Design of a New T-Way Strategy for Test Case Generation Supporting Sequence-Less and Sequence Input Interaction

©This item is protected by original copyright

by

Md Mostafijur Rahman

(1040210511)

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

School of Computer and Communication Engineering
UNIVERSITI MALAYSIA PERLIS

2017

ACKNOWLEDGEMENT

Alhamdulillah, all praises to Allah (subhanahu wa ta'ala) who gives me the ability to complete this research work.

I would like to extend my gratitude to my respectful supervisors Dr. Rozmie Razif Othman, Professor Ir. Dr. R Badlishah Ahmad and Dr. Md Mijanur Rahman. Without them this research and thesis would not exist. They have been a continual source of support and motivation and have provided me with insights from differing points of view. Their excellent guidance, motivation, caring, patience, and providing me with an excellent facilities and environment for doing this research. A special thanks to Dr Rick Kuhn, NIST, USA for his prompt help to share some valuable documentation on T-way testing.

I would like to thanks all of my colleagues have worked closely with me at Embedded Computing Research Cluster as well as all staff of the School of Computer and Communication Engineering, Universiti Malaysia Perlis, for the help and sharing knowledge and ideas during my research. My appreciation also goes to laboratory technicians who help me to use the laboratory equipments.

Thank you very much to my beloved wife, for her never ending support and always stands by beside me. I would like to express my gratitude and happiness to my parents, brothers, sisters, brother-in-law (Assoc. Prof. Dr. Fazlul Bari) and all of my relatives for their continued support, inspiration, patience and love.

Last but not least, thanks goes to those whoever has helped me either directly or indirectly in accomplishment of my research.

Md Mostafijur Rahman
Universiti Malaysia Perlis (UniMAP)
mostafijur.unimap@gmail.com

© 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 FIGURES

©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 ABBREVIATIONS

ASP	Answer Set Programming
ACTS	Advanced Combinatorial Testing Suite
AI	Artificial Intelligence
ACA	Ant Colony Algorithm
ACS	Ant Colony System
AETG	Automatic Efficient Test Generator
AETG-SAT	Automatic Efficient Test Generator using SAT solver
BA	Bee Algorithm
BTS	Bat-Inspired t-way Strategy
BTI	Banbara, Tamura, and Inoue method
BR	Brain method
CA	Covering Array
CAD	Computer-Aided Design
CIIT	Combinatorial Input Interaction Testing
CE	Combinatorial Explosion
CP	Control Programming
CV	Condition Value
CSS	Cuckoo Search Strategy
D	Down/Lower Bund
D_R	Reverse Down/Lower Bund
ER	Erdem Encodings
GUI	Graphical User Interface
GTWay	Generalized T-Way Test Suite Generator
GIPOG	Grid In Parameter Order General
GA	Genetic Algorithm
GA-N	Nie Implementation of GA
HHH	High level Hyper Heuristic
HSS	Harmony Search Strategy

©This item is protected by original copyright

HS	Harmony Search
HMS	Harmony Memory Size
HM	Harmony Memory
HMCR	Harmony Memory Consideration Rate
ITTSG	Integrated T-Way Test Suite Generator
ISTQB	International Software Testing Qualifications Board
IPOG	In Parameter Order General
IPO	In Parameter Order
IPOG-C	In Parameter Order General with Constraint support
IPOG-D	In Parameter Order General Double
LAHC	Late Acceptance Hill Climbing
MII	Mix Input Interaction
MARS	Military Affiliate Radio System
MCA	Mix Covering Array
MIPOG	Modified In Parameter Order General
MC-MIPOG	Multi Core Modified In Parameter Order General
mAETG	Modified Automatic Efficient Test Generator
MDI	Multiple Document Interface
NIST	National Institute of Standard and Technology
OA	Orthogonal Array
OPAT	One Parameter At A Time
OTAT	One Test Case At A Time
PICT	Pairwise Independent Combinatorial Testing
PAR	Pitch Adjustment Rate
PSTG	Particle Swarm Test Generator
PSO	Particle Swarm Optimization
STLC	Software Test Life Cycle
SDLC	Software Development Life Cycle
SUT	System Under Test
SA	Simulated Annealing
SAT	Satisfiability

