PAIRWISE TESTING APPLIED WITH ONLINE REGISTRATION

WASAN UTHAILEANG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
(TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2014

Thesis
entitled

PAIRWISE TESTING APPLIED WITH ONLINE REGISTRATION

…….....
Mr. Wasan Uthaileang
Candidate

.....
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Major advisor

.....
Asst. Prof Adisorn Leelasantitham,
Ph.D. (Electrical and Computer Engineering)
Co-advisor

.....
Lect. Waranyu Wongseree,
Ph.D. (Electrical Engineering)
Co-advisor

…….....
Assoc. Prof. Sombat Thanawan, Ph.D,
Acting Dean
Faculty of Graduate Studies,
Mahidol University

…….
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Program Director
Master of Science Program in Technology of Information System Management
Faculty of Engineering
Mahidol University

Thesis
entitled
PAIRWISE TESTING APPLIED WITH ONLINE REGISTRATION

was submitted to the Faculty of Graduate Studies, Mahidol University
for the Degree of Master of Science
(Technology of Information System Management)
on
March 24, 2014

…….....
Mr. Wasan Uthaileang
Candidate

.....
Lect. Sotarat Thammaboosadee,
Ph.D. (Information Technology)
Chair

…….....
Lect. Waranyu Wongseree,
Ph.D. (Electrical Engineering)
Member

…….
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Member

…….....
Lect. Surapong Pongyupinpanich,
Ph.D. (Electrical and Informatic Engineering)
Member

…….
Asst. Prof Adisorn Leelasantitham,
Ph.D. (Electrical and Computer Engineering)
Member

…….....
Assoc. Prof. Sombat Thanawan, Ph.D,
Acting Dean
Faculty of Graduate Studies,
Mahidol University

…….
Lect. Worawit Israngkul,
M.S. (Technical Management))
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENTS

This thesis is not successful If not both advisors Dr.Supaporn Kiattisin and Dr.Adisorn Leelasantitham recommended guidance and kindness to me everything. Thanks for Assoc. Rangsit Sirirangsri You are a teacher the patron the cause of this thesis. Especially, various information is useful. This allows a much more complete a thesis. I would like to express my deepest gratitude to him.

Besides my advisor and my co-advisor, I would like to thank to my thesis committee, Lect. Dr. Waranyu Wongseree For theirs insightful comments and questions to fulfill my thesis.

My thanks also go to all of teacher and staff in Information Technology Faculty of Science Maejo University for their help in everything.

My thanks also go to all of staff and my friends in Faculty of Technology Information System Management Mahidol University for their help in everything.

Finally, I would like to give my thanks to my mother Omsin Uthaileang and my wife Rattawadee setthajit for their supporting me throughout my life.

Wasan Uthaileang

PAIRWISE TESTING APPLIED WITH ONLINE REGISTERATION

WASAN UTHAILEANG 5536270 EGTI/M

M.Sc. (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)

THESIS ADVISORY COMMITTEE: SUPAPORN *KIATTISIN,* Ph.D., ADISORN LEELASANTITHAM*,* Ph.D., WARANYU WONGSEREE, Ph.D.

ABSTRACT

Pair-wise Testing is a Software testing technique used for the design of test cases to test every possibilities input parameter that pair interaction together (2 way-Interaction). This resulted in a decrease in the number of test cases, time and increase percent coverage test case up to 80-100%. Every combination of valid values of these two parameters was covered by at least one test case. This research, presents decreasing the number of input parameters with In-Parameter-Order (IPO) Framework applications to online registration (Thai language) generate a test suite of all possibilities input parameters. The result is a test suite of software testing much more effective than exhaustive testing (All-pair testing) to 60-80 % and the size of test suites decreased over 1-20%.

Pair wise testing requires a major reason for software testers who lacked the skills to use the tool to decrease the number of input parameters before the testing. However, Pair wise testing is an important process in software development testing of Thailand.

KEY WORDS: PAIRWISE TESTING/ COMBINATION TESTING/ INPUT PARAMETER/ ORTHOGONAL ARRAY

67 pages

การประยุกต์การทดสอบ PAIRWISE กับเว็บลงทะเบียนออนไลน์
PAIRWISE TESTING APPLIED WITH ONLINE REGISTERATION

วสันต์ อุทัยเลี้ยง 5536270 EGTI/M

วท.ม. (เทคโนโลยีการจัดการระบบสารสนเทศ)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์ : สุภาภรณ์ เกียรติสิน, Ph.D., อดิศร ลีลาสันติธรรม, Ph.D., วรัญญู วงษ์เสรี, Ph.D.

บทคัดย่อ

Pair wise testing เป็นเทคนิคการทดสอบซอฟแวร์ในการออกแบบ Test Case ทุกความเป็นไปได้ของอินพุตพารามิเตอร์ เป็นการจับคู่การทำงานของอินพุตพารามิเตอร์ (2 Way-Interaction) ผลลัพธ์ที่ได้จะลดจำนวน Test Case, เวลาและเพิ่มความครอบคลุม Test Case ถึงร้อยละ 80-100% โดยทุกๆการรวมกันของอินพุตพารามิเตอร์ที่ถูกต้องของการจับคู่ จะต้องครอบคลุมการทดสอบอย่างน้อยหนึ่งครั้ง ในงานวิจัยนี้นำเสนอ การลดจำนวนของอินพุตพารามิเตอร์ด้วย Pair wise testing โดยใช้ IPO Framework ประยุกต์กับเว็บไซต์ของการลงทะเบียนออนไลน์ (ภาษาไทย) เพื่อสร้างชุดการทดสอบในทุกความเป็นไปได้ของอินพุตพารามิเตอร์ ผลที่ได้คือ ชุดการทดสอบซอฟต์แวร์ที่มีประสิทธิภาพมากขึ้นกว่าการทดสอบแบบ Exhaustive testing (All-pair testing) ถึง 60-80% และขนาดของชุดการทดสอบลดลงกว่า 1-20%

เหตุผลสำคัญที่ต้องใช้ Pair wise testing เพราะว่านักทดสอบซอฟแวร์ขาดทักษะการใช้เครื่องมือในการลดจำนวนของอินพุตพารามิเตอร์ก่อนการทดสอบ อย่างไรก็ตามความสำคัญของ Pair wise testing เป็นกระบวนการที่สำคัญในกระบวนการทดสอบซอฟต์แวร์ในประเทศไทย

67 หน้า

CONTENTS

CONTENTS(cont.)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Currently software testing in Thailand are not focused on the design of test cases to cover the test which affect the quality of the software greatly because the error was found late. Thus the cost to fix and including the time required to increase. Software testing is a process of try the software guidelines by using the technical knowledge to be able to identifier or search for the error software that could be hidden in provides appear and to identify solutions to the problem and the assumption of errors occur.

Software Testing is to examine the empirical truth need to follow a requirements of the beneficiary. The quality of the product or service under test Information on the quality of the product or service on the test. With what is defined in the context of provides implementation of the plan. Software Testing define the objectives show the independence of software to license. Business is to realize Preparing and understand the risks of software. Testing techniques to gather but no boundaries the process of implementation of the program application with the intent of finding software faults processes can be defined from checking that the system can work according to the requirements of the user, or verifying that the system performed correctly or not.

The primary function of the test is finding defects in all the possibilities within the system to be free from defects for a minimum prior to delivery to the user. modern testing will focus on three commandments goal is to reduce the amount of time spent in testing finding defects with the highest number of test case is minimal and to cover the most testing with minimal test case. Such a goal will find what they want for the test in each case is to reduce the number of tests to a minimum. Such as browser type, operating system that is running in a different way as well as integration with a different databases. function tests of this nature could cause problems such as

the need for the test case as many to cover every possibility in all tests which may affecting of system resources may not be sufficient in to test all the possible total.

The primary function of the test is to find fault in every possible inside the system. For the system to be free from defects before delivery to the user. By testing strategy will focus on three major goals include reducing the amount of time spent in testing. To find the number of defects with the highest number of Test Case minimal and to cover the most testing by Test Case smallest of such goals to see what the requirements for testing in every case is to reduce the test minimum. Especially in the case of a test of the accuracy of the system is a collaboration between several input parameters such as browser type that are running on different operating systems. As well as working with different database systems. Which tests to run in this way may cause problems such as the need to use a lot of Test Case to cover every possibility in all tests. As a result, the use of system resources may not be sufficient for running all the possibilities in all tests. There also may be risk occurs in cases that do not test at all the possibilities. Although the technique used to test the Equivalence Partitioning and Boundary-Value Analysis can reduce the number of Test Case to a certain Education, but for testing in all possibilities are still a lot of them. Therefore, to solve such problems, testers need to choose the most suitable method of testing to find bugs as possible within minimal time itself. And one of the test methods were developed and have been very popular. Pairwise testing method called the All Pair, which is intended to reduce the number of Test Case to test itself. In addition, the potential risk of not test all possibilities of the system as well. Although technical testing Equivalence Partitioning and Boundary-Value Analysis can reduce the number of tests at a certain level. But also alternatively testing can find defects and test all the possibilities are many. Was developed and has been popularity in the last 5 years, called Pair-wise testing to reduce the test case used in the test.

- **Software Testing Life Cycle**

In every organization, testing is an important phase in the development of a software product. However, the way it is carried out differs from one organization to another. It is advisable to carry out the testing process from initial stage with regard to the SDLC to avoid any complications.

The Need for a Constant Testing Process Software Testing Phase show in figure 1.1.

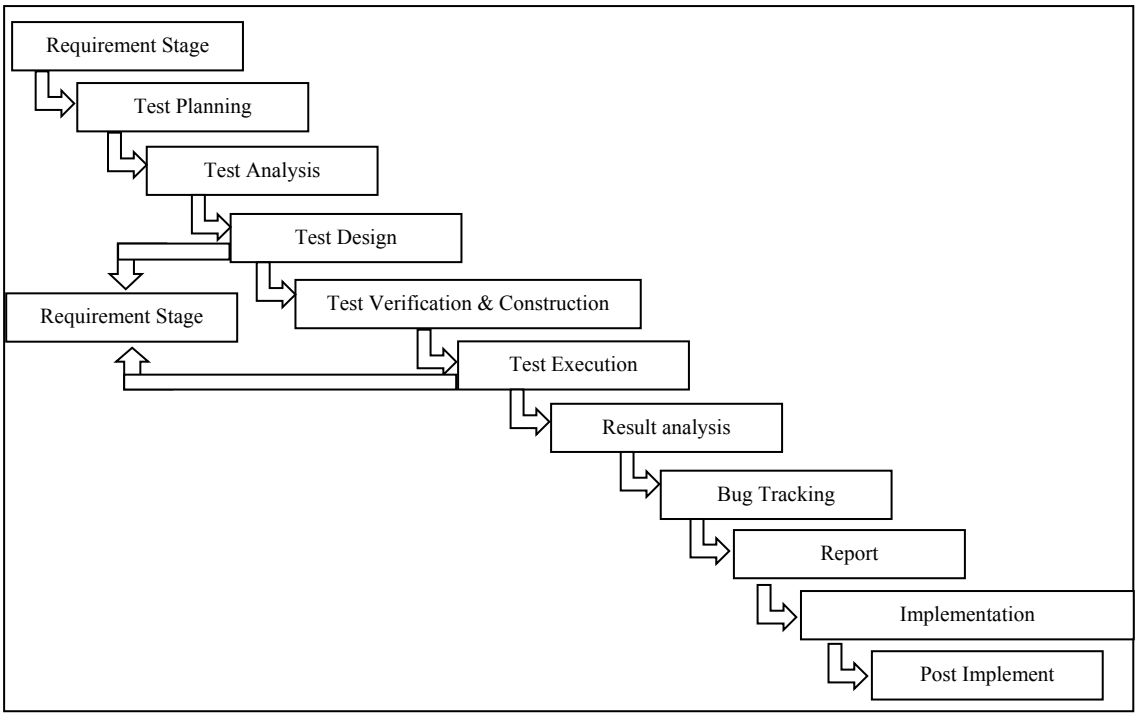

Figure 1.1 Software Testing Life Cycle flow chart.

