered in the prioritization process, each test case will get a different priority. The business value of test cases presented in 10 is computed and depicted in Table 3.7. The sum of business values of requirements covered by test case T1 is 9.49 because it covers requirements R1 and R2 having business values of 5 and 4.49 respectively, and the cost of T1 is 2, therefore the business value of T1 is 4.745. The sum of business values of requirements covered by test case T2 is 4.61 because it covers faults R2 and R3 having business values of 4.49 and 4.61 respectively, but the business value of requirement R2 is not added to the sum because it is already covered by test case T1. The cost of test case T2 is 1 so the business value of T2 is calculated as 4.61. The sum of business values of requirements covered by test case T3 is 0 because the requirements covered by T3 are already covered by T1 and T2. The cost of T3 is 3 and its business value is computed as 0.

Table 3.7: Test cases with business value

Test Case	$\frac{BVK_1 + BVK_2 + BVK_3 + \cdots + BVK_n}{C}$	TCBV
T1	$\frac{5 + 4.49}{2}$	4.745
T2	$\frac{4.61}{1}$	4.61
T3	$\frac{0}{3}$	0

Hence the best order of test cases will be T1, T2, and T3. In value-neutral TCP, test cases T1, T2, and T3 have equal priority but in value-based TCP, all three test cases have different priorities. Hence business value makes the difference.

3.5.3. Proposed Value-Based TCP and Evaluation Metrics

The previous section outlines how business value can be incorporated into the TCP process. In this section, two value-cognizant TCP approaches are proposed including Value-Cognizant Faults Detection-Based Test Case Prioritization (VCFDB-TCP) and Value-Cognizant Requirements Coverage-Based Test Case Prioritization (VCRCB-TCP). For performance evaluation of the proposed techniques, two value-cognizant metrics are proposed in this section including the $APFD_v$, and the $APRC_v$. In a value-based context, it is imperative to know that different test cases can have different execution times. Similarly, each requirement, statement, function, loop, or condition may have a different value from another requirement, statement, function, loop, or condition, respectively. The proposed TCP techniques are presented in this value-based context. The proposed techniques and performance metrics are published in [135].

As per Pareto Analysis, it is widely believed that 80 percent of faults lie in 20 percent of the modules or code [130]. Pareto Analysis is a technique to find out the parts of the software that contain more bugs or possess more business value. This analysis indicates that fewer factors are responsible for most of the bugs and generate most of the business value. A TCP technique should target most value-covering test cases first. The idea behind this is to cover those parts of the application that possess greater business value to achieve the set performance goal. Therefore, we are focusing on those test cases first which cover most of the business value. We are not applying Pareto distribution, but our approach is similar to it in the context of the outcome.

3.5.3.1. Value-Cognizant Fault Detection-Based TCP (VCFDB-TCP)

a) Description

In this technique, test cases are prioritized based on their value of Severity Detection Score (SDS). The measure "SDS" for a test case is the fault severity detection per execution time of a test case. The proposed VCFDB-TCP is depicted in Figure 3.5.

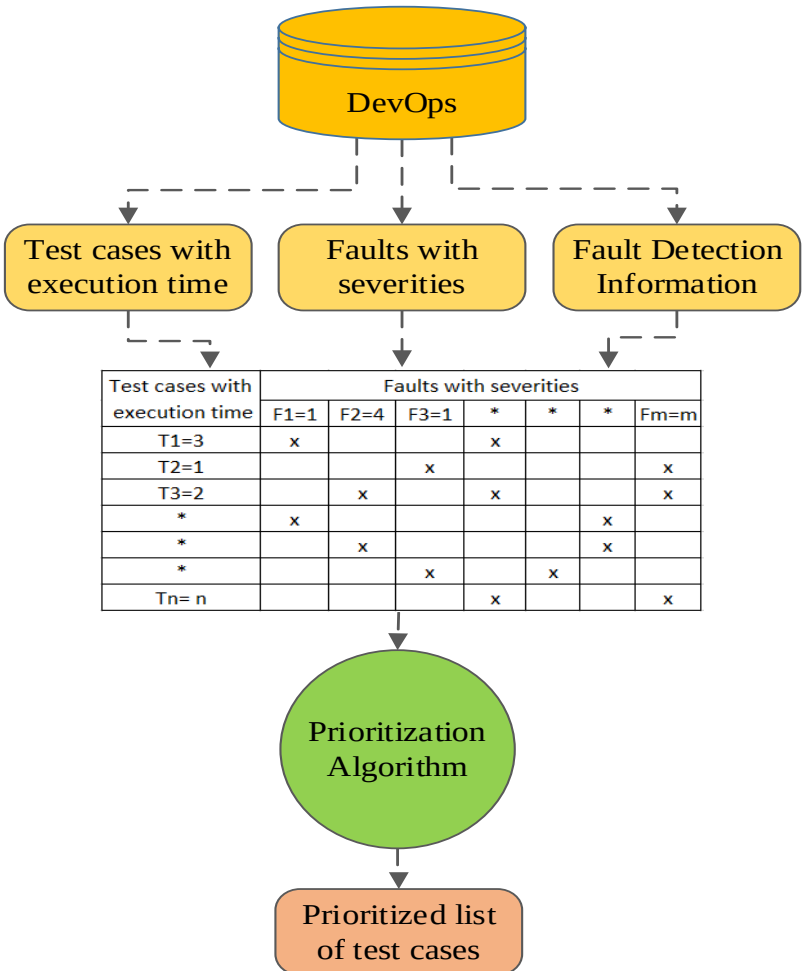


Figure 3.5: Overview of value-based TCP

The formula for the severity detection score can be represented by equation 18.

$$\text{Severity detection score} = \frac{\text{Sum of fault severity detected}}{\text{Test case execution time}} \qquad (18)$$

For a test case, the total fault severity detection is the sum of the severity of faults detected that are not already detected by any other test case. In equation 18, if the sum of fault severities detected is S, and the test case execution time or cost is C then for the test cases presented in Table 3.8, the measure SDS is presented in Table 3.9.

Table 3.8: Test cases with cost VS faults with severity

Test cases with cost	Faults with severities			
	F1= 2	F2= 1	F3= 4	F4= 3
A= 3	x		x	
B= 1		X		
C= 2	x			x

Table 3.9: Test cases fault severity detection score

Test case	S/C	SDS
A	6/3	2
B	1/1	1
C	3/2	1.5

Test case A detects faults F1 and F3 having severities 2 and 4 respectively and the execution time of A is 3. Dividing total severity 6 by cost 3 results in 2. Test case B detects fault F2 having severity 1 and execution time of B is 1. Dividing severity 1 by cost 1 results in 1. Test case C detects faults F1 and F4 but fault F1 is already detected by test case A, so we consider only the severity of fault F4. The severity of fault F4 is 3 and the cost of test case C is 2. Dividing severity 3 by cost 2 results in 1.5. Now test cases can be prioritized by the greatest to least severity detection score value. According to severity detection score criteria A, C, B is the best order of execution. Table 3.10 shows the performance results of different orders of test cases in the test case suite evaluated in terms of $APFD_v$. The value of $APFD_v$ is calculated by using the formula given in Equation 19.

Table 3.10: Performance in terms of $APFD_v$

Test case order	**$APFD_v$**
B, A, C	92.00%
B, C, A	91.00%
C, B, A	92.00%
C, A, B	94.00%
A, B, C	94.00%
A, C, B	95.00%

b) Average Percentage of Fault Detection per Value ($APFD_v$)

In this thesis, a new metric $APFD_v$ is proposed. This metric provides the measure of the average percentage detection of fault severity ratio test case execution time for a given order of test cases in a test suite. For test case execution the term test case cost is used. The metric formula is given in Equation 19.

$$APFD_v = 1 - \frac{\sum_{i=1}^{n} (TF_i \times \frac{S_i}{C_i})}{\sum_{i=1}^{n}(C_i) \times \sum_{j=1}^{m}(S_i)} + \frac{1}{2 \times \sum_{i=1}^{n} c_i} \qquad (19)$$

In this equation, TF_i is the order of the first test case that detects fault F_i, S_i is the severity of fault F_i, and C_i is the cost of the test case. The metric $APFD_v$ accommodates varying test case costs and fault severity. It is also applicable when test case cost and fault severities are the same.

The metric $APFD_v$ is supposed to evaluate the performance of different prioritization orders. It is not supposed to estimate fault severities and test case costs. It works when fault severities and test case costs are already known. If severity and cost values are not known, it still works by considering that each test case has a cost of 1 and each fault has a severity of 1. It deals with cost and severity as equal units.

This formula for $APFD_v$ is derived from the classical formula of APFD proposed by Sebastian Elbaum et al. in 2000 [112]. The mathematical base of $APFD_v$ is the same as that of APFD. The metric APFD uses bug count, test cases count, and order of test cases as parameters. Whereas $APFD_v$ uses two additional parameters including test case cost and fault severity. The minimum value of APFD can be 0 and the maximum can be 1. The metric $APFD_v$ is established on the same mathematical base. Its minimum value can be 0 and the maximum can be 1. To prove its generalization, we performed more than a hundred trials. APFD is based on the frequency of faults and treats all faults at an equal level whereas $APFD_v$ is based on severity detection per cost consumption considering critical faults more important than other faults.

To understand the working of the proposed metric, consider the example given in Table 3.11 containing test cases with cost vs faults with a severity which is depicted in Figure 3.6. We calculate the $APFD_v$ value for the orders T5, T2, T1, T4, and T3.

Table 3.11: Test cases with cost and fault with severity

Test cases with cost	Faults with severity			
	F1= 2	F2= 4	F3= 1	F4= 3
T1= 4	X			
T2= 1			x	
T3= 2			x	
T4= 1		x		
T5= 3	X			x

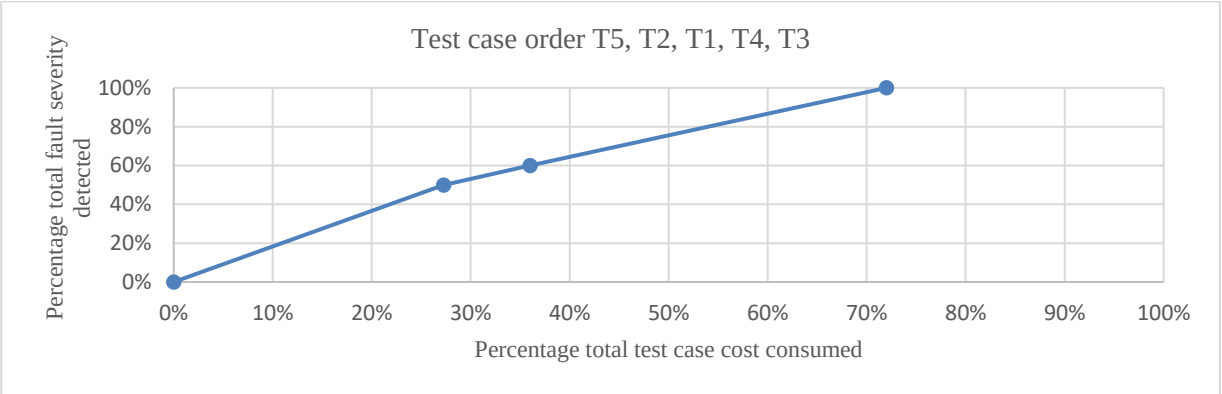


Figure 3.6: Cost vs Severity

$$APFD_v = 1 - \frac{3 \times \frac{2}{4} + 4 \times \frac{4}{1} + 2 \times \frac{1}{1} + 1 \times \frac{3}{3}}{(4+1+2+1+3) \times (2+4+1+3)} + \frac{1}{2 \times (4+1+2+1+3)}$$

$APFD_v = 0.86$

In percentage, the resultant value of $APFD_v$ is 86.00%. The minimum value of $APFD_v$ can be 0 and the maximum can be 1.

Now we consider four different cases to understand the working of the proposed metric. Table 3.12 shows case A where all test cases have the identical cost and all faults have same severity. Table 3.13 shows case B where all test cases have the identical cost, but fault severity varies. In this case, fault F2 has severity 2, and all other faults have severity 1. Table 3.14 presents case C where all faults have the same severity, but test case cost varies. In this case, test case T2 cost 2, and all other test cases cost 1. Table 3.15 presents

case D, where both test cases' cost and fault severities vary. In this case, test case T3 cost 3 and all other test cases cost 1. Similarly, fault F1 has a severity of 2, and all other faults have a severity of 1.

Case A

Table 3.12: Same severity and same cost

Test cases with cost	Faults with severities		
	F1= 1	F2=1	F3=1
T1=1		X	
T2=1	x		
T3=1			X

For order (T1, T2, T3) $APFD_v$= 0.50

For order (T2, T1, T3) $APFD_v$= 0.50

Case B

Table 3.13: Varying severity and same cost

Test cases with cost	Faults with severities		
	F1= 2	F2=1	F3=1
T1=1		x	
T2=1	x		
T3=1			X

For order (T1, T2, T3) $APFD_v$= 0.50

For order (T2, T1, T3) $APFD_v$= 0.59

In Case A, the value of $APFD_v$ is equal for both orders T1, T2, T3, and T2, T1, T3. The exchange of T1 and T2 orders does not affect the result because they both detect 1 fault each. In Case B, the value of $APFD_v$ for the orders T2, T1, and T3 is higher than that of the orders T1, T2, and T3 because the detection per cost of faults severities of T2 is higher than T1. The $APFD_v$ metric distinguished the order T2, T1, and T3 from T1, T2, and T3 and demonstrated better results for it.

Case C

Table 3.14: Varying costs and the same severity

Test cases with cost	Faults with severities		
	F1= 1	F2=1	F3=1
T1=1		X	
T2=2	x		
T3=1			X

For order (T1, T2, T3) $APFD_v$= 0.71

For order (T2, T1, T3) $APFD_v$= 0.67

Case D

Table 3.15: varying severity and varying cost

Test cases with cost	Faults with severities		
	F1= 2	F2=1	F3=1
T1=1		x	
T2=1	x		
T3=3			X

For order (T1, T2, T3) $APFD_v$= 0.80

For order (T3, T2, T1) $APFD_v$= 0.73

In case C presented in Table 3.14, the value of $APFD_v$ for order T1, T2, T3 is higher than that of order T2, T1, T3 because the detection per cost of faults severities of T1 is higher than T2. The $APFD_v$ metric distinguished the order T1, T2, T3 from T2, T1, T3 and demonstrated better results for it. In case D presented in Table 3.15, the value of $APFD_v$ for order T1, T2, T3 is higher than that of order T3, T2, and T1 because the detection per cost of faults severities of T1 is higher than T3. The $APFD_v$ metric distinguished the order T1, T2, T3 from T3, T2, T1 and demonstrated better results for it.

c) Implementation Using GA

Let us start with a few definitions in the context of value-based TCP using GA. GA is a subclass of evolutionary computation algorithms and is a form of metaheuristic search based on Darwin's theory of guiding search by considering "survival of fittest" [47]. In evolutionary search-based problems like TCP, the application of GA is effective [131]. Other approaches like the greedy algorithm produce suboptimal results since they are based on the "next best" philosophy. The additional greedy, 2-optimal algorithm and hill climbing are three other algorithms.

Gene: A test case available in the test suite to be prioritized is a gene.

Chromosome: A single order or permutation of test cases is identified as a chromosome.

Population: A collection of all test cases in the test suite is identified as population.

Parents: Two permutations that are combined to create a new permutation.

Mating pool: A collection of parents that are utilized to create the next generation of permutations.

Fitness Function: A function that tells how much closer the permutation result is to the ideal result.

Mutation: A method to generate a new permutation by randomly swapping the position of two test cases.

The proposed GA works in the following steps:

1. Create the population
2. Calculate fitness
3. Crossover
4. Mutate

5. Repeat

Input: Test cases suite T= (t1, t2, t3, ..., tn)

Execution cost of test cases (c1, c2, c3, ..., cn)

The severity of faults (f1, f2, f3, ..., fm)

Test cases vs fault coverage matrix

The performance goal is maximizing $APFD_v$

Result: Permutation of test cases with maximum $APFD_v$

Genetic Algorithm

Genetic Algorithm

Input: Test cases suite ***T= (t1, t2, t3, ..., tn)***

Execution cost of test cases ***(c1, c2, c3, ..., cn)***

Severity of faults ***(f1, f2, f3, ..., fm)***

Test cases vs fault coverage matrix *coverage*

Performance goal is maximizing $\boldsymbol{APFD_v}$

Result: Permutation of test cases with maximum $\boldsymbol{APFD_v}$

1 Begin

/*Initialization*/

2 Read test cases vs fault coverage matrix from the sheet in ***coverage***

3 Calculate total severity of 'm' faults ***TotSev***

4 Calculate the total cost of 'n' test cases ***TotCost***

5 Record test case order in ***TestOrder***

6 Read termination criteria from the user in ***Iterations***

7 while (not Termination condition) do

8 begin

9 Generate n random permutations in the iteration

10 Evaluate fitness function against each permutation

11 Assign the best $APFD_v$ value to ***fitmax1*** and assign its permutation to ***parent A***

12 Assign the second-best $\mathbf{APFD_v}$ value to ***fitmax2*** and assign permutation to ***parent B***

13 Update $\boldsymbol{BestAPFD_v}$ and its permutation with ***fitmax1*** and ***parent A***

14 Perform order crossover on *permutation with* $\boldsymbol{BestAPFD_v}$

15 Perform swap mutation after crossover

16 end

17 Display permutation with $\boldsymbol{BestAPFD_v}$

18 end

Test case prioritization is a sequence-based problem where the order or position of a test

case in the test suite is important. The value of the fitness function calculation is dependent on the position of a test case in the test case sequence. The details of the GA parameter setting, its operators, and the fitness function used in the case studies reported in this thesis are as follows.

Input Parameters:

Set of test cases in test suite T (t1, t2, t3, … tn).

Cost (execution time) of test cases in test suite T (c1, c2, c3, … cn).

The severity of faults detected by test cases in test suite T (f1, f2, f3, …, fn).

Test cases vs fault coverage matrix.

Population:

The collection of possible permutations (solutions) of the given test suite in the search space is called population.

Encoding:

TCP is a sequence problem therefore order chromosomes are used instead of binary encoding.

Chromosome:

For a given test suite with n test cases, a chromosome is encoded as an array of test case elements.

Gene:

A test case element in a chromosome is called a gene.

Order:

The position of a test case (Gene) in the array of test cases (Chromosome) is called order.

Selection:

Selection is dependent on the fitness value calculation of individual test cases.

Fitness Calculation:

The fitness of a test case is calculated as the total fault severity detected by a test case that is not already detected by any other test case divided by the cost of that test case. It is represented by equation 20.

$$\text{fitness(t)} = \frac{\text{total fault severity detected}}{\text{total cost}} \qquad (20)$$

Fitness Function

The performance goal of the proposed technique is to maximize the value of the $APFD_v$. Therefore, $APFD_v$ is the fitness function for the proposed technique implemented through GA and is represented by equation 19.

Termination criteria:

The termination criteria are the number of iterations and are taken as user input while executing the genetic algorithm.

GA Workflow

Evaluation:

In each iteration *n* number of random solutions/permutations are generated and for each solution, the fitness function is evaluated. In each iteration, the permutation with the maximum fitness function value is assigned to parent A and the permutation with the second maximum fitness function value is assigned to parent B. An array is defined to store the permutation with the best $APFD_v$ value. At the end of each iteration, if the best permutation APFDv value is less than the $APFD_v$ value of maximum fitness function value then the permutation with maximum $APFD_v$ value is assigned to the best permutation with Best $APFD_v$.

Crossover

At the end of each iteration, two solutions are selected as parent A and parent B with maximum fitness function value and second maximum fitness function value, respectively. Parent A is selected as a suitable chromosome to create a new generation of chromosomes. An order crossover is performed to generate a new chromosome. An order crossover is performed at the middle position. The first half of genes are shifted to the second half and the second half of genes are shifted to the first half. We followed the order crossover of genes within a chromosome and this style was adopted by Antoniol et al [132].

Mutation:

The mutation operator is used to swap the position of two genes within a chromosome. This process is known as swap mutation. The order crossover and swap mutation process

are depicted in Figure 3.7.

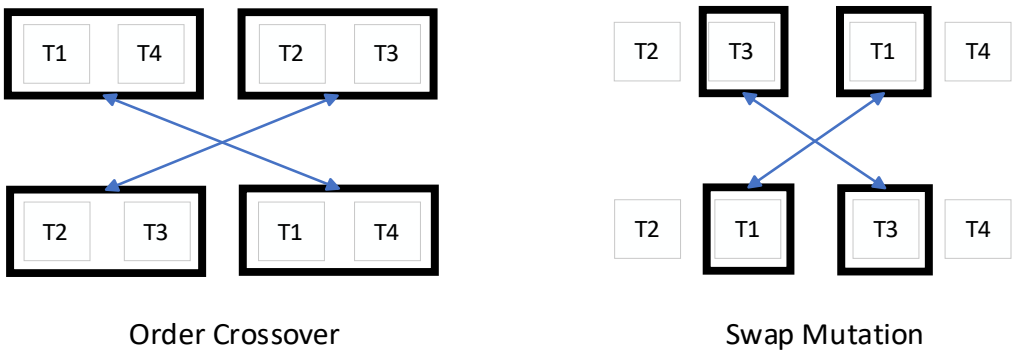


Figure 3.7: Crossover and mutation process

d) Flowchart of Value-Based TCP Implementation using GA

The flowchart of value-based TCP using a genetic algorithm is depicted in Figure 3.8.

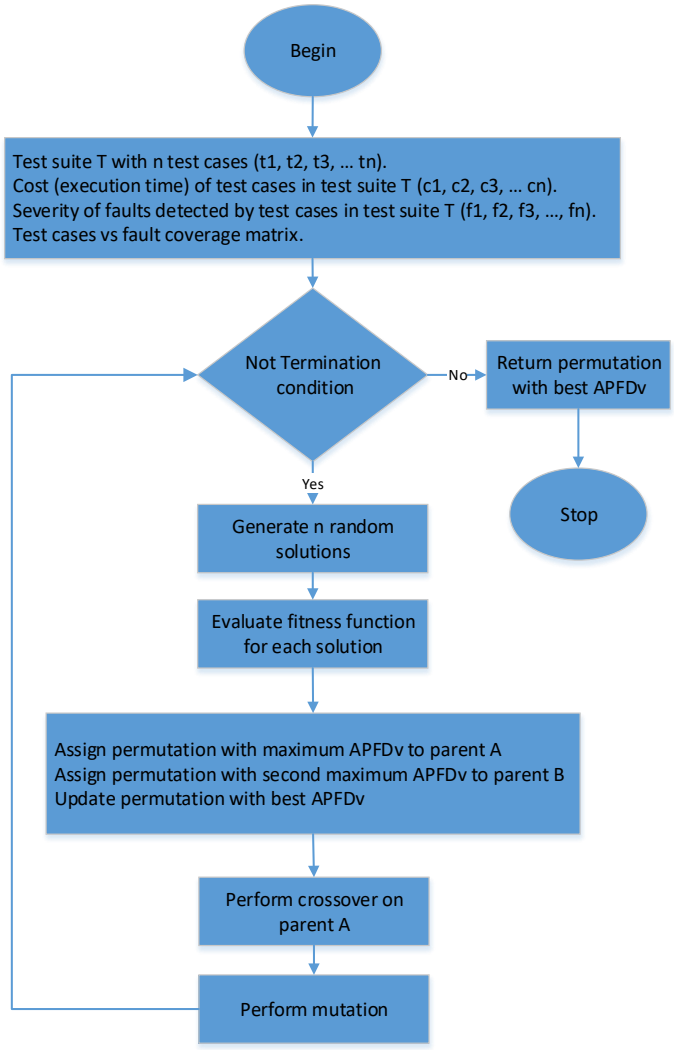


Figure 3.8: Flowchart of value-based TCP using GA

The time complexity of the proposed solution using GA depends upon the number of iterations performed and the computational cost of evaluating the fitness function for each permutation in a population.

3.5.3.2. Value-Cognizant Requirements Coverage-Based TCP (VCRCB-TCP)

a) Description

In this technique, test cases are prioritized based on their business value coverage. The measure of "Business value coverage" for a test case is the ratio of requirements for business value coverage per execution time of a test case. The value-cognizant requirements coverage-based TCP is depicted in Figure 3.9.