©This item is protected by original copyright

SCA	Sequence Covering Array
TVG	Test Vector Generator
TConfig	Test Configuration
TIIT	T-way Input Interaction Testing
TA	Tarui Method
TIDG	Test Input Data Generator
TTG	T-way Tuple Generator
TCG	Test Case Generator
TSIDG	Test Sequence Input Data Generator
TSTTG	T-way Sequence Tuple Tree Generator
TCG	Test Case Generator
TWIIT	T-way Input Interaction Testing
TID	Test Input Data
TID	Test Input Data
TSID	Test Sequence Input Data
U	Upper Bound
U_R	Reverse Upper Bound

©This item is protected by original copyright

Reka bentuk Strategi T-hala Baharu bagi Kes Ujian Generasi Penyokong Urutan Kekurangan dan Urutan Input Interaksi

ABSTRAK

Strategi t-hala adalah kombinasi interaksi ujian input kes penjana (CIIT) yang berkesan dan berjaya. input kombinasi ujian interaksi (CIIT). Dalam CIIT, interaksi boleh berlaku di antara nilai-nilai input parameter (input interaksi urutan kekurangan) atau di antara urutan input parameter (interaksi input urutan). Banyak strategi t-hala yang berguna telah dibangunkan dalam tempoh dua puluh tahun yang lalu. Banyak pembangunan kepada strategi t-hala yang sedia ada telah dipertimbangkan sama ada urutan-kurang (HHH, HSS, CSS, PSTG, SA, GA, BPR, BTS, LAHC, GA-N, IPO-N, AETG, mAETG, IPOG, MIPOG, ITCH, TVG, GTWay, Density, ParaOrder, PICT, TITG, dan lain-lain) atau urutan (BA, Kuhn pengekodan, ASP dengan Clasp, CP dengan Gula, ER, TA, BR, dan lain-lain) interaksi input. Dalam banyak scenario ujian, adakalanya perlu pengujian untuk kedua-dua urutan kekurangan dan urutan operasi dalam perisian yang sama di bawah ujian (SUT). Jika strategi t-hala dapat diperluaskan dan disediakan, satu strategi bersepadu akan dicari kerana ia melegakan jurutera ujian daripada bebanan mempelajari alat ujian yang lain. Tambahan pula, sebagai generasi ujian t-hala yang sesuai (untuk kedua-dua urutan-kekurangan dan urutan input interaksi), ia memiliki satu masalah iaitu NP-keras, tiada sebarang cabaran strategi untuk menjana bilangan minimum kes-kes ujian untuk setiap konfigurasi sistem. Menangani isu-isu yang dinyatakan di atas, kertas kerja ini mencadangkan satu strategi bersepadu yang baharu untuk menyokong urutan-kekurangan dan urutan input interaksi, yang dinamakan sebagai, t-hala ujian interaksi input (TWIIT) strategi. Strategi TWIIT yang dicadangkan terdiri daripada tiga modul, iaitu penjana data input ujian, penjana tuple dan penjana kes ujian. Modul input ujian penjana data menjana data ujian (kedua-dua untuk urutan-kekurangan dan urutan input interaksi), modul penjana tuple menjana t-hala tuple (untuk urutan-kekurangan input interaksi) dan t-hala pokok tuple (untuk urutan input interaksi) dan ujian modul kes penjana menjana ujian suite. Di samping itu, strategi TWIIT untuk interaksi input urutan-kekurangan menyokong kedua-dua nilai parameter yang seragam dan tidak seragam. Dalam kajian ini penilaian yang luas dengan penanda aras yang berbeza dan kes-kes eksperimen telah dibentangkan untuk menentukan kekuatan dan ancaman strategi yang dicadangkan. Keputusan menunjukkan bahawa, strategi TWIIT (untuk urutan-kekurangan dan urutan input interaksi) mampu menjana jumlah minimum bagi kes-kes ujian berbanding dengan strategi seangkatannya.