- **Requirement Stage**

This is the initial stage of the software testing process. In this phase, developers take part in analyzing the requirements for designing a product. The role of software testers is also necessary in this phase, as they can think from the 'users' point of view, which developers may not. Thus a team of developers, testers, and users can be formed to analyze the requirements of the product. Formal meetings of the team can be held in order to document the requirements, which can further be used as software requirements specification or SRS.

Pair-wise Testing or 2-way interaction was invented the algorithm to reduce the size of a number of test cases. There are 2 algorithm; IPO In-parameter-order [1] and the AETG which recognized and be the most reference. [2] Principles of test suite are to find errors or defects that hard to find and input parameters without Combinatorial Software Testing. [3] Software testing in the

modern world has 3 objects: reducing amount of time used in experiment. Find the maximum number of defects by using a minimum number of test cases [4]. Especially to test the accuracy of the system. With collaboration in various input parameters such as browser, database systems. That has to work on different operating systems. These may be cause of problems as the number of test case to cover every possibility (not included, impossible case) all resources of computer system cannot compile all the possibility in all tests. Moreover, the risk of software testing cannot cover every possibility. Although some of test cases decreased with Equivalence Partitioning and Boundary-Value Analysis can reduce the number of tests at a certain level [5] However, there are a lot of test case for testing every possibility are exists.

The following examples are conclude the method; in this example we have three parameters which can have following values as *A, B, C; 1, 2; X, Y* which means that the number of combinations 3 x 2 x 2 = 12. If we want to test just 2-pairs or pairwise testing which covers every possible combination of two parameters, there should be 6 combinations. The highlighted test cases *TC2, TC3, TC5, TC8, TC10, and TC11* are not necessary. Follow figure 1.2

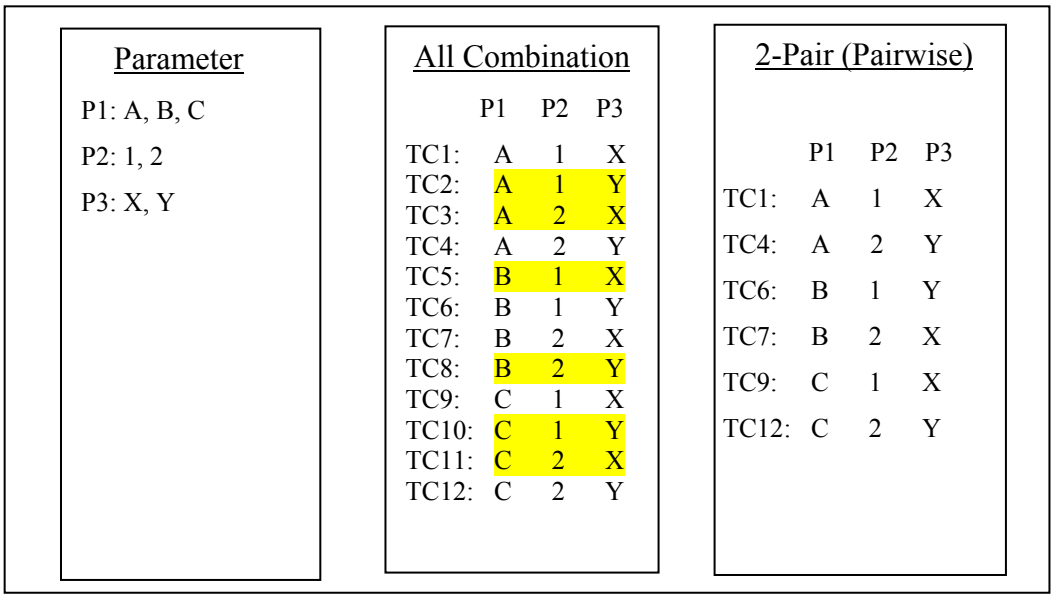

Figure 1.2 Show parameter before combination to All pair and Pair wise

While this example resulted in a relatively small savings in test cases, as the number of factors and levels increase, the benefits of HTT (High Throughput Testing) will be quite substantial. The table expresses the savings for various scenarios between All Combinations and Pairwise Combinations show in table1.1

Table 1.1 Combination to All pair and Pair wise with number of factor and level

Factor & Levels	All Combination	Pairwise Combinations
5 Factor at 3 Levels each	243 Test case	11 Test case
6 Factor at 4 Levels each	4096 Test case	23 Test case
7 Factor at 6 Levels each	279,936 Test case	56 Test case
10 Factor at 7 Levels each	282,475,289 Test case	89 Test case

1.2 Objectives of study

1.2.1 Applied to develop and optimization of software testing process with Pair wise testing.

1.2.2 Compare efficiency of the All pair testing and Pair wise testing in matter of a reduced number of parameters, Time spent on testing and percent coverage testing test case after use to reduce a number of pair wise testing.

1.3 Scope of work

1.3.1 Studying Pair wise testing process working with algorithm to reduce the number of tests for website registration online. (Thai language)

1.3.2 Use data of input parameter from Test plan: Training Registration System in a FillRegisterProfile page.

1.3.3 Use input parameter Thai language**.**

1.4 Expected result

Make testing software more effective. In terms of reducing the time spent in testing software, find the maximum defects, most test coverage to test the website to register online (Thai language) and better than an existing tests. By the application of the website to register online (Thai language) is a useful for social tester program in Thailand.

CHAPTER 2
LITERATURE REVIEW

2.1 Pair wise method in many researches

Testing complex systems have a number of challenges: test cases take too long to create, too long to automate, too long run, too long verify, for new builds there is no easy way to know which test case need to be re-run, there is no objective measure of test effective ness, and if testers/developers are fortunate enough to be using test automation then there are too many test cases to maintain.[2] From such research, Software engineering researchers have invented a way to reduce test cases with different methods, until researchers have discovered how wide variety of input parameter to pair following example.

- In parameter order (IPO): The extension of an existing pairwise test set for an additional parameter contains the following 2 steps. [1]

- Horizontal growth, which extends each existing test by adding one value of the new parameter.
- Vertical growth, which adds new tests, if necessary, to the test set produce by horizontal growth.

Algorithm *IPO_H*($\mathcal{T}$, p_i)
// $\mathcal{T}$ is a test set. But $\mathcal{T}$ is also treated as a list with elements in arbitrary order.
{ assume that the domain of p_i contains values $v_1, v_2, \ldots$, and v_q;
  π = { pairs between values of p_i and values of $p_1, p_2, \ldots$, and p_{i-1} };
  if ($|\mathcal{T}| \leq q$)
  { for $1 \leq j \leq |\mathcal{T}|$, extend the jth test in $\mathcal{T}$ by adding value v_j and
    remove from π pairs covered by the extended test;
  }
  else
  { for $1 \leq j \leq q$, extend the jth test in $\mathcal{T}$ by adding value v_j and
    remove from π pairs covered by the extended test;
    for $q < j \leq |\mathcal{T}|$, extend the jth test in $\mathcal{T}$ by adding one value of p_i
    such that the resulting test covers the most number of pairs in π, and
    remove from π pairs covered by the extended test;
  }
}

Figure 2.1 Framework of the IPO Strategy (Horizontal Growth)

- **AETG:** Commercial n-way test generation's tool. This methodology can create a minimal set of test vectors which will thoroughly exercise a system under test. The term "test vectors" is used as opposed to test case because in most practical situations. AETG does not create actual test case, but a table of the input values needed to execute the desired test case [2] follow table 2.1

Table 2.1 AETG test suit for example test problem.

	Parameter		
test case	P_1	P_2	P_3
1	1	1	1
2	2	2	2
3	3	1	2
4	3	2	1
5	2	1	1
6	1	2	2

- **Hexa-wise:** Combinatorial testing can help detect problems like this early in the testing life cycle. The key insight underlying combinatorial testing is to test the interaction between the different parameters that effect the system, and that not every possible parameter is a contributor to every fault. Most faults are caused by interactions between a relatively small numbers of parameters [11] .

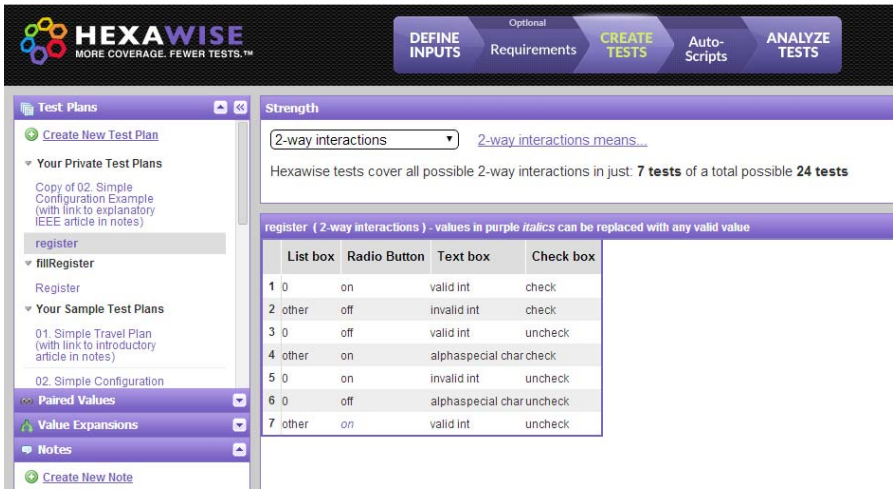

Figure 2.2 Hexa-wise tools test plans and combination application.

- **PUTs**: a new method to remove the effective pairwise test cases from combinatorial test suit. The method is based on the using the dependent relationship of input parameter to achieve the aim of reducing pair wise combinatorial test suit. [13] follows table 2.2 and Figure 2.3

Table 2.2 These result of algorithm of removing ineffective combinations

No.	Subject	Num. of 2-way combination	Num. of ineffective 2-way combination	Time(s)
1	4^3	54	5	<0.1
2	4^4	96	7	<0.1
3	5^4	160	15	<0.1
4	5^5	250	28	<0.1
5	6^4	240	27	<0.1
6	$3^3*4^4*2^5$	148	7	<0.1

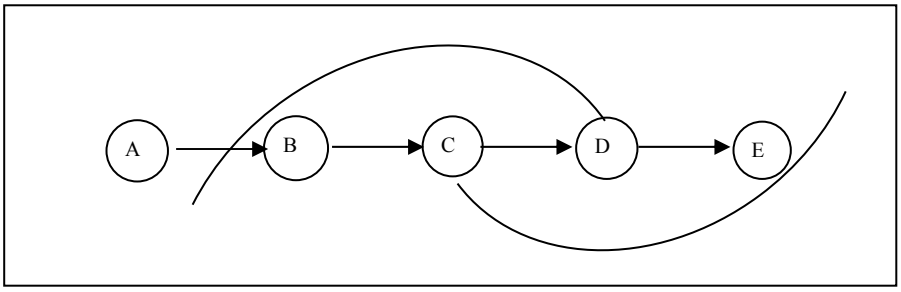

Figure 2.3 PUTs method is based on the using the dependent relationship of input parameter

2.2 All-pairs testing

Is a combinatorial method of software testing that, for *each pair* of input parameters to a system (typically, a software algorithm), tests all possible discrete combinations of those parameters. Using carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by "parallelizing" the tests of parameter pairs [15].

Table 2.3 Generate a multi set of All pair testing

Parameter Name Value	1Value	2Value	3Value	4Value
Enabled	True	False	*	*
Choice	**1**	**2**	**3**	*
Category	**a**	**b**	**c**	**d**

2.3 Pair wise Testing and Combination testing approach

Pairwise testing is a specification-base testing criterion, which requires that for each pair of input parameter of systems, ever combination of valid values of these two parameters be covered by at least one test cases.[1] [2][13]

Pairwise testing (or 2-way interaction) is a specification-based testing criterion, which requires that for each pair of input parameters of system, every combination of valid values of these two parameters be covered by at least one test case. Empirical results show that pair wise testing is practical and effective for various types of software systems [1].

Is a criteria requirement testing which requires each pair of input parameters in the system, besides, it also combine all of the possible input parameters at two parameters to cover at least one case2-way interaction [1].to test the Pair-wise efficiently, therefore, using a concept "*Orthogonal Array"* to reduce amount of test case process[6].This is a way of statistical methods that applied to test the pair- wise. Orthogonal Array is ideally suited to test every possibility [2].