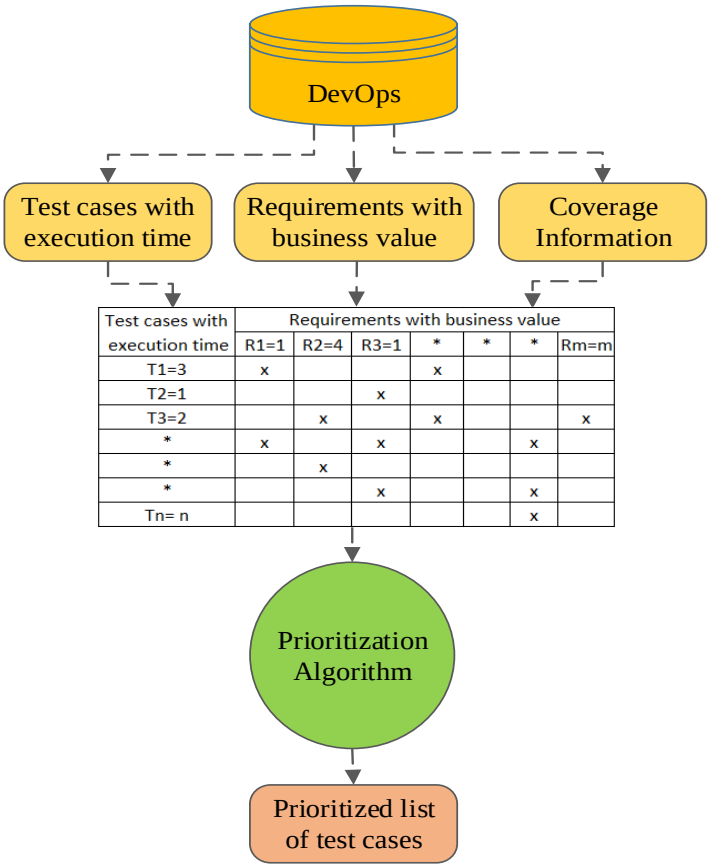


Figure 3.9: Value-cognizant requirements coverage-based TCP

The formula for business value coverage can be represented by equation 21.

$$\text{Business value coverage} = \frac{\text{Business value covered}}{\text{Test case execution time}} \quad (21)$$

The requirements total business value coverage of a test case is the sum of the covered business value of requirements by the test case that is not already covered by any other test case. In equation 21, if the business value covered is V, and the test case execution time is C then for the test cases presented in Table 3.16, the Business Value Coverage (BVC) is presented in Table 3.17.

Table 3.16: Test cases vs requirements

Test case	V/C	BVC
T1	4/3	1.33
T2	3/1	3
T3	0/2	0

Table 3.17: Test cases business value coverage

Test cases with cost	Requirements with business value		
	R1= 2	R2= 1	R3= 4
T1= 3			X
T2= 1	x	X	
T3= 2	x		

The test case T1 covers R3 which has a business value of 4 and the execution time of T1 is 3. Diving 4 by 3 results in 1.33 so the BVC of T1 is 1.33. The test case T2 covers R1 and R2 having business values 2 and 1, respectively. The total business value covered by T2 is 3 and its cost is 1 so its BVC is 3. The test case T3 covers R1 but it is already covered by T2 therefore BVC of T3 is 0. Now test cases can be prioritized by greatest to least business value coverage. According to coverage score criteria, T2, T1, T3 is the best order of execution. Table 3.18 shows the performance results of different orders of test cases in the test case suite in terms of the $APRC_v$. The value of $APRC_v$ is calculated by using the formula given in Equation 22.

Table 3.18: Performance of test orders in terms of $APRC_v$

Test case order	$APRC_v$
T2, T1, T3	94.00%
T2, T3, T1	91.00%
T3, T2, T1	84.00%
T3, T1, T2	80.00%
T1, T2, T3	90.00%
T1, T3, T2	83.00%

B) Average Percentage of Requirement Coverage ($APRC_v$)

In this thesis, we propose a new metric $APRC_v$. It is a value-cognizant metric that provides the measure of the average percentage of business value coverage of requirements per cost of test cases for a given order of test cases in a test suite. The metric formula is given in the following equation 22.

$$APRC_v = 1 - \frac{\sum_{i=1}^{n}(TR_i \times \frac{R_i}{C_i})}{\sum_{i=1}^{n}(C_i) \times \sum_{j=1}^{m}(R_i)} + \frac{1}{2 \times \sum_{i=1}^{n} c_i} \quad (22)$$

In the above formula, TR_i is the order of the first test case that covers requirement *i*, R_i is the business value of requirement *i*, and C_i is the cost of the test case. The expanded form of equation 22 is given in equation 23.

$$APRC_v = 1 - \frac{TR_1 \times \frac{R_1}{C_1} + TR_2 \times \frac{R_2}{C_2} + TR_3 \times \frac{R_3}{C_3} + \ldots + TR_n \times \frac{R_n}{C_n}}{\sum_{i=1}^{n}(C_i) \times \sum_{j=1}^{m}(R_i)} + \frac{1}{2 \times \sum_{i=1}^{n}(c_i)} \quad (23)$$

The metric $APRC_v$ is supposed to evaluate the performance of different test case orders for the requirements coverage-based TCP technique. It assumes that the business value of requirements is already known. According to the test case set and requirements set presented in Table 3.18, the value of $APRC_v$ for the test case order T2, T1, T3 is 94.00% which is the best order.

While proposing performance metrics $APFD_v$ and $APRC_v$, we performed more than 100 trials to generalize the derived equations. This process ensured that the minimum value of $APFD_v$ can be 0 and the maximum value can be 1. Similarly, the minimum value of $APRC_v$ can be 0 and the maximum can be 1.

CHAPTER 4

DATA ANALYSIS / RESULTS / FINDINGS

4.1. Introduction

Every testing technique, approach, and methodology is proposed to add value to the testing process. This is important to analyze and evaluate what value a new technique has added. This chapter describes the results and discussion of two working examples to validate the proposed business value quantification model and two case studies to validate the proposed TCP techniques and performance evaluation metrics.

To evaluate the effectiveness of the proposed techniques, there are two performance evaluation metrics options, one is $APFD_c$, and the other is $APFD_v$. We utilized the $APFD_v$ metric for the performance evaluation of the proposed technique because it is better than $APFD_c$. To validate this claim we took two already published example cases. The same already published test data have been used and the value of $APFD_c$ and $APFD_v$ is computed for the performance evaluation of the proposed technique. Section 4.2 shows the performance of $APFD_c$ vs $APFD_v$.

4.2. $APFD_c$ vs $APFD_v$

$APFD_c$ is an existing cost-cognizant metric for the performance evaluation of TCP techniques that incorporates varying test case costs and fault severity [19]. This was introduced to overcome the limitations of the APFD metric. $APFD_c$ is not derived from APFD and is a little bit complex. On the other hand, $APFD_v$ is a proposed metric for the performance evaluation of TCP techniques. $APFD_v$ is derived from the native evaluation metric APFD. It is as simple as APFD. From the performance point of view, $APFD_v$ is better than $APFD_c$ and it produces better results. For the performance comparison of $APFD_v$ and $APFD_c$, let us take two examples.

Example 1

Example 1 shown in Table 4.1 is reported in [19]. It contains five test cases and ten faults. The authors assumed that test case B has cost 2 and all other test cases have cost 1. Similarly, faults F6 and F7 have a severity of 3 and all other faults have a severity of 1.

Table 4.1: Test cases with cost vs faults with severity

Test cases with cost	Faults with severities									
	F1= 1	F2= 1	F3= 1	F4= 1	F5= 1	F6= 3	F7= 3	F8= 1	F9= 1	F10= 1
A= 1	x				x					
B= 2						x	x			
C= 1	x	x	x	x	x	x	x			
D= 1					x					
E= 1								x	x	X

Table 4.2 shows the comparison of results of $APFD_c$ and $APFD_v$ for the test case orders A, B, C, D, E, and B, A, C, D, E. It demonstrates that $APFD_v$ has better results than $APFD_c$.

Table 4.2: Results comparison of $APFD_c$ and $APFD_v$

Test case order	$APFD_c$	$APFD_v$
A, B, C, D, E	52.38%	70.00%
B, A, C, D, E	54.76%	72.00%

Example 2

Example 2 shown in 4.3 is reported in [133]. It contains five test cases and four faults. The test case cost and severity of faults are shown in Table 4.3.

Table 4.3: Test cases with cost vs faults with severity

Test cases with cost	Faults with severities			
	F1= 2	F2= 1	F3= 4	F4= 3
A= 3	x		x	
B= 1		x		
C= 2	x			X
D= 1		x		
E= 4	x		x	X

Table 4.4 shows the comparison of the results of $APFD_c$ and $APFD_v$ for the test case orders E, D, C, B, A, and A, B, C, D, E. It demonstrates that $APFD_v$ provides better results than $APFD_c$.

Table 4.4: Results comparison of $APFD_c$ and $APFD_v$

Test case order	$APFD_c$	$APFD_v$
E, D, C, B, A	75.90%	88.00%
A, B, C, D, E	75.00%	97.00%

Both Example 1 and Example 2 proved that the performance of the $APFD_v$ metric is better than the $APFD_c$ metric. Both example datasets are taken from already published papers. Hence $APFD_v$ is the right metric for performance evaluation of value-cognizant TCP techniques. In this study, the $APFD_v$ metric is utilized for the performance evaluation of proposed techniques through case studies.

4.3. Validation Of Proposed Business Value Quantification Model For VB-TCP

To evaluate the feasibility of the proposed model, two working examples are designed. The proposed model is applied to a healthcare-related software system "ACO Healthcare Solution" (ACO-HCS) developed by a US-based healthcare IT company. An ACO is an Account Care Organization regulated by CMS (Center for Medicare and Medicaid Services) for the management of Medicare population healthcare in the United States of America [134]. ACO-HCS is being used by more than 40 ACOs for the healthcare management of more than 180000 patients across the USA.

The major objective of the proposed model is to estimate fault severities and test case costs through business value computation of software features as well as quality attributes that serve value-based TCP processes for regression testing. First, the dataset is finalized. The dataset contains the requirements set against which the application is required to be tested. Then the proposed model is applied to compute the business value of requirements. Afterward, we extracted the traceability matrix of test cases vs requirements and requirements vs faults from the DevOps system. Test case cost and severity of faults are derived from the business value of requirements. Once requirements business value, faults severities, and test case costs are extracted, two coverage matrixes are developed as refined data sets that serve to calculate the business value of test cases. These coverage matrix datasets are presented in Table 4.5 and Table 4.7. Working Example 1 is based on the test data presented in Table 4.5 and Working Example 2 is based on the test data presented in Table 4.7.

4.3.1. Working Example 1

4.3.1.1. Unit of Analysis and Method

The unit of analysis for working example 1 is a software fault set of the healthcare system ACO-HCS. The example consists of the following steps:

- A monthly data update project containing medical claims data is taken as an example case to empirically evaluate the proposed FDC-TCP.
- Finalize a list of requirements, their associated test cases, and faults.

- Five business success factors are listed. The list is taken from section 2.3 of this thesis.
- The working of the proposed model was explained to the business analysis team.
- Data is taken from the business analysis team. They applied the Delphi technique to assign a value to five success factors against each requirement (feature/quality attribute) of the above-mentioned system.
- As a result, a list of requirements is achieved with their computed business value.
- Requirements are managed in the DevOps system and test cases are linked with the requirement
- Test cases with their cost and faults with their severity are extracted from DevOps along with the coverage metrics
- The GA-based algorithm is applied to prioritize the test cases by using severity detection as a prioritization criterion.
- Results are evaluated in terms of the $APFD_v$

4.3.1.2. Test Data 1: Test Cases VS Faults Coverage Matrix

Test data 1 contains the list of 20 faults taken against a monthly release of a data update project for an ACO-based patient care software application. The fault set also contains their severities. The test case cost and their detection of faults are also presented in the dataset. The test data is given in Table 4.5.

Table 4.5: Test data 1-test cases vs faults

Faults & severity / Test Cases & cost		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
		S = 3	S = 4	S= 2	S= 1	S= 1	S= 2	S= 1	S= 2	S= 3	S= 5	S= 2	S= 1	S= 4	S= 2	S= 5	S= 2	S= 2	S= 3	S= 3	S= 5
T1	C= 2		x																		x
T2	C= 1	x																			
T3	C= 3				x										x						
T4	C= 5							x	x												
T5	C= 4					x													X		
T6	C= 2										x										

T7	C= 1						x													x
T8	C= 2			x											x					
T9	C= 5												x				x			
T10	C= 1				x															
T11	C= 2																	x		
T12	C= 4											x		x						
T13	C= 3									x									x	
T14	C= 5					x										x				

4.3.1.3. Adequacy Criteria for Test Prioritization

Fault severity detection is adopted as the adequacy criteria for test case prioritization. The test cases with higher severity detection get more priority.

4.3.1.4. Prioritization Algorithm

TCP is a permutation-based search optimization problem; therefore, a Genetic Algorithm (GA) is utilized as a prioritization algorithm.

4.3.1.5. Evaluation Metric

APFD is the most used evaluation metric for the performance evaluation of fault detection-based TCP techniques, but this is a value-neutral metric and cannot be used for the performance evaluation of value-oriented fault detection-based TCP techniques. Therefore, $APFD_v$ is adopted as an evaluation metric because it incorporates varying fault severities and test case costs in the TCP process.

4.3.1.6. Comparison Techniques

The performance of the proposed business value-based TCP approach is compared with four different state-of-the-art TCP approaches including Greedy Order (GO), Reverse Order (REVO), Original Order (OO), and Random Order (RO). These four approaches have been used for performance comparison in different studies [24] [135].

4.3.1.7. Results

The performance comparison results of the proposed TCP with existing approaches are shown in Table 4.6. The results are presented in terms of $APFD_v$ and are also depicted in Figure 4.1.

Table 4.6: Performance comparison of proposed vs existing approaches in terms of $APFD_v$

Evaluation Metric	Existing TCP Approaches				Proposed Approach
	GO	RO	REO	OO	FDB-TCP
$APFD_v$	0.9524	0.9187	0.9214	0.9501	0.916

The results presented in Table 4.6 show that FDB-TCP has produced better results in terms of $APFD_v$ as compared to other existing techniques. Greedy order is the second-best approach. Random order is the least-performing approach. As per the proposed approach, the best-prioritized order provides an $APFD_v$ value of 0.9616, and the order is as follows.

Best order:

[['T1', 1], ['T2', 4], ['T3', 10], ['T4', 12], ['T5', 11], ['T6', 3], ['T7', 5], ['T8', 7], ['T9', 13], ['T10', 14], ['T11', 9], ['T12', 8], ['T13', 2], ['T14', 6]]

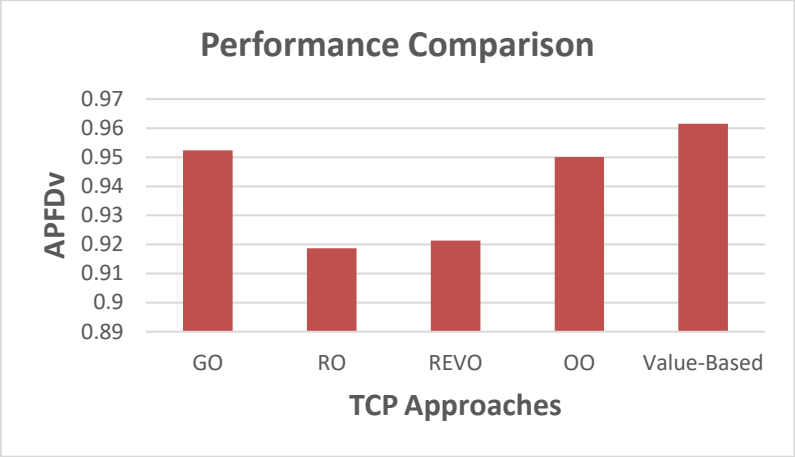


Figure 4.1: $APFD_v$ of proposed vs existing TCP Approaches

4.3.1.8. Discussion of Cost Consumption VS Severity Detection

Apart from the performance comparison of the proposed TCP approach with other TCP approaches, it is observed that the test cases appearing earlier in the prioritization order are detecting more severity of faults. This trend is depicted in Figure 4.2. In the first five prioritized test cases, the severity detection percentage of the test is higher than their cost consumption percentage. The test cases with a lower severity detection percentage than the cost consumption percentage appear later in the prioritized order. This trend validates the effectiveness of value-based FDB-TCP.

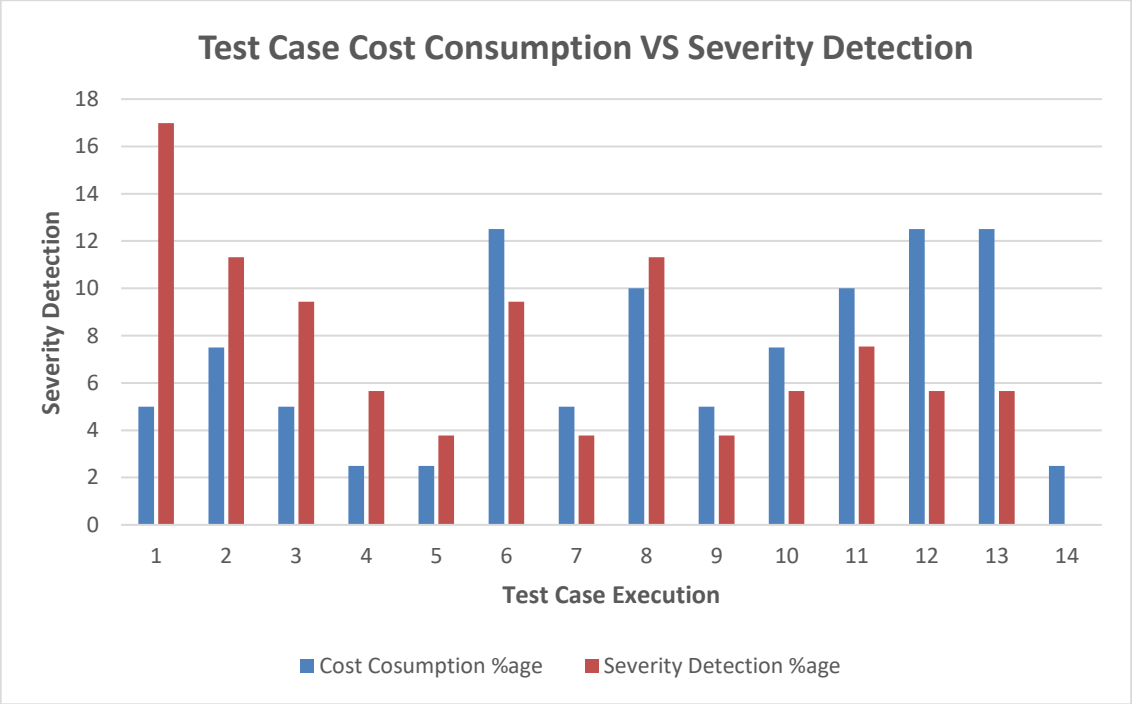


Figure 4.2: Test cases cost consumption vs severity detection trend

4.3.2. Working Example 2

4.3.2.1. Unit of Analysis and Method

The unit of analysis for working example 2 is a software feature set and quality attributes set of the healthcare system ACO-HCS. The case consists of the following steps:

- A monthly data update project containing medical claims data is taken as an example case to empirically evaluate the proposed model.
- Finalized the list of important features of the ACO-HCS system. This list contains eight major features.
- Finalized the list of software quality attributes that are important for the specified system. This list contains twelve quality attributes and four were the most critical for the system.
- We combined eight features and four quality attributes resulting in a list of fourteen requirements.
- Five business success factors are listed. The list is taken from section 2.3 of this thesis.

- The working of the proposed model was explained to the business analysis team.
- Data is taken from the business analysis team. They applied the Delphi technique to assign a value to five success factors against each feature, and quality attribute of the above-mentioned system.
- As a result, a list of requirements is achieved with their computed business value.
- Requirements are managed in the DevOps system and test cases are linked with the requirement
- A traceability matrix of test case vs requirement was prepared.
- The GA-based algorithm is applied to prioritize the test cases by using business value coverage as prioritization criteria.
- Results are evaluated in terms of the $APRC_v$

4.3.2.2. Test data 2: Test Cases VS Requirements Coverage Matrix

Test data 2 contains the list of 14 requirements taken against a monthly release of a data update project for an ACO-based patient care software application. The requirements set contains both functional requirements (features) and non-functional requirements (quality attributes) along with their business values computed through the proposed model. The test case cost and their coverage of requirements are also presented in the dataset. The dataset is given in Table 4.7 and is used to calculate the business value of test cases.

Table 4.7: Test data 2-test cases vs requirements

Requirements & business value / Test Cases & cost		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14
		BV=2	BV=4	BV=1	BV=3	BV=2	BV=4	BV=5	BV=1	BV=3	BV=2	BV=3	BV=4	BV=1	BV=3
T1	C= 2			x											
T2	C= 1	x													
T3	C= 3									x					x
T4	C= 5							x							
T5	C= 4			x											
T6	C= 2					x									
T7	C= 1													x	
T8	C= 2				x										x
T9	C= 5												x		
T10	C= 1						x								
T11	C= 2								x						
T12	C= 4										x				
T13	C= 3											x			
T14	C= 5		x												

4.3.2.3. Adequacy Criteria for Test Prioritization

Business value coverage is adopted as an adequacy criterion for test case prioritization. The test cases with higher business value coverage get more priority.

4.3.2.4. Prioritization Algorithm

The prioritization algorithm is the same as utilized in the example case 1.

4.3.2.5. Evaluation Metric

Average Percentage of Requirements Coverage (APRC) is the most used evaluation metric for the performance evaluation of requirement coverage-based TCP techniques, but this is a value-neutral metric and cannot be used for the performance evaluation of value-based requirement coverage-based TCP techniques. Therefore, $APRC_v$ is adopted as an evaluation metric because it incorporates the varying business value of requirements and test case costs in the TCP process.

4.3.2.6. Comparison Techniques

The comparison techniques are the same as those utilized in example case 1.

4.3.2.7. Results

The performance comparison results of the proposed TCP with existing approaches are shown in Table 4.8. The results are presented in terms of $APRC_v$ and are also depicted in Figure 4.3.

Table 4.8: Performance comparison of proposed vs existing approaches in terms of $APRC_v$

Evaluation Metric	Existing TCP Approaches				Proposed Approach
	GO	RO	REVO	OO	RCB-TCP
$APRC_v$	0.9467	0.9270	0.9312	0.9374	0.9671

The results presented in Table 4.8 show that RCB-TCP has produced better results in terms of $APRC_v$ as compared to other existing techniques. GO is the second-best approach. RO is the least-performing approach. As per the proposed approach, the best-prioritized order provides an $APRC_v$ value of 0.9671, and the order is as follows.

Best order:

[['T1', 13], ['T2', 3], ['T3', 2], ['T4', 6], ['T5', 11], ['T6', 8], ['T7', 4], ['T8', 12], ['T9', 9], ['T10', 1], ['T11', 5], ['T12', 14], ['T13', 10], ['T14', 7]]

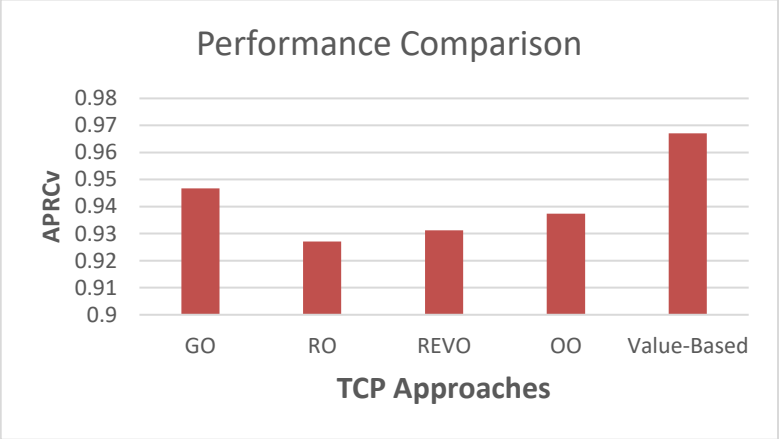


Figure 4.3: $APRC_v$ Comparison of proposed vs existing TCP approaches

4.3.2.8. Discussion of Cost Consumption VS Business Value Coverage

Apart from the performance comparison of the proposed RCB-TCP approach with other TCP approaches, it is observed that the test cases appearing earlier in the prioritization order cover more business value of requirements. This trend is shown in Figure 4.4. In the first three prioritized test cases, the business value coverage percentage of test cases is much higher than their cost consumption percentage. The test cases with lower business value coverage percentages than cost consumption percentages appear later in the prioritized order. This trend validates the effectiveness of value-based RCB-TCP.

