The Design of T-way Test Suite Generator Supporting Uniform Strength, Variable Strength and Input-Output Based Relations

by

Nuraminah Bt Ramli
(1540211784)

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

School of Computer and Communication Engineering
UNIVERSITI MALAYSIA PERLIS

2019

ACKNOWLEDGMENT

In the name of Allah, the Most Gracious, the Most Merciful. All praise to Allah for the blessing that lead to my successfulness in completing my research. First and foremost, I would like to express my profound gratitude to my supervisor, Associate Prof. Dr. Rozmie Razif Othman for his support and encouragement. I also would like to thank him for the many hours he spent helping me through the difficult phase of this research. Special thanks to my co-supervisor, Dr. Zahereel Ishwar Abdul Khalib for his kind words of guidance, motivations and criticism from the commencement until the end of the research. Their words of encouragement kept me motivated throughout this research and I have learned a lot from them and the achievement would not have been possible without their help.

My deepest appreciation to my beloved husband, Shukor Sanim Mohd Fauzi for his endless love, support and encouragement, whom understands and cares about me during my ups and downs. To my daughter, Safiya Nayli Sanim, thank you for the unconditional love, happiness and joy you bring, which give me strength to finish my study. My hearties thanks to my parents, parents in laws, siblings, sisters-in-laws, nieces and nephews for all the support, who continuously pray for my success throughout this journey.

My sincere appreciation also goes for my friends at ENAC, Pasca Lab, my ex-colleagues and lecturers for the fruitful ideas and sharing opinions. Last but not least, I would like to acknowledge the Malaysian Ministry of Higher Education and Universiti Malaysia Perlis for financially support my research.

Nuraminah Ramli

.

@This item is protected by original copyright

TABLE OF CONTENTS

@This item is protected by original copyright

@This item is protected by original copyright

@This item is protected by original copyright

@This item is protected by original copyright

LIST OF TABLES

@This item is protected by original copyright

@This item is protected by original copyright

LIST OF FIGURES

@This item is protected by original copyright

LIST OF ABBREVIATIONS

ACA	Ant Colony Algorithm
ACO	Ant Colony Optimization
ACS	Ant Colony System
CA	Covering Array
CS	Cuckoo Strategy
GA	Genetic Algorithm
GVS	General Variable Strength
DC	Don't care
HHH	High Level Hyper-Heuristic
HSS	Harmony Search Strategy
IOR	Input-Output Based Relations
IPO	In-Parameter-Order
IPOG	In-Parameter-Order-General
ITTDG	Integrated T-way Test Data Generation
MCA	Mixed Covering Array
NA	Not Available
NP	Non-deterministic Polynomial Time
OPAT	One-parameter-at-a-time
OTAT	One-test-at-a-time
PSO	Particle Swarm Optimization
PSTG	Particle Swarm Test Generator
STLC	Software Testing Life Cycle
SA	Simulated Annealing
SUT	Software Under Test
TTSGA	T-way Test Suite Generation based on Ant Colony Algorithm
TVG	Test Vector Generation
VCA	Variable Strength Covering Array

@This item is protected by original copyright

LIST OF SYMBOLS

η	Heuristic value
E	Edges
i	The current nodes
j	The next nodes
p	Probability value
$\boldsymbol{\tau}$	Pheromone value
ρ	Pheromone decay
Δ	Fitness function
α	alpha value
β	beta value
$\forall$	for all / for each

@This item is protected by original copyright

Rekabentuk Penjana Sut Ujian T-Hala Yang Menyokong Kekuatan Seragam, Kekuatan Berubah Atau Hubungan Berdasarkan Input-Output

ABSTRAK

Pengujian perisian adalah sangat penting dalam semua projek perisian bagi mengesan ralat. Banyak teknik reka bentuk kes ujian menawarkan pelbagai kaedah untuk menghasilkan kes ujian yang efektif. Saiz sut ujian telah menjadi fokus utama teknik reka bentuk ujian untuk mengatasi masalah ujian menyeluruh disebabkan oleh kekangan masa dan sumber. Kebanyakan ralat adalah berpunca daripada interaksi di antara beberapa parameter, jadi, ujian *t*-hala sangat sesuai untuk digunakan. Dalam kerja ini, taksonomi ujian *t*-hala dan strategi sedia ada yang menyokong ujian *t*-hala telah diterokai. Strategi sedia ada ini telah dibandingkan dengan faktor-faktor pada taksonomi *t*-hala. Walaupun bermanfaat, strategi sedia ada cenderung menyokong jenis interaksi sokongan *t*-hala tertentu sama ada interaksi dengan kekuatan seragam, kekuatan berubah atau hubungan berdasarkan Input-Output (IOR). Walau bagaimanapun, masalah pengujian datang dalam pelbagai situasi. Oleh kerana semua jenis interaksi sokongan berfungsi dengan sempurna dalam situasi yang berbeza, satu strategi yang fleksibel dan dapat digunakan pada semua jenis interaksi sokongan membolehkan penguji perisian untuk berkompromi dalam memilih jenis interaksi sokongan *t*-hala yang paling sesuai berdasarkan masalah pengujian yang ada. Selain itu, penghasilan ujian *t*-hala telah dikategorikan sebagai masalah kesukaran-NP. Tiada satu strategi yang boleh memberikan hasil optimum sepanjang masa. Oleh itu, Generasi Pengujian *T*-hala berdasarkan algoritma Ant Colony (TTSGA) telah dicadangkan dalam kajian ini. Selain menyokong ketiga-tiga jenis interaksi sokongan, strategi ini menggunakan teknik pencarian metaheuristik, yang tidak pernah digunakan oleh strategi lain yang menyokong kesemua jenis interaksi sokongan dan telah terbukti menghasilkan saiz sut ujian yang minimum berbanding dengan teknik pencarian melalui pengiraan. Seperti namanya, strategi ini menggunakan algoritma Ant Colony untuk teknik carian. Prestasi strategi TTSGA telah diuji dengan menggunakan beberapa eksperimen yang diukur untuk setiap jenis interaksi sokongan. Selain daripada itu, dua analisis statistik telah dilaksanakan, iaitu ujian Friedman Test and Wilcoxon Rank. Keputusan telah dibandingkan dengan strategi metaheuristik dan komputasi lain untuk semua jenis interaksi sokongan. Bagi saiz sut ujian, TTSGA mendapat pangkat pertama bagi kekuatan berubah-ubah dan konfigurasi seragam IOR dengan pangkat purata Ujian Friedman masing-masing adalah 1.49 dan 2.00. Strategi ini juga menghasilkan keputusan yang menggalakkan bagi kekuatan seragam (2.75 di pangkat ketiga) dan konfigurasi IOR bukan seragam (3.25 di pangkat kedua). Pada umumnya, TTSGA menghasilkan hasil yang baik dalam kekuatan dan konfigurasi interaksi yang tinggi.