Orthogonal Array is resulted from the interaction of the input parameters to the system by pairing the values of any two columns within the array [7]. All possible values of the pair column will be used to test interoperability between the input parameters defined. In case of test the interaction with the input parameters in the form of components.

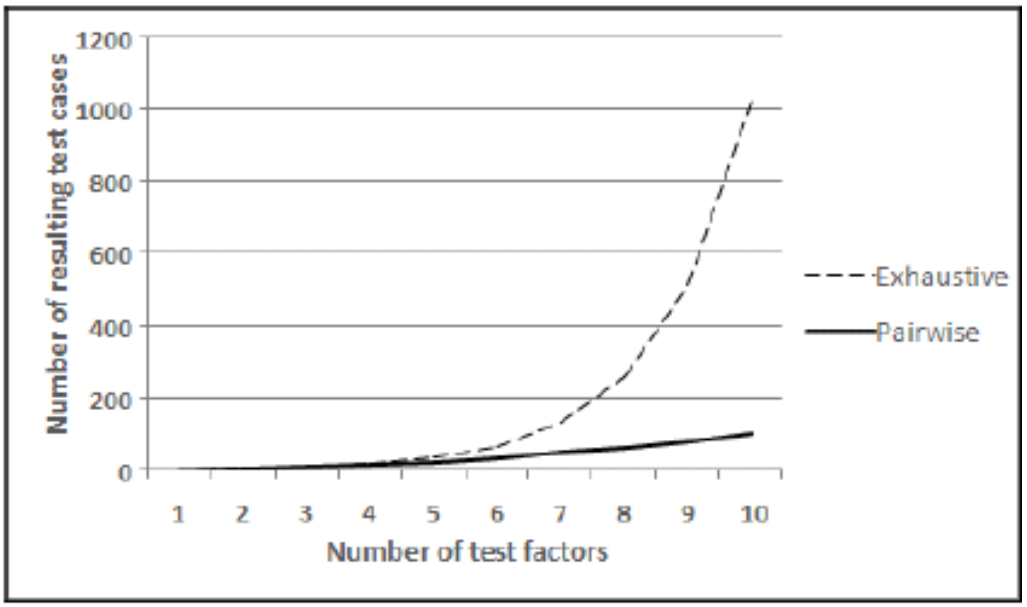

Figure 2.4 Increase in number of exhaustive and pairwise tests with number of test levels

Combination testing is a versatile methodology which is useful in a broad range of situations to detect faults in software.[12] It is based on the sight that while the behavior of a software system may be affected by a large number of factors, only a few factors are involve in a failure-inducing fault. Combinatorial testing began as pair-wise testing in which first orthogonal arrays and then covering arrays were used to make sure that all pairs of the test setting were tested. Subsequent investigations of actual software failures showed that pair wise (2-way) testing may not always be sufficient and combinatorial *t*-way testing for *t* greater than 2 may be needed. Until recently efficient tools for generating test suites for combinatorial *t*-way testing were not widely available.

2.4 *t*-way interaction testing

Combinatorial (or *t*-way) testing requires every combination of any *t* parameter values to be covered by at least one test [14]. Combinatorial methods can be highly effective because empirical data suggest that nearly all failures involve the interaction of a small number of parameters (1 to 6).

- Select 2-way coverage to generate the most powerful few dozen tests.
- Select 3-way coverage to generate the most powerful few hundred tests.
- Select 4-way, or 5-way coverage to generate the most powerful few thousand tests.
- Select 6-way to select the most powerful tens of thousands of tests.

2-way interaction plan the input parameters include the following parameters.

A = "A_1", B = "B_1", C = "C_1"

That in the generated set of **2-way** tests, at least one test that covers **every** possible interaction involving values from 2 different parameters. One example **2-way** test is:

So **every** valid **2-way** combination is covered, but it is possible that a valid **3-way** combination is **not** covered. For example, there might **not** be a single test that includes all 3 of the following:

A = "A_1" together with **B = "B_1"** together with **C = "C_1"**

Thus, all of the 2-way interaction that is required to cover all the input parameters. But it is possible that the 3-way Interaction correct input parameters may not cover all of the following:

A = "A_1" with B = "B_1" with C = "C_1"

Compared to 2-way interaction test plan with 3-wayinteraction is more thorough and better able to detect defects that are hard to find, but it will take longer to execute the test plan.

For example, given three parameters A, B (two values each), and C (three values), and pair-wise generation, three parameter-interaction structures are set up: AB, AC, and BC. Each of these has a number of slots that correspond to possible value combinations that participate in a particular parameter interaction—four slots for AB, and six slots each for AC and BC. (See Figure 2.5)

			AB	AC	BC
			00	00	00
A:	0, 1		01	01	01
B:	0, 1	Translates to	10	02	02
C:	0, 1, 2		11	10	10
				11	11
				12	12

Figure 2.5 Parameter-interaction structures

Table 2.4 Shows the Orthogonal Array of equation $L_4(2^3)$.

	Factor		
Row	**A**	**B**	**C**
1	A1	B1	C1
2	A1	B2	C2
3	A2	B1	C3
4	A2	B2	C1

$$L_4(2)^3$$

L_4= Number of rows

(2)= Maximum value=2, 3…, n

^{3}=Number of columns

Figure 2.6 Shows the equations used to Orthogonal Array.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Pairwise testing approach.

Given any pair of input parameters of a system, every combination of valid values of the two parameters be covered by at least one test [1], [2]. A special case of combinatorial testing that requires *n*-way combinations be tested. *n* can be 1, 2, …, or the total number of parameters in the system and based on simple specifications, and does not need to look into the implementation details.

Suppose that A and B are parameters. Then input parameters A (A_1, A_2) and input parameters B (B_1, B_2) Matching parameter A and B and result is {(A_1, B_1), (A_1, B_2), (A_2, B_1), (A_2, B_2)} When the parameter extended and input parameters C (C_1, C_2, C_3) respectively. So when the combination between the three parameters are input parameters (A_1, B_1, C_1), (A_1, B_2, C_2), (A_2, B_1, C_3) but not cover the matching of input parameters. {(A_2, C_1), (B_2, C_1), (A_2, C_2), (B_1, C_2), (A_1, C_3), (B_2, C_3)} Thus, (A_2, B_2) on the match with the input parameters (C_1, C_2, C_3) input parameters C_1 will get (A_2, B_2, C_1) covers only match (A_2, C_1) And (B_2, C_1), but if input parameters C_2 will get (A_2, B_2, C_2) covers matching only (A_2, C_2), but if input parameter C_3 will get (A_2, B_2, C_3) covering the match specific (B_2, C_3) the result shown that the three input parameters (C_1, C_2, C_3) and C_1 have the more Comprehensive tests than C_2, C_3 therefore select Test set of (A_2, B_2, C_1) and will earn 4 the capture pair that not cover (A_2, C_2), (A_1, C_3), (B_1,. C_2), (B_2, C_3) to take generate new test to cover all four pairs, the next step is to test the Pair-Wise Testing by algorithm of IPO (Input Parameter Order), is divided by two al algorithm is IPO. (In-Order-Parameter) follows table 3.1,3.2.

Table 3.1 Input parameter from Equivalence partitioning in orthogonal array

A	**B**	**C**
A1	B1	C1
A2	B2	C2
		C3

Table 3.2 Combination all possibilities (Exhaustive testing)

	A	**B**	**C**
1	A1	B1	C1
2	A1	B1	C2
3	A1	B1	C3
4	A2	B1	C1
5	A2	B1	C2
6	A2	B1	C3
7	A1	B2	C1
8	A1	B2	C2
9	A1	B2	C3
10	A2	B2	C1
11	A2	B2	C2
12	A2	B2	C3

For example, to test the interaction with input parameters in the form of components with Collaboration such as online registration page FillRegisterProfile input parameters needed to create Test Case 2x10x100 = 2000 Test Case. (Does not include terms Negative) If the input parameters will affect the test was not successful in a short time.

3.2 In parameter order (IPO) Algorithm.

IPO_H: definition of al algorithm assumes that T {p1, p2, ... pi-1} horizontal grow the T = pi is extended in each test in T by inserting into the pi. IPO_V: definition of al algorithm assumes that T {p1, p2,...pi-1} horizontal grow of T = pi for is a set of pairs not covered by T when | π |> 0 let generate new test sets in π. π and take the T

to the "-" Unspecified Value of the parameter as if pi is the last parameter, each value "-" set as pk1 ≤ k ≤ I replace every value of pk, otherwise "-" These are replaced by the parameters of Horizontal Growth.

Algorithm *IPO_H*($\mathcal{T}$, p_i)
// $\mathcal{T}$ is a test set. But $\mathcal{T}$ is also treated as a list with elements in arbitrary order.
{ assume that the domain of p_i contains values $v_1, v_2, \ldots,$ and v_q;
  π = { pairs between values of p_i and values of $p_1, p_2, \ldots,$ and p_{i-1} };
  if ($|\mathcal{T}| \leq q$)
  { for $1 \leq j \leq |\mathcal{T}|$, extend the jth test in $\mathcal{T}$ by adding value v_j and
    remove from π pairs covered by the extended test;
  }
  else
  { for $1 \leq j \leq q$, extend the jth test in $\mathcal{T}$ by adding value v_j and
    remove from π pairs covered by the extended test;
    for $q < j \leq |\mathcal{T}|$, extend the jth test in $\mathcal{T}$ by adding one value of p_i
    such that the resulting test covers the most number of pairs in π, and
    remove from π pairs covered by the extended test;
  }
}

Figure 3.1: Algorithm IPO_H

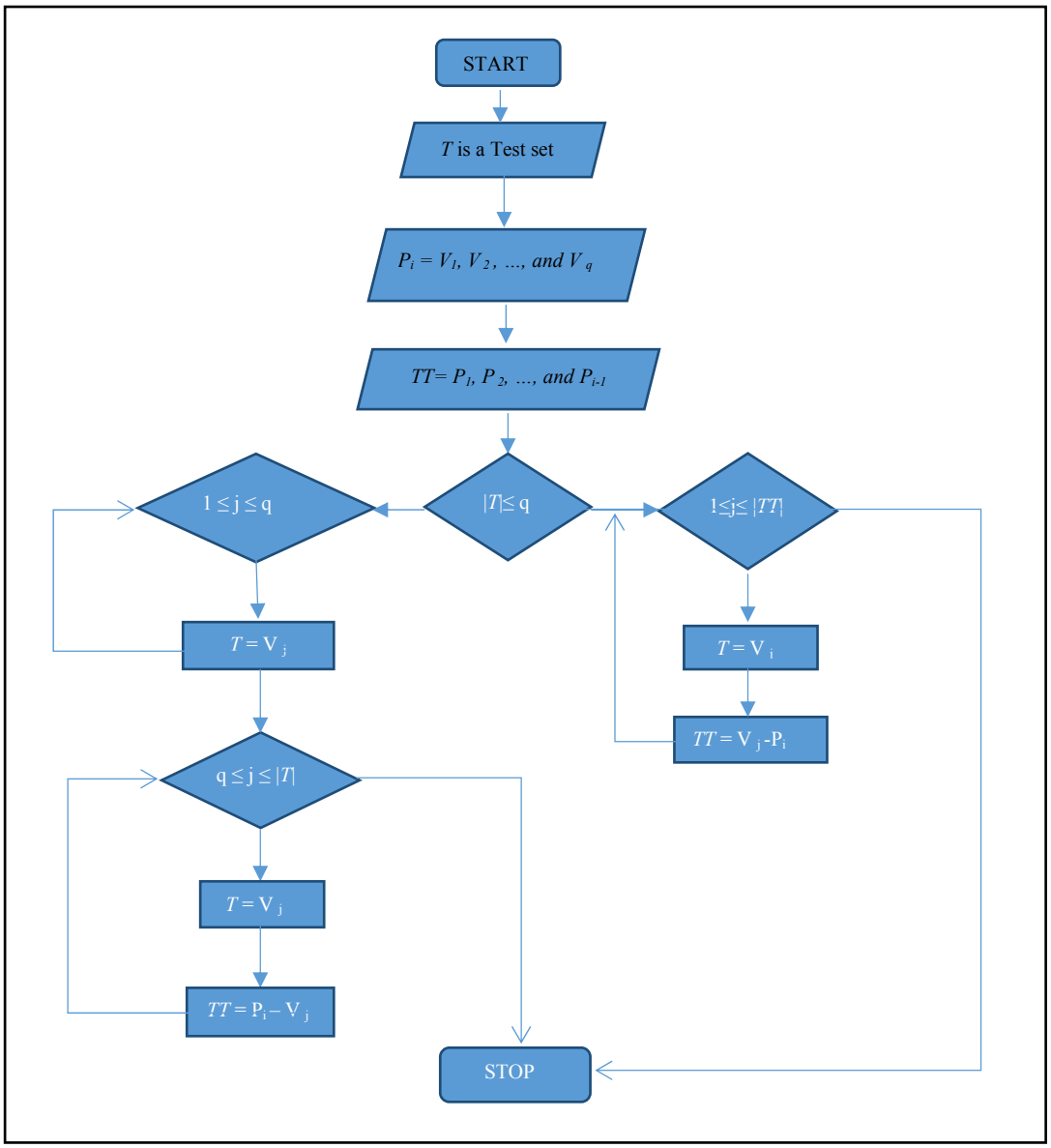

Figure 3.2 Flow Chart of Algorithm IPO_H

```
Algorithm IPO_V(T, π)
{ let T' be an empty set;
  for each pair in π
  { assume that the pair contains value w of p_k, 1 ≤ k < i, and value u of p_i;
    if (T' contains a test with "–" as the value of p_k and u as the value of p_i)
      modify this test by replacing the "–" with w;
    else
      add a new test to T' that has w as the value of p_k, u as the value of p_i,
      and "–" as the value of every other parameter;
  };
  T = T ∪ T';
};
```

