A HYBRID INPUT-OUTPUT BASED RELATION TEST SUITE GENERATION STRATEGY

By

ZIDAN SALEM ALI MOUSSA
1632322002

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science (Embedded System Design Engineering)

School of Computer and Communication Engineering
UNIVERSITI MALAYSIA PERLIS

2017

UNIVERSITI MALAYSIA PERLIS

DECLARATION OF THESIS

Author's full name	:	**ZIDAN SALEM ALI MOUSSA**
Date of birth	:	02 / 03 / 1973
Title	:	**A HYBRID INPUT-OUTPUT BASED RELATION TEST SUITE GENERATION STRATEGY**
Academic Session	:	**2016/2017**

I hereby declare that this thesis becomes the property of University Malaysia Perlis (UniMAP) and to be placed at the library of UniMAP. This thesis is classified as:

☐ **CONFIDENTIAL** (Contains confidential information under the Official Secret Act 1972)

☐ **RESTRICTED** (Contains restricted Information as specified by the organization where research was done)

[✓ **OPEN ACCESS**] I agree that MY thesis is to be made immediately available as hard copy or online open access (full text)

I, the author, give permission to the UniMAP to reproduce this thesis in whole or in part for the purpose of research or academic exchange only (except during a period of ____ years, if so requested above).

Certified by:

SIGNATURE	**SIGNATURE OF SUPERVISOR**
(1632322002 / 947866)	**Dr. Rozmie Razif Othman**
(NEW IC NO. / PASSPORT NO.)	**NAME OF SUPERVISOR**
Date:	Date:

NOTES: ***** If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentially or restriction.

ACKNOWLWDGEMENT

I would first of all like to thank ALLAH to support me to finish my dissertation, also I would like to thank my desertion advisor A hybrid input-output based relation test suite generation strategy for Application Testing, Dr. Rozmie Razif Othman of the school of computer and communication systems engineering at University Malaysia Perlis. The door to Dr. Othman office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this dissertation to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge all School of computer and communication systems engineering staff and classmateat Unimapfor their help and support me to do best, and I am gratefully indebted to my Dean and chairman school, prof. Dr. Hadramy and Dr.Hasliza Rahem for their help and guide me in all my study.

Finally, I must express my very profound gratitude to my parents and to my wife, children for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NO. **PAGE**

LIST OF ABBREVIATIONS

ACTS	Advanced Combinatorial Testing Suite
CPU	Central Processing Unit
AETG	Automatic Efficient Test Generator
AI	Artificial Intelligence
CA	Covering Array
DDA	Deterministic Density Algorithm
DOE	Design of Experiment
GA	Genetic Algorithm
GA-N	Nie Implementation of GA
GT Way	Generalized T-Way Test Suite Generator
GUI	Graphical User Interface
IOR	Input-Output Based Relations Covering Array
IPO	In Parameter Order
IPOG	In-Parameter-Order-General
IPOG-D	In-Parameter-Order-General Double
IPOG-F	Heuristic Based IPOG
IPOG-F2	Heuristic Based IPOG version 2
IPO-N	Nie Implementation of IPO
ITTSG	Integrated T-Way Test Suite Generator
MAETG	Modified Automatic Efficient Test Gene
MUX	Multiplexer
OA	Orthogonal Array

OATS	Orthogonal Array Test System
OPAT	One Parameter At A Time
OTAT	One Test Case At A Time
Paraorder	Parameter Order
PICT	Pairwise Independent Combinatorial Testing
PSO	Particle Swarm Optimization
RAM	Random Access Memory
Reqorder	Requirement Order
SA	Simulated Annealing
SUT	System Under Test
TC	Test Configuration
TCGTest	Case Generator
TVG	Test Vector Generator
VCA	Variable – Strength Covering Array

Analisis Hibrid Berasaskan Input-Output Berdasarkan Ujian Hibrida Untuk Ujian Aplikasi Berasaskan bstrak