© This item is protected by original copyright

Design of a New T-way Strategy for Test Case Generation Supporting Sequence-less and Sequence Input Interaction

ABSTRACT

The t-way strategy is a successful and efficient combinatorial input interaction test (CIIT) case generator. In CIIT, the interactions may occur either between input parameter values (sequence-less input interaction) or between sequences of input parameters (sequence input interaction). Many useful t-way strategies have been developed in the last twenty years. Much of existing works on t-way strategy has considered either sequence-less (HHH, HSS, CSS, PSTG, SA, GA, ACA, BTS, LAHC, GA-N, IPO-N, AETG, mAETG, IPOG, MIPOG, ITCH, TVG, GTWay, Density, ParaOrder, PICT, TIITG, etc.) or sequence (BA, Kuhn encoding, ASP with Clasp, CP with Sugar, ER, TA, BR, etc.) input interaction. In many testing scenario, there is sometimes a need testing for both sequence-less and sequence operations within the same software under test (SUT). If t-way strategy is going to be pervasively made available, an integrated strategy is much sought after as it relieves the test engineers from the burden of learning another testing tool. In addition, as the generation of t-way test suite (for both sequence-less and sequence input interaction), it is an NP-hard problem, no single strategy challenges to generate minimum number of test cases for every system configuration. Addressing the aforementioned issues, this work proposes a new integrated strategy supporting sequence-less and sequence input interactions, named as, t-way input interaction test (TWIIT) strategy. The proposed TWIIT strategy consists of three modules, such as, test input data generator, tuple generator and test case generator. The test input data generator module generates test data (both for sequence-less and sequence input interaction), the tuple generator module generates t-way tuple (for sequence-less input interaction) and t-way tuple tree (for sequence input interaction) and the test case generator module generates test suite. In addition, the TWIIT strategy for sequence-less input interaction supports both uniform and non-uniform parameter values. In this research an extensive evaluation with different benchmarks and experimental cases has been presented to determine the strengths and threats of the proposed strategy. The results show that, the TWIIT strategy (for sequence-less and sequence input interactions) able to generate minimum number of test cases compared with its counterpart strategies.

©This item is protected by original copyright

CHAPTER 1

INTRODUCTION

1.1 Overview

In this century, the modern society is becoming fully dependent on software controlled systems. The multi-functional and complex designed software (e.g. 3D games, GUI based web applications, etc.) are growing tremendously with the function of multiple inputs. Therefore, adequate software testing is desired for software verification and validation while considering the multiple numbers of input interactions.

Inadequate software testing may cause blunders. In the past history, a number of software failures had led to embarrassment, massive financial losses, injuries and even death. In 1962, the mission control destroyed the rocket after 293 seconds of lift off. It costs around $18.5 million because of missing a single superscript bar in the program code (Xiong, 2011). In 1978, the Hartford coliseum was collapsed. It costs around $70 million and added $20 million damage to the local economy. The reason is that the coliseum is designed incorrectly using CAD software (Xiong, 2011).

In 1985, the Canada's Therac-5 radiation therapy machine malfunctioned due to software bug (Leveson & Turner, 1993). The cost was around $500 millions. It was delivered six massive overdoes of radiation to six cancer patients and caused three

©This item is protected by original copyright

deaths and sever injuries. In 1996, a software bug caused the Ariane 5 Flight 501 military satellite launcher exploded 40 seconds after initiation of the flight. A simultaneous register overflow occurred on one processor and its associated backup processor; which was caused shut down both of the processors. In the meantime, the satellite mission terminated itself. It was the costliest accident in the history (Lions, 1996).

In 1999, the MARS climate orbital failure was another mishap. The root cause was to use metric units in the coding of a ground software file. The investigation report says that the verification and validation process did not adequately address in the ground level of the software (Stephenson, 1999). The Pentium floating point division bug was found as a hardware bug for Pentium processors. This bug caused an error in the accuracy of a small number of division computations. To identify this problem, Intel Corporation costs to millions of dollars (Edelman, 1997).