Figure 3.3 Algorithm IPO_V

IPO_V: definition of al algorithm assumes that the Horizontal growth p_i has Produced a test set T for p1, p2, ... p_{i-1} Let π be the set of pairs not covered by T. Each pair in π contains a values of p_i and a value of p1, p2, ... p_{i-1}. Assume that | π | >0. The vertical growth of T according to π is to construct new tests for covering pairs in π and add these new tests to T. Thus, the resulting T is a pairwise test set for p1, p2, ... p_{i-1} .

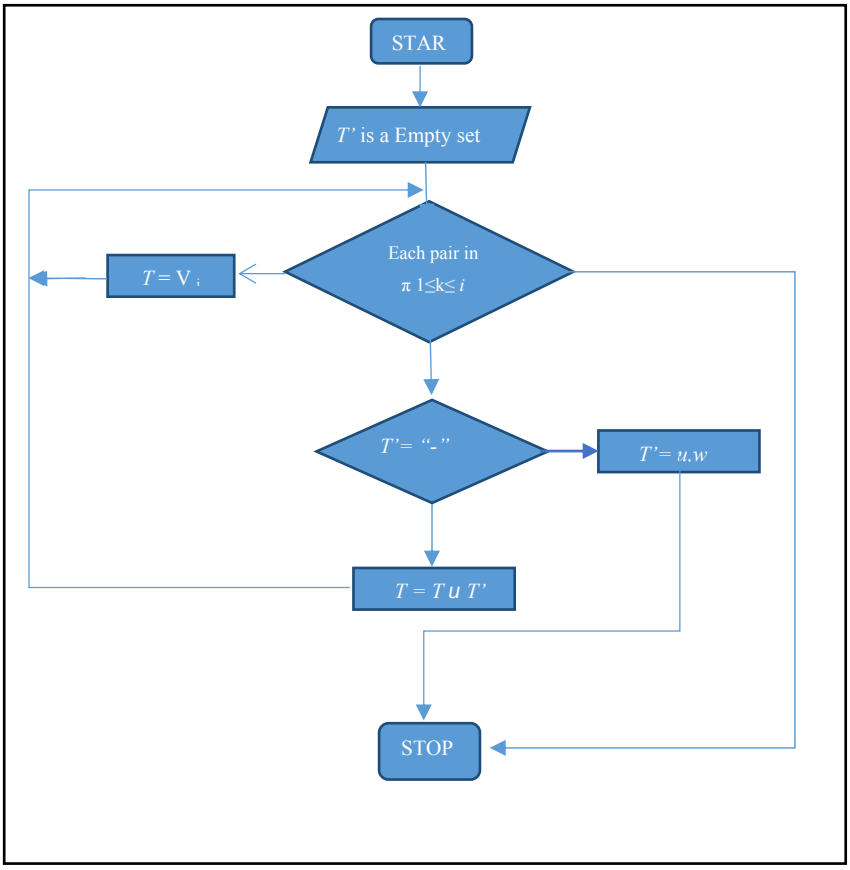

Figure 3.4 IPO_V Flow Chart

3.2 Pair wise applied with online registration.

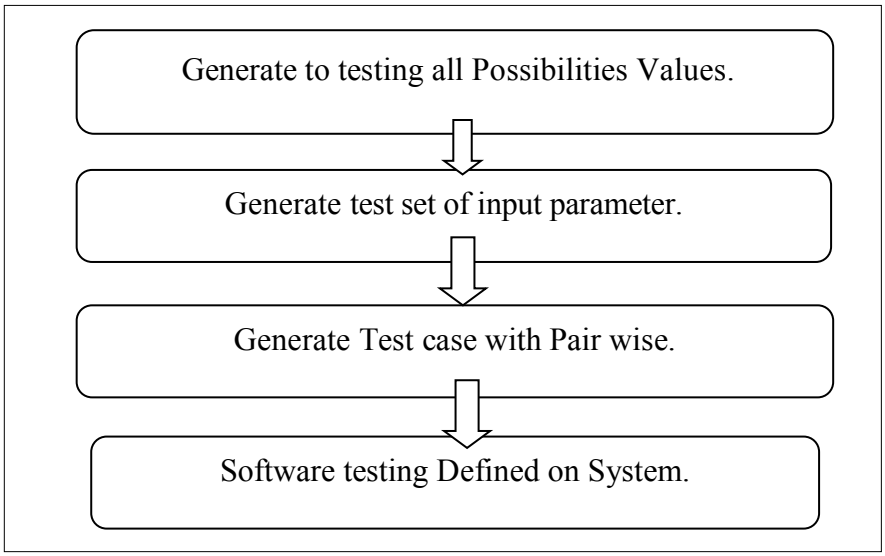

Figure 3.5 Approach to pair wise applied with online registration flow chart

3.2.1 Fill Registration Profile page Use case

Use cases are where users fill in the details for registration. System displays the registration form participants include Profile of participants are Title, Name and Surname, Education.

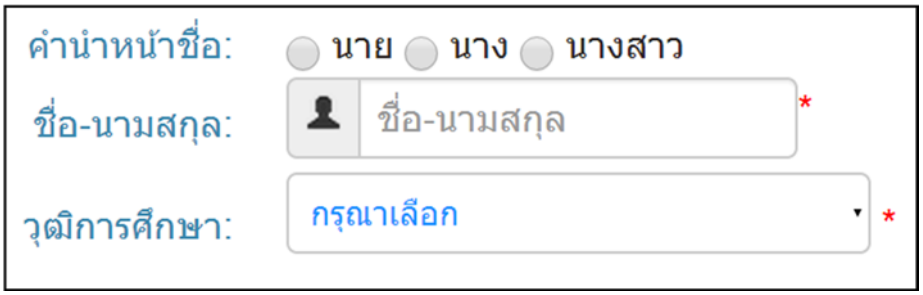

Figure 3.6 Partial of FillRegisterProfile page from Test plan: Training Registration

First, the example in Pairwise Testing of registration online. When taken the test case with FillRegisterProfile page by Equivalence Partition includes of the following parameter. An application with simple list box with 10 elements (Let's say 0,1,2,3,4,5,6,7,8,9) along with a List box, Radio button, Text Box and OK Button. The

Constraint for the Text box is it can accept values only between 1 and 100. Below are the values that each one of the GUI objects can take follows:

- *Text Box* Constraint assigned a value from 1 to 100
- *Radio Button* comprising Status on or off
- *List Box* contains a value from 0 to 9

Text Box = 100
Radio Button = 2
List Box = 2
Total Number of Test Cases using Cartesian Method: 2*2*100 = 2000
Total Number of Test Cases including Negative Cases will be > 2000

Figure 3.7 Shows Equivalence partitioning includes of the following parameter

System requirements in the Title

A system requirement to testing in the Title is as follows.

1. Not null.

Test conditions in the Title

1. In case the user does not select Title. System displays message and a notification to users is “ERROR: Please select Title” When users fill valid system allowed to click submit button and records into database system requirements is Comprehensive testing, Follows table 3.1,3.2,3.3.

Table 3.3 Test conditions in the a title for testing

Valid EC	Invalid EC	Invalid EC
VEc1-Not null	IVc1.0: Null	

Table 3.4 Test case for testing a title

Test Case ID	Class	Input Condition	Expected Result
1	VEc1-Not null	นาย	System displays to users interface
2	IVc1.0: Null	“”	System displays message and a notification to users is “ERROR: Please select Title”

System requirements in the Name

A system requirement to testing in the Names is as follows.

1. Must be characters Thai language only [ก-ฮ].
2. Without Numbers in Names.
3. Must be space between Name and Surname at least 1
4. Characters have at least 5 and not more 30 characters.
5. Not null

Test conditions in the Names

1. In case the user does filled valid Name. System displays message and a notification to users is “ERROR: Please filled valid Name and surname”

2. In case the user does filled Name System displays message and a notification to users is “ERROR: Please filled valid Name and surname” When user filled valid system allowed to click submit button and records into database system requirements is Comprehensive testing.

Table 3.5 Test conditions in a Names for testing

Valid EC	Invalid EC	Invalid EC
VEc2- Must be characters Thai language only [ก-ฮ]	IVc2.0: Other language	
VEc3- Without Numbers in Names	IVc3.0: Have a Numbers in Names	
VEc4- Must be space between Name and Surname at least 1	IVc4.0: Not space between Name and Surname at least 1	
VEc5- Characters have at least 5 and not more 30 characters	IVc5.0: Characters have less 5 characters	IVc5.1: Characters have more 30 characters
VEc6- Not null	IVc6.0:Null	

Table 3.6 Test case for testing a Names

TC ID	Class	Input Condition	Expected Result
3	VEc2- Must be characters Thai language only [ก-ฮ]	วสันต์ อุทัยเลี้ยง	System displays to users interface
4	IVc2.0: Other language	Wasan Uthaileang	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”
5	IVc3.0: Have a Numbers in Names	วสันต์ อุทัยเลี้ยง2577	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”

TC ID	Class	Input Condition	Expected Result
6	IVc4.0: Not space between Name and Surname at least 1	วสันต์อุทัยเลี้ยง	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”
7	IVc5.0: Characters have less 5 characters	วสัน	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”
8	IVc5. 1: Characters have more 30 characters	รัตนวดี ศรีสมรอมรเทพเจริญวิวัฒนาการณ์กรสกุลเกียรติยั่งยืนยง	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”
9	IVc6.0: Null	“”	System displays message and a notification to users is “ERROR: Please filled valid Name and surname”

System requirements in the Educations.

A system requirements to testing in the Educations are as follows.

1. Not null.

Test conditions in the Educations.

1. In case the user does not select Educations. System displays message and a notification to users is “ERROR: Please select Educations” When users fill valid system allowed to click submit button and records into database system requirements is Comprehensive testing.

Table 3.7 Test conditions in the Educations for testing

Valid EC	Invalid EC	Invalid EC
VEc1-Not null	IVc1.0: Null	

Table 3.8 Test case for testing the Educations

Test Case ID	Class	Input Condition	Expected Result
1	VEc1-Not null	ปริญญาตรี	System displays to users interface
2	IVc1.0: Null	“”	System displays message and a notification to users is “ERROR: Please select Educations”

3.2.2 Step of PairTest to generate input parameter.

Step 1: applied algorithm IPO strategy by Mapping with principle of Orthogonal Array (Textbox, Radio button, List box) Mapping Parameter A, B, C and set capabilities to support the value of each input parameter in Orthogonal Array equation. $L_4 (3^3)$ shown in Table 3.9.