ABSTRACT

Oleh sebab punca ruang pengiraan yang besar, data ujian sampel sistematik mempunyai penyelidikan aktif untuk memenuhi strategi ujian t-cara dalam tempoh 20 tahun yang lalu. Oleh itu, bahawa (t) menunjukkan kekuatan interaksi. Walaupun terdapat kegunaan, penggunaan strategi t-cara sedia ada dalam perisian tidak akan digunakan di kalangan komuniti kejuruteraan. Mengikut kaji selidik mengenai strategi t-cara sedia ada, interaksi t-cara yang seragam ditentukan di mana interaksi kekuatan berubah-ubah dikenakan dalam beberapa strategi. Oleh kerana hubungan berasaskan input-input terdapat juga strategi yang menetapkan interaksi Meningkatkan kebolehgunaan dan menaik taraf penggunaan strategi t-cara sedia ada, terdapat keperluan untuk strategi yang lebih fleksibel dan mampu untuk melibatkan semua bentuk kemungkinan interaksi Ujian Jurutera (sebagai pakar domain) harus mempunyai peluang memilih dari berbagai kemungkinan interaksi. Dalam hal ini, jurutera ujian boleh bersedia untuk menghasilkan kesilapan kreatif mereka bergantung kepada masalah pengujian. Tidak sampai baru-baru ini, beberapa strategi telah muncul untuk muncul (mis. Density, ParaOrder dan TVG). Walau bagaimanapun, sebagai generasi ujian t-cara dianggap sebagai masalah serius NP, Tidak ada strategi tunggal yang tunggal yang dapat menguasai sejauh ukuran ujian ukuran optimum. Motivasi cabaran yang disebutkan di atas, kajian ini akan membentangkan strategi t-cara baru yang dinamakan ITTSG yang dapat menyokong semua kemungkinan interaksi termasuk kekuatan seragam, kekuatan ubah, dan input berdasarkan output.

A HYBRID INPUT-OUTPUT BASED RELATION TEST SUITE GENERATION STRATEGY

ABSTRACT

The cause of large enumeration space the systemized sample test data have the active research to meet t-way testing strategies during the previous 20 years Hence that (t) indicates to interaction strength Despite of usefulness, the adoption of existing t-way strategies in the software lacks to be used among engineering community According to a survey on existing t-way strategies, that uniform t-way interactions are dictated whereas variable strength interaction are imposed in some strategies input-output based relationships there also strategies that prescribe interactions Improving the applicability and upgrading the existing t-way strategies adoption there is a requirement for a strategy which is more flexible and is capable to engage all form of possibilities of interaction. The test engineer (as domain experts) should be having the chance to select from various interaction possibilities in this regard, test engineer can be prepared to generate their creative-ambling relying on the testing problem Not until recently a number of strategies have set out to have appeared (e.g. Density, ParaOrder and TVG). However, as t-way test suite generation is considered as NP had problem, no existing strategy can dominance as far as test size of optimality size is concerned. The Motivation of the aforementioned challenge this study will present t-way strategy which is called ITTSG that can support all possibilities of interaction includes input output based relationship further of the t – way test suite has been tested in application.

CHAPTER 1

INTRODUCTION

1.1 Overview

The Modernized software is basically designed to support different platforms and as well as environments. In various cases, the same software is normally being installed in different platforms and with different hardware setup. As such, the software is sometimes upgraded so that it can be customized to be suitable for different requirements of users and their needs. In regard of flexible and highly configurable software engineers have to develop and also accommodate such as requirements, software.

Nonetheless, as software become have high configuration, they tend to be more complicated in terms of design, number of inputs, size and implementation. Here, there are crucially many intertwined dependencies between different parameter inputs, thus, exposing tremendous software fault possibilities. For these aforementioned rationales, software testing becomes an important stage in software developmental lifecycle which Consumes 40 to 50 percent of cost of software development software testing is one of the most resource 1 consuming activity in software development lifecycle. While contributing importantly to cost of software development, lack testing may lead to greater cost. In the line with published report by National Institute for Standards, lack of software testing methods and tools in United States cost between 22.2 to 59.5 billion US Dollars every year It should be into consideration that the loss figure excludes the losses in mission critical software as it is impossible to assign value for the live loss. Given this large magnitude of problem, software testing practices and strategies desperately need even a small enhancement for contribution. Executing a program with the intent to find error is a common practice in software testing stage. With multiple number of inputs, several execution processes with different configurations of inputs (also refer as test cases) are typically required to test a particular software of interest. Therefore, to aim for bug free software, a set of quality test cases (i.e. called test suite) must be first generated.

Nowadays, software usage is entering into every corner of social life, from the television remote control and mobile phone applications to the airplane control systems. Smart software facilitates our daily chores; however, unwarranted glitches can lead to undesirable consequences. Software testing is the main activity for ensuring the quality of any software (i.e. ensuring that the software meets the user's needs, business and technical requirements. Software testing refers to the process of finding errors/defects and/of ensuring that a particular software of interest meets its specification. Perhaps, the ideal way for testing any software system is to test all possible combinations of input parameters. However, exhaustively testing all possible input/output possibilities are extremely difficult in practice owing to the huge number of combinations needs to be considered. Often, a sampling mechanism is needed to find a subset of effective test case for testing consideration. As a result, a number of test design techniques have been proposed to reduce the size of test cases and save time and effort such as Random. Recently, researchers are turning to t-way testing (Ahmed & Zamli, 2011), as complementary alternatives to the aforementioned sampling techniques. The central concern of many researchers into t-way testing is on how to obtain the minimal number of test cases. One of the interesting properties of t-way testing is that all interaction possible of input values with strength t are involved in generated test suite.