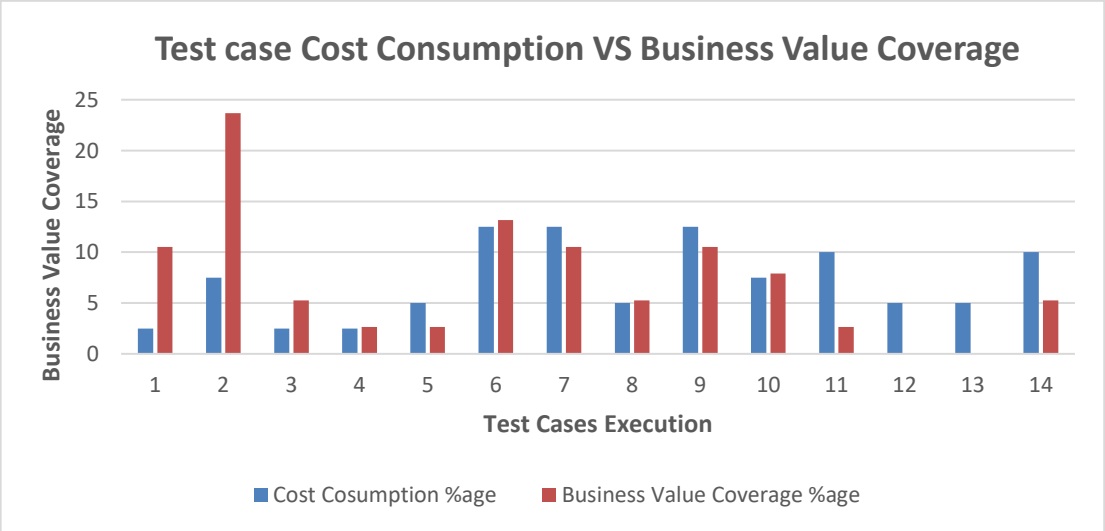


Figure 4.4: Test cases cost consumption vs severity detection trend

It is noted that normally business guys are specialists in marketing, sales, and business

growth but they do not go into the domain of IT units that are developing software to support the businesses. Similarly, IT professionals are experts in solution findings, algorithms, and logic building. They are not used to crossing their limits and do not go into the business domain. This creates a huge gap in understanding business needs. The proposed model will bridge this gap and provide insight into the business worth of IT deliverables. When IT guys are aware of the business worth of each software feature and quality attribute, they will better prioritize their decisions.

This model will be used by software professionals as well as software clients to finalize their quality requirements. By utilizing it, business-critical quality attributes and software features will be identified and addressed during the early phases of the software development life cycle. Software regression testing is a big challenge. Due to limited time and resources, this is not possible to re-execute all test cases. The proposed value-based prioritization helps testing teams consume their regression testing time to execute the most valuable test cases.

4.4. Validation of Proposed VB-TCP Techniques and Performance Metrics

This section presents two case studies designed to validate the proposed VB-TCP techniques and their performance evaluation metrics. This section addressed RQ4.

4.4.1. Case Study 1

This section presents a case study conducted to evaluate the performance of the proposed VCFDB-TCP technique using a proposed novel performance evaluation metric.

4.4.1.1. Context of Study

As an object of our study, two software products are selected from the healthcare domain that is developed by a US-based Healthcare IT company. These are developed to support the Account Care Organizations (ACO) business. An ACO is an Account Care Organization that comprises a group of doctors, healthcare providers, and physicians who voluntarily join to manage high-quality coordinated care of Medicare patients attributed to them [136]. An ACO has threefold goals including better care, lowering care costs, and

improving patient experience [136]. We followed the guidelines provided by Per Runeson, in [137], for performing case study research in software engineering.

The test data used for case study 1 is described in 4.9 which is related to two healthcare products to evaluate the performance of the proposed VCFDB-TCP technique in terms of specified performance goals. The test data is collected from the Azure DevOps system in which the population care management projects are managed through the scrum model. Product development is managed in the form of epics, features, requirements, and tasks. Each release is comprised of a set of features. A feature can have many requirements. Test cases are designed against requirements. Bugs or faults are filed against the execution of test cases. Each fault is associated with a test case and each test case is associated with a requirement. A complete traceability matrix is developed through a fully automated process. Microsoft configuration management tools Azure DevOps, and Microsoft Test Manager (MTM) are used for the automated testing life cycle. The extracted dataset comprises test cases, faults, and fault detection information. The test cases also include their execution time or cost, and faults include their severities. Test case execution time is recorded by Microsoft Test Manager and fault severities are defined by test experts using standard guidelines. The test cases and fault data are collected for the last three releases against Product A and Product B. Product A is a healthcare management system developed for care analytics. Its current version is V7, it has 87132 lines of code and was developed using .Net/MVC technologies. Product B is a healthcare-related application developed for the care management of patients. Its current version is V7, it has 97450 LOC and was developed using .Net/MVC technologies. The test data specifications are given in Table 4.9.

Table 4.9: Test data for Product A and Product B

Product A			**Product B**		
Release	Regression Test cases	Faults	Release	Regression Test cases	Faults
R1	102	156	R1	108	135
R2	114	188	R2	115	165
R3	123	212	R3	101	124

4.4.1.2. Testing Criteria, Evaluation Algorithm, and Evaluation Metric

To answer the established research question, fault detection capability is used as the testing criteria. This fault detection capability is taken in a value-based fashion where the severity of faults and test case cost are considered. The fault detection capability of a test case is the ratio value of total severity detected and total cost consumed. We extracted the test cases vs faults matrix along with test cases cost and severity of faults. Fault severity detection is the coverage criteria to be optimized. The performance of the proposed technique is compared with four other state-of-the-art techniques including Original Order (OO), Reverse Order (REV-O), Random Order (RO), and Greedy algorithm. The comparison techniques were utilized in [24] [135].

A greedy Algorithm is a search-based algorithm that is implemented to find the "next best" [47]. The element with the highest weight is selected first followed by the second highest, third highest, and so on. A greedy algorithm is used to solve TCP problems in many research papers [15], [138]. Consider test cases are required to be ordered based on fault detection rate. Using the Greedy approach, the test cases with the higher number of detected bugs will come earlier in the test case order in a test suite. For instance, there is a test suite having five test cases t1, t2, t3, t4, t5 and assume that t4 detects 3 bugs, t5 detects 2 bugs, t3 detects 4 bugs, t1 detects 2 bugs, and t2 detects 1 bug, then prioritized test case order will be (t3, t4, t1, t5, t2).

The dominant metric for performance evaluation of TCP techniques is APFD but this is not applicable for value-based TCP where test case execution time, the severity of faults, or the business value of elements may vary. There are two value-based cost-cognizant $APFD_c$ and $APFD_v$ for the performance evaluation of our proposed value-based TCP technique [135]. Example case 1, and example case 2 proved that $APFD_v$ is producing better results than $APFD_c$. Therefore, we used $APFD_v$ as a performance evaluation metric because the performance goal of the proposed VCFDB-TCP is to increase the average percentage of fault severity detection per value.

4.4.1.3. Results of the Study

In this section, the results have been presented to answer the defined research question. The results of the case study are compared in terms of $APFD_v$ for the proposed and

existing state-of-the-art approaches and are presented in Table 4.10 and Table 4.11.

Table 4.10: $APFD_v$ of products A releases

Release	Regression Test cases	Faults	$APFD_v$					Execution Time of Value-Based GA
			Original order	Reverse Order	Random	Greedy	Value-Based GA	
R1	102	156	0.9313	0.9193	0.9178	0.9279	0.9408	35.1121
R2	114	188	0.9332	0.9262	0.9263	0.9275	0.9445	38.5359
R3	123	212	0.9221	0.9356	0.9313	0.9286	0.9437	48.1182
Average of All Releases			**0.9288**	**0.9271**	**0.9251**	**0.9280**	**0.9430**	**40.5887**

The performance results of all three releases are averaged out. 4.11 indicates that the performance results of the proposed technique is better than RO, OO, Greedy Order, and REVO approaches in terms of $APFD_v$ against all releases of Product A. The proposed technique outperformed state-of-the-art techniques. The performance of the RO approach was the worst among all the techniques. The performance of OO was second best. The results are presented in a box plot chart in Figure 4.5.

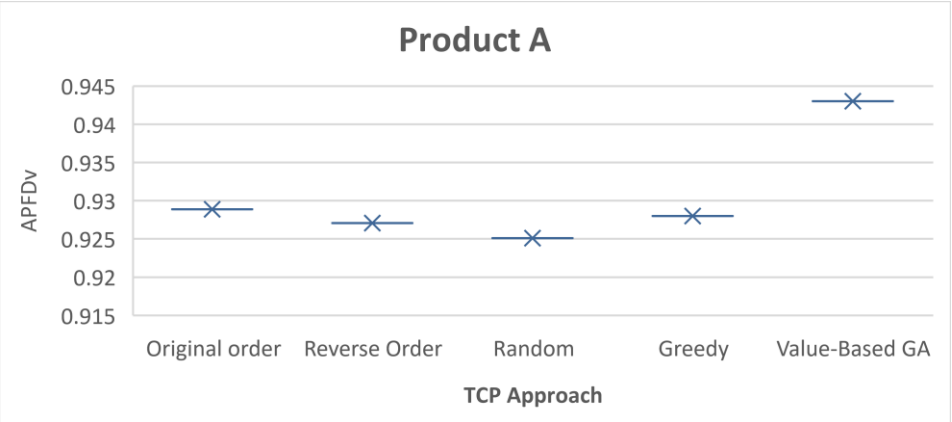


Figure 4.5: Performance results of product A releases in terms of $APFD_v$

Table 4.11 depicts the performance results of different releases of product B. It indicates that the performance of the proposed technique is better than RO, OO, Greedy Order, and

REVO approaches in terms of $APFD_v$ against product B.

Table 4.11: $APFD_v$ of the three releases for two products B

Release	Regression Test cases	Faults	$APFD_v$					Execution Time of Value-Based GA
			Original order	Reverse Order	Random	Greedy	Value-Based GA	
R1	101	124	0.9385	0.9339	0.9367	0.9374	0.9477	40.8578
R2	108	135	0.9456	0.9188	0.9278	0.9336	0.9502	55.6960
R3	115	165	0.9348	0.9409	0.9386	0.9423	0.9472	44.7882
Average of All Releases			**0.9397**	**0.9313**	**0.9344**	**0.9378**	**0.9484**	**47.1140**

A pictorial representation of the results is given in the box plot chart in Figure 4.6. The proposed technique outperformed existing state-of-the-art techniques in terms of $APFD_v$. OO is the second-best performer and REVO is the least performer.

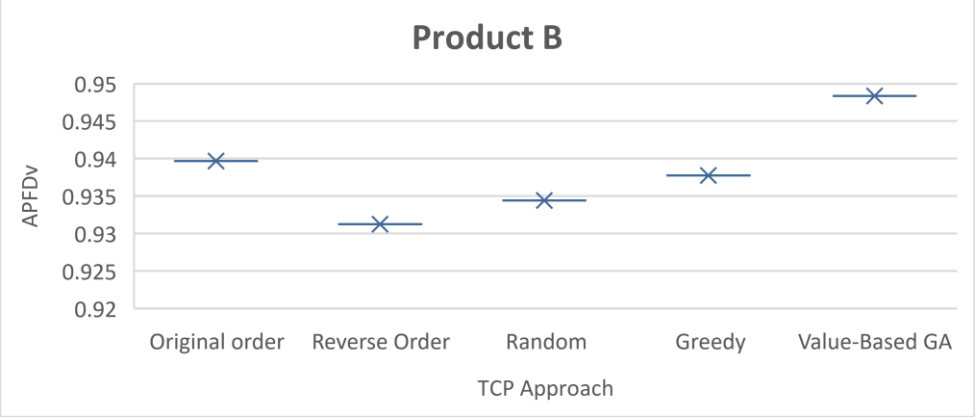


Figure 4.6: Performance results of product B release in terms of $APFD_v$

The proposed technique is the GA-based search optimization TCP technique implemented

using Python language. The execution of GA is based on some termination criteria. In this study, termination criteria are based on the number of iterations. The number of iterations is a value taken as user input. Each iteration processes five permutations and calculates $APFD_v$ against the dataset. The execution time of the implemented technique is based on the number of iterations. Against each release dataset, we executed it with 25, 50, 100, 200, 400, and 800 iterations and recorded its $APFD_v$ and maximum execution time. Table 4.12, and Table 4.13 shows the value of $APFD_v$ and maximum execution time for a different number of iterations against each release. The average maximum execution time of products A and B are **40.0237 seconds** and **47.1140 seconds,** respectively.

Table 4.12: $APFD_v$ of value-based TCP using GA against different numbers of iterations for releases of products A

Release	**Regression Test cases**	**Faults**	**$APFD_v$ of Value-Based TCP Using GA per Iterations**						**Maximum Execution Time**
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	102	156	0.9339	0.9359	0.9372	0.9375	0.9400	0.9407	33.4168
R2	114	188	0.9415	0.9425	0.9425	0.9436	0.9445	0.9445	38.5359
R3	123	212	0.9390	0.9415	0.9418	0.9433	0.9433	0.9437	48.1182
Average of All Releases			**0.9382**	**0.9399**	**0.9405**	**0.9415**	**0.9426**	**0.9430**	**40.0237**

The $APFD_v$ value trend with the different numbers of iterations for Product A is depicted in Figure 4.7. The graph shows that as the number of iterations increases, the $APFD_v$ value increases gradually. The growth trend in all three releases of product A is almost consistent.

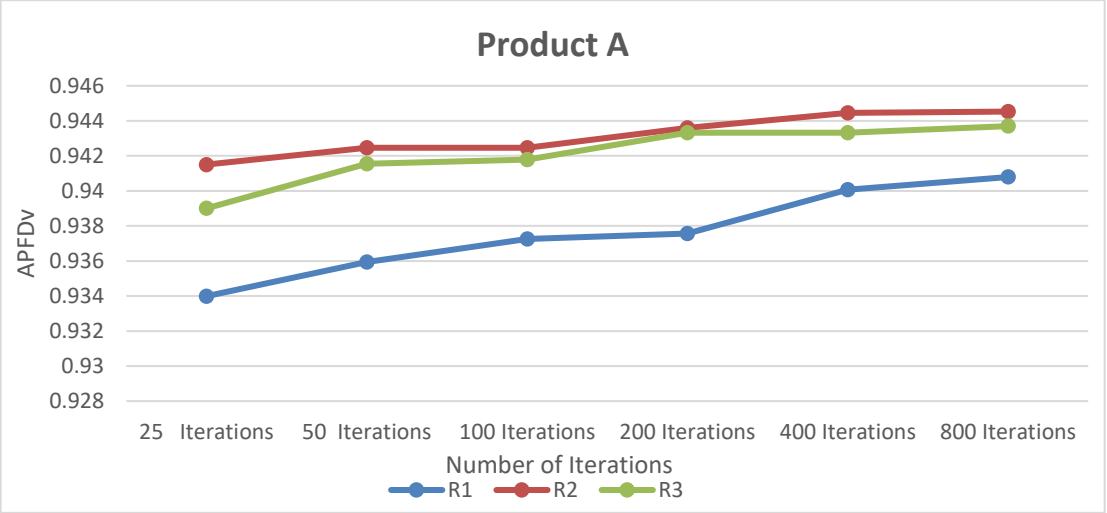


Figure 4.7: $APFD_v$ trend per number of iterations for product A

Table 4.13: $APFD_v$ of Value-based TCP using GA against different numbers of iterations for releases of products B

Release	Regression Test cases	Faults	$APFD_v$ of Value-Based TCP Using GA per Iterations						Maximum Execution Time
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	101	124	0.9437	0.9435	0.9464	0.9466	0.9467	0.9476	40.8578
R2	108	135	0.9445	0.9440	0.9480	0.9454	0.9493	0.9501	55.6959
R3	115	165	0.9442	0.9438	0.9450	0.9464	0.9461	0.9472	44.7881
Average of All Releases			**0.9442**	**0.9438**	**0.9465**	**0.9461**	**0.9474**	**0.9483**	**47.1139**

Similarly, the $APFD_v$ value trend with the different numbers of iterations for Product B is depicted in Figure 4.8. The graph shows that as the number of iterations increases, the $APFD_v$ value increases gradually. The growth trend in releases R1 and R3 of product B is consistent. For R2, the $APFD_v$ value declined with 200 iterations and then improved with 400, and 800 iterations gradually.

The overall trend of increase in $APFD_v$ value with a greater number of iterations is consistent. The different number of iterations are exercised for different releases of

Product A and Product B. With 800 iterations, the results are almost mature.

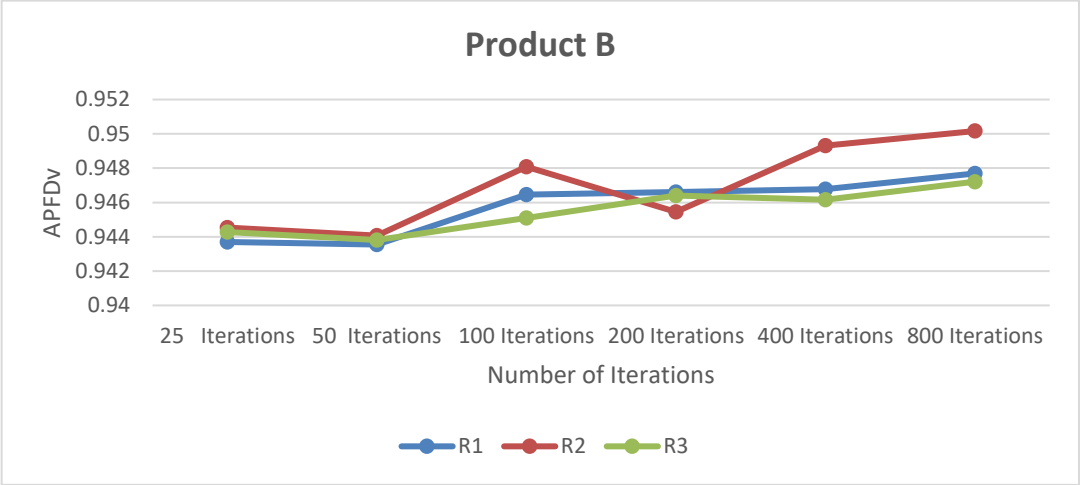


Figure 4.8: $APFD_v$ trend per number of iterations for product B

Tables 4.14 and 4.15 show the execution times of the different numbers of iterations against different releases along with the maximum value of $APFD_v$. The average maximum $APFD_v$ values of products A and B are **0.9430** and **0.9484,** respectively.

Table 4.14: Execution time of value-based TCP using GA for different numbers of iterations for releases of products A

Release	Regression Test cases	Faults	Execution Time of Value-Based TCP Using GA per Iterations						Maximum $APFD_v$
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	102	156	1.1180	2.2330	4.4951	9.2402	17.5999	35.1120	0.9408
R2	114	188	1.5169	2.7855	5.3598	10.4341	20.5091	38.5358	0.9445
R3	123	212	1.4989	3.8237	6.2855	14.8163	28.3415	48.1182	0.9437
Average of All Releases			**1.3780**	**2.9474**	**5.3801**	**11.4969**	**22.1501**	**40.5887**	**0.9430**

Tables 4.14 and 4.15 and Figures 4.9 and 4.10 show that execution time is directly proportional to the number of iterations.

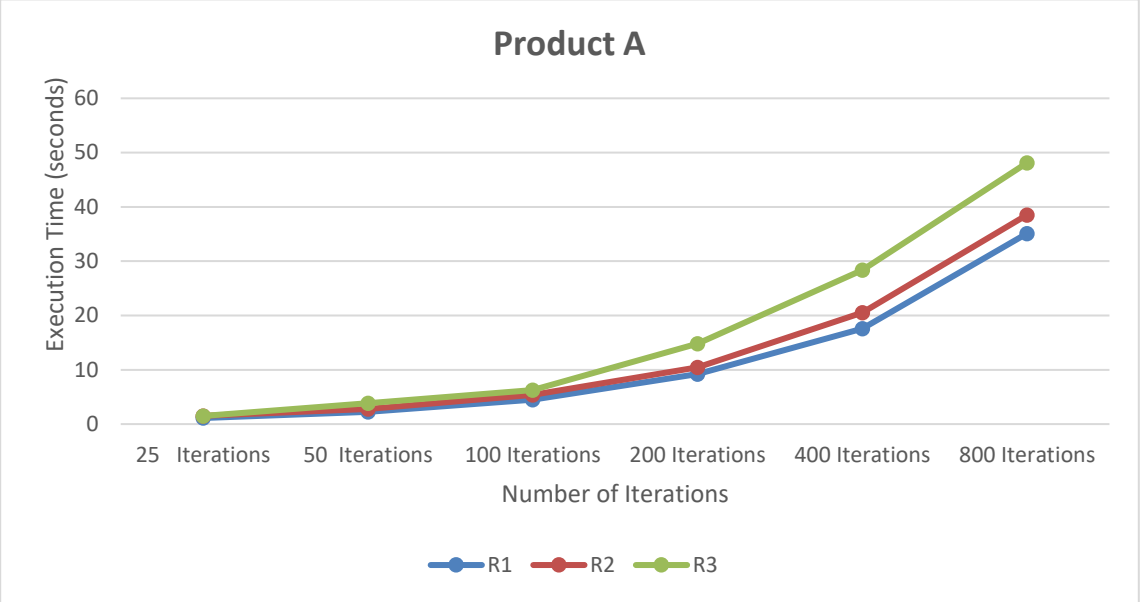


Figure 4.9: Execution time trend per number of iterations for product A

Table 4.15: Execution time of value-based TCP using GA for different numbers of iterations for releases of products B

Release	Regression Test cases	Faults	**Execution Time of Value-Based TCP Using GA per Iterations**						**Maximum APFD$_v$**
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	101	124	0.9214	1.9816	4.3108	9.9952	15.7253	40.8578	0.9476
R2	108	135	0.9245	3.7579	3.9753	7.8290	14.9545	55.6959	0.9501
R3	115	165	1.1888	2.7127	10.2575	9.8386	19.5523	44.7881	0.9472
Average of All Releases			**1.0116**	**2.8175**	**6.1813**	**9.2209**	**16.7440**	**47.1139**	**0.9483**

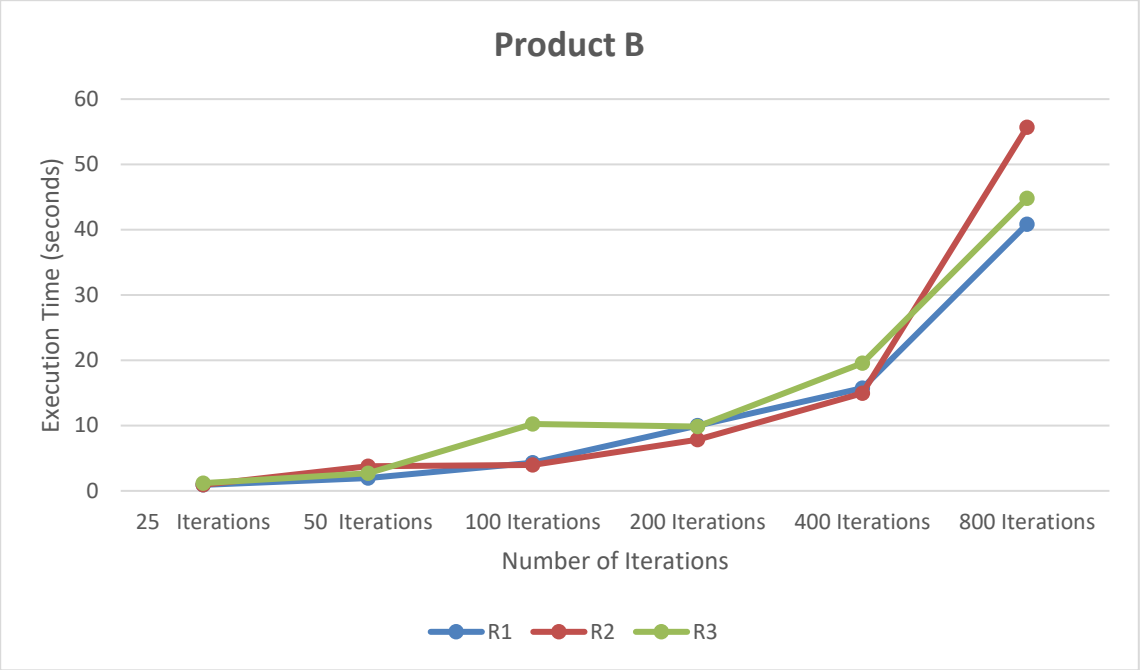


Figure 4.10: Execution time trend per number of iterations for product B

The growth trend in execution time of all three releases of product A is consistent. Similarly, the growth trend of all releases of product B is also consistent. The execution time for 800 iterations of product B quickly increased. The consistency in the execution time of the algorithm makes it reliable.