Table 3.9 Input parameter set capabilities to support the value of each input parameter in Orthogonal Array

Radio Button(2)	List Box(2)	Text Box(3)
ON	0	Valid Int
OFF	Other	Invalid Int
		AlphaSpecial Char

Step 2: Define values the possibility of the input parameters in table {(On, 0, Valid Int), (On, other, Invalid Int), (Off, AlphaSpecial, Char), (Off, Other, Valid Int}) shown in Table 3.10.

Table 3.10: Define values the possibility of the input parameters in table

Run	Radio Button(2)	List Box(2)	Text Box(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5			
6			

Step 3: Table 3.11 shows that the input parameters in the table does not cover the test. To pairing the other 4 pairs follows {(Off, Invalid Int), (On, AlphaSpecial Char), (0, Invalid Int), (other, AlphaSpecial Char)} These can performed using the algorithm Input Parameter Oder (IPO) apply without having to input parameter is missing [1], but we can use to generate new test can take input parameters disappear into the Run 5. The Radio Button (On) and putting - in the ListBox by algorithm IPO in the TextBox values AlphaSpecial Char (On, AlphaSpecial Char) Run 6 the Radio Button (Off) and put - in the Listbox based algorithm in the Text Box input Invalid Int (Off, Invalid Int).

Table 3.11 Define – follows algorithm IPO_V of the input parameters in table

Run	Radio Button(2)	List Box(2)	Text Box(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5	On	-	AlphaSpecial Char
6	Off	-	Invalid Int

Step 4: Put the first input parameters (Off, Invalid Int),(On, AlphaSpecial Char) are found in the input parameters ListBox marked with required to out the remaining input parameter (0, Invalid Int), (other, AlphaSpecial Char) Run 5 rows by the values 0 ListBox and TextBox input values the rows Invalid Int Run 6 the ListBox and other Text Box values Invalid Int values by generating a new input parameter. (On, 0, Invalid Int) and in the Run (Off, Other, AlphaSpecial Char).

Table 3.12 Define all input parameter follows algorithm IPO_V

Run	Radio Button(2)	ListBox(2)	TextBox(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5	On	0	Invalid Int
6	Off	other	AlphaSpecial Char

When Table is arranged into fillRegisterProfile Test case. It found that of Run 1-6 a input parameters in the Radio button compared to the Title of the table fillRegisterProfile Test case the List Box comparable to Position and Text Box compared with Name and Sure name, When the input parameters in the similarity of both tables will have values in the table fillRegisterProfile Test case which can be put to the test in Manual test and Automated test follow table 3.13.

Table 3.13 Compared input parameter values from table 3.12

Run	**Radio Button(2) & Title**		**List Box(2)& Position**		**Text Box(3)&Name**	
1	On	นาย	0	0	Valid Int	วสันต์ อุทัยเลี้ยง
2	On	นางสาว	Other	1	Invalid Int	Mariya Kajana
3	Off	“”	0	0	AlphaSpecial Char	mol@
4	Off	“”	Other	3	Valid Int	รัตนวดี เศรษฐจิตร
5	On	นาย	0	0	Invalid Int	ธีระยุทธอุทัยเลี้ยง
6	Off	“”	other	2	AlphaSpecial Char	Tee$$

Table 3.14 fillRegisterProfile test case example from Pair wise Testing to Reducing input parameter values.

TC	Title	Name&Surname	Position	Expected Result
1	นาย	วสันต์ อุทัยเลี้ยง	0	Pass/Fail
2	นางสาว	Mariya Kajana	1	Pass/Fail
3	“”	mol@	0	Pass/Fail
4	“”	รัตนวดี เศรษฐจิตร	3	Pass/Fail
5	นาย	ธีระยุทธอุทัยเลี้ยง	0	Pass/Fail
6	“”	Tee$$	2	Pass/Fail

3.3 Pair test: An IPO-Based Testing Generation Tool.

An IPO-base test generation tool, called PairTest, that includes algorithm IPO_H for horizontal growth and algorithm IPO_V for vertical growth. PairTest also supports the reuse of test sets when systems are modified due to changes of input parameter/or values.

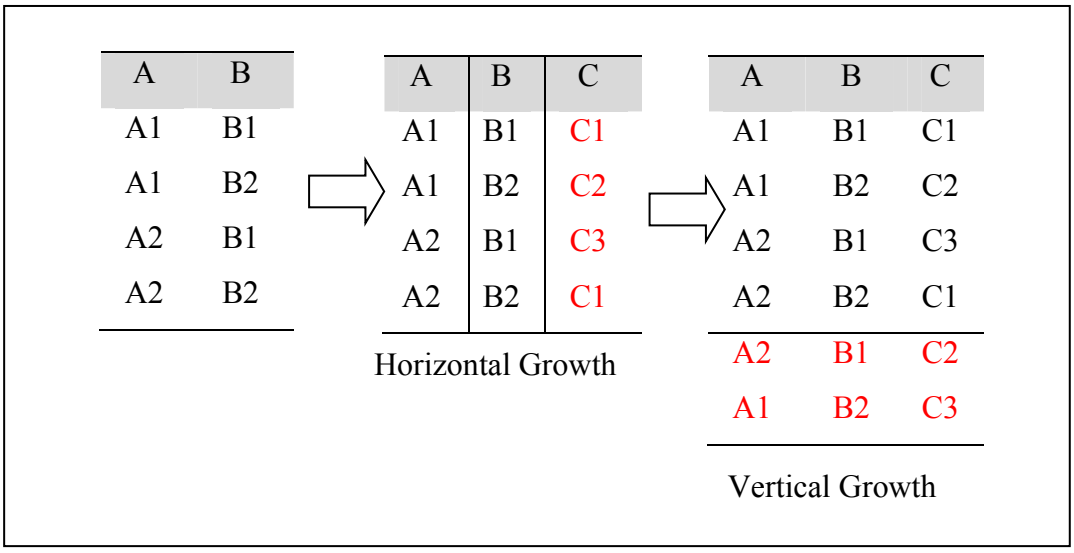

Figure 3.8 Consider a system with the following parameters and values

Another test generation tool for pair wise testing is AETG (Automatic Efficient Test Generate)[1] [2] . And used AETG[2] to produce pair wise test sets for the same six system. Table 3.1 shows the size information produced by AETG and PairTest to generate pairwise test sets for same six systems.

Table 3.15 Sizes of Pairwise Test Sets Generated by AETG and PairTest

System	S1	S2	S3	S4	S5	S6
AETG	11	17	35	25	12	193
Pairtest	9	17	34	26	15	212

S1: 4 3-values parameters

S2: 13 3-values parameters

S3: 61 parameters (15 4-values parameters, 17 3-values parameters, 29 2-values parameters)

S4: 75 parameters (1 4-values parameters, 39 3-values parameters, 35 2-values parameters)

S5: 4 3-values parameters

S6: 4 3-values parameters

It was shown that, for a system with n parameters, each having d values, the size of a minimum pairwise test set grows at most logarithmically in n and quadratically in d [2]. Empirical results based on AETG indicate that when the number of candidate tests case for a new test case is 50. The number of test case grows logarithmically in n. We have carried out empirical studies to determine the growth function for the size of pairwise testing set generate by PairTest in terms of n and d.

Table 2 shows the sizes of test sets generated by PairTest for System with d=4 and difference values on n. Table 3.1 shows the sizes of test sets generated by PairTest for systems with n=10 and difference values of d. According to statistical analysis[3], the values of s (number of tests).

Table 3.16 Result of PairTest for Systems with n 4-value parameters

n (#of parameter)	10	20	30	40	50	60	70	80	90	100
s (# of tests)	31	34	41	42	48	48	51	51	51	53
t (time in seconds)	0.11	0.16	0.22	0.44	0.77	0.99	1.37	1.81	2.23	2.96

In Table 2 and 3 grow in $O(\log (n))$ and $O(d^2)$, respectively. These empirical results match the theoretical results mentioned earlier.

Table 3.17 Result of PairTest for Systems with 10 parameters, each having *d* values

d (# of values)	5	10	15	20	25	30
s (# of values)	47	169	361	618	956	1355
t (time in seconds)	0.05	0.28	0.72	1.54	2.96	5.16

Table and also show time information for test sets generated by PairTest. The Execution time information was collected when PairTest was compiled and run on PC with Intel 450MHZ Pentium II Processor, Windows 98, and JDK 1.2.2.According to statistical analysis, the values of *t* (times for test generation) in table and grow in $O(n^2\log (n))$ and $O(d^3)$, respectively show in table . Based on the observation that the size of a test set generated by PairTest is $O(d^2\log (n))$.

Table 3.18 Growth Rates Compared

	n=1	**n=2**	**n=4**	**n=8**	**n=16**	**n=32**
1	1	1	1	1	1	1
logn	0	1	2	3	4	5
n	1	2	4	8	16	32
nlog n	0	2	8	24	64	160
n^2	1	4	16	64	256	1024
n^3	1	8	64	512	4096	32768
2^n	2	4	16	256	65536	4294967296
$n!$	1	2	24	40320	20.9T	Don’t ask!

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Result of test case from pair wise.

The results obtained from the use of input parameters from pair-wise testing process. When put to the test with Quick Test Professional (QTP) test case with the 6 input parameters by a conditions specified in the test document SRS Test plan In a FillRegisterProfile page.

4.1.1 Test Case ID 1

Test field Title by “นาย” Name&Surname by “วสันต์ อุทัยเลี้ยง” Position by “0” in the SRS test plan conditions has an expected result is pass.

4.1.2 Test Case ID 2

Test field Title by “นางสาว” Name&Surname by “Mariya Kajana” and Position by “1” in the SRS test plan conditions has an expected result is pass.

4.1.3 Test Case ID 3

Test field Title by “” Name&Surname by “mol@” and Position by “0” in the SRS test plan conditions has an expected result is pass.

4.1.4 Test Case ID 4

Test field Title by “” Name&Surname by “รัตนวดี เศรษฐจิตร” and Position by “3” in the SRS test plan conditions has an expected result is pass.

4.1.5 Test Case ID 5

Test field Title by “นาย” Name&Surname by “ธีระยุทธอุทัยเลี้ยง” and Position by “0” in the SRS test plan conditions has an expected result is pass.

4.1.6 Test Case ID 6 Test field Title by “” Name&Surname by “Tee$$” and Position by “2” in the SRS test plan conditions has an expected result is pass.

The results of such Show that the output from a test is valid or not. Depending on set a conditions in SRS test plan. Therefore, the results from testing the software with this approach. Show that Input parameters are generated and Combinations of Pair wise. Can be used to test the software, the results are acceptable. Can be written as the following table 4.1

Table 4.1 Expect results of input parameter to testing in a conditions of SRS Test plan.