Many good techniques for testing generation cases (or loosely called strategies) have been proposed in literature (e.g. equivalence class partitioning, cause and effect graphing, boundary value analysis and interaction testing (also refer as *t*-way testing)) (Zamli et al., 2008). Equivalence class partitioning technique is a test case generation strategy that can only be used when the inputs and outputs of a system under test (SUT) are well-defined for every input, the possible values will be separated into different partitions in a manner that values within the same partition will be processed equivalently by SUT (Hass, 2008, Sharma and B., 2010). Test suite will be generated in a way that every partition will be tested for at least once. While test suite is equivalently generated by class partitioning technique assumes which faults are occurring within the equivalent partition, some faults can be occurred at the boundary of a partition.

Complementing equivalent class partitioning, boundary value analysis stresses on testing the value at the beginning and the end of partition (i.e. the boundary of a partition). For instance, consider a temperature controller system which activates the cooling device when temperature is higher than 20 ºC. Here there are two partitions which are considered cold partition for the temperature values less than or equal to 20 ºC (i.e. temperature $\leq$ 20 ºC) and hot partition for temperature values greater than 20 ºC (i.e. temperature > 20 ºC). For cold partition, the boundary value is 20 ºC while, for hot temperature being, the boundary value is 21 ºC. Thus, the finalized test suite should be included 20 ºC and 21 ºC immediately.

As for cause and effect graphing, this test cases generation technique is being adopted from hardware testing technique by Elmendorf and further developed by others Cause and effect graphing is attributive to a directed graph that makes a set of causes (i.e. input conditions of input) to a set of effects (i.e. the expected outputs) for a particular SUT (Naik and Tripathy, 2008). From the graph, a decision table can be constructed to systematically represent possible combinations of inputs with the expected outputs. The test cases can be derived from the decision table where the columns of the decision table (i.e. possible combinations of input conditions) which are converted into cases of tests.

Unlikely the aforementioned test cases generation techniques (i.e. equivalence partitioning, boundary value analysis and cause and effect graphing) that stresses on testing individual input parameters, testing interaction (also refers as t-way testing) which is used to discover bugs cause of input parameters interaction (Ahmed and Zamli, 2011a). In *t*-way testing, test cases are generated to cover every combination of *t* parameters (where *t* refers as strength of interaction) for at least once (Younis and Zamli, 2009b, Ong and Zamli, 2011b, In this regard, every possible combination of *t* input parameters is accordingly being tested As the success of software testing importantly relies on the effectiveness of generated test cases, advancement in test cases generation technique can make a somewhat contribution in s field of software testing . Motivated by this challenge, this research work focuses on test cases generation technique (or strategy) that is depended on interaction testing. In the next section, this chapter will raise the problem statements and the thesis roadmap.

1.2 Problem Statements

An increasing of customer demands and the advancement technology of hardware and computing (i.e. processing power and the storage capacity), existing software system has grown enormously in terms of size (i.e. code line (LOC)) and functionalities. And the trend in software development often accommodates many interconnected parts within a single program in order to produce a large integrated software system. For these causes, several combinations of input parameters, hardware configurations, software configurations and system platforms require to be tested and verified against the certain requirements a widely cited reports that inadequate testing methods and tools annually cost the U.S. economy between $22.2 and $59.5 billion, with roughly half of these costs borne by software developers in the form of extra testing and half by software users in the form of failure avoidance and mitigation efforts. The same study notes that between 25 and 90 percent of software development budgets are often spent on testing. This posting, the first in a two-part series, highlights result of an analysis that documents problems that commonly occur during testing. Specifically, this series of posts identifies and describes 77 testing problems organized into 14 categories, lists potential symptoms by which each can be recognized, potential negative consequences, potential causes, and makes recommendations for preventing them or mitigating their effects. To test all combinations which are possible, full testing is perhaps the most effective testing strategy. However, due to constrain factors of time and cost, full testing is impractical as well as the cost is not efficient Here, t-way strategies assist to search and to produce a set of sampled test cases to finish the test suite that covers the required interaction strength from a typically large possible test values space.

drawbacks in order to enhance applicability. Within the context of t-way strategies, test engineers (as domain experts) have often not given enough flexibility to select amongst all possibilities of interaction from a single t-way strategy. Some strategies dictates only, called Integrated T-way Test Suite Generation (ITTSG) which incorporate all possibilities of interaction including uniform strength, variable strength, and input-output based relations.