4.4.1.4. Cost Consumption VS Severity Detection

The section describes the cost consumption vs severity detection trend in pictorial form. In product A, all three releases R1, R2, and R3 show that the initial set of test cases consumed less and detected greater severity. Later test cases consumed higher costs and severity detection declined. This trend shows that the test cases that are likely to detect higher fault severity are prioritized first and are depicted in Figures 4.11, 4.12, and 4.13. In product B, all three releases R1, R2, and R3 generally show that the higher severity ratio cost test cases are prioritized first. This trend is depicted in Figures 4.14, 4.15, and 4.16.

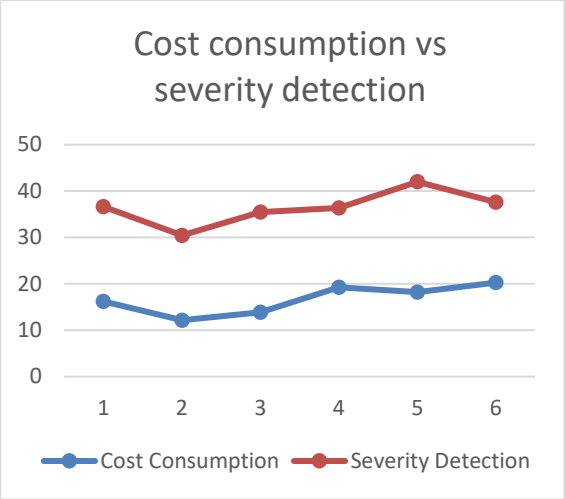


Figure 4.11: Product A, release 1

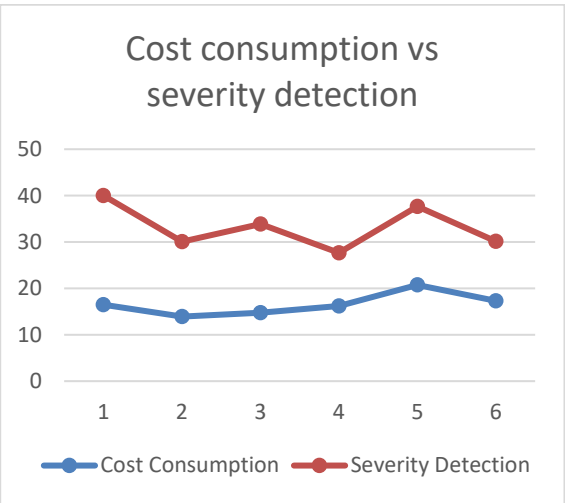


Figure 4.12: Product A, release 2

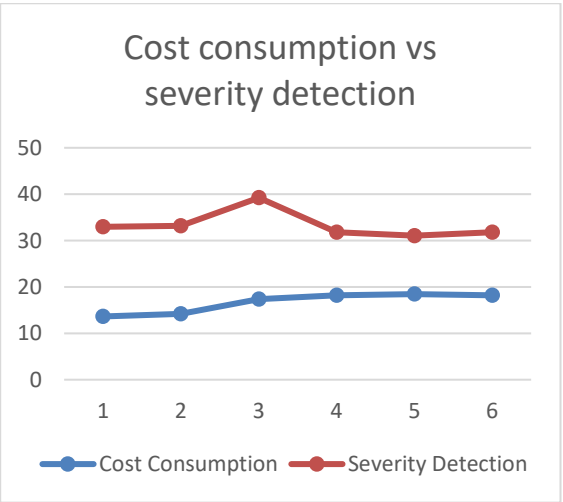


Figure 4.13: Product A, release 3

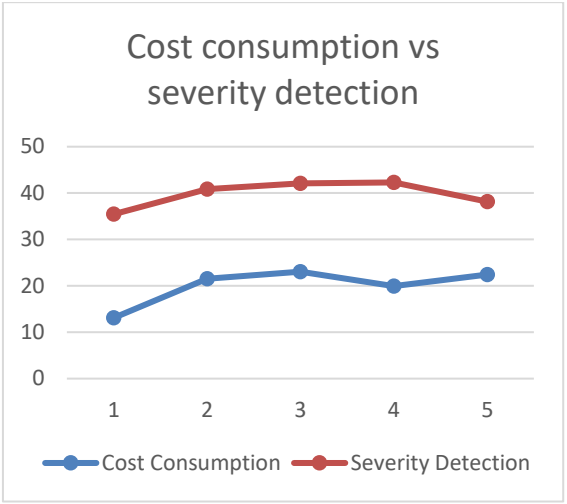


Figure 4.14: Product B, release 1

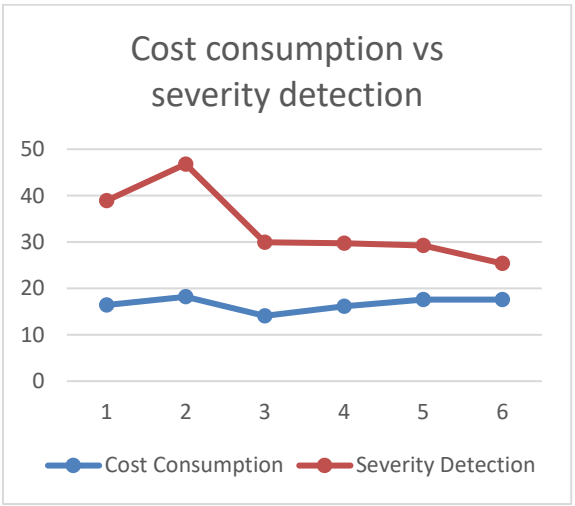


Figure 4.15: Product B, release 2

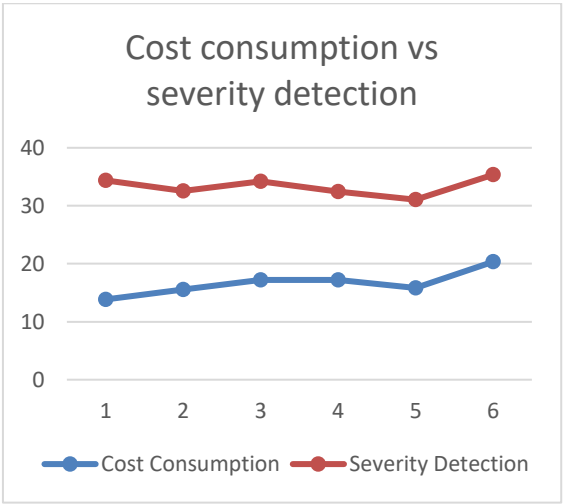


Figure 4.16: Product B, release 3

4.4.1.5. Statistical Analysis

To statistically analyze the results of case study 1, we applied a t-test. A t-test is an inferential statistic utilized to know if there is a significant difference between the means of two different groups. In our case, we compared the $APFD_v$ mean value of our proposed value-based GA with the other four techniques. The first t-test is applied to the mean value of "Original order" and "Value-Based GA".

Null Hypothesis (H_0): $APFD_v$ value of "Original order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APFD_v$ value of "Original order" and "Value-Based GA" is different.

The mean $APFD_v$ value of the Original Order was 0.93425 and it was 0.945683 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned a p-value of 0.004266839. A p-value is the likelihood of finding a mean difference by chance if indeed there is no difference in the population. If the p-value is less than the defined significance level, then the difference between means is statistically significant. In this statistical test the value of p is less than the value of Alpha therefore Null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Original Order.

The second t-test is applied to the mean value of "Reverse order" and "Value-Based GA".

Null Hypothesis (H_0): $APFD_v$ value of "Reverse order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APFD_v$ value of "Reverse order" and "Value-Based GA" is different.

The mean $APFD_v$ value of Revers Order was 0.929116667 and it was 0.945683 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned a p-value of 0.0072552. In this statistical test, the value of p is less than the value of Alpha, therefore, Null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Reverse Order.

The third t-test is applied to the mean value of "Random order" and "Value-Based GA".

Null Hypothesis (H_0): $APFD_v$ value of "Random order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APFD_v$ value of "Random order" and "Value-Based GA" is different.

The mean $APFD_v$ value of Random Order was 0.92975 and it was 0.945683 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned a p-value of 0.001406697. In this statistical test the value of p is less than the value of Alpha therefore Null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Random Order.

The fourth t-test is applied to the mean value of "Greedy order" and "Value-Based GA".

Null Hypothesis (H_0): $APFD_v$ value of "Greedy order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APFD_v$ value of "Greedy order" and "Value-Based GA" is different.

The mean $APFD_v$ value of Greedy Order was 0.932883333 and it was 0.945683 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned a p-value of 0.001041445. In this statistical test the value of p is less than the value of Alpha therefore Null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Greedy Order. The statistical analysis of case study one is given in Table 4.16, 4.17, 4.18, and 4.19.

Table 4.16: Statistical Analysis of Original Order and Value-Based GA

Mean APFDv of Original Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.93425	0.945683	0.05	0.004266839	Significant

Table 4.17: Statistical Analysis of Reverse Order and Value-Based GA

Mean APFDv of Reverse Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**

0.929116667	0.945683	0.05	0.0072552	Significant

Table 4.18: Statistical Analysis of Random Order and Value-Based GA

Mean APFDv of Random Oder	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.92975	0.945683	0.05	0.001406697	Significant

Table 4.19: Statistical Analysis of Greedy Order and Value-Based GA

Mean APFDv of Greedy Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.932883333	0.945683	0.05	0.001041445	Significant

Considering the level of significance as 0.05, the null hypothesis H_0 is rejected in four t-tests. Statistically, the performance of the proposed technique is significantly better than the other four techniques.

4.4.2. Case Study 2

This section describes a case study conducted to evaluate the performance of proposed VCRCB-TCP techniques using a proposed novel performance evaluation metric.

4.4.2.1. Context of Study

This section describes the context of the case study. As an object of our study, two healthcare applications developed by a US-based software company are selected. These applications are developed to support ACO business in the USA. The performance goal of the proposed VCRCB-TCP is to increase the APRC in a value context. The test data for this study is comprised of test cases set, requirements set, and coverage information. The test cases also include their execution time or cost, and requirements include their business value. The test case execution time is recorded by MTM, and requirements business value is defined by the business analysis team by using expert judgment

techniques. Data is collected against three releases. The test data is presented in Table 4.20.

The test data is collected from the Azure DevOps system in which the population care management projects are managed. Test cases are designed against requirements. Each requirement is associated with a test. The test cases set, requirements set, and coverage information for the last three releases are collected against Application A and Application B. Application A is a healthcare management system developed for care analytics. Its current version is V12, it has 65028 lines of code and was developed using .Net/MVC technologies. Application B is a healthcare-related application developed for the care management of patients. Its current version is V12, it has 88210 LOC and was developed using .Net/MVC technologies. The test data specifications are given in Table 4.20.

Table 4.20: Dataset for Application A and Application B

Application A			**Application B**		
Release	**Test Cases**	**Requirements**	**Release**	**Test Cases**	**Requirements**
R1	41	48	R1	36	42
R2	44	52	R2	40	50
R3	48	60	R3	44	53

4.4.2.2. Testing Criteria, Evaluation Algorithm, and Evaluation Metric

To answer the established research question, requirements coverage is used as the testing criteria. This coverage is taken in a value-based fashion where the business value of requirements and test case cost are considered. The business value coverage of a test case is the ratio value of the total requirements business value covered and total cost consumed. The test cases vs requirements coverage matrix along with test cases cost and business value of requirements are extracted. The requirement's business value is the coverage criteria to be optimized. The performance of the proposed technique is compared with

four other state-of-the-art techniques including Original Order (OO), Reverse Order (REV-O), Random Order (RO), and Greedy algorithm. The comparison techniques were also utilized in some other studies [24] [135].

Coverage-based methods are most prominent in TCP, therefore most of the researchers evaluated the performance of their proposed techniques with this method [85]. The metric for performance evaluation of requirements coverage-based TCP techniques is APRC but this is not applicable for value-based requirements coverage-based TCP where test cases execution time and the value of requirements vary. In this study, the performance evaluation metric $APRC_v$ [135] is used for the performance evaluation of VCRCB-TCP. The metric $APRC_v$ is presented by equation 22 in section 3.5.2.

4.4.2.3. Results of the Study

In this section, the results of the study have been presented to answer the defined research question. The results of the case study are compared in terms of $APRC_v$ for the proposed and existing state-of-the-art approaches and are presented in Tables 4.21 and 4.22.

Table 4.21: $APRC_v$ of application A releases

Release	Test cases	Requirements	$APRC_v$					Execution Time of Value-Based GA
			Original order	Reverse Order	Random Order	Greedy	Value-Based GA	
R1	41	48	0.9121	0.8888	0.9012	0.6282	0.9366	38.5386
R2	44	52	0.9510	0.9404	0.9405	0.7976	0.9594	42.8723
R3	48	60	0.9387	0.9048	0.9271	0.7042	0.9410	56.1029
Average of All Releases			**0.9339**	**0.9113**	**0.92297**	**0.7100**	**0.9457**	**45.8380**

The performance results of all three releases are averaged out. Table 4.21 indicates that the performance result of the proposed VCRCB-TCP technique is better than the RO, OO, Greedy Order, and REVO approach in terms of $APRC_v$ against all different releases of Application A. The proposed technique outperformed state-of-the-art techniques. The performance results of the Greedy approach were the worst among all other techniques.

The performance of OO was second best. The results are presented in a box plot chart in Figure 4.17.

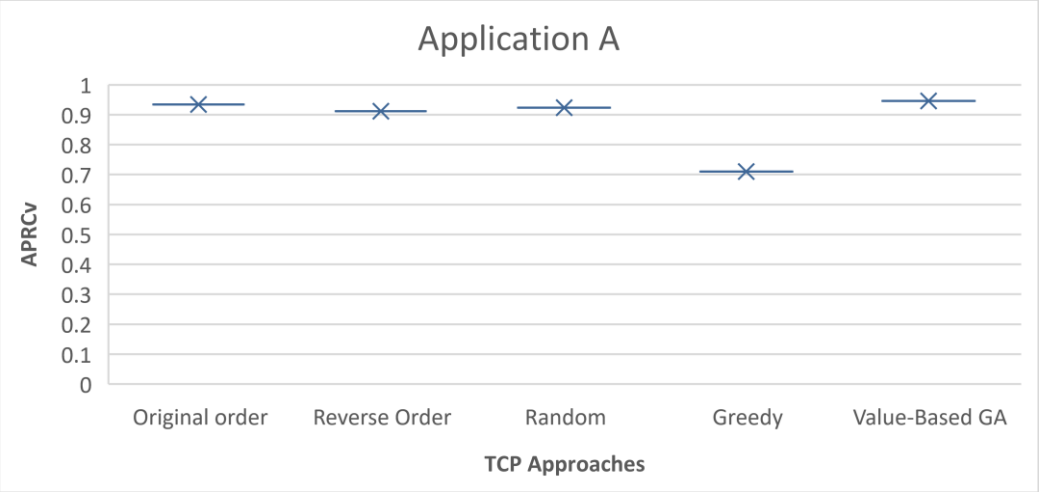


Figure 4.17: Performance results of application A releases in terms of $APRC_v$

Table 4.22 depicts the performance results of different releases of Application B. It indicates that the performance results of the proposed technique are better than RO, OO, Greedy Order, and REVO approaches in terms of $APRC_v$ against Application B.

Table 4.22: $APRC_v$ of the three releases for Application B

Release	Test cases	Requirements	$APRC_v$					Execution Time of Value-Based GA
			Original order	Reverse Order	Random	Greedy	Value-Based GA	
R1	36	42	0.9119	0.9270	0.9087	0.6870	0.9444	51.3133
R2	40	50	0.9289	0.9273	0.9251	0.6961	0.9492	83.0596
R3	44	53	0.9143	0.9214	0.9118	0.6759	0.9454	70.2029
Average of All Releases			**0.9183**	**0.9252**	**0.9152**	**0.6863**	**0.9463**	**68.1919**

A pictorial representation of the results is given in the box plot chart in Figure 4.18. The proposed technique outperformed existing state-of-the-art techniques in terms of $APRC_v$. REVO is the second-best performer and Greedy is the worst among all other techniques.

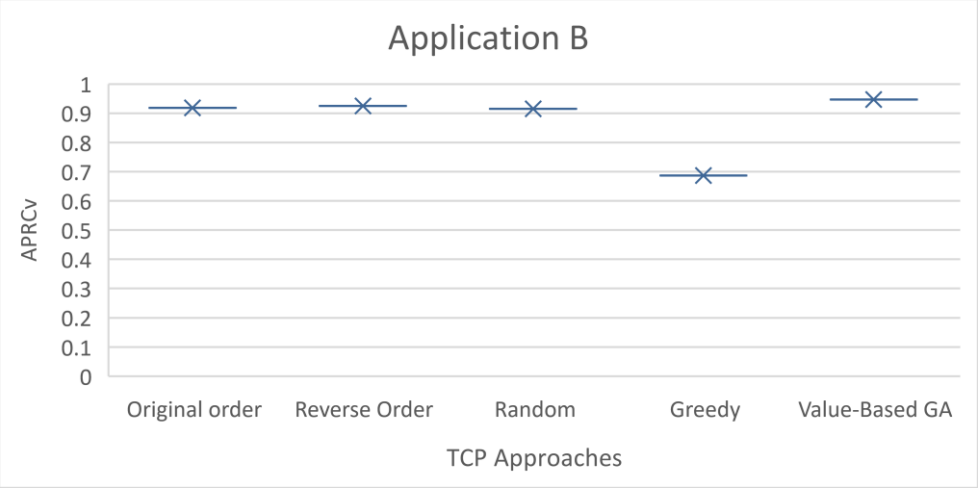


Figure 4.18: Performance results of application B releases in terms of $APRC_v$

The proposed technique is the GA-based search optimization TCP technique implemented using Python language. The execution of GA is based on some termination criteria. In this thesis, termination criteria are based on the number of iterations. The number of iterations is a value taken as user input. Each iteration processes five permutations and calculates $APRC_v$ against the dataset. The execution time of the implemented technique is based on the number of iterations. Against each release dataset, we executed it with 25, 50, 100, 200, 400, and 800 iterations and recorded its $APRC_v$ and maximum execution time. Table 4.23, and 4.24 shows the value of $APRC_v$ and maximum execution time for the different number of iterations against each release. The average maximum execution time of Application A and B are **45.8379 seconds** and **68.1919 seconds,** respectively.

Table 4.23: $APRC_v$ of value-based TCP using GA for different numbers of iterations for releases of applications A

Release	Test cases	Requirements	**$APRC_v$ of Value-Based TCP Using GA per Iterations**						**Maximum Execution Time**
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	41	48	0.9191	0.9228	0.9242	0.9260	0.9278	0.9366	38.5386
R2	44	52	0.9545	0.9587	0.9566	0.9569	0.9590	0.9594	42.8723
R3	48	60	0.9385	0.9376	0.9378	0.9391	0.9393	0.9409	56.1029
Average of All Releases			**0.9373**	**0.9397**	**0.9395**	**0.9407**	**0.9420**	**0.9456**	**45.8379**

The $APRC_v$ value trend with the different number of iterations for application A is depicted in Figure 4.19. The graph shows that as the number of iterations increases, the $APRC_v$ value increases gradually. The growth trend in all three releases of Application A is almost consistent.

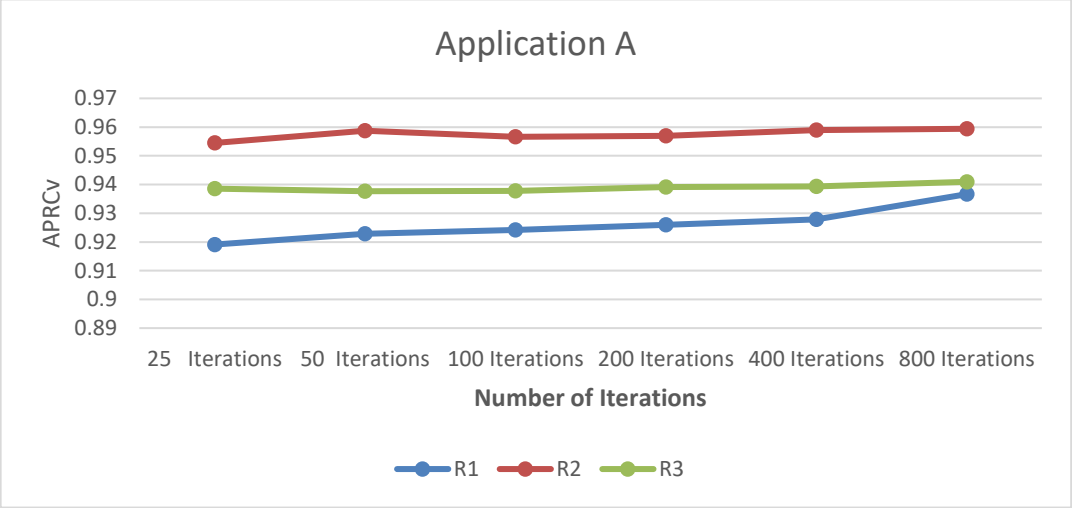


Figure 4.19: $APRC_v$ trend per number of iterations for application A

Table 4.24: $APRC_v$ of value-based TCP using GA for different number of iterations for releases of applications B

Release	Test cases	Requirement	**$APRC_v$ of Value-Based TCP Using GA per Iterations**						**Maximum Execution Time**
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	36	42	0.9334	0.9367	0.9367	0.9381	0.9374	0.9444	51.3133
R2	40	50	0.9424	0.9448	0.9452	0.9471	0.9471	0.9492	83.0596
R3	44	53	0.9372	0.9406	0.9406	0.9420	0.9449	0.9453	70.2029
Average of All Releases			**0.9376**	**0.9407**	**0.9408**	**0.9424**	**0.9432**	**0.9464**	**68.1919**

Similarly, the $APRC_v$ value trend with the different number of iterations for Application B is depicted in Figure 4.20. The graph shows that as the number of iterations increases, the $APRC_v$ value increases gradually. The growth trend in releases R2 and R3 of Application B is consistent. For R1, the $APRC_v$ value declined with 400 iterations and then significantly improved by 800 iterations.

The overall trend of increase in $APRC_v$ value with a greater number of iterations is consistent. The different number of iterations for different releases of Application A and Application B is exercised. With 800 iterations, the results are almost mature.