@This item is protected by original copyright

The Design of T-Way Test Suite Generation Supporting Uniform Strength, Variable Strength and Input-Output Based Relations

ABSTRACT

Software testing is important in any software project to detect faults. Many test case design techniques offer various methods to produce effective test cases. Test suite size is the biggest concern of test design technique to overcome the exhaustive testing problem due to time and resources. As most of the faults are caused by interactions of few parameters, *t*-way testing is an appropriate technique to be used. In this work, *t*-way testing taxonomy and existing strategies that support *t*-way testing has been explored. The existing strategies have been compared against factors in the taxonomy. Although beneficial, the existing strategies tend to support a specific type of *t*-way support interactions either uniform strength, variable strength or Input-Output based Relations (IOR). Those types of support interactions function in different situations perfectly. However, software tester received testing problems in various situations. Therefore, a single flexible strategy that allows software tester to choose the most suitable support interactions based on testing problems on hand is required. In addition to that, *t*-way test suite generation has fallen under the NP-hard problem. There is no single strategy that can guarantee the optimal results at all times. As a consequence, *T*-way Test Suite Generation based on Ant Colony algorithm (TTSGA) was proposed in this research. Besides supporting all three types of support interactions, this strategy applies metaheuristic search technique, which is never been applied by any other strategies that support all types of support interactions. Metaheuristic search technique also is proven to produce minimum test suite size as compared to computational search technique. As the name suggested, the strategy adopts the ant colony algorithm for the search technique. Performance of the TTSGA strategy was implemented using several benchmarked experiments for each type of support interactions. Besides that, two statistical analysis tests, Friedman Test and Wilcoxon Rank test has been performed. The results have been compared to other metaheuristic and computational strategies for all types of support interactions. As far as test suite size is concerned, TTSGA achieved the first rank for variable strength and IOR uniform configurations with the mean rank of Friedman Test is 1.49 and 2.00 respectively. The strategy also produces encouraging results for uniform strength (2.75 in third rank) and IOR non-uniform configurations (3.25 in second rank). Generally, TTSGA produces good results in high interactions strength and configurations.

@This item is protected by original copyright

CHAPTER 1 : INTRODUCTION

1.1 Overview

In this day and age, software application has become a crucial element in our lives which is widely used in various fields such as business, education and management to support and facilitate daily tasks. The use of this software is also helpful in critical fields like finance and security. For example, online banking has seen a rapid rise as it allows banking transactions to be carried out anytime and anywhere, as long as users have access to Internet connection. Therefore, it is important to ensure the use of software application to be reliable and efficient. Software applications are also utilised by companies to run their businesses. The expenses used on software applications are quite high (Ahmad & Abd Samat, 2019; Charette, 2005; Rai, Mehfuz, & Sahoo, 2013). This reinforces the need for software applications in many areas of society today as they have become the backbone of the community.

Due to the growth of the software applications usage, the applications become more complex as it involves a lot of interactions among other software, hardware, modules, or users. The complexity of the software leads to a high tendency to crash if the development stage is not managed effectively. Hence, this is a risky consequence for anyone who is directly involved with this software. Besides, the financial standing and economic growth of the company can also be affected (Taherdoost & Keshavarzsaleh, 2015). In extreme cases, it can be life threatening and can turn future operations of the company into chaos.

@This item is protected by original copyright

Therefore, software developers need to direct their full attention on software testing to guarantee the reliability and quality of the software. More time and bigger budget should be allocated to software testing activities (Sajad, Sadiq, Naveed, & Iqbal, 2016) and the allocation is increasing (Ahmad & Abd Samat, 2019). The same amount of resources should also be devoted to a critical system to ensure that any detected fault is rectified before the system is deployed (Sommerville, 2001). A higher percentage of cost allocation reflects greater importance of software testing in the software development life cycle.

Software testing is not only concerned with detecting faults but is aimed at ensuring the software under test (SUT) is well-functioning and conforms to requirements or specifications. As such, defining a set of test cases is the essence in software testing. Black box testing or functional testing is one of the software testing approaches. It is used to identify a set of test cases based on specifications of a programme from a behavioural or functional perspective (Jorgensen, 2014). Hence, a software tester is responsible to establish the best set of tests to ensure the quality of the software without neglecting costs and time limits.

In order to determine effective test cases, numerous test case design techniques such as equivalence class partitioning, boundary value analysis and *t*-way testing have been introduced in the literature. Each technique has a different method of identifying test cases. The purpose of these techniques is to select a set of test cases that are able to detect the most number of faults.

@This item is protected by original copyright

Equivalence class partitioning (Myers, Badgett, & Sandler, 2012) works by dividing test conditions or input values into a partition that have similar or equivalent conditions. The partitions will include both valid and invalid partitions. Any picked value for each partition represents the whole partition as they have equivalent conditions. This technique assumes that if one value from as specific partition is passed, all values in the partition are passed and vice versa. For example, for an application that accept an input between 1 and 10, there will be one valid partition (i.e. 1 - 10) and two invalid partitions (i.e. less than 1 and greater than 10). Then, any value for each partition is chosen for testing.

As compared to equivalence class partitioning, boundary value analysis (Myers et al., 2012) focuses on a boundary of a partition. Test cases must include boundaries between partitions of the input values. The partitions may include the minimum and maximum value of each partition. For example, a shopping centre announces that to obtain a voucher, the minimum value purchase must be RM 30. In this situation, there are two partitions which are value of below RM 30 and value of more than RM30. Thus, the test case must include RM 29 and RM 31 for at least once.