TC	Title	Name&Surname	Position	Expected Results
1	นาย	วสันต์ อุทัยเลี้ยง	0	Pass (Requirement 1 satisfied)
2	นางสาว	Mariya Kajana	1	Pass (Requirement 2 satisfied)
3	“”	mol@	0	Pass (Requirement 3 satisfied)
4	“”	รัตนาวดี เศรษฐจิตร	3	Pass (Requirement 4 satisfied)
5	นาย	ธีระยุทธอุทัยเลี้ยง	0	Pass (Requirement 5 satisfied)
6	“”	Tee$$	2	Pass (Requirement 6 satisfied)

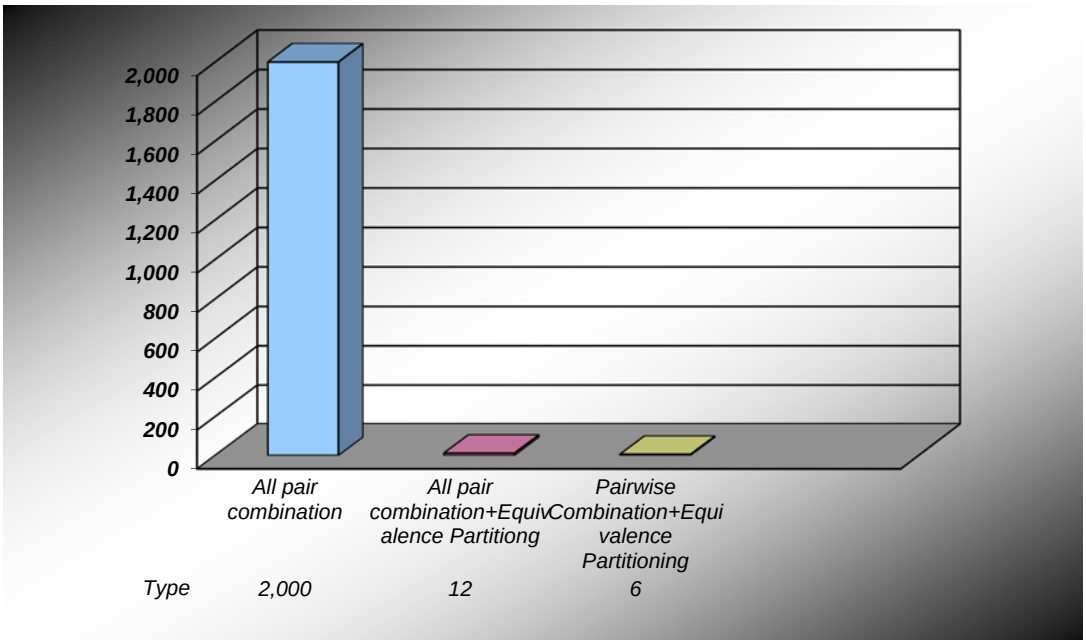

Figure 4.1 Shows the number of input parameters for each method in software testing.

Exhaustive Combination results in >2000 Test Cases. Conventional Software Testing technique (Equivalence Partitioning) result in 12 test cases. Pair wise Software Testing technique result in 6 test cases.

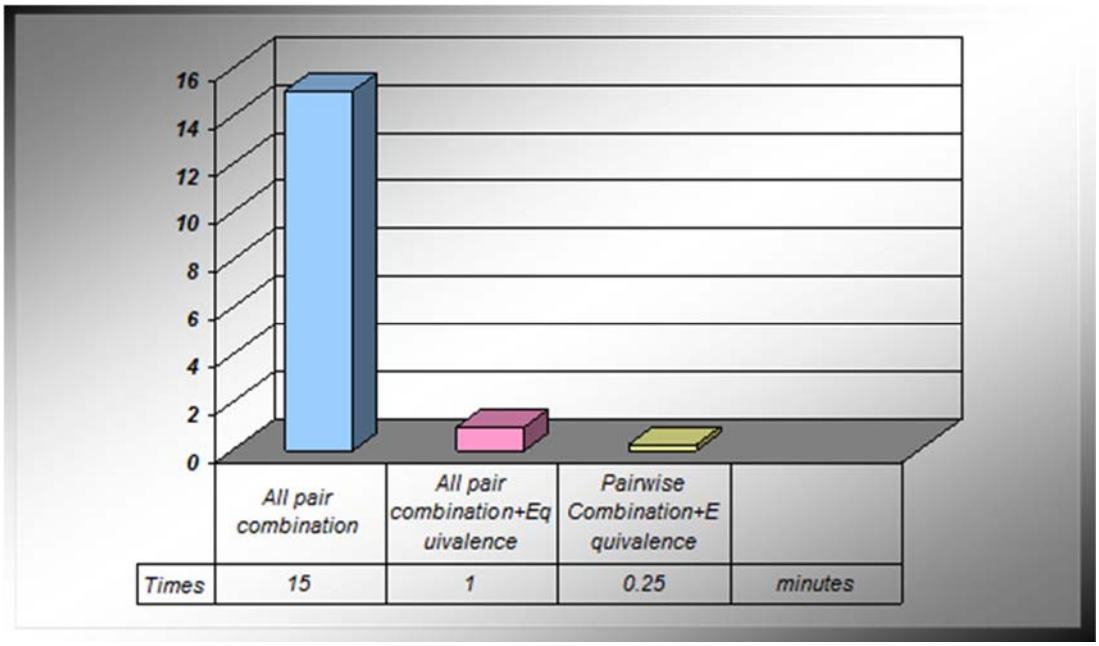

Figure 4.2 Shows times to spending of testing for each method in software testing.

Exhaustive Combination results in Times = 15 Minute. Conventional Software Testing technique (Equivalence Partitioning) result in Times= 1 Minute. Pair wise Software Testing technique result in Times=0.25 Minute.

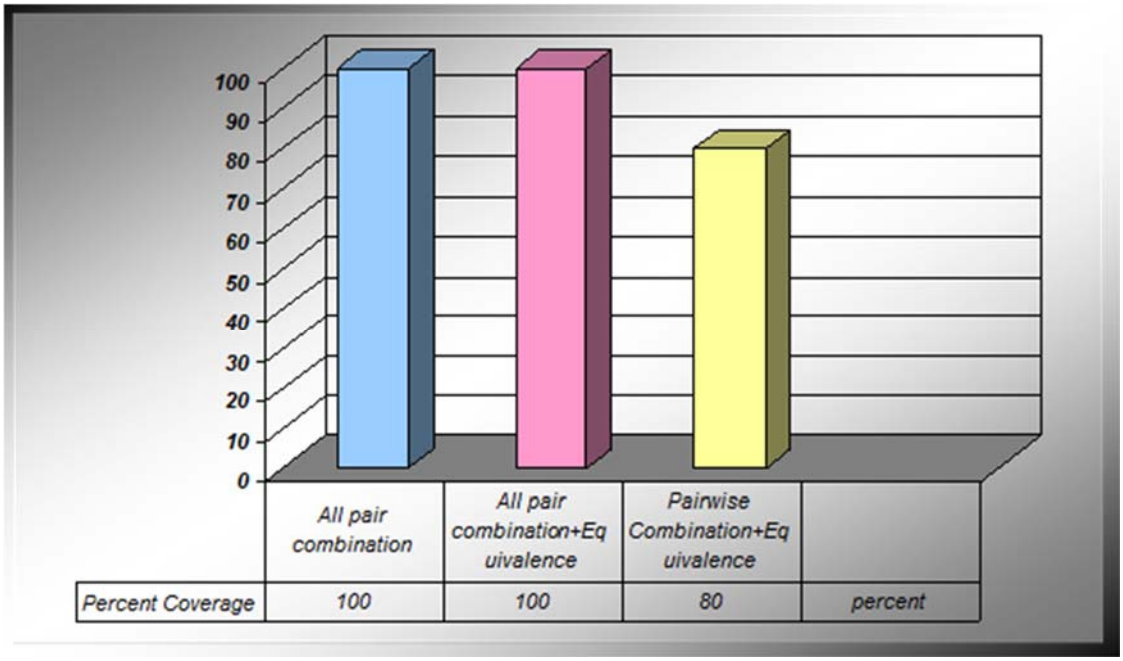

Figure 4.3 Shows percent coverage of input parameter in testing for each method.

Exhaustive Combination results in Percent Coverage = 100%. Conventional Software Testing technique (Equivalence Partitioning) result in Percent Coverage = 100%. Pair wise Software Testing technique result in Percent Coverage = >80%.

From the figure 4.1 if input parameters generated by All pair testing, must be take the time to test all the possible input parameters approximately 15 minutes. However, when the input parameters are generated with Pair-wise remaining input parameters, only 1 in 4 of all input parameters. That corresponds to the time spent in each test set to test the possibility of the input parameters. In table 8 show that taking time less compared to the input parameters were generated using All pair testing. However, a covered test suite was generated with Pair-wise Testing is no different from the Manual Testing for the input parameters have been set by the algorithm IPO, makes percentage coverage is still in an acceptable range as shown in Fig. 4.4

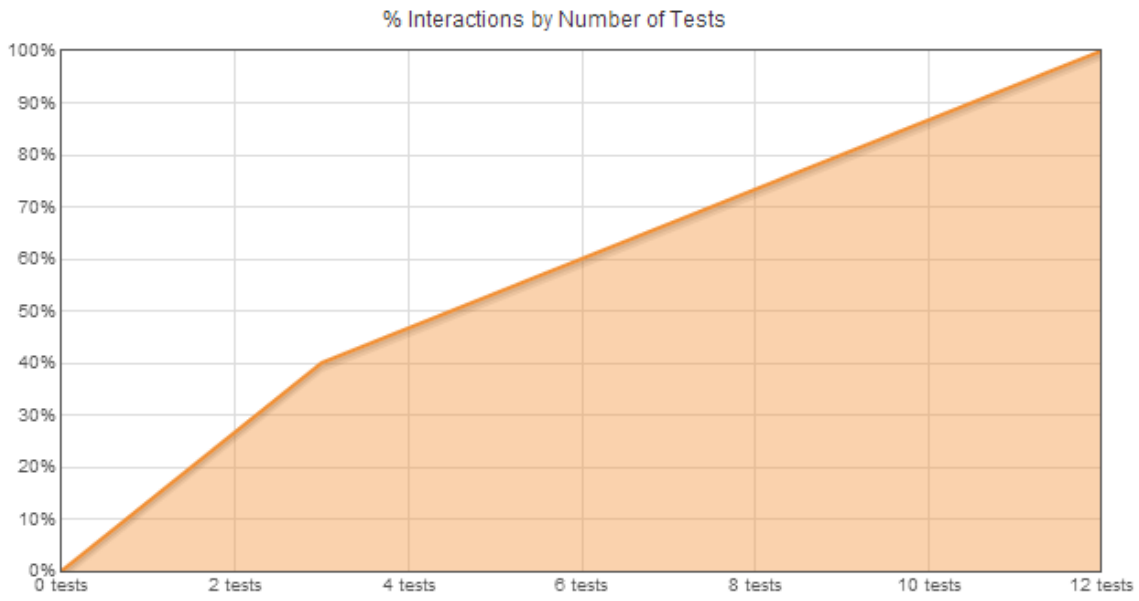

Figure 4.4 Test Analysis for FillRegisterProfile (3-way interaction) by www.hexawise.com

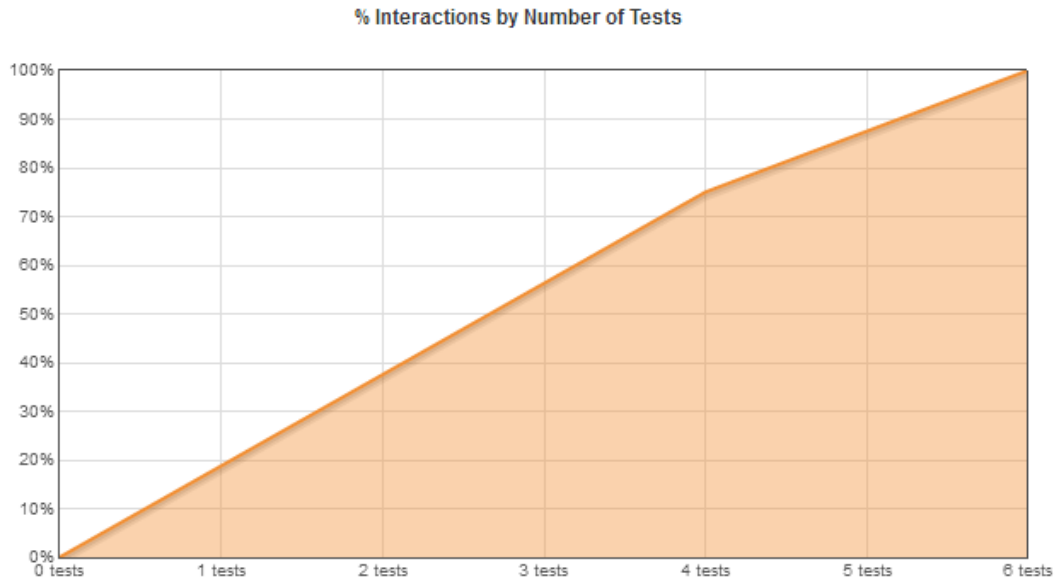

Figure 4.5 Test Analysis for FillRegisterProfile (2-way interaction) by www.hexawise.com

4.2 Discussion

In number of parameter can be reducing the number of input parameters of all parameters of Exhaustive Combination accounted for 3 percent of the time for the test less proportionally.