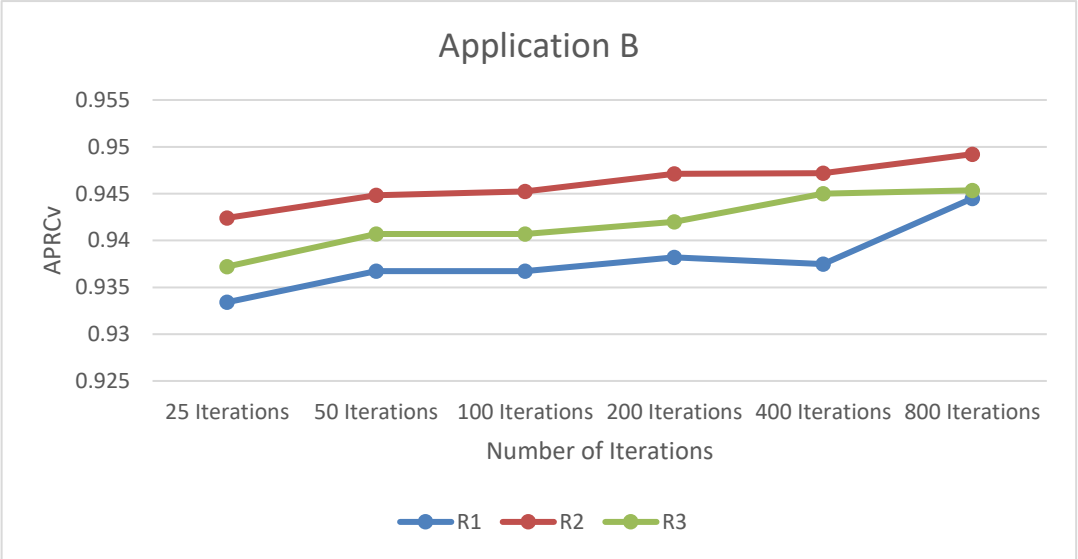


Figure 4.20: $APRC_v$ trend per number of iterations for application B

Tables 4.25 and 4.26 show the execution times of the different numbers of iterations aga inst different releases along with the maximum value of $APRC_v$. The average maximum $APRC_v$ values of Application A and B are **0.9447** and **0.9464,** respectively.

Table 4.25: Execution time of value-based TCP using GA for different numbers of iterations for releases of applications A

Release	Test cases	Requirements	Execution Time of Value-Based TCP Using GA per Iterations						Maximum $APRC_v$
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	41	48	1.2664	2.6299	5.3586	10.4505	20.8860	38.5386	0.9366
R2	44	52	1.8494	4.9856	8.0136	15.1965	29.3816	42.8723	0.9566
R3	48	60	2.7455	3.8984	8.8448	16.4846	29.0515	56.0150	0.9409
Average of All Releases			**1.9538**	**3.8380**	**7.4057**	**14.0439**	**26.4397**	**45.8086**	**0.9447**

Tables 4.25 and 4.26 and Figures 4.21 and 4.22 show that execution time is directly proportional to the number of iterations.

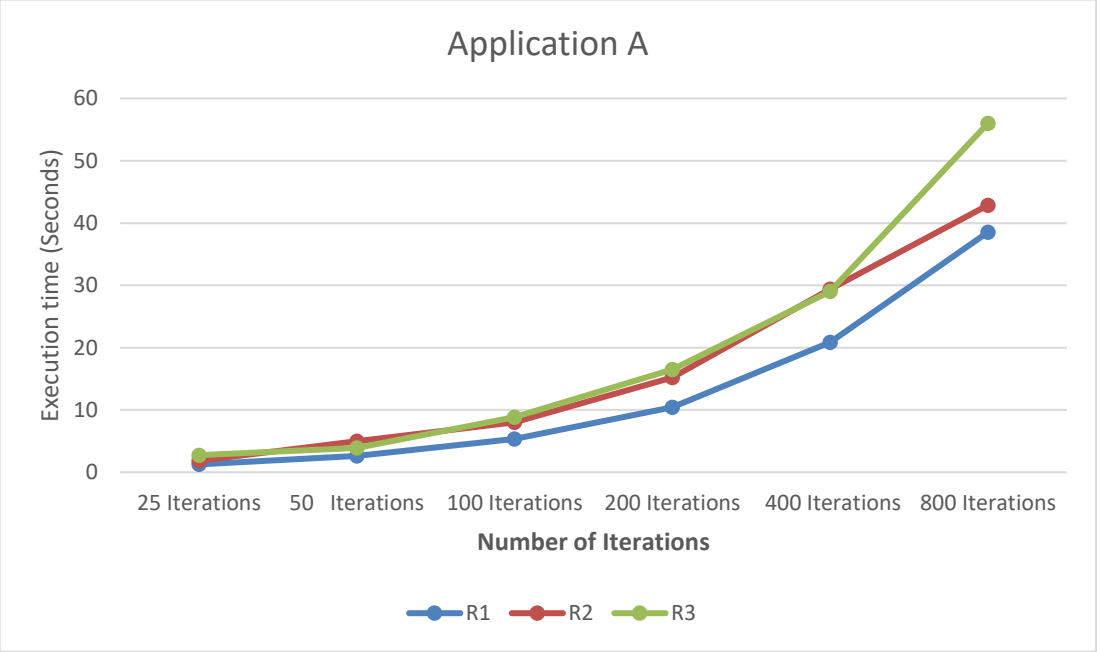


Figure 4.21: Execution time trend per number of iterations for application A

Table 4.26: Execution time of value-based TCP using GA against different numbers of iterations for releases of applications B

Release	Test cases	Requirement	Execution Time of Value-Based TCP Using GA per Iterations						Maximum $APRC_v$
			25 Iterations	50 Iterations	100 Iterations	200 Iterations	400 Iterations	800 Iterations	
R1	36	42	2.5989	3.4699	6.2440	13.7613	26.8584	51.3133	0.9444
R2	40	50	2.6731	6.1350	11.0016	22.1467	42.3335	83.0596	0.9492
R3	44	53	1.5286	2.7596	5.5091	11.7314	21.5678	70.2029	0.9453
Average of All Releases			**2.2668**	**4.1215**	**7.5849**	**15.8798**	**30.2532**	**68.1919**	**0.9463**

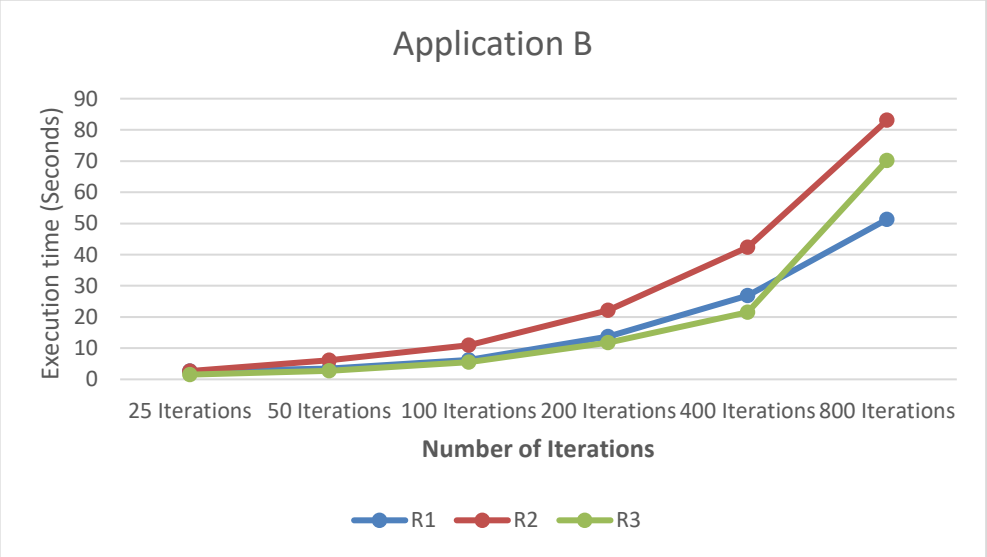


Figure 4.22: Execution time trend per number of iterations for application B

The growth trend in execution time of all three releases of Application A is consistent. Similarly, the growth trend of all releases of Application B is also consistent. The consistency in the execution time of the algorithm makes it reliable.

4.4.2.4. Cost Consumption VS Business Value Coverage

The section describes the cost consumption vs business value coverage trend in graphical form. In Application A, all three releases R1, R2, and R3 show that the initial set of test cases consumed less cost and covered greater business value. Later test cases consumed higher costs and their business value coverage declined. This trend shows that the test cases that are likely to cover higher requirements business value are prioritized first and are depicted in Figures 4.23, 4.24, and 4.25. In Application B, all three releases R1, R2, and R3 generally show that the higher business value coverage ratio cost test cases are prioritized first. This trend is depicted in Figures 4.26, 4.27, and 4.28.

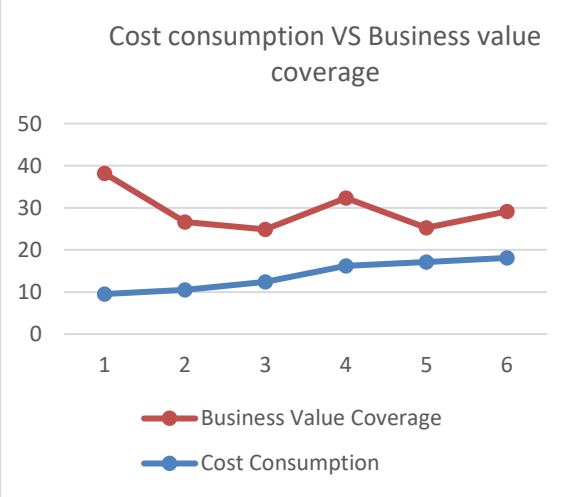


Figure 4.23: Application A – R1

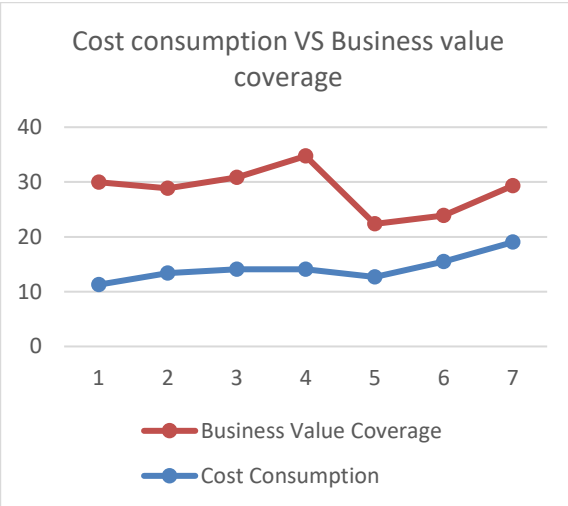


Figure 4.24: Application A – R2

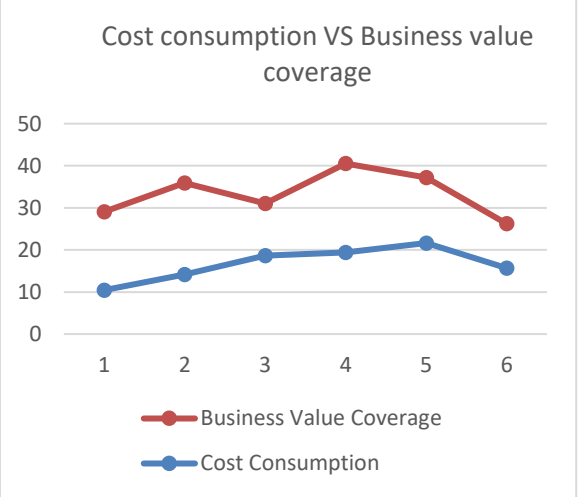


Figure 4.25: Application A – R3

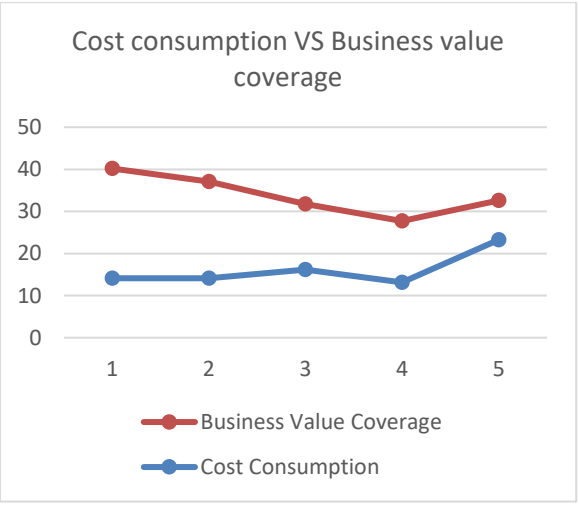


Figure 4.26: Application B – R1

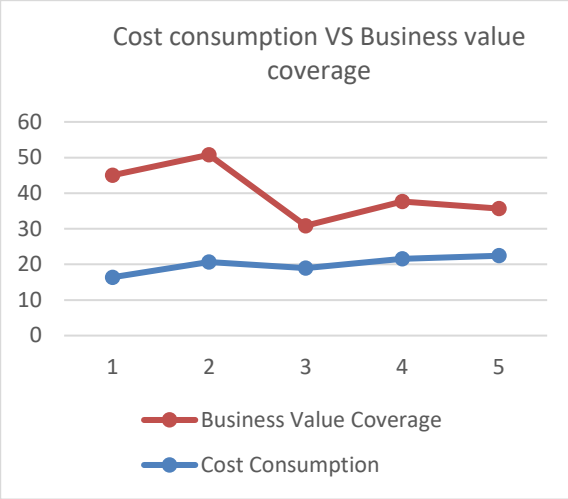


Figure 4.27: Application B- R2

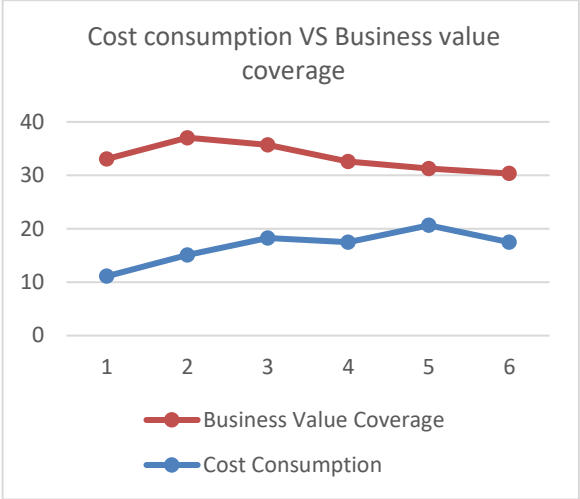


Figure 4.28: Application B- R3

4.4.2.5. Statistical Analysis

To statistically analyze the results of case study 2, we applied a t-test. In our case, we compared the $APRC_v$ mean value of the proposed value-based GA with the other four techniques. The first t-test is applied to the mean value of "Original order" and "Value-Based GA".

Null Hypothesis (H_0): $APRC_v$ value of "Original order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APRC_v$ value of "Original order" and "Value-Based GA" is different.

The mean $APRC_v$ value of the Original Order was 0.9121 and it was 0.946 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned a p-value of 0.010566514. Here, the value of p is less than the value of Alpha therefore null hypothesis H_0 is rejected. It indicated that Value-Based GA is significantly better than Original Order.

The second t-test is applied to the mean value of "Reverse order" and "Value-Based GA".

Null Hypothesis (H_0): $APRC_v$ value of "Reverse order" and "Value-Based GA" is the same.

Alternate Hypothesis (H_1): $APRC_v$ value of "Reverse order" and "Value-Based GA" is

different.

The mean $APRC_v$ value of Revers Order was 0.918283333 and it was 0.946 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned the p-value as 0.002290953. Here, the value of p is less than the value of Alpha therefore null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly better than Reverse Order.

The third t-test is applied to the mean value of "Random order" and "Value-Based GA".

Null Hypothesis (H_0): $APRC_v$ value of "Random order" and "Value-Based GA" are the same.

Alternate Hypothesis (H_1): $APRC_v$ values of "Random order" and "Value-Based GA" are different.

The mean $APRC_v$ value of Random Order was 0.919066667 and it was 0.946 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned p-value of 0.000875866. Here, the value of p is less than the value of Alpha therefore null hypothesis H0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Random Order.

The fourth t-test is applied to the mean value of "Greedy order" and "Value-Based GA".

Null Hypothesis (H_0): $APRC_v$ values of "Greedy order" and "Value-Based GA" are the same.

Alternate Hypothesis (H_1): $APRC_v$ values of "Greedy order" and "Value-Based GA" are different.

The mean $APRC_v$ value of Greedy Order was 0.698166667 and it was 0.946 for Value-Based GA. The level of significance "Alpha" was set as 0.05. The statical test returned p value of 0.000057984631. Here, the value of p is less than the value of Alpha therefore null hypothesis H_0 is rejected. It indicated that the performance of Value-Based GA is significantly different from Greedy Order. The statistical analysis of case study one is given in Table 4.27, 4.28, 4.29, and 4.30.

Table 4.27: Statistical Analysis of Original Order and Value-Based GA

Mean APFDv of Original Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.9121	0.946	0.05	0.010566514	Significant

Table 4.28: Statistical Analysis of Reverse Order and Value-Based GA

Mean APFDv of Reverse Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.918283333	0.946	0.05	0.002290953	Significant

Table 4.29: Statistical Analysis of Random Order and Value-Based GA

Mean APFDv of Random Oder	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.919066667	0.946	0.05	0.000875866	Significant

Table 4.30: Statistical Analysis of Greedy Order and Value-Based GA

Mean APFDv of Greedy Order	**Mean APFDv of Value-Based GA**	**Level of Significance 'α'**	**p-value**	**Difference**
0.698166667	0.946	0.05	0.000057984631	Significant

Considering the level of significance as 0.05, the null hypothesis H_0 is rejected in all four t-tests. Statistically, the performance of the proposed technique is significantly better than the other four techniques.

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Discussion

The TCP is a vital approach for regression testing to meet time and budget constraints. There are two major classes of TCP techniques 1) Value-neutral TCP techniques and 2) Value-based TCP techniques. Both classes have many other categories like coverage-based, history-based, and risk-based. The value-neutral TCP techniques assume that all elements like statements, requirements, test cases, use cases, methods, and bugs are equally important. This assumption sldom holds therefore VN TCP techniques are likely to produce unreliable results. Due to this major limitation of the TCP process, value-based cost-cognizant TCP techniques are gaining popularity.

There was no comprehensive literature review available on VB TCP techniques. In this study, a detailed literature review of VB TCP techniques is performed. The objective is to see the current state of research in this field. The literature review is evident that there is very limited work on value-based test prioritization. It is needed to realize that without value considerations in the TCP process, its intended results cannot be achieved. An enhanced taxonomy of TCP techniques has been devised in this work for further advancements in the value-based cost-cognizant TCP process. This taxonomy yields that there is great potential in value-based cost-cognizant TCP.

From the literature review, it is evident that no quantitative definition of business value is available for estimating test case cost and severity of bugs. All existing definitions are practice-oriented. A business value quantification model is proposed in this work. This model can be helpful to know the notion of business value through its quantitative definition. To make software initiatives aligned with client business success, the business value of different features and quality attributes must be known to IT units. To address this need; the proposed model can help software stakeholders to identify and meet

business expectations. The core objective of this study is to propose a business value quantification mechanism for software requirements both functional and non-functional. Secondly to propose a method to estimate fault severities and test case costs that serves as an input in value-based cost-cognizant TCP for regression testing. We have incorporated business value in the TCP process through this model.

Most of the existing work related to the TCP is done in a value-neutral fashion and has many limitations. The existing techniques are based on the coverage of code components, and it is evident that 100% coverage does not guarantee 100% fault detection. They assume that each fault has an equal cost. Another limitation is that they only considered functional aspects of the application and non-functional aspects have been ignored in the existing work. This research presents a value-based TCP technique (VB-TCP) for value-based regression testing. The major contribution of this work is to re-order test cases for improving the performance of prioritization using business value. It focuses on business value coverage instead of traditional coverage metrics. The performance of the proposed fault detection-based prioritization technique and other comparison techniques is evaluated in terms of $APFD_v$. The performance of the proposed requirements coverage-based prioritization technique and other comparison techniques are evaluated in terms of $APRC_v$. The results show that the use of the value-based approach provides better performance as compared to random, value-based, and dependency-based approaches.

Two value-based TCP techniques and two novel performance evaluation metrics are proposed in this work. The performance of the proposed VCFDB-TCP technique and other comparison techniques is evaluated in terms of $APFD_v$. The performance of the proposed VCRCB-TCP technique and other comparison techniques is evaluated in terms of $APRC_v$. Two case studies are performed for results evaluation. A statistical analysis is performed, and the statistical results show that the use of the value-based approach provides better performance as compared to RO, OO, REVO, and the Greedy approaches.

5.2 Implications of the Study

The proposed business value quantification mechanism can better elaborate the business goals of each software feature. If IT teams are equipped with such a clear insight into the business worth associated with their software initiatives, they can better align their efforts and resources. This can help to eliminate ambiguities while defining quality-related

parameters. This can result in a good client-vendor contract through better perception and understanding of the client's product quality expectations. The proposed value-based model can be applied to address different challenges in the software development life cycle. We applied this model for TCP for value-based regression testing and it produced satisfactory and reliable results. Incorporating business value in the TCP process will prioritize those test cases first which cover overall higher severity of faults and higher business value of requirement. This way testing time and resources will be utilized to cover high business value features or modules of software products. This fact supports the generality and practicality of the findings.

5.3 Research Contribution

- An SLR has been performed on value-based cost-cognizant TCP techniques. Existing TCP techniques have been analyzed in terms of the algorithm used, performance metric used, result validation method adopted, and open research problems. An enhanced taxonomy of TCP techniques has been devised after a comprehensive literature review for further advancements in the value-based TCP process. This contribution addressed our RQ1.
- A Business Value Quantification Model has been proposed for the measurement of the business value of software requirements (functional/non-functional) by using five business success factors including profitability, productivity, operational efficiency, client satisfaction, and time to market. A mechanism has been proposed to estimate the severity of faults and cost of test cases based on the business value of requirements for value-based test case prioritization. This contribution has answered our RQ2.
- Two value-based cost-cognizant TCP techniques for regression testing using GA have been proposed. These techniques include Value-Cognizant Fault Detection-Based TCP (VCFDB-TCP) and Value-Cognizant Requirements Coverage-Based TCP (VCRCB-TCP). This contribution has addressed our RQ3.
- Two novel value-based performance evaluation metrics are also introduced for value-based TCP techniques including the $APFD_v$ and $APRC_v$. $APFD_v$ deals with varying test case costs and fault severity whereas $APRC_v$ deals with varying test

case costs and requirements for business value. RQ4 has been answered through this contribution.

5.4 Threats to Validity

In this section, we identified a few known threats to the validity of this study's results.

Construct Validity Threat:

Our defined research questions may not include all aspects of value-based cost-cognitive TCP techniques. We addressed it through discussions. We believe that our research questions are well-designed and mapped with the goals of the study.

Internal Validity Threats:

Ensuring perfection in the data collection process is a difficult task. We cannot guarantee that our data collection is complete. Imperfect data collection can be a threat to the validity of the literature review. We carefully selected our search keywords to fetch more relevant studies from the research repositories. Our paper search was limited to a few prominent research repositories. There might be more relevant publications available in other search repositories. To minimize this problem, we utilized those research repositories that were utilized by previous literature reviews of TCP techniques. Validation of the study relevancy evaluation process is also a major threat to any literature review. To address this issue, an independent reviewer also evaluated the selected studies' relevance. The second author (supervisor) played this role as an independent reviewer. The data extraction process may be imprecise, and this may affect the validity of this research. This is due to the unsystematic data extraction process. To reduce this risk, we applied manual data extraction through expert judgment. The study assumes that one unit of severity is equivalent to one unit of test case cost or execution time. Similarly, one unit of the business value of a requirement is equivalent to one unit of the test case time to cover it.

External Validity Threat:

This study assumes that the business value of requirements and test case execution time is already known in the case of VCRCB-TCP. Similarly, it assumes that the severity of faults and test case cost is already known in the case of VCFDB-TCP. In this study, it is considered that both test case cost and fault severity are equally important. However, in

some scenarios, there might be a tradeoff between cost and severity. Similarly, there might be a tradeoff between test case cost and the business value of requirements. Thirdly the datasets used for the study are of smaller size and are collected against different products/applications from a single company. The results may vary with the variety of other software applications developed by different software development organizations.

5.5 Future Work

Value-based TCP still has many dimensions to be investigated in the future. A few future directions include the performance evaluation of VB TCP techniques and the development of novel performance evaluation standard metrics for it.