Another test case design technique is combinatorial tests or known as *t*-way testing. As compared to previously described test case design technique, *t*-way is not designed for single input parameters. It is introduced to generate test cases that find faults due to interactions of input parameters (Othman & Zamli, 2011b). *T*-way testing works by interacting input parameter values based on strength, *t*. It covers every possible combination of *t* input parameters at least once (Kuhn, Kacker, & Lei, 2013). Even though *t*-way generates smaller number of test cases, it is able to detect faults

@This item is protected by original copyright

effectively (Borazjany, Yu, Lei, Kacker, & Kuhn, 2012; Raunak, Kuhn, & Kacker, 2017; Wu, Nie, Kuo, Leung, & Colbourn, 2015).

Research on *t*-way test case design technique is growing because of its significance in generating effective test cases in software testing. It does not only contribute by reducing time but also lowers the cost of testing implementation. This research focuses on *t*-way testing because of its capability in fault detection is very encouraging.

1.2 Motivation

Test design technique plays a vital role in software testing. Software tester is tasked with the duty to perform testing activities on all possible test cases based on possible scenarios, permutations or combinations of input to ensure the SUT is free of fault. This situation is called exhaustive testing. Exhaustive testing may lead to too many test cases to be performed by software tester. It can cause more time and budget to be spend (Zakaria & Zamli, 2016).

As a result, test design techniques were developed to reduce the number of test cases produced by exhaustive testing. In previous subsection, three examples of test design technique have been discussed. All of these are useful in finding faults. Nonetheless, most faults are caused by interactions between a few parameters and not necessarily caused by every parameter (Kuhn, Kacker, Lei, & Hunter, 2009). Equivalence class partitioning, boundary value analysis and other traditional test designs do not detect faults caused by interaction (Zamli, Othman, Younis, & Mohamed

@This item is protected by original copyright

Zabil, 2011). Therefore, *t*-way testing that interact the input parameters based on specified strength is a solution to the issue. This testing form also provides higher fault coverage with a smaller test suite size (Bryce, Lei, Kuhn, & Kacker, 2010; Nie & Leung, 2011). This approach can reduce the amount of time, effort and cost in testing activity.

As *t*-way involves interactions of input parameters, *t*-way testing offers three types of support interactions, which include uniform strength, variable strength and input-output based relations (IOR). Each of these has distinct characteristics and is used in different situations. The characteristics of each support interactions are explained in the subsequent section. As SUT usually comes with different features, these different types of support interactions can be adapted with any type of SUT.

Uniform strength combines input parameters uniformly. It is used whenever the software tester has no available information of the SUT. The growth of uniform strength research has been increasing as there is a need for strength value to be greater than two (Bryce et al., 2010; Kuhn et al., 2009). It is also has been explored widely by researchers (Othman, Zamli, & Mohamad, 2013) and many strategies has been developed such as In-Parameter Order General (IPOG) (Lei, Kacker, Kuhn, Okun, & Lawrence, 2007), TConfig (Williams, 2000), GTWay (Zamli, Klaib, Younis, Isa, & Abdullah, 2011) and HHH (Zamli, Alkazemi, & Kendall, 2016).

Meanwhile, variable strength is used by software testers who have partial information of the SUT. As opposed to uniform strength, variable strength offers interactions of multi-strength. Several strategies have been developed such as Simulated

@This item is protected by original copyright

Annealing (SA) (Cohen, Gibbons, Mugridge, Colbourn, & Collofello, 2003), Harmony Search Strategy (HSS) (Alsewari & Zamli, 2012) and Particle Swarm Test Generator (PSTG) (Ahmed & Zamli, 2010).

Another type of support interactions is IOR which is slightly different compares to other types of support interactions. A software tester must have adequate information of the SUT and the actual factors must be taken into consideration. This would mean covering the actual interactions rather than all interactions. A few strategies have been developed to support IOR, such as Union (Schroeder, 2001), Greedy (Cheng, Dumitrescu, & Schroeder, 2003) and ParaOrder (Ziyuan, Nie, & Baowen, 2007).

All the three types of support interactions are used in different situations depending on information available of the SUT. Although the SUT can be exercised with only one type of support interactions, it is best to have the precise type of support interactions to guarantee validity of test cases. Some SUTs have their own factors, relation of parameters and strength. By ignoring those components, some of the generated test cases can be invalid. The real valid combinations may not be discovered and hence, can fail to detect certain faults (Li, Cui, & Yao, 2010). This situation entails the need to establish a strategy that could support all three types of support interactions.

Motivated by all the aforementioned challenges and situation, this research focuses on test design technique that involves interactions of parameter inputs to generate the minimum test suite size. *T*-way testing area that could support all three types of support interactions has been chosen as the research work.

@This item is protected by original copyright

1.3 Problem Statements

In the previous section, motivation towards the research has been presented. Based on the challenges and issues, this research intends to determine the problems in *t*-way testing area.

Producing the best test cases is important to ensure the test activities are performed perfectly. Conducting exhaustive testing is time consuming as it produces too large test suite size and uses a lot of testing resources. It is impractical for the software tester to perform the exhaustive testing (Bryce et al., 2010; Esfandyari & Rafe, 2018; Raunak et al., 2017). This form of testing has also been declared as impossible to perform by the International Software Testing Quality Board (ISTQB) principles. Thus, it presents a real challenge for a software tester to reduce the number of test cases to a subset that can be executed within the scope of available resources while able to adequately detects majority of the defects (Cheng, Dumitrescu, & Schroeder, 2015, Kire & Malhotra, 2014).

T-way testing emerged as one of the test design techniques intends to reduce the number of test cases. The advantage of *t*-way is it caters on faults due to interactions of input parameters as compared to other test design techniques. Many *t*-way strategies have been developed to tackle on this issue. Existing strategies in the market are limited to certain support interactions only except that all strategies support uniform strength (Othman et al., 2013). These strategies are insufficient to support the need of a software tester who has to perform separate strategies for different types of support interactions. It is not an effective way to have more than one strategy to produce test

@This item is protected by original copyright

cases. Therefore, a strategy that provide flexibility to the software tester is required (Othman, 2012). Thus in turn, enables a software tester to choose any type of support interactions based on the available information regarding the SUT.