©This item is protected by original copyright

The causes of the above accidents were addressed to inadequate software testing. Therefore, the software testing is a critical element for software quality assurance (Harrold, 2000). Each of the above catastrophic outcomes offers lesson to the project teams (particular for the software test engineers) regarding the possible way of failures and their affects, which could be set high price for the software testing. In this sense, the National Institute of Standards and Technology (NIST) was estimated annual cost of insufficient software testing tools and methods is 22.2 to 59.5 billion USD (Carroll, 2003a; Carroll, 2003b). Most of the cases, the testing costs are even higher than 30%-50% of software development (Tassey, 2002). Hence the Software Testing Process (STP) is a crucial part of Software Development Life Cycle (SDLC). The fundamental

test process suggested by the International Software Testing Qualifications Board (ISTQB) (Morgan et al., 2015) consists of five parts (shown in Fig. 1.1) that cover all the features of testing. From the aforementioned discussion it can be seen that, in software test engineering, only fault detection and prevention are not effective for tackling bugs at input interaction level.

Combinatorial Input Interaction Testing (CIIT) is an effective fault detection technique. In CIIT, early fault detection can be improved by ordering test cases on the basis of input interaction. In CIIT, possible combination of input can be used to find the software configuration error(s). The efficiency of CIIT is to detect failures effectively which is triggered by the input interactions among input parameters. CIIT offers such an effective testing technique which has been studied well and widely used in the last 20 years (Othman, 2012). For the increasing number of input based system, CIIT is considered as a practical and efficient testing strategy (Cohen et al., 1997; Cohen et al., 2003a; Cohen et al., 2003b; Lei et al., 2007a).

©This item is protected by original copyright

The t-way (is a CIIT) strategy generate test suite with the aim to cover every possible combination produced by the interacting parameters. Given any P input parameters, every combination of values of the t strength must be covered in at least one test case. For example, a 3-wise coverage requires every triple need to be covered in at least one test case. The strength t value can be set 2 (known as pairwise interaction), 3, 4, 5, 6, 7 and so on. The t-way strategy can be shown by Covering Array (CA). The mathematical notation of a t-way test configuration can be shown as CA(N, t, v^p) where, N, t, v and p indicate the number of test cases, input interaction strength, values and number of parameters respectively (Cohen, 2004). The values can be distributed for

each parameter either in uniform or non-uniform basis. The mathematical notation of a t-way test configuration for non-uniform values can be shown as MCA(N, t, $v_j^{p_i}$) where, i=1,2,3,4…n and j=0,1,2,3…m (Cohen, 2004). The t-way strategy can be deployed in sequence-less system and sequence system. In sequence-less system all input parameters interactions are observed on parallel input. On the other hand, in sequence system all input parameters interaction are observed in sequence. The mathematical notation of a t-way test configuration for sequence input interaction system can be shown as SCA(N, t, p) where, N, t and p indicate the generated number of test cases, input interaction strength and number of parameters respectively (Kuhn et al., 2012). The SCA indicates sequence covering array. The detail description is shown in chapter 2.

©This item is protected by original copyright

1.2 Problem Statement

In CIIT, Combinatorial Explosion (CE) is a great issue for software test design. In CE the increasing numbers of input sizes are caused by the exponential growth of test input interaction and create a large input space. The problem examine is needed to do so fast that even the fastest computers require an intolerable amount of time. It limits the ability of computers to solve large input space problems. Furthermore, caused by time and cost constraints, the exhaustive input interaction testing is not fully practiced by the test engineers (Williams & Probert, 2001). It is found that in the most realistic problems of interest, the number of combinations is typically very large. For example, a system is needed to make *n* decisions and for each decision there are 10 possible options. That means there are all together 10^n combinations of solutions. It indicates the number of

combinations grows exponentially as *n* increases. In CIIT the CE problem can be minimized by the t-way input interacted test case generation strategy; which is able to select test cases with a tractable size and can manage bugs effectively.