Value orientation can be applied to different coverage-based TCP techniques like it can be applied to statement coverage, branch coverage, function coverage, or any other element coverage.

The right metric selection for the performance evaluation of TCP techniques is essential to get reliable results. Popularity-based metric selection is not a valid justification, and it cannot produce reliable results. This is a big area for further improvement. The efficiency and effectiveness of TCP approaches are strongly dependent on the correct evaluation metric because a researcher usually targets an improvement in a metric value while proposing a TCP technique.

The value-cognizant performance evaluation metrics can be derived as the Average Percentage of Branch Coverage per value ($APBC_v$), Average Percentage of Loop Coverage per Value ($APLC_v$), or Average Percentage of Function Coverage per value ($APFC_v$. It can bring a shift from value-neutral TCP to value-based TCP.

The proposed business value quantification model can be validated for different dimensions of the software development life cycle. We believe cost and time-to-market for any software product are the most important factors. Software budget and time should be utilized in a value-based manner for software activities. This work can provide a base for other SDLC phases to be considered in a value context.

BIBLIOGRAPHY

[1] S. R. Faulk, R. R. Harmon, and D. M. Raffo, “Value-Based Software Engineering (VBSE),” in Software Product Lines, P. Donohoe, Ed., Boston, MA: Springer US, 2000, pp. 205–223. DOI: 10.1007/978-1-4615-4339-8_12.

[2] D. Zhang, “Machine Learning in Value-Based Software Test Data Generation,” in 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’06), Nov. 2006, pp. 732–736. DOI: 10.1109/ICTAI.2006.77.

[3] B. W. Boehm, “Value-Based Software Engineering: Overview and Agenda,” in Value-Based Software Engineering, S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 3–14. DOI: 10.1007/3-540-29263-2_1.

[4] T. Dingsøyr and C. Lassenius, “Emerging themes in agile software development: Introduction to the special section on continuous value delivery,” Inf. Softw. Technol., vol. 77, pp. 56–60, 2016.

[5] Srikanth H. and Williams L. “On the economics of requirements-based test case prioritization,” ACM SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, pp. 1–3, Jul. 2005, https://DOI.org/10.1145/1082983.1083100

[6] D. Saff and M. D. Ernst, “Reducing wasted development time via continuous testing,” in 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003., Nov. 2003, pp. 281–292. DOI: 10.1109/ISSRE.2003.1251050.

[7] X. Zhang, C. G. Onita, and J. S. Dhaliwal, “The impact of software testing governance choices,” J. Organ. End User Comput. JOEUC, vol. 26, no. 1, pp. 66–85, 2014.

[8] R. Ramler, S. Biffl, and P. Grünbacher, “Value-Based Management of Software Testing,” in Value-Based Software Engineering, S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 225–244. DOI: 10.1007/3-540-29263-2_11.

[9] R. Ramler, T. Kopetzky, and W. Platz, “Value-based coverage measurement in requirements-based testing: Lessons learned from an approach implemented in the tosca test suite,” in 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, IEEE, 2012, pp. 363–366.

[10] B. Boehm, “Value-Based Software Engineering,” ACM SIGSOFT, vol. 28, no. 2, p. 12.

[11] S. Biswas, R. Mall, M. Satpathy, and S. Sukumaran, “Regression test selection techniques: A survey,” Informatica, vol. 35, no. 3, 2011.

[12] M. R. N. Dobuneh, D. N. A. Jawawi, and M. V. Malakooti, “Web Application Regression Testing: A Session Based Test Case Prioritization Approach,” in The International Conference on Digital Information Processing, E-Business and Cloud Computing (DIPECC), Society of Digital Information and Wireless Communication, 2013, p. 107.

[13] A. Orso and G. Rothermel, “Software testing: a research travelogue (2000–2014),” in Proceedings of the on Future of Software Engineering, ACM, 2014, pp. 117–132.

[14] N. Gupta, A. Sharma, and M. K. Pachariya, “An Insight Into Test Case Optimization: Ideas and Trends With Future Perspectives,” IEEE Access, vol. 7, pp. 22310–22327, 2019, DOI: 10.1109/ACCESS.2019.2899471.

[15] G. Rothermel, R. H. Untch, and M. J. Harrold, “Prioritizing Test Cases For Regression Testing,” IEEE Trans. Softw. Eng., vol. 27, no. 10, p. 20, 2001.

[16] Z. Sultan, R. Abbas, S. N. Bhatti, and S. A. A. Shah, “Analytical Review on Test Cases Prioritization Techniques: An Empirical Study,” Int. J. Adv. Comput. Sci. Appl. IJACSA, vol. 8, no. 2, 2017.

[17] E. Ashraf, A. Rauf, and K. Mahmood, “Value based Regression Test Case Prioritization,” p. 5, 2012.

[18] B. Boehm and L. G. Huang, “Value-based software engineering: A case study,” Computer, vol. 36, no. 3, pp. 33–41, 2003.

[19] S. Elbaum, A. Malishevsky, and G. Rothermel, “Incorporating varying test costs and fault severities into test case prioritization,” in Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, Toronto, Ont., Canada: IEEE Comput. Soc, 2001, pp. 329–338. DOI: 10.1109/ICSE.2001.919106.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Sensing as a service model for smart cities supported by Internet of Things,” Trans. Emerg. Telecommun. Technol., vol. 25, no. 1, pp. 81–93, 2014, DOI: 10.1002/ett.2704.

[21] V. Mandić, V. Basili, L. Harjumaa, M. Oivo, and J. Markkula, “Utilizing GQM+ Strategies for business value analysis: An approach for evaluating business goals,” in Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, 2010, p. 20.

[22] Z. Racheva, M. Daneva, and K. Sikkel, “Value Creation by Agile Projects: Methodology or Mystery?,” in Product-Focused Software Process Improvement, vol. 32, F. Bomarius,

M. Oivo, P. Jaring, and P. Abrahamsson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 141–155. DOI: 10.1007/978-3-642-02152-7_12.

[23] A. F. Minkiewicz, "The evolution of software size: A search for value," Softw. Eng. Technol., pp. 23–26, 2009.

[24] H. Jahan, Z. Feng, and S. M. H. Mahmud, "Risk-Based Test Case Prioritization by Correlating System Methods and Their Associated Risks," Arab. J. Sci. Eng., Apr. 2020, DOI: 10.1007/s13369-020-04472-z.

[25] R. Huang, Q. Zhang, T. Y. Chen, J. Hamlyn-Harris, D. Towey, and J. Chen, "An Empirical Comparison of Fixed-Strength and Mixed-Strength for Interaction Coverage Based Prioritization," IEEE Access, vol. 6, pp. 68350–68372, 2018, DOI: 10.1109/ACCESS.2018.2879638.

[26] A. Bajaj and O. P. Sangwan, "A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms," IEEE Access, vol. 7, pp. 126355–126375, 2019, DOI: 10.1109/ACCESS.2019.2938260.

[27] B. Miranda and A. Bertolino, "An assessment of operational coverage as both an adequacy and a selection criterion for operational profile based testing," Softw. Qual. J., vol. 26, no. 4, pp. 1571–1594, Dec. 2018, DOI: 10.1007/s11219-017-9388-0.

[28] G. Luque and E. Alba, Parallel Genetic Algorithms: Theory and Real World Applications. Springer, 2011.

[29] H. Krasner, "The Cost of Poor Software Quality in the US: A 2020 Report," p. 46, 2020.

[30] "11 of the most costly software errors in history," Raygun Blog. Accessed: Feb. 02, 2022. [Online]. Available: https://raygun.com/blog/costly-software-errors-history/

[31] F. S. Ahmed, A. Majeed, T. A. Khan, and S. N. Bhatti, "Value-based cost-cognizant test case prioritization for regression testing," PLOS ONE, vol. 17, no. 5, p. e0264972, May 2022, DOI: 10.1371/journal.pone.0264972.

[32] I. Hooda and R. S. Chhillar, "Software test process, testing types and techniques," Int. J. Comput. Appl., vol. 111, no. 13, 2015, Accessed: Oct. 12, 2023. DOI=0fbe1b5515e747025d950658fbc039e98b29b801

[33] J. A. Prado Lima and S. R. Vergilio, "Test Case Prioritization in Continuous Integration environments: A systematic mapping study," Inf. Softw. Technol., vol. 121, p. 106268, May 2020, DOI: 10.1016/j.infsof.2020.106268.

[34] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann, “Test Generation and Test Prioritization for Simulink Models with Dynamic Behavior,” IEEE Trans. Softw. Eng., vol. 45, no. 9, pp. 919–944, Sep. 2019, DOI: 10.1109/TSE.2018.2811489.

[35] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. Mohamed Suffian, “Test Case Prioritization Using Firefly Algorithm for Software Testing,” IEEE Access, vol. 7, pp. 132360–132373, 2019, DOI: 10.1109/ACCESS.2019.2940620.

[36] S. Tahvili, R. Pimentel, W. Afzal, M. Ahlberg, E. Fornander, and M. Bohlin, “sOrTES: A Supportive Tool for Stochastic Scheduling of Manual Integration Test Cases,” IEEE Access, vol. 7, pp. 12928–12946, 2019, DOI: 10.1109/ACCESS.2019.2893209.

[37] R. Mukherjee and K. S. Patnaik, “Prioritizing JUnit Test Cases Without Coverage Information: An Optimization Heuristics Based Approach,” IEEE Access, vol. 7, pp. 78092–78107, 2019, DOI: 10.1109/ACCESS.2019.2922387.

[38] C. Lu, J. Zhong, Y. Xue, L. Feng, and J. Zhang, “Ant Colony System With Sorting-Based Local Search for Coverage-Based Test Case Prioritization,” IEEE Trans. Reliab., pp. 1–17, 2019, DOI: 10.1109/TR.2019.2930358.

[39] Md. Abdur, Md. Abu, and Md. Saeed, “Prioritizing Dissimilar Test Cases in Regression Testing using Historical Failure Data,” Int. J. Comput. Appl., vol. 180, no. 14, pp. 1–8, Jan. 2018, DOI: 10.5120/ijca2018916258.

[40] D. Hao, L. Zhang, L. Zang, Y. Wang, X. Wu, and T. Xie, “To be optimal or not in test-case prioritization,” IEEE Trans. Softw. Eng., vol. 42, no. 5, pp. 490–505, 2016.

[41] Y. Bian, Z. Li, R. Zhao, and D. Gong, “Epistasis Based ACO for Regression Test Case Prioritization,” IEEE Trans. Emerg. Top. Comput. Intell., vol. 1, no. 3, pp. 213–223, Jun. 2017, DOI: 10.1109/TETCI.2017.2699228.

[42] S. Eghbali and L. Tahvildari, “Test Case Prioritization Using Lexicographical Ordering,” IEEE Trans. Softw. Eng., vol. 42, no. 12, pp. 1178–1195, Dec. 2016, DOI: 10.1109/TSE.2016.2550441.

[43] A. Marchetto, Md. M. Islam, W. Asghar, A. Susi, and G. Scanniello, “A Multi-Objective Technique to Prioritize Test Cases,” IEEE Trans. Softw. Eng., vol. 42, no. 10, pp. 918–940, Oct. 2016, DOI: 10.1109/TSE.2015.2510633.

[44] Md. Abu Hasan, Md. Abdur Rahman, and Md. Saeed Siddik, “Test Case Prioritization Based on Dissimilarity Clustering Using Historical Data Analysis,” in Information, Communication and Computing Technology, vol. 750, S. Kaushik, D. Gupta, L. Kharb,

and D. Chahal, Eds., Singapore: Springer Singapore, 2017, pp. 269–281. DOI: 10.1007/978-981-10-6544-6_25.

[45] H. Mei, D. Hao, L. Zhang, L. Zhang, J. Zhou, and G. Rothermel, “A Static Approach to Prioritizing JUnit Test Cases,” IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1258–1275, Nov. 2012, DOI: 10.1109/TSE.2011.106.

[46] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, “The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments,” IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 593–617, Sep. 2010, DOI: 10.1109/TSE.2010.58.

[47] Z. Li, M. Harman, and R. M. Hierons, “Search Algorithms for Regression Test Case Prioritization,” IEEE Trans. Softw. Eng., vol. 33, no. 4, pp. 225–237, Apr. 2007, DOI: 10.1109/TSE.2007.38.

[48] N. Chen and S. Kim, “Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles,” in Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ACM, 2012, pp. 140–149.

[49] R. Wu, H. Zhang, S.-C. Cheung, and S. Kim, “CrashLocator: locating crashing faults based on crash stacks,” in Proceedings of the 2014 International Symposium on Software Testing and Analysis, ACM, 2014, pp. 204–214.

[50] H. Do and G. Rothermel, “A controlled experiment assessing test case prioritization techniques via mutation faults,” in Software Maintenance, 2005. ICSM’05. Proceedings of the 21st IEEE International Conference on, IEEE, 2005, pp. 411–420.

[51] H. Stallbaum, A. Metzger, and K. Pohl, “An automated technique for risk-based test case generation and prioritization,” in Proceedings of the 3rd international workshop on Automation of software test, ACM, 2008, pp. 67–70.

[52] G. Chaurasia, S. Agarwal, and S. S. Gautam, “Clustering based novel test case prioritization technique,” in Engineering and Systems (SCES), 2015 IEEE Students Conference on, IEEE, 2015, pp. 1–5.

[53] H. Kumar and N. Chauhan, “A Coupling effect based test case prioritization technique,” in Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, IEEE, 2015, pp. 1341–1345.

[54] M. R. N. Dobuneh, D. N. Jawawi, M. Ghazali, and M. V. Malakooti, “Development test case prioritization technique in regression testing based on hybrid criteria,” in Software Engineering Conference (MySEC), 2014 8th Malaysian, IEEE, 2014, pp. 301–305.

[55] B. Jiang and W. K. Chan, “Input-based adaptive randomized test case prioritization: A local beam search approach,” J. Syst. Softw., vol. 105, pp. 91–106, Jul. 2015, DOI: 10.1016/j.jss.2015.03.066.

[56] B. Miranda, “FAST Approaches to Scalable Similarity-Based Test Case Prioritization,” p. 11, 2018.

[57] A. Nanda, S. Mani, S. Sinha, M. J. Harrold, and A. Orso, “Regression testing in the presence of non-code changes,” in Verification and Validation 2011 Fourth IEEE International Conference on Software Testing, Mar. 2011, pp. 21–30. DOI: 10.1109/ICST.2011.60.

[58] “An empirical study on clustering approach combining fault prediction for test case prioritization,” in 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), May 2017, pp. 815–820. DOI: 10.1109/ICIS.2017.7960104.

[59] P. Konsaard and L. Ramingwong, “Total coverage based regression test case prioritization using genetic algorithm,” in 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Jun. 2015, pp. 1–6. DOI: 10.1109/ECTICon.2015.7207103.

[60] T. B. Noor and H. Hemmati, “Test case analytics: Mining test case traces to improve risk-driven testing,” in 2015 IEEE 1st International Workshop on Software Analytics (SWAN), Mar. 2015, pp. 13–16. DOI: 10.1109/SWAN.2015.7070482.

[61] D. Marijan, M. Liaaen, A. Gotlieb, S. Sen, and C. Ieva, “TITAN: Test Suite Optimization for Highly Configurable Software,” in 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), Mar. 2017, pp. 524–531. DOI: 10.1109/ICST.2017.60.

[62] R. Lachmann, “12.4 - Machine Learning-Driven Test Case Prioritization Approaches for Black-Box Software Testing,” AMA Serv. GmbH Von-Münchhausen-Str 49 31515 Wunstorf Ger., 2018, DOI: 10.5162/ettc2018/12.4.

[63] A. Ansari, A. Khan, A. Khan, and K. Mukadam, “Optimized Regression Test Using Test Case Prioritization,” Procedia Comput. Sci., vol. 79, pp. 152–160, 2016, DOI: 10.1016/j.procs.2016.03.020.

[64] M. M. Öztürk, “A bat-inspired algorithm for prioritizing test cases,” Vietnam J. Comput. Sci., vol. 5, no. 1, pp. 45–57, Feb. 2018, DOI: 10.1007/s40595-017-0100-x.

[65] D. Marijan, “Multi-perspective Regression Test Prioritization for Time-Constrained Environments,” in 2015 IEEE International Conference on Software Quality, Reliability and Security, Aug. 2015, pp. 157–162. DOI: 10.1109/QRS.2015.31.

[66] S. Wang, J. Nam, and L. Tan, “QTEP: quality-aware test case prioritization,” in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017, Paderborn, Germany: ACM Press, 2017, pp. 523–534. DOI: 10.1145/3106237.3106258.

[67] M. Aggarwal and S. Sabharwal, “Combinatorial Test Set Prioritization Using Data Flow Techniques,” Arab. J. Sci. Eng., vol. 43, no. 2, pp. 483–497, Feb. 2018, DOI: 10.1007/s13369-017-2631-y.

[68] E. Ashraf, K. Mahmood, T. Ahmed, and S. Ahmed, “Value based PSO Test Case Prioritization Algorithm,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 1, 2017, DOI: 10.14569/IJACSA.2017.080149.

[69] M. Bagherzadeh, N. Kahani, and L. Briand, “Reinforcement Learning for Test Case Prioritization,” IEEE Trans. Softw. Eng., pp. 1–1, 2021, DOI: 10.1109/TSE.2021.3070549.

[70] Q. Luo, K. Moran, D. Poshyvanyk, and M. Di Penta, “Assessing Test Case Prioritization on Real Faults and Mutants,” ArXiv180708823 Cs, Sep. 2018, Accessed: Dec. 15, 2019. [Online]. Available: http://arxiv.org/abs/1807.08823

[71] A. Askarunisa, L. Shanmugapriya, and D. N. Ramaraj, “Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques,” INFOCOMP Journal of Computer Science, Vol. 9 No. 1, p. 10, March, 2010

[72] X. Wang and H. Zeng, “History-based Dynamic Test Case Prioritization for Requirement Properties in Regression Testing,” in Proceedings of the International Workshop on Continuous Software Evolution and Delivery, in CSED ’16. New York, NY, USA: ACM, 2016, pp. 41–47. DOI: 10.1145/2896941.2896949.

[73] C.-T. Lin, C.-D. Chen, C.-S. Tsai, and G. M. Kapfhammer, “History-Based Test Case Prioritization with Software Version Awareness,” in 2013 18th International Conference on Engineering of Complex Computer Systems, Jul. 2013, pp. 171–172. DOI: 10.1109/ICECCS.2013.33.

[74] D. Marijan, “Multi-perspective Regression Test Prioritization for Time-Constrained Environments,” in 2015 IEEE International Conference on Software Quality, Reliability and Security, Vancouver, BC, Canada: IEEE, Aug. 2015, pp. 157–162. DOI: 10.1109/QRS.2015.31.

[75] T. Noor and H. Hemmati, “Test case analytics: Mining test case traces to improve risk-driven testing,” in 2015 IEEE 1st International Workshop on Software Analytics (SWAN), Montreal, QC, Canada: IEEE, Mar. 2015, pp. 13–16. DOI: 10.1109/SWAN.2015.7070482.

[76] B. Busjaeger and T. Xie, “Learning for test prioritization: an industrial case study,” in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016, Seattle, WA, USA: ACM Press, 2016, pp. 975–980. DOI: 10.1145/2950290.2983954.

[77] L. Xiao, H. Miao, W. Zhuang, and S. Chen. “An Empirical Study on Clustering Approach Combining Fault Prediction for Test Case Prioritization", IEEE, ICIS 2017, May 24–26, 2017, Wuhan, China.

[78] J. A. do Prado Lima and S. R. Vergilio, “A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments,” IEEE Trans. Softw. Eng., pp. 1–1, 2020, DOI: 10.1109/TSE.2020.2992428.

[79] Z. Q. Zhou, C. Liu, T. Y. Chen, T. H. Tse, and W. Susilo, “Beating Random Test Case Prioritization,” IEEE Trans. Reliab., pp. 1–22, 2020, DOI: 10.1109/TR.2020.2979815.

[80] M. L. Mohd-Shafie, W. M. N. Wan-Kadir, M. Khatibsyarbini, and M. A. Isa, “Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion,” PLOS ONE, vol. 15, no. 2, p. e0229312, Feb. 2020, DOI: 10.1371/journal.pone.0229312.

[81] Y. Venugopal, P. Quang-Ngoc, and L. Eunseok, “Modification Point Aware Test Prioritization and Sampling to Improve Patch Validation in Automatic Program Repair,” Appl. Sci., vol. 10, no. 5, p. 1593, Feb. 2020, DOI: 10.3390/app10051593.

[82] H. Wang, M. Yang, L. Jiang, J. Xing, Q. Yang, and F. Yan, “Test Case Prioritization for Service-Oriented Workflow Applications: A Perspective of Modification Impact Analysis,” IEEE Access, vol. 8, pp. 101260–101273, 2020, DOI: 10.1109/ACCESS.2020.2998545.

[83] Saqib Iqbal, Issam Al-Azzoni , “Test case prioritization for model transformations | Elsevier Enhanced Reader.”, Volume 34, Issue 8, Part B, September 2022, Pages 6324-6338, https://DOI.org/10.1016/j.jksuci.2021.08.011

[84] R. Cheng, L. Zhang, D. Marinov, and T. Xu, “Test-case prioritization for configuration testing,” in Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Denmark: ACM, Jul. 2021, pp. 452–465. DOI: 10.1145/3460319.3464810.

[85] C. Catal and D. Mishra, “Test case prioritization: a systematic mapping study,” Softw. Qual. J., vol. 21, no. 3, pp. 445–478, Sep. 2013, DOI: 10.1007/s11219-012-9181-z.

[86] Y.-C. Huang, K.-L. Peng, and C.-Y. Huang, “A history-based cost-cognizant test case prioritization technique in regression testing,” J. Syst. Softw., vol. 85, no. 3, pp. 626–637, Mar. 2012, DOI: 10.1016/j.jss.2011.09.063.

[87] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, “Cost-cognizant Test Case Prioritization,”, Technical Report TR-UNL-CSE-2006-0004, p. 41, Department of Computer Science and Engineering, University of Nebraska–Lincoln, Lincoln, Nebraska, U.S.A., 12 March 2006

[88] H. Park, H. Ryu, and J. Baik, “Historical Value-Based Approach for Cost-Cognizant Test Case Prioritization to Improve the Effectiveness of Regression Testing,” in 2008 Second International Conference on Secure System Integration and Reliability Improvement, Jul. 2008, pp. 39–46. DOI: 10.1109/SSIRI.2008.52.

[89] B. Kitchenham, “Procedures for Performing Systematic Reviews,” p. 33, Keele, UK, Keele University, 2004, Citeseer

[90] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering – A systematic literature review,” Inf. Softw. Technol., vol. 51, no. 1, pp. 7–15, Jan. 2009, DOI: 10.1016/j.infsof.2008.09.009.

[91] M. Khatibsyarbini, M. A. Isa, D. N. Jawawi, and R. Tumeng, “Test case prioritization approaches in regression testing: A systematic literature review,” Inf. Softw. Technol., vol. 93, pp. 74–93, 2018.

[92] R. Mukherjee and K. S. Patnaik, “A survey on different approaches for software test case prioritization,” J. King Saud Univ. - Comput. Inf. Sci., p. S1319157818303616, Oct. 2018, DOI: 10.1016/j.jksuci.2018.09.005.

[93] J. Ahmad and S. Baharom, “A Systematic Literature Review of the Test Case Prioritization Technique for Sequence of Events,” vol. 12, no. 7, p. 7, 2017.

[94] H. de S. Campos Junior, M. A. P. Araújo, J. M. N. David, R. Braga, F. Campos, and V. Ströele, “Test case prioritization: a systematic review and mapping of the literature,” in Proceedings of the 31st Brazilian Symposium on Software Engineering - SBES’17, Fortaleza, CE, Brazil: ACM Press, 2017, pp. 34–43. DOI: 10.1145/3131151.3131170.

[95] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” Softw. Test. Verification Reliab., vol. 22, no. 2, pp. 67–120, 2012.

[96] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, and for the PRISMA Group, “Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement,” BMJ, vol. 339, no. jul21 1, pp. b2535–b2535, Jul. 2009, DOI: 10.1136/bmj.b2535.