In order to overcome the problem, a strategy must consider uniform strength, variable strength and IOR support interactions. In comparison to existing strategies of uniform and variable strength, not much research has been conducted on the topic of IOR. In addition, none of the existing strategies apply metaheuristic search technique which has been used by uniform and variable strength based test case generation strategies (Alsewari, Tairan, & Zamli, 2015). In fact, the metaheuristic search technique has been proven to yield a smaller test suite size for uniform and variable strength support interactions (Chen, Gu, Qi, & Chen, 2010; Mahmoud & Ahmed, 2015; Rahman, Othman, Ahmad, & Rahman, 2014; Zabil, Zamli, & Lim, 2018; Zamli et al., 2016).

Moreover, results from experiments on IOR uniform and non-uniform configurations indicate that none of developed strategies have been able to generate the best results for both configurations. ITTDG produces the smallest test suit size for only uniform configurations while Density produces the smallest test suite size for most of the non-uniform configurations (Othman & Zamli, 2011a). This calls for an opportunity for other researchers to design better strategies.

@This item is protected by original copyright

In addition, generating a near-optimal test suite size has fall under NP-hard problem (Alsewari & Zamli, 2014; Yeh & Zamli, 2011). It is hard to find one single strategy that able to produce optimal results at all times (Alsewari & Zamli, 2011). Therefore, it has become motivation to other researchers to introduce new strategies.

Hence, this thesis proposed a *t*-way test suite generation strategy, TTSGA. The strategy is developed to cater all three types of support interactions; uniform strength, variable strength and IOR. Software testers have the flexibility in choosing which support interaction to use. The strategy employs metaheuristic algorithm, Ant Colony Optimization to assists in generating a near-optimal test suite size.

1.4 Objectives

The objectives of this research are:

i. To investigate the current progress of *t*-way testing that support uniform strength, variable strength and IOR *t*-way testing.

ii. To design a T-way Test Suite Generation Strategy based on Ant Colony algorithm (TTSGA) that supports the interaction of uniform strength, variable strength and IOR *t*-way testing.

iii. To evaluate the performance of the strategy against other existing *t*-way strategies in terms of generated test suite size using benchmarked experiments.

@This item is protected by original copyright

1.5 Scope of Study

Software Testing Life Cycle (STLC) is a sequential process that consists of six phases (Dalal & Chhillar, 2012). The phases are requirements analysis, test planning, test case development, environment setup, test execution and test cycle closure as depicted in Figure 1.1. In the requirements analysis phase, system requirements is thoroughly reviewed and analysed. Next, a test plan is prepared which includes strategies, resources, responsibilities and approach to be implemented. Test cases are designed, created and validated during test case development. After this stage, environment of the testing such as software and hardware is installed and setup. Subsequently, test cases are executed in the test execution phase. The last phase, known as test cycle closure, test results are evaluated and analysed.

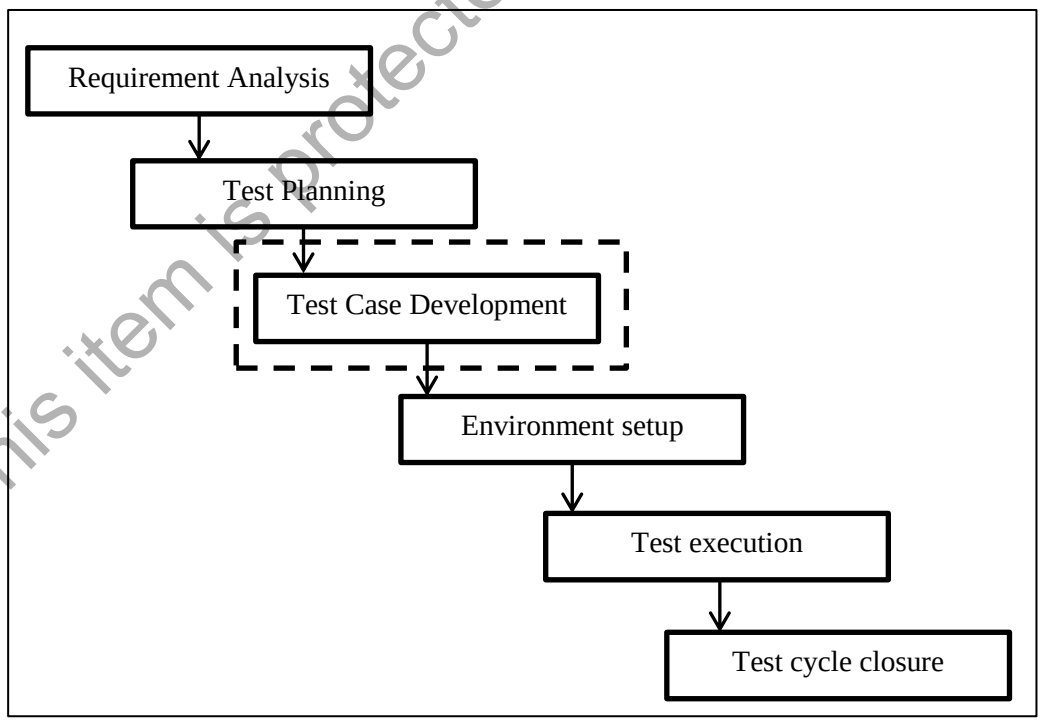


Figure 1.1 Software testing life cycle

@This item is protected by original copyright

All the phases in the STLC are important to ensure testing meets it objectives. Specifically, this research focuses on the test case development phase and emphasises on designing test suite generation strategy by applying one-test-at-a-time (OTAT) approach and metaheuristic search technique. The strategy supports uniform strength, variable strength and IOR *t*-way testing to produce near-optimal test cases.

1.6 Research Contribution

This research contributes towards the design, implementation and evaluation of a new strategy named TTSGA. The strategy supports all three types of support interactions. The support interactions are uniform strength, variable strength and IOR.