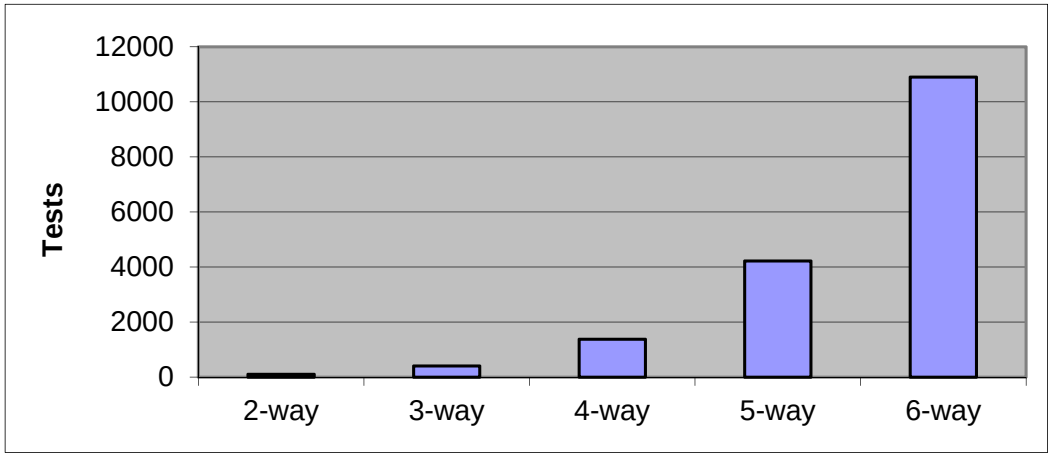

Figure 4.6 Shows *t*-way interactions of input parameter in testing

Pairwise or 2-way analysis is actually pretty effective because the combinatorial fault model suggests that most errors that result from the interaction of input variables occurs between the simple interactions of 2 inputs. Also, many studies demonstrate that a pairwise or 2-way analysis of input values is very effective in exposing a high percentage of multi-modal (and single mode) defects. But, although pairwise testing is effective in revealing more than 50% of the multi-modal errors there could still be bugs lurking that are caused by more complex interactions of input variables. In a found that some additional multi-modal bugs were exposed when 3 or more input values interacted as illustrated in the graph below.

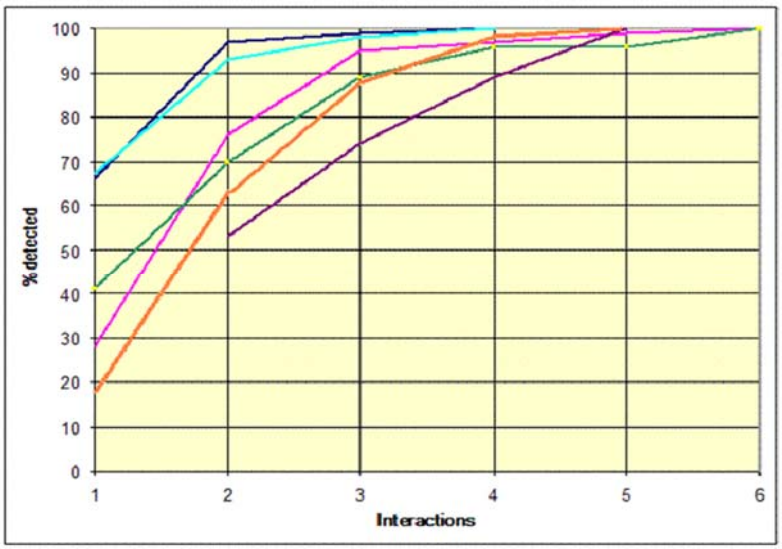

Figure 4.7 Combinatorial Methods for Cyber-security Testing

Fortunately, in order to increase the order of *n*-way combinations being tested. But, it do need a tool capable of performing a (2 + *n*)-way analysis of input parameters, and PICT (and other tools) can do just that. Passing the /o:*n* switch to the PICT command line (where *n* is the order up to the maximum number of parameters) will cause PICT to generate a *n*-way analysis of the input values. Why don't we just start with 3-way or 4-way analysis? The simple answer is cost. Each increment in the n-way order causes an approximate quadratic increase in the number of test combinations as illustrated in the table below. Suppose, this particular feature had 421,200 possible combinations.

Table 4.2 Blocks Covered of n-way interaction

	2-way (pairwise)	**3-way**	**4-way**
Number of Tests	136	800	3533
Blocks Covered	979	994	1006

Even if test is automated we might not want to immediately do a 3-way or higher analysis of value interactions. (Automation! = free testing.) The strategy we use is to start with 2-way interactions, then generate different 2-way sets until no more bugs are found. Then generate an output for 3-way interactions to look for more complex issues. If no bugs are found with 3-way interactions we might jump to 4-way or not depending on the complexity and criticality of the feature, and the confidence of the tester.

The bottom line is that sometimes pairwise or 2-way analysis of values for interdependent input parameters is not sufficient to find bugs cause by more complex interactions. To help resolve this problem testers need a tool that can generate *t*-way combinations beyond a basic pairwise analysis.

4.3 Summary

- Pairwise testing is the execution of a test data set that covers all combinations of the *selected test data values* for every pair of a system's input variables.
- Based on the observation that the size of a test set generated by PairTest is $O(d^2 \log(n))$.
- It is used on combinatorial testing problems that appropriate for pairwise testing (i.e., program outputs influenced by at most two inputs).

The strength of pairwise testing is that it systematically generates small test sets that combine many test data values.

The weakness of the pairwise technique is that is does not take into account the combinatorial characteristics of the system under test.

CHAPTER 5
CONCLUSION

5.1 Conclusion

In studies in Software testing process is found the researcher oriented reducing the number of inputs process, time spent and increase the coverage and find bugs in the testing process to increase the efficiency and capabilities of the process of finding defects in the program. This research is focus on studies of method of pairwise and applied testing process by referring algorithm has been invented by researchers testing the software. To reduce the number of input parameters in the system. Which follow with the conditions prescribed of the SRS test Plan. The results obtained when compared with all pair testing will find that if the number of input parameters much more. For searching defects and errors in the testing process, time to testing it more. Therefore, the study process to reduce the number of input parameters and find bugs in programs focused on education Pair wise in other research found.

5.2 Recommendation

This method has the ability to reduce the number of input parameters and discover faults or defects in the program are on average higher. Therefore, the study process to reduce the number of input parameters and find bugs in programs focused on studies Pair wise in other research found. This method has the ability to reduce the number of input parameters and discover faults or defects in the program are on average higher more than other methods. But the invented many an algorithm related methods Pair wise testing, Hexa-wise, Combinatorial Method, Combination Strategies Relationship Method. That the software tester can be applied to test the software. In the present, automated testing the present used in the software testing process. Software testing process is included implementation and testing together to reduce the

test procedure and repeated the test until the end of the testing process. Follows in figure 5.1

Test automation is an activity that aims to test the most frequently used and least frequently changed parts of an application automatically. This can be done by test automation tools by the help of programming languages supported. Therefore, related test cases are written as a piece of code. Thought test automation activity is criticized under testing activity, it is a software development activity. Every change in the application should be reflected to the test automation scripts so, even the cleverest test scripts are produced, and maintenance activity comes out as problem of test automation since the application is a live object.

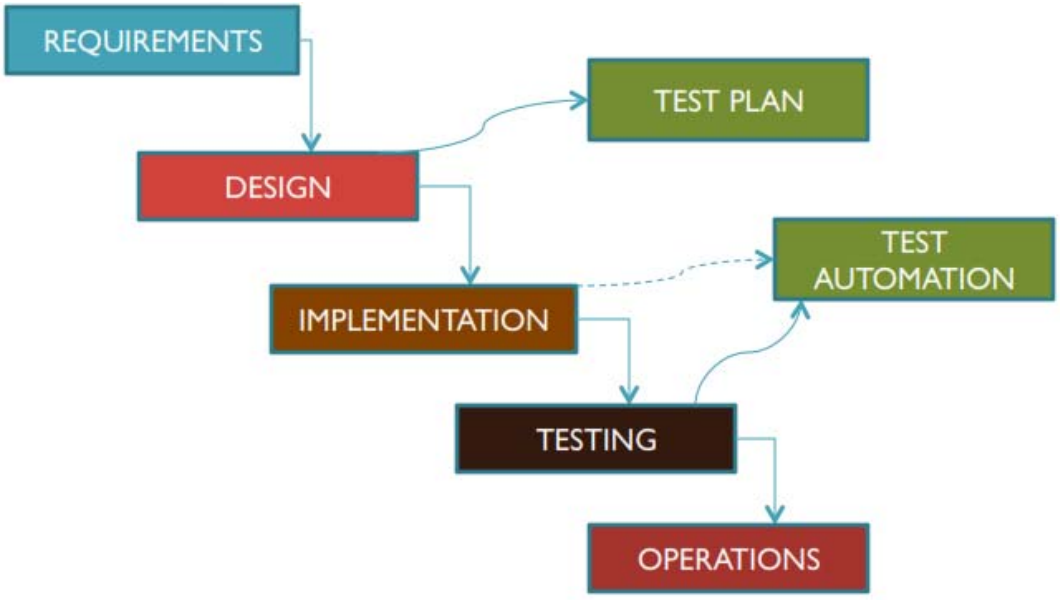

Figure 5.1 Test Automation fit in the Software Life Cycle.

5.3 Future work

A testing applied to the platform other such IOS and Android and others to test the usability of the software. Optimize performance of operating systems on mobile device in the next step.

REFERENCES

1. Y. Lei and K. Tai. In-Parameter-Order: A test generation Strategy for pairwise testing. In Proceedings of 3rd IEEE Intl. Symp. on High Assurance Systems Engineering, pages254–261, 1998. **Retrieved from** http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=731623&tag=1
2. D. Cohen, S. Dalal, M.Fredman, and G. Patton. The AETG system: An approach to testing based on combinatorial design. IEEE Trans. Software Eng., 23(7):437–443, 1997. **Retrieved from** http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=605761
3. Rick Kuhn and Raghu Kacker, Y.Lie, JustinHunter.CombinatorialTesting. IEEEComputor Society 2009. **Retrieved from** http:*//csrc.nist.gov/groups/SNS/.../kuhn-kacker-lei-hunter09.pdf*
4. Shiba T., Tsuchiya T., &Kikuno T. (2004). Using artificial life techniques to generate test cases for combinatorial Testing. Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC'2004), Hong Kong, China, 72-78. **Retrieved from** http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1342808
5. Rangsit Sirirangsri. (2013) Software testing ISBN: 978-974-8445-29-8 page 149-163, 2013.
6. Mandl R. (1985) Orthogonal Latin squares: An application of experimental design to compiler testing. Communications of the ACM, 28(10), 1054-1058. **Retrieved from** http://dl.acm.org/citation.cfm?id=4375
7. Shubhra Banerji (2012) Orthogonal Array Approach for Test Case Optimization. Inter National Journal of Advanced Research and Communication Engineering Vol.1, Issue9, November 2012. **Retrieved from** http://www.ijarcce.com/upload/november/5.Orthogonal%20Array%20Approach%20for%20Test.pdf
8. Test Tools, Justin Hunter: www.hewawise.com

9. Relevant standard-IEEE Std.839-1998 for Software test plan.
10. Templates for testing: http://www.tuffy.aust.com/tcs20003.htm
11. Functional and System Test plan Standard: Version 2.1 Copyright© 1992 and 1993 by Alan R. Weiss and Steven Menyhurt.
12. Raghu N.Kacker (2013) Combinatorial testing for software: An adaptation of design experiments. Measurement (2013), http://dx.doi.org/10.1016/j.measurement.2013.02.021. **Retrieved from** http://www.sciencedirect.com/science/article/pii/S0263224113000596
13. Lixin Wang, Renxia Wan (2010) **A *New Method of Reducing Pair-wise*** Combinatorial Test Suite. Computer and Information Science, vol.3. No.1Retrieved from http://www.ccsenet.org/journal/index.php/cis/article/download/5119/4296
14. D.R. Kuhn, R. Kacker, Y.Lei, **"Random vs. Combinatorial Methods for Discrete Event Simulation of a Grid Computer Network"**, Proceedings, Mod Sim World 2009, Oct. 14-17 2009, Virginia Beach, pp. 83-88, NASA CP-2010-216205, National Aeronautics and Space Administration.
15. Software testing –all pair testing: http://www.tutorialspoint.com/Software_testing_dictionary/all_pairs_testing.htm

APPENDICES

APPENDIX A

TESTPLAN TEMPLATE

Introduction

Document overview

This document is the software test plan of the XXX software development project. It contains the list of tests, which are executed during the phases of XXX integration and verification:

Software Integration tests,

Software Verification tests.