[97] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” J. Syst. Softw., vol. 80, no. 4, pp. 571–583, Apr. 2007, DOI: 10.1016/j.jss.2006.07.009.

[98] Z. Yu, F. M. Fahid, T. Menzies, G. Rothermel, K. Patrick, and S. Cherian, “TERMINATOR: Better Automated UI Test Case Prioritization,” Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. - ESECFSE 2019, pp. 883–894, 2019, DOI: 10.1145/3338906.3340448.

[99] B. Miranda and A. Bertolino, “Scope-aided test prioritization, selection and minimization for software reuse,” J. Syst. Softw., vol. 131, pp. 528–549, Sep. 2017, DOI: 10.1016/j.jss.2016.06.058.

[100] Y. Wang, X. Zhao, and X. Ding, “An effective test case prioritization method based on fault severity,” in 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China: IEEE, Sep. 2015, pp. 737–741. DOI: 10.1109/ICSESS.2015.7339162.

[101] M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke, “Empirical evaluation of pareto efficient multi-objective regression test case prioritisation,” in Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015, Baltimore, MD, USA: ACM Press, 2015, pp. 234–245. DOI: 10.1145/2771783.2771788.

[102] A. Rauf and A. I. AlSalem, “Intelligent Web Application Systems Testing through Value Based Test Case Prioritization,” in Progress in Systems Engineering, vol. 366, H. Selvaraj, D. Zydek, and G. Chmaj, Eds., in Advances in Intelligent Systems and Computing, vol. 366. , Cham: Springer International Publishing, 2015, pp. 765–768. DOI: 10.1007/978-3-319-08422-0_110.

[103] B. Hoq, S. Jafrin, and S. Hosain, “Dependency Cognizant Test Case Prioritization,” p. 5, Department of Electrical Engineering and Computer Science North South University, Dhaka, Bangladesh , 2011.

[104] Q. Li and B. Boehm, “Improving scenario testing process by adding value-based prioritization: an industrial case study,” in Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013, San Francisco, CA, USA: ACM Press, 2013, p. 78. DOI: 10.1145/2486046.2486061.

[105] D. Marijan, A. Gotlieb, and S. Sen, “Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study,” in 2013 IEEE International Conference on Software Maintenance, Eindhoven, Netherlands: IEEE, Sep. 2013, pp. 540–543. DOI: 10.1109/ICSM.2013.91.

[106] X. Zhang and B. Qu, “An Improved Metric for Test Case Prioritization,” in 2011 Eighth Web Information Systems and Applications Conference, Chongqing, China: IEEE, Oct. 2011, pp. 125–130. DOI: 10.1109/WISA.2011.31.

[107] R. C. Bryce, S. Sampath, J. B. Pedersen, and S. Manchester, “Test suite prioritization by cost-based combinatorial interaction coverage,” Int. J. Syst. Assur. Eng. Manag., vol. 2, no. 2, pp. 126–134, Jun. 2011, DOI: 10.1007/s13198-011-0067-4.

[108] X. Zhang, C. Nie, B. Xu, and B. Qu, “Test Case Prioritization Based on Varying Testing Requirement Priorities and Test Case Costs,” in Seventh International Conference on Quality Software (QSIC 2007), Oct. 2007, pp. 15–24. DOI: 10.1109/QSIC.2007.4385476.

[109] H. Srikanth, L. Williams, and J. Osborne, “System test case prioritization of new and regression test cases,” in 2005 International Symposium on Empirical Software Engineering, 2005., Nov. 2005, p. 10 pp.-. DOI: 10.1109/ISESE.2005.1541815.

[110] H. Srikanth and L. Williams, “On the economics of requirements-based test case prioritization,” ACM SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, pp. 1–3, Jul. 2005, DOI: 10.1145/1082983.1083100.

[111] S. Elbaum, A. Malishevsky, and G. Rothermel, “Incorporating varying test costs and fault severities into test case prioritization,” in Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, Toronto, Ont., Canada: IEEE Comput. Soc, 2001, pp. 329–338. DOI: 10.1109/ICSE.2001.919106.

[112] S. Elbaum, A. G. Malishevsky, and G. Rothermel, “Prioritizing Test Cases for Regression Testing,” in Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis, in ISSTA ’00. New York, NY, USA: ACM, 2000, pp. 102–112. DOI: 10.1145/347324.348910.

[113] X. Wang, H. Zeng, H. Gao, H. Miao, and W. Lin, “Location-Based Test Case Prioritization for Software Embedded in Mobile Devices Using the Law of Gravitation,” Mob. Inf. Syst., vol. 2019, pp. 1–14, Jan. 2019, DOI: 10.1155/2019/9083956.

[114] A. Jatain and G. Sharma, “A Systematic Review of Techniques for Test Case Prioritization,” Int. J. Comput. Appl., vol. 68, no. 2, pp. 38–42, Apr. 2013, DOI: 10.5120/11554-6833.

[115] G. L. Geissler, “Building customer relationships online: the Web site designers’ perspective,” J. Consum. Mark., vol. 18, no. 6, pp. 488–502, Jan. 2001, DOI: 10.1108/EUM0000000006154.

[116] J. Azar, R. Smith, and D. Cordes, “Value-Oriented Requirements Prioritization in a Small Development Organization,” IEEE Softw., vol. 24, no. 1, pp. 32–37, Jan. 2007, DOI: 10.1109/MS.2007.30.

[117] K. Mossakowska and A. Jarzębowicz, “A Survey Investigating the Influence of Business Analysis Techniques on Software Quality Characteristics,” in Towards a Synergistic Combination of Research and Practice in Software Engineering, in Studies in Computational Intelligence. , Springer, Cham, 2018, pp. 135–148. DOI: 10.1007/978-3-319-65208-5_10.

[118] S. Raju, “Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm,”, European Journal of Scientific Research ISSN 1450-216X Vol.74 No.3 (2012), pp. 389-402

[119] G. Issac, C. Rajendran, and R. N. Anantharaman, “An instrument for the measurement of customer perceptions of quality management in the software industry: An empirical study in India,” Softw. Qual. J., vol. 14, no. 4, pp. 291–308, 2006.

[120] J. Offutt, “Quality attributes of web software applications,” IEEE Softw., vol. 19, no. 2, pp. 25–32, 2002.

[121] N. Iqbal, W. Nadeem, and A. Zaheer, “Impact of BPR critical success factors on inter-organizational functions: an empirical study,” Ment Rev., vol. 6, no. 1, p. 152, 2015.

[122] A. Stefani and M. Xenos, “E-commerce system quality assessment using a model based on ISO 9126 and Belief Networks,” Softw. Qual. J., vol. 16, no. 1, pp. 107–129, Mar. 2008, DOI: 10.1007/s11219-007-9032-5.

[123] H.-W. Jung, S.-G. Kim, and C.-S. Chung, “Measuring software product quality: A survey of ISO/IEC 9126,” IEEE Softw., vol. 21, no. 5, pp. 88–92, 2004.

[124] R. A. Asaka, G. H. S. Mendes, and G. M. D. Ganga, “Factors Influencing Customer Satisfaction in Software as a Service (SaaS): Proposal of a System of Performance Indicators,” IEEE Lat. Am. Trans., vol. 15, no. 8, pp. 1536–1541, 2017, DOI: 10.1109/TLA.2017.7994803.

[125] T. Menzies and A. Marcus, “Automated severity assessment of software defect reports,” in 2008 IEEE International Conference on Software Maintenance, Sep. 2008, pp. 346–355. DOI: 10.1109/ICSM.2008.4658083.

[126] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, “Predicting the severity of a reported bug,” in 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), May 2010, pp. 1–10. DOI: 10.1109/MSR.2010.5463284.

[127] Y. Tian, D. Lo, and C. Sun, “Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction,” in 2012 19th Working Conference on Reverse Engineering, Oct. 2012, pp. 215–224. DOI: 10.1109/WCRE.2012.31.

[128] Y. Tian, N. Ali, D. Lo, and A. E. Hassan, “On the unreliability of bug severity data,” Empir. Softw. Eng., vol. 21, no. 6, pp. 2298–2323, Dec. 2016, DOI: 10.1007/s10664-015-9409-1.

[129] A. K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma, “Regression testing in an industrial environment,” Commun. ACM, vol. 41, no. 5, pp. 81–86, May 1998, DOI: 10.1145/274946.274960.

[130] N. E. Fenton and N. Ohlsson, “Quantitative analysis of faults and failures in a complex software system,” IEEE Trans. Softw. Eng., vol. 26, no. 8, pp. 797–814, Aug. 2000, DOI: 10.1109/32.879815.

[131] P. Husbands, “Genetic algorithms for scheduling,” AISB Q., vol. 89, pp. 38–45, 1994.

[132] G. Antoniol, M. Di Penta, and M. Harman, “Search-based techniques applied to optimization of project planning for a massive maintenance project,” in 21st IEEE International Conference on Software Maintenance (ICSM’05), Sep. 2005, pp. 240–249. DOI: 10.1109/ICSM.2005.79.

[133] M. Tulasiraman, N. Vivekanandan, and V. Kalimuthu, “Multi-objective Test Case Prioritization Using Improved Pareto-Optimal Clonal Selection Algorithm,” 3D Res., vol. 9, no. 3, p. 32, Sep. 2018, DOI: 10.1007/s13319-018-0182-y.

[134] R. Matulis and J. Lloyd, “The History, Evolution, and Future of Medicaid Accountable Care Organizations,” Center for Health Care Strategies, p. 22, February 2018.

[135] H. Jahan, Z. Feng, S. M. H. Mahmud, and P. Dong, “Version specific test case prioritization approach based on artificial neural network,” J. Intell. Fuzzy Syst., vol. 36, no. 6, pp. 6181–6194, Jan. 2019, DOI: 10.3233/JIFS-181998.

[136] K. J. Kelleher et al., “Cost Saving and Quality of Care in a Pediatric Accountable Care Organization,” PEDIATRICS, vol. 135, no. 3, pp. e582–e589, Mar. 2015, DOI: 10.1542/peds.2014-2725.

[137] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empir. Softw. Eng., vol. 14, no. 2, pp. 131–164, Apr. 2009, DOI: 10.1007/s10664-008-9102-8.

[138] S. Yoo and M. Harman, “Pareto efficient multi-objective test case selection,” in Proceedings of the 2007 international symposium on Software testing and analysis - ISSTA ’07, London, United Kingdom: ACM Press, 20s07, p. 140. DOI: 10.1145/1273463.1273483.

s

APPENDIX A

Source Code Snippets

1) Library Files

```
#*************** Python Library Files ********************************************************************************
import numpy as np
import pandas as pd
import pygad
import ga
import time
```

2) Reading Test Data

```
df = pd.read_excel(r'C:\Users\HP\OneDrive\Software testing\My PhD\3rd semester\Publications\Pub 4- TCP techn

coverage = df.to_numpy()
coverageTemp = df.to_numpy() #New
print(coverage)

Iterations = input("Please enter number of iteration:\n")
Iterations = int(Iterations)
start = time. time()
print(f'Number of Iterations are: {Iterations}')

TotSev = np.sum(coverage[0])
print('\nTotal Severity of Faults\n----------\n', TotSev)
TotCost = np.sum(coverage[:,0], axis = 0)
print('\nTotal Cost of Test cases\n----------\n', TotCost)
num_rows = np.shape(coverage)[0]
num_columns = np.shape(coverage)[1]
n = num_rows-1
m = num_columns-1
print ('\nTotal no. of test cases----------\n', n)
print ('\nTotal no. of Faults----------\n', m)
print ('\nList of Test cases in the Test Suite to be Prioritized\n')
```

3) Objective Function

```
def Result(coverage1, FaultSeverity1, TestCost1, Permutation1, TotSev1, TotCost1):
    res1 = 0.0000
    for p in range(1, m+1):
        #print(FaultSeverity[p+1])
        for q in range(1, n+1):
            #print('\n')
            #print(FaultSeverity[p])
            #print(TestCost[q])
            #print(Permutation[q-1][1])
            #print(coverage[q][p])
            if coverage1[q][p] == 1:
            #print(Permutation[q-1][1])
                res1 = res1 + ((FaultSeverity1[p]/TestCost1[q]) * Permutation1[q-1][1])
                break
    #print(res1)
    APFDv = 1 - (res1/(TotSev1*TotCost1)) + (1.0/(2.0*TotCost1));
    #print(APFDv)
    return APFDv
```

4) Crossover Function

```
def Crossover(Parent1, x):
    crossover_point = int(x/2)
    Off1 = np.append(Parent1[crossover_point:], Parent1[:crossover_point])
    return Off1
```

5) Mutation Function

```
def Mutate(OffM1, n):
    #ind = []
    print(OffM1)
    idx = len(OffM1)
    #for i in range(np.random.choice(range(idx), 1)[0]):
        ## Two exclusive indices, False for replacement so they are exclusive
    x1, x2 = np.random.choice(range(idx), 2, False)
        ## Swap the genes at the indices
    print(x1)
    print(x2)
    OffM1[x1], OffM1[x2] = OffM1[x2], OffM1[x1]
    return OffM1
print('\n')
```

6) Fitness Function

```
print ('Fitness of individual test cases- A ratio of total severity detected by a test case and cost of that test case')
sevcov = 0
#count = 1
for i in range(1, n+1):
    severity = 0
    for j in range(1, m+1):
        #print("Severity:",FaultSeverity[j])
        if coverageTemp[i][j] == 1:
            severity = severity + FaultSeverityTemp[j]
            for k in range(i+1, n+1):
                coverageTemp[k][j] = 0
    Fitness1 = severity/TestCost[i]
    print(severity)
    sevcov = sevcov + severity
    FitnessOfTests.append(Fitness1)
print("Sum=", sevcov)
print(FitnessOfTests)
print(coverage)
```

7) Main Body of the Program

```
print("Generation : ", p)
print("Test Order =", TestOrder)
#print("Order : ","                                          ","             ")
for i in range(5):   # Loop counter defines the number of random permutations of a given order.
    NewPermutationA =''
    NewOrder = np.random.permutation(TestOrder)
    for j in range(n):
        NewPermutation[j][1] = NewOrder[j]
    NewPermutationA = Cloning(NewOrder)
    fit = Result(coverage, FaultSeverity, TestCost, NewPermutation, TotSev, TotCost)
    fitness.append(fit)
    #NewPermutation.sort(key = sortSecond)
    print("Permutation:", "  ",i+1)
    print("Order :", "  \n  ",NewPermutation)
    print("\n")
    print("APFDv:", "   ",fit)
    print("\n\n")

    if fit > fitmax1:
        Tempmax = fitmax1
        Tempparent = parentA[:]
        fitmax1 = fit
        parentA = NewPermutationA
        if Tempmax > fitmax2:
            fitmax2 = Tempmax
            parentB = Tempparent[:]
    elif fit > fitmax2:
          fitmax2 = fit
          parentB = NewPermutationA
print("The selected parents are")

print(parentA, "                    ",fitmax1)
print("\n")
print(parentB, "                    ",fitmax2)

print("\n")
for k in range(n):
    NewPermutation[k][1] = int(parentA[k:k+1])
# NewPermutation.sort(key = sortSecond)
print(NewPermutation, "                    ",fitmax1)
for l in range(n):
    NewPermutationX[l][1] = int(parentB[l:l+1])
#NewPermutationX.sort(key = sortSecond)
print("\n")
print(NewPermutationX, "                    ",fitmax2)

print('\n')
#print("The best order:", BestOrder)
```

```
if fitmax1 > BestAPFDv:
    BestAPFDv = fitmax1
    BestOrder = CloningBest(parentA)
    for k in range(n):
        BestOrder1[k][1] = int(BestOrder[k:k+1])
#BestOrder1.sort(key = sortSecond)
    #for k in range(n):
    #    BestOrder[k][1] = int(parentA[k:k+1])
print("*************************The best order:*********************************", BestOrder)
```

8) Function Calling

```
print("Result before crossover")
print(p1)
OffSpring1 = Crossover(p1, n)
print("Crossover result")
print(OffSpring1)

print('Mutation Results')
OffSpringM1 = Mutate(OffSpring1, n)
print(OffSpringM1)
#print(OffSpringM2)
TestOrder = OffSpringM1
#NewPermutation.sort(key = sortFirst)
```

9) Printing Results

```
print("The Best Prioritized Order")
print(BestOrder1)
print('\n')
print("The Best APFDv\n")
print(BestAPFDv)
```

10) Execution Time

```
start = time. time()
```

```
end = time. time()
print("The Execution Time :  ", end - start)
print("Hello")
```

APPENDIX B

1. Example 1: ACO Healthcare Solution for Validation of Business Value Estimation Model- Fault-Based TCP

Test Data: Test Cases Vs Fault Traceability Matrix

Faults & severity / Test Cases & cost		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
		S = 3	S = 4	S= 2	S= 1	S= 1	S= 2	S= 1	S= 2	S= 3	S= 5	S= 2	S= 1	S= 4	S= 2	S= 5	S= 2	S= 2	S= 3	S= 3	S= 5
T1	C= 2		x																		x
T2	C= 1	x																			
T3	C= 3				x										x						
T4	C= 5							x	x												
T5	C= 4					X													x		
T6	C= 2										x										
T7	C= 1						x														x
T8	C= 2			x											x						
T9	C= 5												x				x				
T10	C= 1				x																
T11	C= 2																	x			
T12	C= 4											x		x							
T13	C= 3									x										x	
T14	C= 5					X										x					

2. Example 2: ACO Healthcare Solution for Validation of Business Value Quantification Model for Requirements-Based TCP

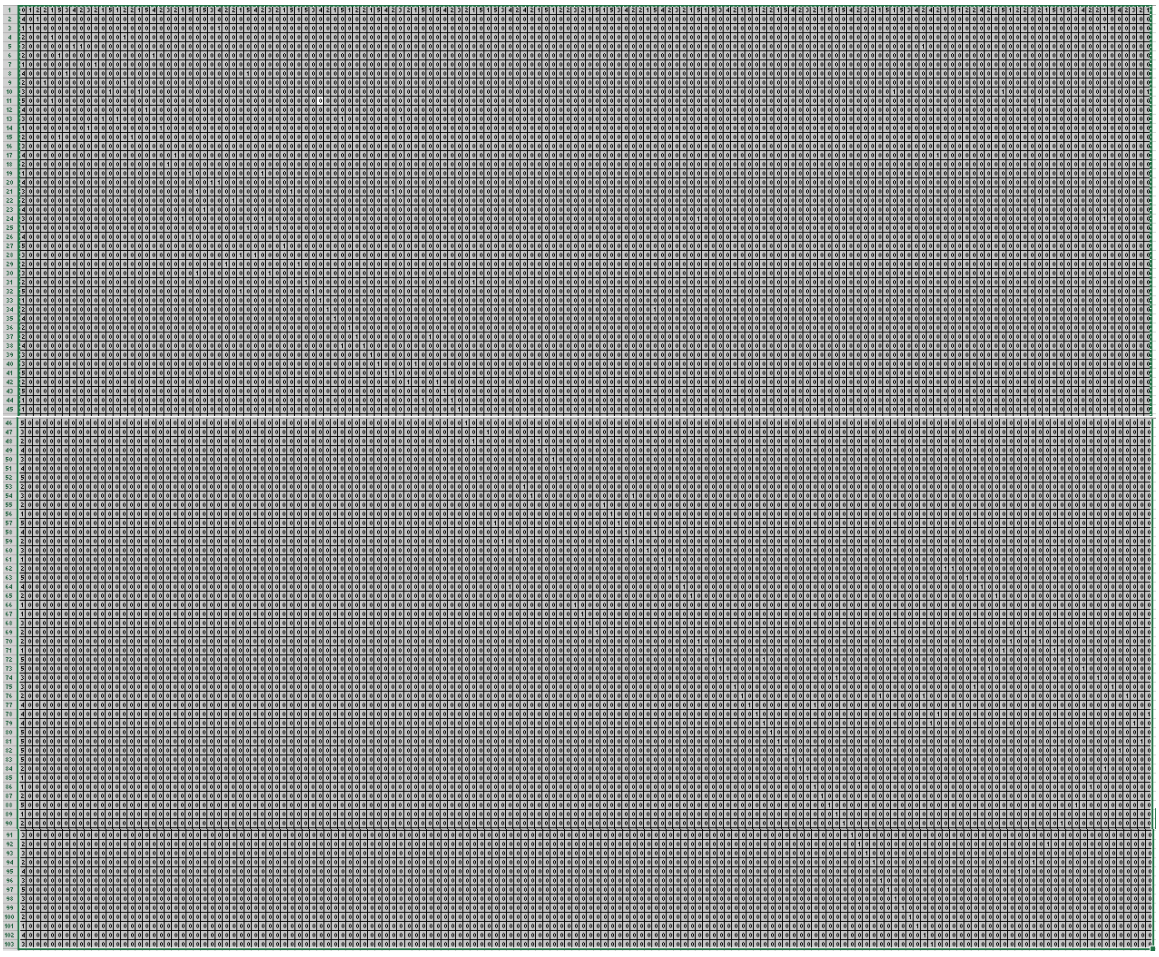
Test Data: Test Cases Vs Requirements Coverage Traceability Matrix

Requirements & Business value / Test Cases & cost		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14
		BV=2	BV=4	BV=1	BV=3	BV=2	BV=4	BV=5	BV=1	BV=3	BV=2	BV=3	BV=4	BV=1	BV=3
T1	C= 2														
T2	C= 1	x													
T3	C= 3									x					x
T4	C= 5							x							
T5	C= 4			x											
T6	C= 2					x									
T7	C= 1													x	
T8	C= 2				x										x
T9	C= 5												x		
T10	C= 1						x								
T11	C= 2								x						
T12	C= 4										x				
T13	C= 3											x			
T14	C= 5		x												