Besides that, the strategy contributes on the use of metaheuristic search technique towards IOR support interactions. As metaheuristic search technique is still new for IOR support interactions, a new fitness function is introduced. The formula for fitness function could support not only IOR, but also uniform and variable strength.

1.7 Thesis Outline

This thesis comprises five chapters. The subsequent sections of this thesis are presented as follows. Chapter 2 provides a review of the strategy approach for *t*-way testing, search techniques used by *t*-way testing strategies and types of interactions supported. Existing strategies in the literature are also presented in this chapter.

@This item is protected by original copyright

In Chapter 3, the design and implementation of the new strategy, TTSGA is discussed. Components of TTSGA and algorithms for each component are described in details in this chapter including any formula used by the strategy. Parameter settings used during execution are also being presented in the chapter.

Results and discussions are explained in Chapter 4. The results of experiments are divided into types of support interactions which are uniform strength, variable strength and IOR. This chapter also includes comparisons between TTSGA and existing strategies for all the three types support interactions.

Last but not least, Chapter 5 offers conclusion and summary of the entire research. Besides that, the chapter proposed recommendations of possible future research direction.

@This item is protected by original copyright

CHAPTER 2 : THEORETICAL CONCEPTS & LITERATURE REVIEW

2.1 Introduction

Overview of software testing has been presented in the previous chapter. In addition of that, software test case design techniques including some examples of the techniques have been discussed. One of the techniques is *t*-way testing which is the main topic of this research. As discussed earlier, *t*-way which involves interactions of parameter inputs is able to overcome exhaustive testing problem.

To further elaborate on *t*-way testing, this chapter explains the theoretical concepts of *t*-way testing. It starts with a case study to support explanation of *t*-way testing. After that, the chapter explains in detail on the structure and characteristics of *t*-way testing. There are 18 existing *t*-way testing strategies from the literature presented in this chapter. Finally, a brief analysis and discussion related to existing *t*-way testing is discussed at the end of the chapter. The theoretical concepts, strength and drawbacks from existing strategies, problems faced by the software tester are the basis to develop new test suite generation strategy.

2.2 *T*-way Testing Overview

T-way testing involves Software Under Test (SUT) to be tested. To further discuss on *t*-way testing, consider an application of College Consultation System as an example. The system suggests which programme is suitable for the students based on their examination results. Table 2.1 illustrates subjects taken by the students on the

@ This item is protected by original copyright

previous examination. The subjects will be used as input to the College Consultation System. Results for subject Science, Mathematics, Physics and English are pass or fail. To ease the discussion, a representation of parameters and its values is used as in Table 2.2. The subjects are represented by letter 'S' for Science, 'C' for Physics, 'M' for Mathematics and 'E' for English and the values (i.e. pass or fail) are represented by lowercase letter of input parameters and a number. For instance, s1 is referring to input parameter 'S' and the value is 1. For College Consultation System input as in Table 2.1, s1 is referring to Science and the value is "pass". Another possible value for input parameter 'S' is s2 which means Science and the value is "fail".

Table 2.1 College Consultation System input

Science	**Physics**	**Mathematics**	**English**
pass	pass	pass	pass
fail	fail	fail	fail

Table 2.2 Representation of SUT's input parameters and its values

Input Parameters	**S**	**C**	**M**	**E**
Values	s1	c1	m1	e1
	s2	c2	m2	e2

From the information given, the exhaustive combination of testing is generated to see the performance of *t*-way testing. The exhaustive combinations of the SUT occurred by having full strength of interactions. Full strength interactions means all parameters and their values interact with each other. For the College Consultation System, it consists of four input parameters. Thus, the full strength of interaction, *t* is 4.

@This item is protected by original copyright

Table 2.3 shows the exhaustive combination of the College Consultation System. There are 16 test cases generated from the exhaustive testing.

Table 2.3 Exhaustive combination of the SUT

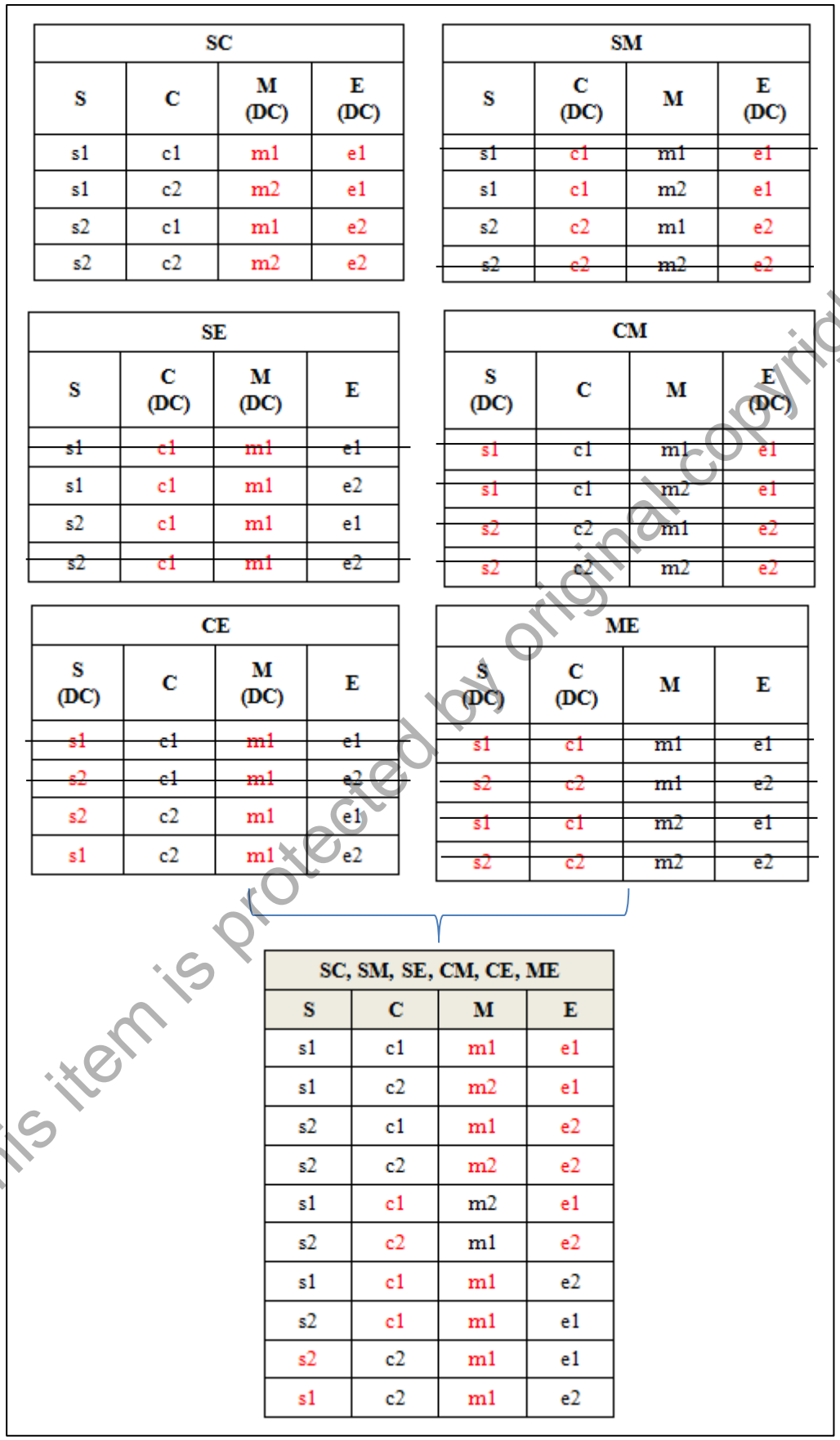
S	**C**	**M**	**E**
s1	c1	m1	e1
s1	c1	m1	e2
s1	c1	m2	e1
s1	c1	m2	e2
s1	c2	m1	e1
s1	c2	m1	e2
s1	c2	m2	e1
s1	c2	m2	e2
s2	c1	m1	e1
s2	c1	m1	e2
s2	c1	m2	e1
s2	c1	m2	e2
s2	c2	m1	e1
s2	c2	m1	e2
s2	c2	m2	e1
s2	c2	m2	e2