As compared the empirical evidence existing strategies appoints that ITTSG produces competitive test size on a number of benchmark configurations.

1.3 Aim and Objectives

The aim of this dissertation to improve the performance of ITTSG by hybridising it with simulated annealing. To achieve this aim, the following objectives adopted:-

1. To improved ITTSG strategy performance in terms of generated test suite case by hybridizing it will simulated annealing.
2. To evaluate the performance of proposed strategy against other competing strategies regarding the generated test suite size.
3. To implement the proposed strategy application.

1.4 Research Scope

1.5 The software testing cycle typically consists of three main stages which are test planning, test execution and test evaluation as shown in Fig. 1.1.

This dissertation will focus on test planning stage (i.e. test suite generation). especially, this research does focus on the improve and evaluate of a t-way strategy enable of supporting interaction of uniform strength, the generating of the test suite variable strength interaction and input-output based relations. the generated test suite idly intended to be executed during the execution test stage and assessed during evaluation of test.

1. Test planning: plan for future testing.
2. Test evaluation: type of assessment in that seek to determine is the best possible for the needs of given.
3. Test execution: the processes of execution of the code comparing expected and actual results.

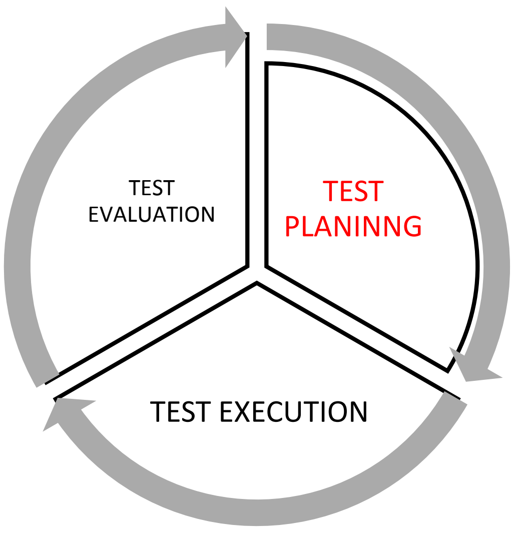

Figure 1.1:Software Testing Cycle.

1.6 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 will present a survey on recently proposed t-way strategies which recently have been proposed. The survey will start with the common characteristics introduction which are shared among t-way strategies. Then, each introduced characteristic will be presented its importance discussed. Thereafter, the chapter will present a survey on recently proposed t-way strategies based on the characteristics introduced. Towards the end, an accurate analysis on existing t-way strategies will be presented. Here, the strengths, crucial omissions, and possible enhancements of each surveyed strategy will be discussed, thereby providing the requirements and justification for the ITTSG development ITTSG. Chapter 3 will discuss the implementation of ITTSG strategy in details.

The discussion will cover the algorithms data structure of ITTSG strategy as well as. In addition to Chapter 3 will also present the system architecture for simulated annealing deploying ITTSG this chapter will also elaborate the ITTSG + SA prototype implementation in order to highlights its use. Chapter 4 presents will finally evaluate ITTSG performance (i.e. regarding to generated test suite size) as well as the generated test suite effectiveness. Here, ITTSG as uniform strength the performance, variable strength and input-output based relations will be benchmarked against publicly available known strategy implementations. Then, the applicability of uniform strength, variable strength and input-output based relations test suite generated by ITTSG is demonstrated using two case studies involving software and hardware design. Finally, the conclusion of this research work is presented in Chapter 5. Here, the achievements, contributions and problems of this research work are summarized. In addition, the characteristics introduced in Chapter 2 are revisited in order to appropriately position ITTSG with its existing counterparts. In the end, this chapter highlights the possibilities for future.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The previous chapter presents the importance of software testing. It also outlines the purpose of the study as well as it additionally summaries that formulated the problem, Research objectives, Research question. Finally, the significance of t-way has been highlighted.

In recent decades, various research has been conducted on combinatorial test data generation strategy with various approaches to provide solution to combinatorial combination problem of test suit generation (Nie *et al.* 2015) while most of the available combinatorial test suit support uniform and variable interaction strength, however all this are found helpful but the inability to perform non-uniform cases in a situation where system configuration define some certain parameters interaction which can be counted as Input-Output based relations(IOR) needs to be observed.