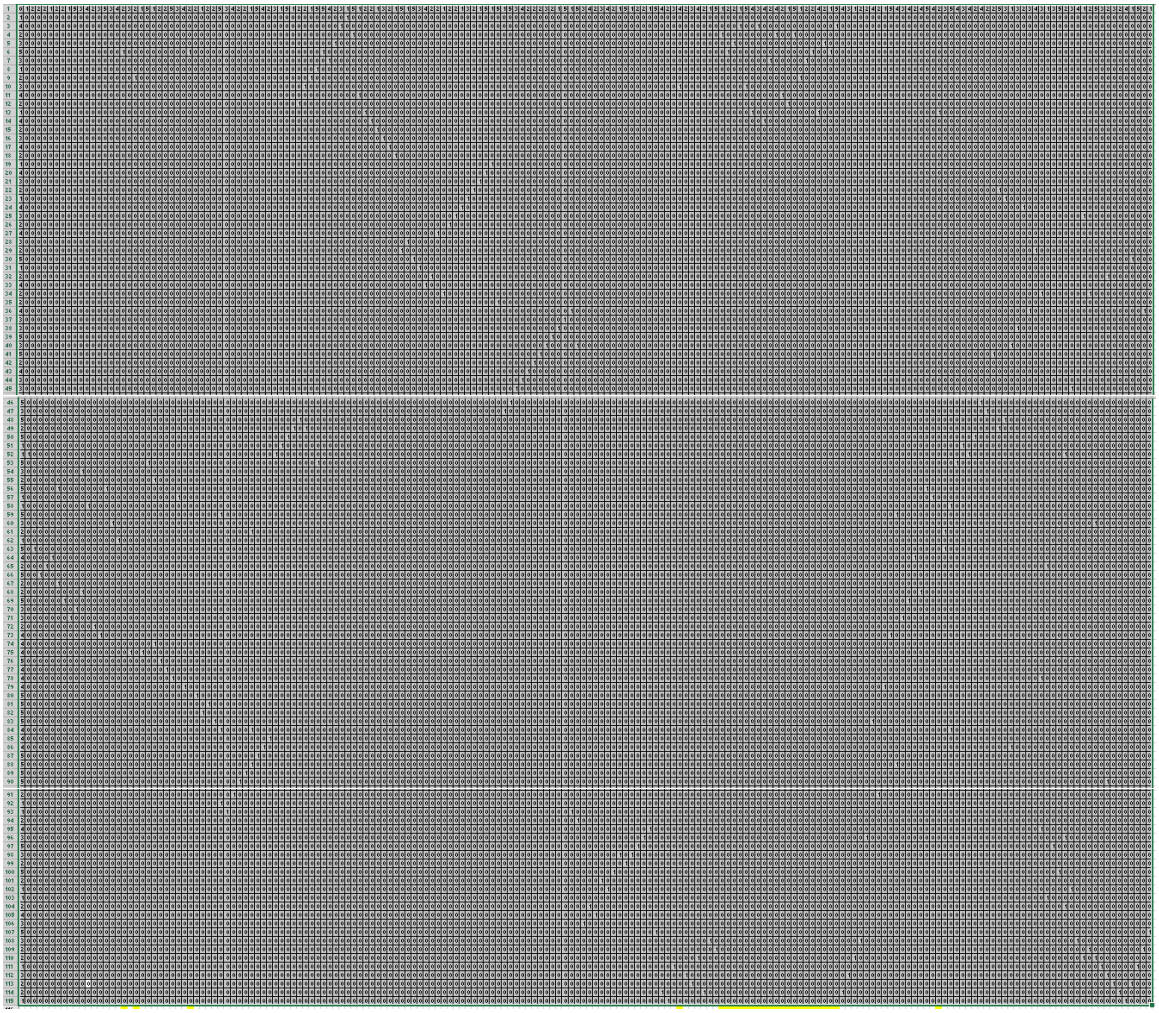
APPENDIX C

Case Study 1: ACO Healthcare Solution for Validation of VCFDB-TCP

1. Test Data 1: Test Cases Vs Faults Traceability Matrix

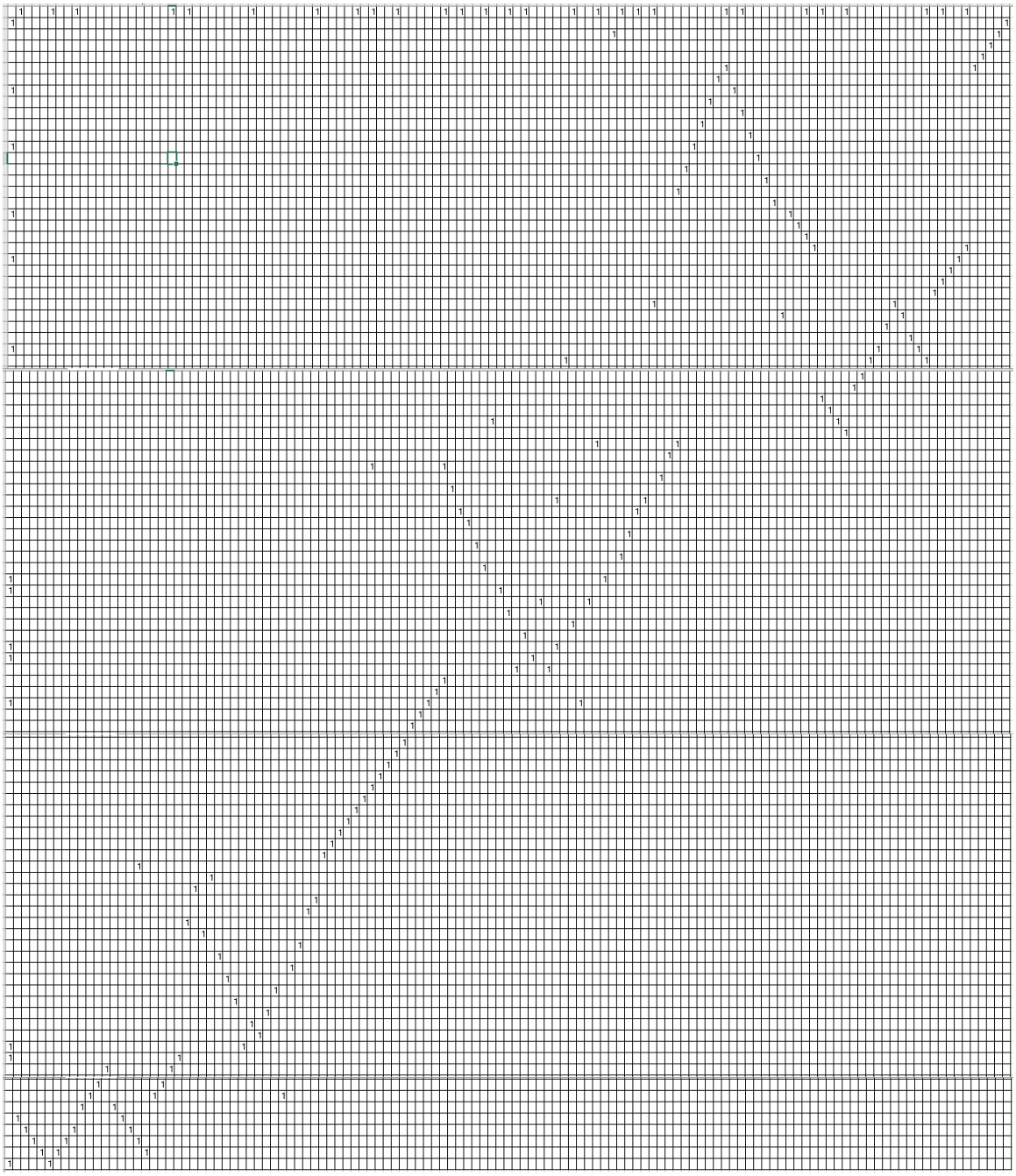


2. Test Data 2: Test Cases Vs Faults Traceability Matrix



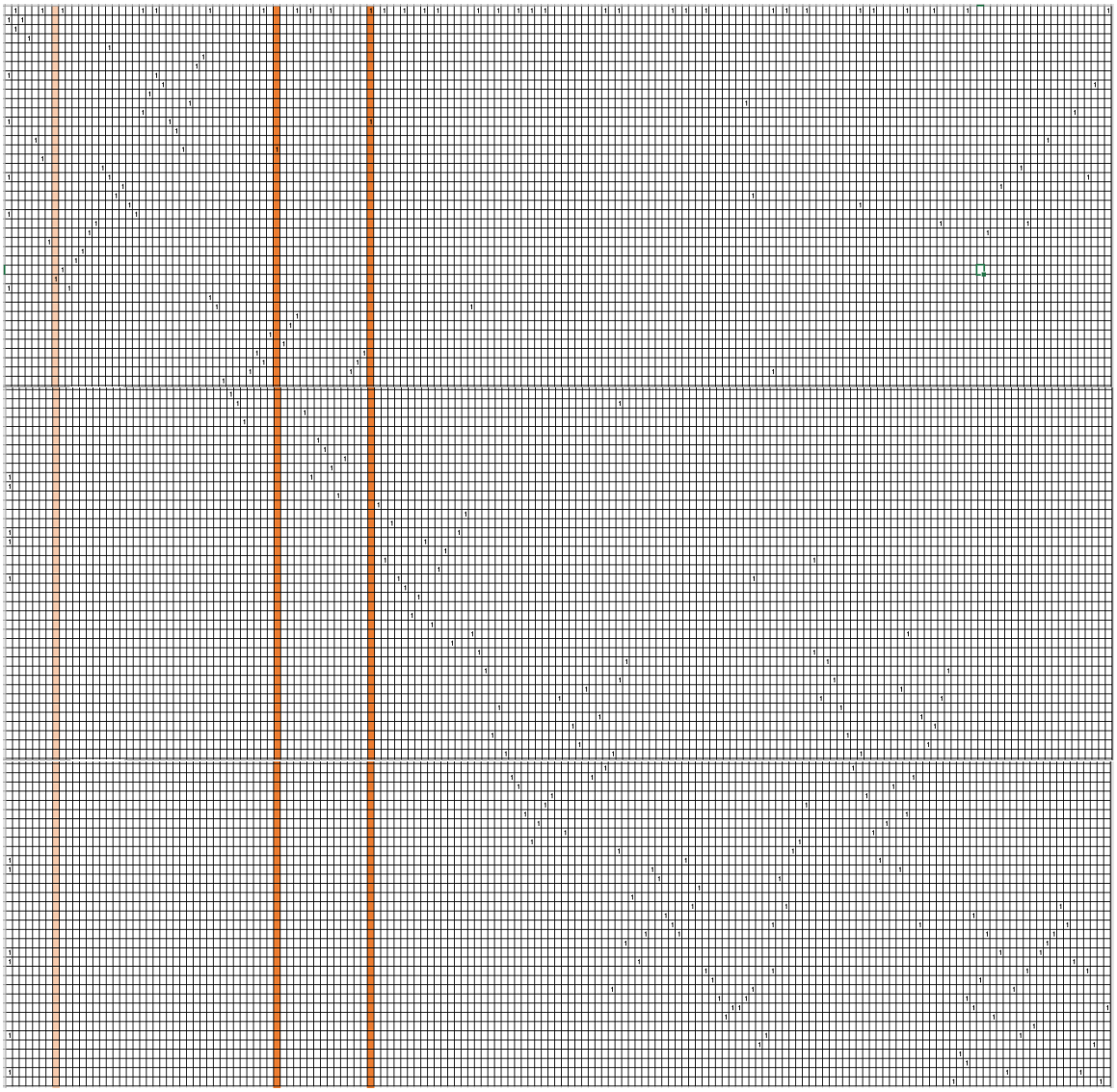
3. Test Data 3: Test Cases Vs Faults Traceability Matrix

4. Test Data 4: Test Cases vs Requirements Traceability Matrix



5. Test Data 5: Test Cases Vs Requirements Traceability Matrix

6. Test Data 6: Test Cases Vs Requirements Traceability Matrix



APPENDIX D

Case Study 2: ACO Healthcare Solution for Validation of VCRCB-TCP Technique

1. Test Data 1: Test Cases Vs Faults Traceability Matrix

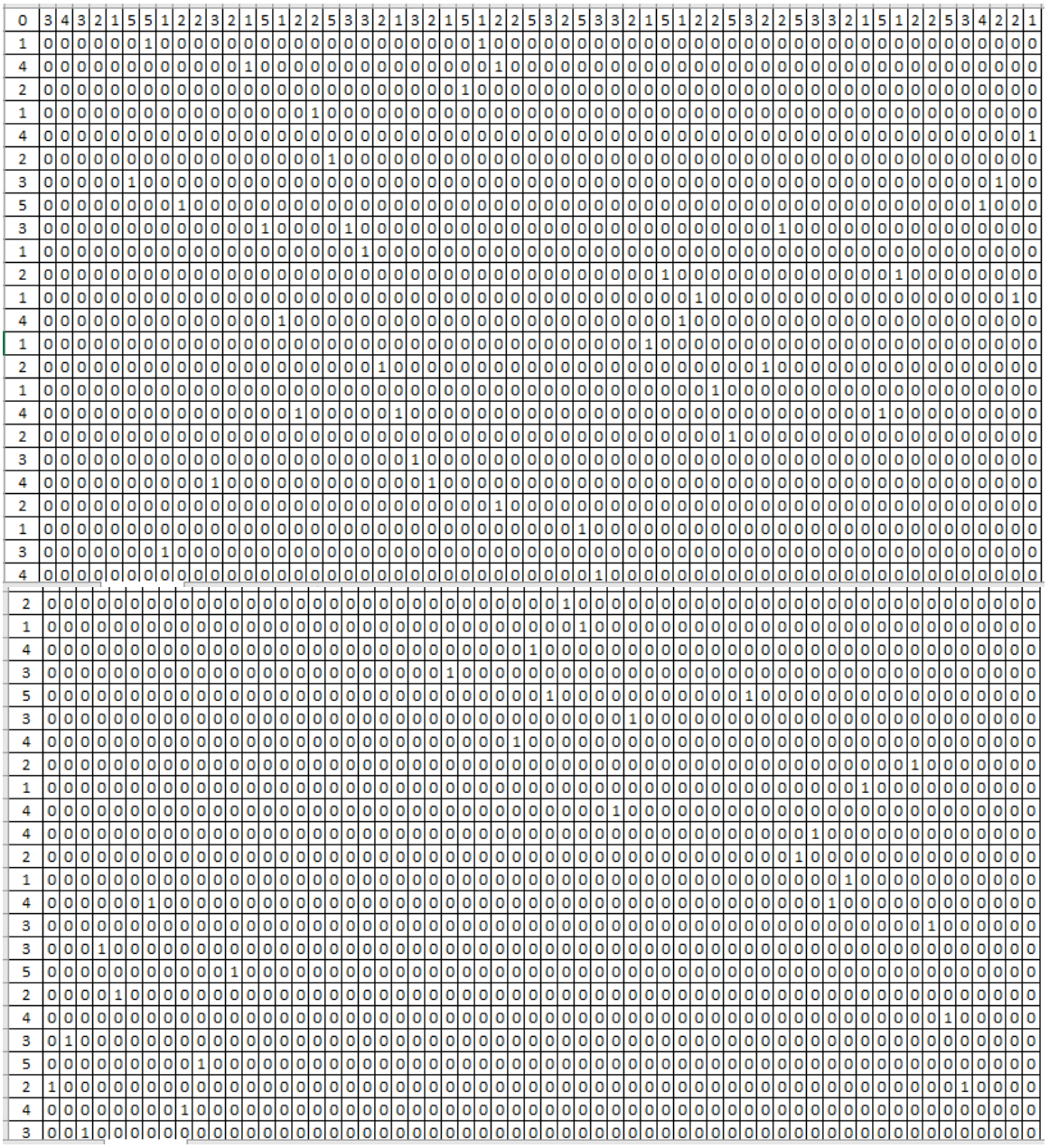
0	1	5	3	5	1	5	3	4	3	2	1	5	1	3	4	2	1	5	3	4	2	1	5	1	5	3	4	2	2	2	1	5	1	5	3	4	5	1	5	3	4	2	2	1	1	2	2	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2. Test Data 2: Test Cases Vs Faults Traceability Matrix

0	1	5	3	4	2	3	5	1	5	3	4	3	5	5	3	2	2	5	3	4	2	2	1	5	3	4	2	2	5	3	4	2	2	1	5	3	4	2	2	1	5	3	2	2	2	1	5	3	2	2	1	4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3. Test Data 3: Test Cases Vs Faults Traceability Matrix



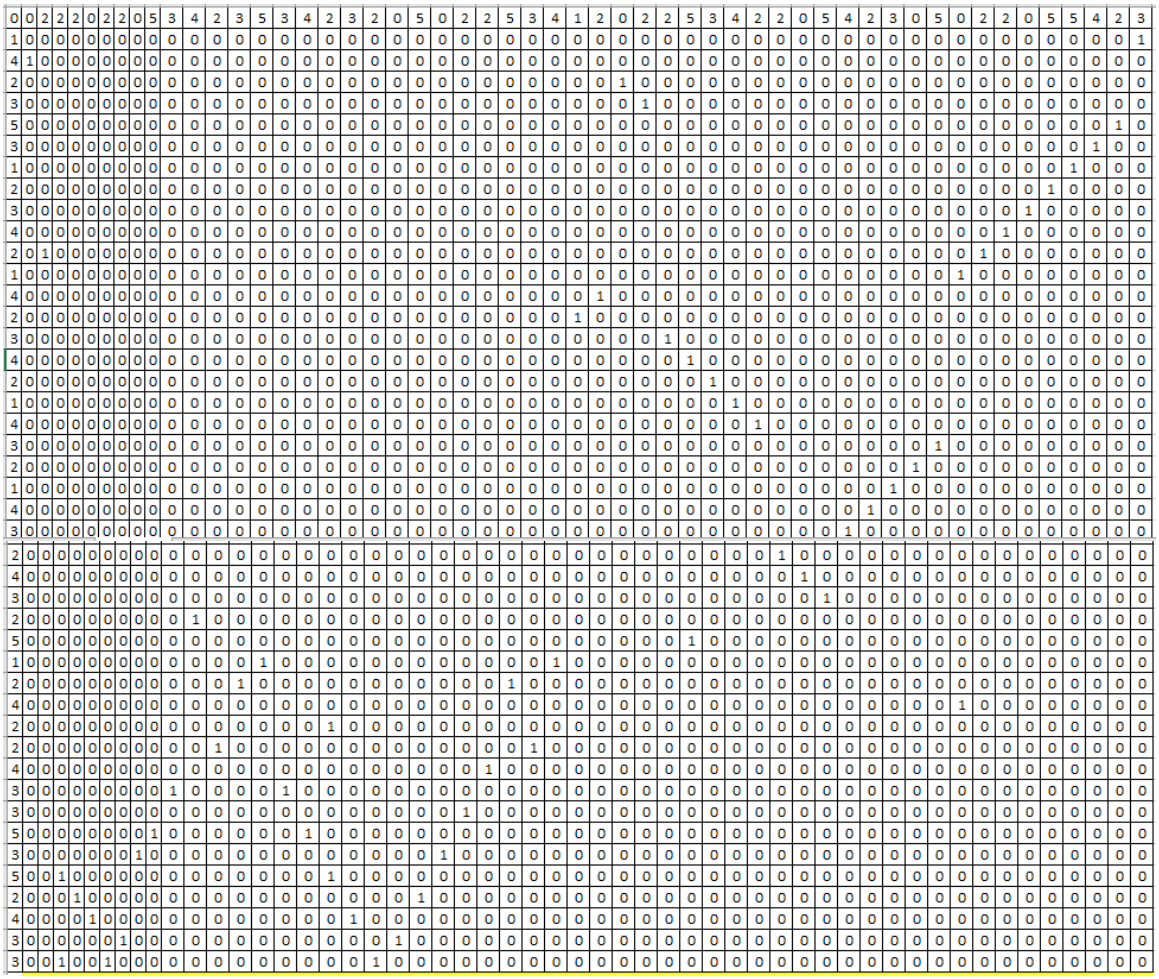
4. Test Data 4: Test Cases Vs Requirements Traceability Matrix

0	1	1	2	2	1	2	2	5	2	2	2	2	1	2	2	1	2	2	5	3	2	1	2	2	5	3	4	2	2	1	2	2	5	2	2	2	3	4	2	2	1	3
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5. Test Data 5: Test Cases Vs Requirements Traceability Matrix

0	2	2	5	3	4	2	2	2	2	5	3	4	2	2	3	2	2	5	3	4	2	2	2	2	2	2	5	3	4	2	2	5	2	2	5	3	4	2	2	2	2	5	3	4	2	2	2	5
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6. Test Data 6: Test Cases Vs Requirements Traceability Matrix



APPENDIX E

1. Case Study 1 Statistical Analysis Results

Test 1: Original Order and VB-GA $APFD_v$			
Product	Release	Original order	Value-Based GA
Product A	R1	0.9313	0.9408
	R2	0.9332	0.9445
	R3	0.9221	0.9437
Product B	R1	0.9385	0.9477
	R2	0.9456	0.9502
	R3	0.9348	0.9472

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Original Order	*VB-GA*
Mean	0.93425	0.9456833
Variance	6.0923E-05	1.119E-05
Observations	6	6
Pearson Correlation	0.769182461	
Hypothesized Mean Difference	0	
df	5	
t Stat	-4.954909371	
P(T<=t) one-tail	0.002133419	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	**0.004266839**	
t Critical two-tail	2.570581836	

Test 2: Reverse Order vs VB-GA $APFD_v$			
Product	Release	Reverse order	Value-Based GA
Product A	R1	0.9193	0.9408
	R2	0.9262	0.9445
	R3	0.9356	0.9437
Product B	R1	0.9339	0.9477
	R2	0.9188	0.9502
	R3	0.9409	0.9472

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Reverse order	*VB-GA*
Mean	0.9291167	0.945683333
Variance	8.301E-05	1.11897E-05
Observations	6	6
Pearson Correlation	0.1276449	
Hypothesized Mean Difference	0	
df	5	
t Stat	-4.365127	
P(T<=t) one-tail	0.0036276	
t Critical one-tail	2.0150484	
P(T<=t) two-tail	**0.0072552**	
t Critical two-tail	2.5705818	

Test 3: Random Order and VB-GA $APFD_v$			
Product	Release	Random order	Value-Based GA
Product A	R1	0.9178	0.9408
	R2	0.9263	0.9445
	R3	0.9313	0.9437
Product B	R1	0.9367	0.9477
	R2	0.9278	0.9502
	R3	0.9386	0.9472

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Random Order	Value-Based GA
Mean	0.92975	0.9456833
Variance	5.7507E-05	1.119E-05
Observations	6	6
Pearson Correlation	0.614933039	
Hypothesized Mean Difference	0	
df	5	
t Stat	-6.37344664	
P(T<=t) one-tail	0.000703349	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	**0.001406697**	
t Critical two-tail	2.570581836	

Test 4: Greedy Order vs VB-GA $APFD_v$			
Product	Release	Greedy order	Value-Based GA
Product A	R1	0.9279	0.9408
	R2	0.9275	0.9445
	R3	0.9286	0.9437
Product B	R1	0.9374	0.9477
	R2	0.9336	0.9502
	R3	0.9423	0.9472

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Greedy order	Value-Based GA
Mean	0.9328833	0.945683333
Variance	3.635E-05	1.11897E-05
Observations	6	6
Pearson Correlation	0.6528059	
Hypothesized Mean Difference	0	
df	5	
t Stat	-6.808254	
P(T<=t) one-tail	0.0005207	
t Critical one-tail	2.0150484	
P(T<=t) two-tail	**0.0010414**	
t Critical two-tail	2.5705818	

2. Case Study 2 Statistical Analysis Results

Test 1: Original Order and VB-GA APRC$_v$			
		Original order	Value-Based GA
Product A	R1	0.9121	0.9366
	R2	0.951	0.9594
	R3	0.9387	0.941
Product B	R1	0.9119	0.9444
	R2	0.9289	0.9492
	R3	0.9143	0.9454

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Variable 1	*Variable 2*
Mean	0.92615	0.946
Variance	0.000264695	6.122E-05
Observations	6	6
Pearson Correlation	0.692919746	
Hypothesized Mean Difference	0	
df	5	
t Stat	-3.976593356	
P(T<=t) one-tail	0.005283257	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	**0.010566514**	
t Critical two-tail	2.570581836	

Test 2: Reverse Order vs VB-GA $APRC_v$			
Product	Release	Reverse order	Value-Based GA
Product A	R1	0.8888	0.9366
	R2	0.9404	0.9594
	R3	0.9048	0.941
Product B	R1	0.927	0.9444
	R2	0.9273	0.9492
	R3	0.9214	0.9454

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Variable 1	*Variable 2*
Mean	0.9182833	0.946
Variance	0.0003414	6.1216E-05
Observations	6	6
Pearson Correlation	0.9046705	
Hypothesized Mean Difference	0	
df	5	
t Stat	-5.716019	
P(T<=t) one-tail	0.0011455	
t Critical one-tail	2.0150484	
P(T<=t) two-tail	**0.002291**	
t Critical two-tail	2.5705818	

Test 3: Random Order and VB-GA $APRC_v$			
		Random order	Value-Based GA
Product A	R1	0.9012	0.9366
	R2	0.9405	0.9594
	R3	0.9271	0.941
Product B	R1	0.9087	0.9444
	R2	0.9251	0.9492
	R3	0.9118	0.9454

Statistical Analysis Results t-Test: Paired Two Sample for Means

	Variable 1	*Variable 2*
Mean	0.919066667	0.946
Variance	0.000207963	6.122E-05
Observations	6	6
Pearson Correlation	0.806947438	
Hypothesized Mean Difference	0	
df	5	
t Stat	-7.069681871	
P(T<=t) one-tail	0.000437933	
t Critical one-tail	2.015048373	
P(T<=t) two-tail	**0.000875866**	
t Critical two-tail	2.570581836	

Test 4: Greedy Order vs VB-GA APRC$_v$			
Product	Release	Greedy order	Value-Based GA
Product A	R1	0.6282	0.9366
	R2	0.7976	0.9594
	R3	0.7042	0.941
Product B	R1	0.687	0.9444
	R2	0.6961	0.9492
	R3	0.6759	0.9454

Properties	***Variable 1***	***Variable 2***
Mean	0.6981667	0.946
Variance	0.0030887	6.1216E-05
Observations	6	6
Pearson Correlation	0.9127873	
Hypothesized Mean Difference	0	
df	5	
t Stat	-12.50658	
P(T<=t) one-tail	2.899E-05	
t Critical one-tail	2.0150484	
P(T<=t) two-tail	**0.0000579846311191025**	
t Critical two-tail	2.5705818	