Since 1995, there are many (refer to Chapter 2) sequence-less t-way test strategies are developed. Mandl (1985), deployed pairwise testing to test Ada Compiler. Khun and Okun (2006), demonstrated the needs of t-way test strategy under CIIT. Klaib (2009), developed GTWay to support automated test data generation. Shiba et. al. (2004) used artificial life techniques to generate test cases. Colbourn et al. (2004) used greedy heuristic algorithm to generate combinatorial test suite termed as Density (Colbourn et al., 2004). Lei et al. (2007a) developed an efficient t-way test strategy for multi-way combinatorial testing termed as In-Parameter-Order General (IPOG) (Lei et al., 2007a). TConfig (Williams, 2002; Williams, 2000; Williams & Probert, 2001) is a t-way test case generation strategy based on recursive block (used to generate pairwise test suite). Jenkins (2010) designed a t-way strategy to generate test cases and termed as Jenny (Jenkins, 2005). Cohen et al. (2008) used simulated annealing (SA) heuristic algorithm in t-way strategy to generate test suite (Cohen et al., 2008). Yamada et al. (2015) developed a combinatorial test strategy using incremental SAT solving in order to optimize the test case (Yamada et al., 2015). Czerwonka (2006) proposed a test case generation strategy termed as Pairwise Independent Combinatorial Testing (PICT) (Czerwonka, 2006; Czerwonka, 2010). Arshem (2010) proposed a strategy to generate t-way test suite and called Test Vector Generator (TVG) (Arshem, 2004). Rabbi and Mamun (2015), designed an effective test data generation strategy (pair wise) based on 'Kids Card' game (Rabbi & Mamun, 2015). Sabharwal and Aggarwal (2015) proposed a strategy to generate 2-way (pairwise) test sets using genetic algorithm (Sabharwal &

©This item is protected by original copyright

Aggarwal, 2015). HHH (Zamli et al., 2016; Zamli et al., 2017) is based on a tabu search hyper-heuristic strategy for t-way test suite generation, LAHC (Zamli et al., 2015) used Late Acceptance Hill Climbing Algorithm for t-way strategy, HSS (Alsewari, & Zamli, 2011) is a harmony search based t-way testing strategy, PSTG (Ahmed & Zamli, 2010; Ahmed et al., 2012a; Ahmed et al., 2012b; Mahmoud & Ahmed, 2015;) is a t-way test data generation strategy based on particle swarm optimization, CSS (Ahmed et al., 2015) is a Cuckoo Search based t-way strategy, ITTSG (Othman, 2012) is an integrated t-way test data generation strategy for interaction testing, BTS (Alsariera et al., 2015) is a bat-inspired strategy for t-way interaction testing. Alsewari & Zamli (2014) compare some t-way strategies adopted by optimization algorithms to explore the strength and limitations of each strategy, and highlighted the possible research for future work in this area. (Alsewari & Zamli, 2014).

©This item is protected by original copyright

During interaction testing most failures are arise because of the interaction between few parameters, not every input parameters caused for every failures. Suggested by NIST (Kuhn, et. al., 2010), most of the failures can be found within the input interaction not more than 6. That's mean the highly effective fault detection is found within the input variable interaction is 6. Another investigation says that at low cost the pair wise combination testing sometimes generate good outcomes in a low cost budget. However it may miss more than 10% of system bugs (Kuhn et al., 2010). Hence pair wise combinatorial testing is not advisable for mission critical system testing purpose. Though the above researches on t-way test case generation focuses on lower to higher strength interactions, Kuhn et al. (2008) and Kuhn and Okun (2006) reported that the higher strength based input interaction is rare, most faults can be found in the strength range 4 to 6. The researchers reported that for rare cases faults may occur in the

higher strengths. Reported by Younis and Zamli (2009), for the software system configuration, there may need for higher than 6-way input interaction. Therefore, the need of enhancement (in terms of drawback and applicability) on t-way strategy still crucial (Othman & Zamli, 2011).

The existing t-way strategy for sequence-less input interaction is useful to handle combinatory explosion. However, they cannot generate test cases for the system that consider the sequence inputs interaction. That's mean the t-way strategies for sequence-less input interaction are not suitable to detect failures in the case of sequence input interaction based system.