There are distinct approaches to IOR and there are some existing work related to IOR which include Density (Bryce et al., 2007), TVG Arshem, 2012), Greedy (Korel and Schroeder, 2000), union (Schroeder, 2001), AURA(Ong and Zamli , 2011), ITTSG(Othman and Zamli , 2011) and Para Oreder (Ziyuan, Changhai et al. 2007). However, other recent existing work try to approach IOR in different ways by adopting nature based algorithm in order to construct a combinatorial test suite, typical examples are genetic algorithm (Shiba et al. 2004), Simulated Annealing (Cohen, et al. 2013) and Particle Swarm Test suite Generation (Jarboui, et al. 2007).

Nature based strategy such as simulated annealing inspired test suite strategy to support non-uniform and uniform interaction including variable interaction strength and will allow optimum satisfactory result in many IOR cases such as when utilizing it for

2.2 Incorporating T- way to Embedded system testing

An embedded system is some combination of computer hardware and software either fixed in capability or programmable, that is designed for a specific function or for specific functions within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys as well as mobile devices are all possible locations for an embedded system for example for embedded software system.

The PlayStation Controller is the first gamepad released by Sony Computer Entertainment for its PlayStation video game console. The original version (model was released alongside the PlayStation Using the simple geometric shapes of a green triangle, a red circle, a blue cross, and a pink square (Triangle, Circle, Cross, Square) triangle =a, circle =b, cross= c and square = d to label its action buttons rather than traditionally used letters or numbers, the PlayStation Controller established a trademark which would be incorporated heavily into the PlayStation brand.

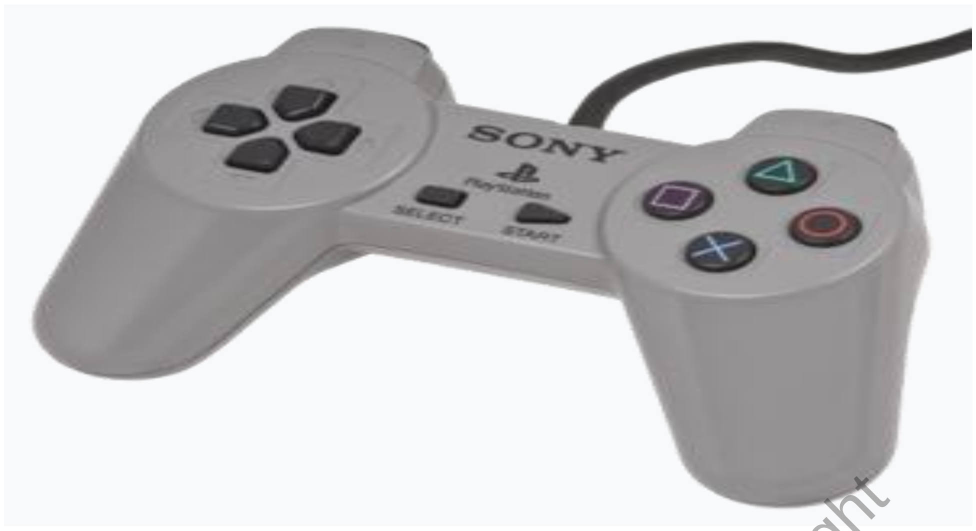

Figure 2.1: The Original PlayStation Controller.

In order to test this 4 button PS4 controller 16 test cases are required to be executed the test cases are show the Table 2.1. data values.

Table 2.1: Data Values.

Input parameter	A	B	C	D
value	Press (a1)	Press b1	Press C1	Press d1
	Not Press (a0)	Not Press b0	Not Press C0	Not Press d0

Table 2.2: Excusive Test Cases.

	A	B	C	D
1	a0	b0	c0	d0
2	a0	b0	c0	d1
3	a0	b0	c1	d0
4	a0	b0	c1	d1
5	a0	b1	c0	d0
6	a0	b1	c0	d1
7	a0	b1	c1	d0
8	a0	b1	c1	d1
9	a1	b0	c0	d0
10	a1	b0	c0	d1
11	a1	b0	c1	d0
12	a1	b0	c1	d1
13	a1	b1	c0	d0
14	a1	b1	c0	d1
15	a1	b1	c1	d0
16	a1	b1	c1	d1

While 4 button PS4 required 16 test cases to completely test it larger number of test must as result this thesis will discuss the potential of implementing t – way testing for embedded system component testing from literature, t – way testing known to reduce number of test cases show Table 2.2 way interaction tuples is need for test such as (A ,B