@This item is protected by original copyright

The strength of interactions in Table 2.3 can be relaxed from $t = 4$ to $t = 2$. By relaxing the strength of interaction to 2, the interaction only combines two input parameters compared to four input parameters. For $t = 2$, each input parameter is paired to other input parameter. The pairs are SC, SM, SE, CM, CE and ME. $t = 2$ is also known as pairwise testing. The purpose of reducing the strength of the interaction is to reduce the number of generated test cases.

For instance, parameter S and C is paired for pairwise interaction testing and become SC. Each value of S (i.e. s1 and s2) is paired to each value of C (i.e. c1 and c2). Therefore, SC combination produces four tuples. The tuples are s1 c1, s1 c2, s2 c1 and s2 c2. To complete a test case, parameter M and E must also have a value. Because of M and E has no relation with SC combination, they take a "don't care" (DC) value. The DC value is any value from the respective parameter. For instance, parameter M consists of two values, m1 and m2. One of the values is chosen as the DC value. The same technique also is used for other pairs of pairwise testing. Figure 2.1 shows the pairwise testing of the College Consultation System. The generated test cases are reduced to only ten test cases as compared to the exhaustive combinations. Pairwise testing is an example of uniform strength support interactions and will be explained in detail in Section 2.3.3.1.

@This item is protected by original copyright



@This item is protected by original copyright

Figure 2.1 Pairwise testing of the SUT

Besides pairwise testing, interactions also may involve more than one strength. It is known as variable strength. Variable strength is explained in detail in Section 2.3.3.2.

For an example of variable strength, revisit Table 2.2 and implemented with multi strengths, $t = 2$ and $t = 3$ for different input parameter. The strength and input parameters are as follows:

i. Strength, t=2 for parameter S, C, M and E
ii. Strength, t=3 for parameter S, C and M

Figure 2.2 illustrates the variable strength interactions for the SUT. Interactions for strength, $t = 2$ is taken from Figure 2.1 as both cases have similar parameters and strength. After removing redundant test cases, the results produce 14 test cases. The result is less than number of test cases generated by exhaustive testing (i.e. 16) as in Table 2.3.