The t-way input interaction test strategy considers effectively generating the feasible test cases for sequence input interaction fault detection (Kuhn, 2012). Numbers of researchers have worked on t-way sequence input test strategy in order to generate test cases. Kuhn et al. (Kuhn et al., 2012) used greedy method (called t-Sec) in order to generate test suite, based on t-way strategy. Zabil et al. (2012) adopted Bee Algorithm (BA) in t-way strategy for sequence input interaction to generate test cases. Investigation shows that for higher interaction, BA algorithm is not profitable to generate t-way test suite while considering the sequence input interaction. Banbara et al. (2012) adopted greedy algorithm in Answer Set Programming (ASP) and Constraint Programming (CP) to implement t-way strategy; which focus on only sequence input interaction to generate test cases. Erdem et al. (2011) used a knowledge-representation formalism from the area of artificial intelligence which was based on logic programming (using ASP) to generate test cases. Brain et al. (2012), also used ASP to implementing t-way strategy for sequence input interaction. Chee et al. (2013)

©This item is protected by original copyright

developed a conditional expectation algorithm which generates sequence covering array with the number of permutations. Since today, for strength 3, 4 and 5, the ASP based t-way strategies for sequence input interaction generate better (most of the cases) test cases than the other t-way strategies for sequence input interaction. The disadvantage of ASP based implementation is, all system configurations cannot run in one program platform, which means that for each system configuration test needs individual program design to generate test cases.

The above researches on t-way input interacted strategies focus on specific strategy either sequence-less or sequence. The integration of both strategies is an ease of exercise on testing in one hand in order to achieve faster and efficient test cases. However, there is a need to have a t-way strategy which is flexible and able to integrate sequence-less and sequence input interaction test possibilities. As the generation of t-way test suite (for both sequence-less and sequence input interaction), it is an NP hard problem, no single strategy challenges to generate minimum number of test cases for every test configuration (Othman, 2012; Afzal et al., 2009). Motivated by the above problems, in this thesis, a new t-way strategy has been proposed.

©This item is protected by original copyright

1.3 Aim and Objectives

The aim of this research is to design and evaluate a new strategy based on t-way strategy, called t-way Input Interaction Test (TWIIT) which integrates both sequence-less and sequence input interaction. To fulfil the aim, the following objectives are taken under consideration:

i. To analyze recent research on t-way sequence-less and sequence input interaction based test strategies in order to identify the potential improvements, in designing t-way test strategy.

ii. To design and implement algorithm for TWIIT supporting sequence-less and sequence input interaction.

iii. To evaluate the performance of TWIIT strategy with other competing strategies on the basis of the generated test suite size.

©This item is protected by original copyright

1.4 Research Scope

This research focus is to design and develop an integrated t-way strategy supporting sequence-less and sequence input interaction to generate test suite. Which fall in the design part of STP in Fig. 1.1. The design processes are concerned with the fine detail of what to test (test conditions), and how to combine test conditions into test cases. Therefore, a small number of test cases can cover as many of the test conditions as possible. These processes are the bridge between planning and test execution processes. However, in the planning process overall test plan and the achievement are prepared; the control process decision is taken during the test activities not match up with the plans. The test implementation and execution activity involves run the tests. The exit criteria is defined during test planning and before test execution started. At the end of test execution, the test manager checks to see if these have been met. Test closure activities concentrate on making sure that everything is organized. In this process, all the defects are written clearly which are deferred with another processes (Morgan et al., 2015).

©This item is protected by original copyright

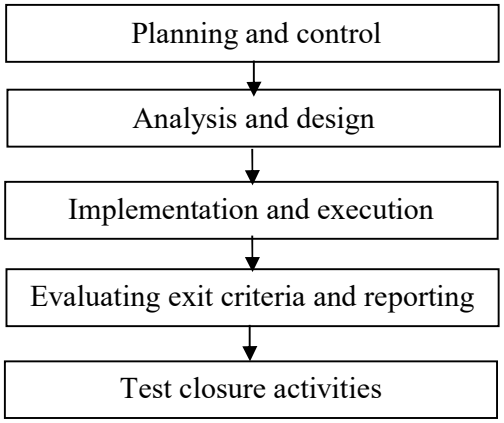

Figure 1.1: Fundamental of test processes (Morgan et al., 2015)