@This item is protected by original copyright

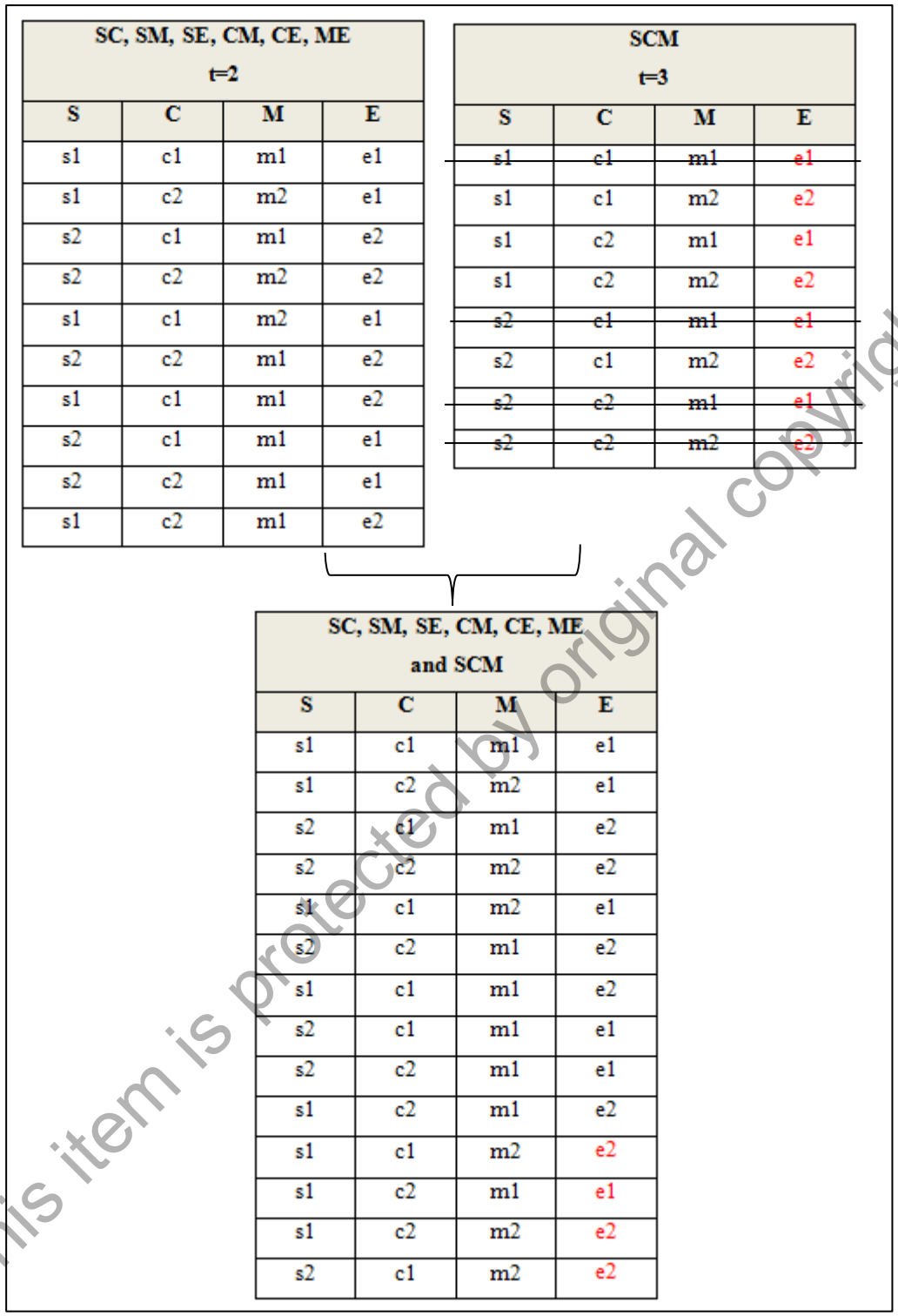


Figure 2.2 Variable strength interaction testing of the SUT

@This item is protected by original copyright

Uniform strength and variable strength interaction have shown a reduction number of test cases. Another possible condition involves input-output relationships or known as IOR. Some of the SUT's characteristics involve specific interaction of parameter inputs to produce an output. In this situation, software tester must have knowledge of the system, particularly the combination of input parameters which will affect an output. Similar SUT characteristic as uniform and variable strength shown in Table 2.1 and Table 2.2 will be used to further discuss for input-output relationships. Assume that College Consultation System offers three programmes. The programmes are Computer Science, Applied Science and Engineering. The system will suggest for:

i. Computer Science programme if the student passes Science and Mathematics subject
ii. Applied Science programme if the student passes Science and English
iii. Engineering programme if the student passes Physics, Mathematics and English subject

The programmes offered by the college are the outputs of the system and represented by f(1) for Computer Science, f(2) for Applied Science and f(3) for Engineering. The relationship of each output and their inputs combination are as follows:

i. f(1) is a function output of parameter S and M
ii. f(2) is a function output of parameter S and E
iii. f(3) is a function output of parameter C, M and E

@This item is protected by original copyright

The IOR interaction based on the suggestion of the College Consultation System programmes is shown in Figure 2.3. The interactions only cater for function output f(1) until f(3). It handles only interactions based on function outputs and ignores other unrelated input parameters or interactions. Similar to uniform and variable strength, the DC value is used to any unrelated input parameters to complete the test case. Only 10 test cases out of 16 test cases need to be performed. The discussion on input-output relationship will be explained in detail in Section 2.3.3.3.

@This item is protected by original copyright

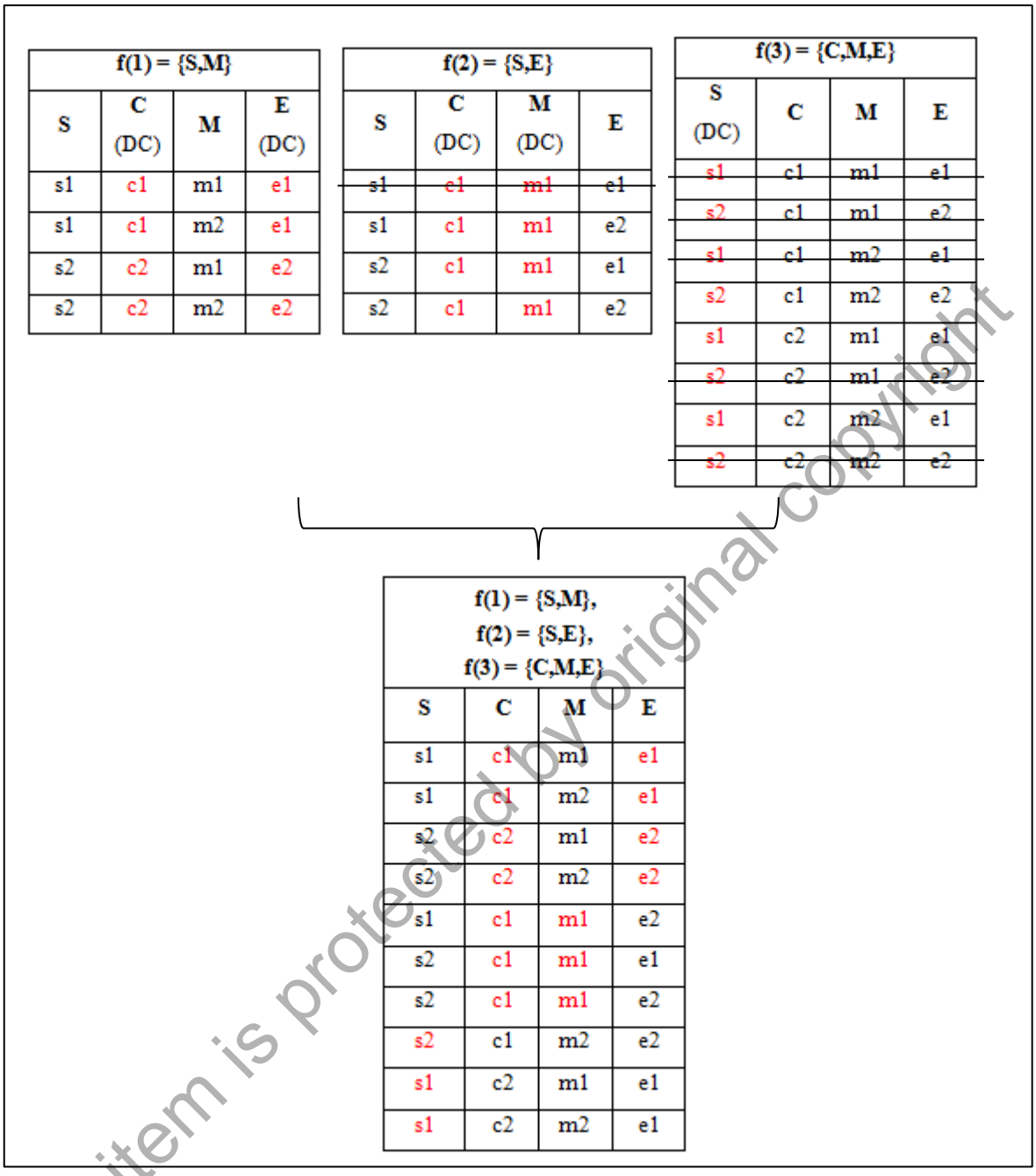


Figure 2.3 IOR Interactions for SUT

@This item is protected by original copyright

The examples discussed in this section are an overview of *t*-way testing and how it works briefly. *T*-way testing structure including the type of interactions discussed earlier will be further described in the next section.

2.3 *T*-way Testing Taxonomy

T-way testing focuses on faults caused by interaction of input parameters. It combines interacted input parameters based on few conditions. Each combination must be covered by at least one test case (Shiba, Tsuchiya, & Kikuno, 2004). *T*-way testing has its own taxonomy to distinguish from other types of test case design techniques.

Figure 2.4 shows taxonomy of *t*-way testing. The taxonomy consists of three factors, which are strategy approach, search technique and support interactions. The strategy approach encompasses of one-parameter-at-a-time (OPAT) and one-test-at-a-time (OTAT). The second factor is search technique which provides computation or metaheuristic technique. To complement *t*-way testing's factor, there are three types of support interactions consisting uniform strength, variable strength and IOR.

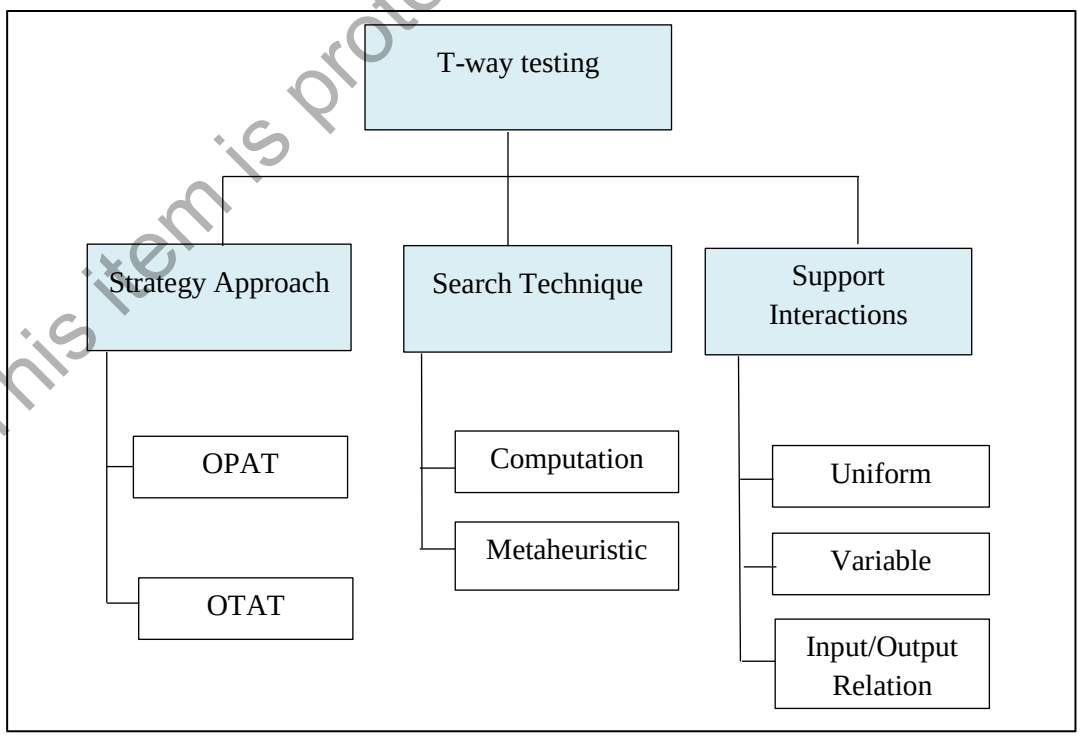


Figure 2.4 *T*-way testing taxonomy

@This item is protected by original copyright

2.3.1 *T*-way Testing Strategy Approach

T-way testing generation algorithm can be divided into two categories. They are parameter based generation or one-parameter-at-a-time (OPAT) and test based generation or one-test-at-a-time (OTAT). Both approaches have different technique in order to produce test suite.

The OPAT strategy generates test suites for *t*-way combination which involves horizontal and vertical extensions. It starts with *t* input parameter to generate the test case. Then, OPAT extends one parameter at a time to complete the test case. This is known as horizontal extension. If there is a combination that has not yet been covered, OPAT needs to extend a new test vertically.

In contrast, OTAT strategy generates one complete test case at a time. Each test case is supposed to cover as many *t*-way combinations as possible for all parameters. The number of rows in the test suite increases as test cases increases. OTAT is more popular among researchers than OPAT in developing tools or algorithm (Khalsa & Labiche, 2014; Nasser, Alsewari, & Zamli, 2018; Zamli et al., 2016).

2.3.2 T-way Testing Search Technique

There are a few search techniques has been explored by researchers. In this sub-section, the popular search techniques will be discussed. They are computational and metaheuristic search technique.

@This item is protected by original copyright