Some sections of this document are about the organization of tests and may already be described in the project management plan. If so, reference the matching section in the project management plan.

Abbreviations and Glossary

Abbreviations

Add here abbreviations

Glossary

Add here words definitions

References

Project References

#	Document Identifier	Document Title
[R1]	ID	Add your documents references. One line per document

Standard and regulatory References

#	Document Identifier	Document Title
[STD1]		Add your documents references. One line per document

Conventions

Add here conventions

This Template is the property of Cyrille Michaud

License terms : see http://blog.cm-dm.com/post/2011/11/04/License

Test environment

This section describes the environment of tests, from the point of view of organization and logistics. It is intented to ensure the smooth progress of tests (bugs apart) on each site.

Assumption : there are two test sites : one in your offices and one in customers offices. Duplicate the sub-sections below if there are more than two sites.

Integration and factory test site

Hardware test Platform

Describe where is located the test platform and opening hours, if necessary.

If by chance there are specific requirements about power supply, room, air conditioning, don't forget them (they may also be described outside this document. That's not really the job of software developers!).

Describe the hardware used to test your software in your offices. Identify accurately the hardware items :

If standard computers and servers

Hardware configuration

Processor

Memory

Hard disk

Network connections

Wireless capabilities : Wifi, bluetooth

If you use specific hardware (hardware simulator of a machine that you don't have, hardware lended by your customer or a 3rd party, electronic card, a medical device …)

Their purpose

Name

Manufacturer

Configuration, version

Firmware version

Lot number, serial number

Anything else …

Consummables

CDROM, memory sticks, tapes …

Printer cartridges , paper …

You may draw a deployment diagram, define a network address plan, electric power supply plan, a room plan …

Software test tools

Identify accurately the software used for test :

OS‘s and service packs

OS drivers (if specific for you)

Backup / recovery tools

Web, blogs, CMS, Databases engines,

Memory, disk usage, CPU, and network analysers,

Test coverage or test management tools

Simulator, data generator of software or hardware that you don’t have

Any tiny (or big) software made by you to do the tests

For simple projects, most of these may be tools provided with the OS (df, du, ps, top, dmesg, taskmanager, control panel …), or consumer products (MS Office, open office …).

Describe also the bug repository tool and the way bugs are collected.

Test Data and documentation

Describe the sets of data used during tests. Their identification, structure, content, location, storage, (structure and content may already be described in the conception documents),

input files,

data files,

scripts to generate data,

Output files, log files

Describe which documentation is delivered for the tests (eg Software tests description, Instructions For Use …), if it is printed or online.

Other test materials

If specific hardware is required : paper in exotic format, a stopwatch, a ruler,

a compass, a willy waller 2006

And also pizzas, bier, red bull …

Installation, set-up, and maintenance

If necessary, describe the installation and set-up of the tests platform, before its use.

Describe also maintenance opérations, if any.

Personnel

If necessary, describe the persons or profesional profiles of persons who do the tests, their number, the special skills required.

Customer/ Field test site

Repeat the pattern above

If your product is tested in an health care centre, or if your customer is a medical device manufacturer, have in mind that you may provide your customer with hardware, software, data and documentation. You may install it and maintain it. His opening hours may be constrained, his personnel shall have specific qualifications …

If you work directly with praticians (of your medical advisory board, for example), who are going to test your product in their offices, some sub-sections may not be relevant, focus on how tests input/output data are managed, how tests logs and bugs reports are collected.

Tests identification

Testing phases

This test plan defines all tests to verify all requirements of XXX software in the following successive testing phases :

Unit tests,

Integration tests,

Factory tests,

End-user or Customer tests.

Change the list to fit your software developement project.

Requirements are defined in SRS, ref XXX.

Test categories

Tests are distributed in categories, depending on the tests performed:

Risk analysis mitigation tests,

Human factors engineering tests,

Main functions,

Response time,

Data exchange…

Add your categories to the list, but keep the first line!

Test progression

The tests are progression depends on the testing phase:

Unit tests:

The testing tool automatically sets the test progression. There is no dependency between unit tests.

Integration tests: tests are executed according to the following rationale:

Integration with interface A alone

Integration with interface B alone

Integration with interface A and B

Factory tests: test progression is defined according to the following rationale:

Inspection tests are done at first,

Tests in category xxx are done afterwards,…End-user tests:

Test progression is defined according operational scenarios.

Describe your own rationale.

Test coverage

Describe tests coverage rationale. Example :

Tests coverage depends on the testing phase:

Automated tests cover all components of XXX software.

Integration tests cover all interfaces requirements of XXX software.

Alpha Tests cover all requirements defined in the SRS, excepted ….

Beta Tests cover all requirements defined in the SRS,

The traceability between tests and requirements is listed in the §6 Requirements traceability.

A requirement may require more than one test to be verified. In this case, it appears in all tests, which verify it.

Data recording, post-processing, and analysis

Describe how raw test data are recorded, if necessary post processed and analyzed.

For example, manual, automatic, and semi-automatic techniques for recording test results.

It may be a list of small procedures, which are launched before/after a session of tests or before/after a subset of tests.

Describe also where tests data is stored (scm repository, shared directory …).

Test identification and content

Each test is unique and contains:

A unique identifier,

The tests category,

A textual description of test objective,

The traceability of the SRS requirement(s),

The verification method (I, A, D, T),

Data recording, post-processing and analysis procedure,

Assumptions and constraints, if any

Safety, security and privacy concerns, if any.

The identifier has the following structure:
Define your own unique identifiers.
For example, concat the chars "T-", the srs requirement ID being tested, "-", and an incremental number (if more than 1 test is need to verify the requirement).

A test identifier is unique. If a test has to be completely redefined between two versions of this test plan, the previous reference is cancelled and a new identifier is attributed to the test.

Planned tests

For each phase, a list of tests is defined with an order of execution if necessary.
Tests Phase xxx
Tests coverage
The tests of phase xxx cover the following range:
For example: interfaces and critical requirements
Requirements of §x and §y of SRS
A functional domain
All requirements
Planned tests
Planned tests of phase xxx are listed in the table below. They are executed in the same order.
Fill the table with your tests,

Identifier	**Description**	**M**	**Category**
T-SRS-REQ-010-1	Verify that XXX …	I	xxx
T-SRS-REQ-010-2	Verify that XXX …	I	yyy
T-SRS-REQ-020-1	Verify that XXX …	D	Yyy
T-SRS-REQ-030-1	Verify that XXX …	D	Yyy

Tests are fully described in the software tests description (STD) document.

Tests Phase yyy

Repeat the pattern for each phase

Tests coverage

The tests of phase yyy cover the following range:

For example: interfaces and critical requirements

Requirements of §x and §y of SRS

A functional domain

All requirements

Planned tests

Planned tests of phase xxx are listed in the table below. They are executed in the same order.

Fill the table with your tests,

Identifier	**Description**	**M**	**Category**
T-SRS-REQ-010-1	Verify that XXX …	I	xxx
T-SRS-REQ-010-2	Verify that XXX …	I	yyy
T-SRS-REQ-020-1	Verify that XXX …	D	Yyy
T-SRS-REQ-030-1	Verify that XXX …	D	Yyy

Tests are fully described in the software tests description (STD) document.

Tests schedules

This either described in the project management plan, or here, or both, if some details were missing when the project management plan was written.

The schedule for conducting the tests is the following:

You may add a graphical representation of the schedule (gantt, …) if

Phase xxx:

Set-up and installation of platform: from yyyy/mm/dd to yyyy/mm/dd

Installation, copy of tests data

Pre-tests, personnel training, dry-run

Tests readiness review

Tests execution

Intermediate reviews

Final test review

Phase yyy:

Set-up and installation of platform: from yyyy/mm/dd to yyyy/mm/dd

Installation, copy of tests data

Pre-tests, personnel training, dry-run

Tests readiness review

Tests execution

Intermediate reviews

Final test review

Requirements traceability

Add here the traceability of SRS requirements.

Identifier	Description	SRS Requirement	M
T-SRS-REQ-010-1	Verify that XXX …	SRS-REQ-010	I
T-SRS-REQ-010-2	Verify that XXX …	SRS-REQ-010	I
T-SRS-REQ-020-1	Verify that XXX …	SRS-REQ-020	D
T-SRS-REQ-030-1	Verify that XXX …	SRS-REQ-030	D

The verification methods (I,A,D,T) in this table shall match the verification methods of SRS requirements in §3 of SRS.

APPENDIX B
TESTCASE OF SRS TESTPLAN

TC ID	Test Steps	Title	Name	Education	Job	User name	Password	Expected Result
1	1.Open IE and go to	นาย	ขจรศักดิ์ ไกลอื่น	ปริญญาตรี	Tester	ItTest	Song2013	Pass
2	2.Enter Title	“ ”	ขจรศักดิ์ ไกลอื่น	ปริญญาตรี	Tester	ItTest	Song2013	Fail
3	3.Enter Name 4.Select List	นาง	Maysa katana	ปริญญาตรี	Tester	ItTest	Song2013	Fail
4	5.Check Check Box	นาย	น้ำใจรักศิ หนึ่ง11	ปริญญาตรี	Tester	ItTest	Song2013	Fail
5	6.Enter Username 7.Enter Password	นาย	กิจติกาญเจริญ	ปริญญาตรี	Tester	ItTest	Song2013	Fail
6	8.Click Register	นาย	มาค	ปริญญาตรี	Tester	ItTest	Song2013	Fail
7	9.Click Message	นาย	อมรเทพเกียรติกุลยิ่งใหญ่ สารัญชานนท์	ปริญญาตรี	Tester	ItTest	Song2013	Fail
8	10.Close Browser	นาย	“ ”	ปริญญาตรี	Tester	ItTest	Song2013	Fail

9	นาย	น้ำใส คงสง่า	“ ”	Tester	ItTest	Song2013	Fail
10	นาย	น้ำใส คงสง่า	ปริญญาศรี	“ ”	ItTest	Song2013	Fail
11	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	%^^*()	Song2013	Fail
12	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	nar	Song2013	Fail
13	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	Songbutterfl er255855	Song2013	Fail
14	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	It test	Song2013	Fail
15	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	“ ”	Song2013	Fail
16	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	Ittester	%^&/*-*/*	Fail
17	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	Ittester	Song	Fail
18	นาย	น้ำใส คงสง่า	ปริญญาศรี	Tester	Ittester	Songbule2013199 890	Fail
19	นาย	น้ำใส คงสง่า	2	Tester	Ittester	So ng2013	Fail

APPENDIX C

Pair-Wise Testing Applied with Online Registration

Wasan Uthaileang, Supaporn Kiattisin and Adisorn Leelasantitham

Technology of Information System Management Program, Faculty of Engineering, Mahidol University

25/25 Puttamonthon, Nakorn Pathom 73170, Thailand

wasan.uth@mahidol.ac.th, supaporn.kit@mahidol.ac.th, adisorn.lee@mahidol.ac.th

***Abstract*—Pair-wise testing is a software testing technique used for design of test cases to test every possibility input parameters pairing interaction together. This result is to decrease in the number of test cases. This paper presents to reduce the number of test cases with pair-wise testing using the reference with algorithm invented and accepted in software testing. It is applied to case in the online registration website (Thai language) to generate test suits in every possible of input parameters. Reducing the number of input parameters is to decrease time consumption of testing and it helps to find highest defects with coverage in testing. The result from the software testing with pair-wise has more effectiveness than the all pair testing up to 60-80%.**

The reason is to use pair-wise testing in test because society of tester in Thailand has a little to use such method to reduce input parameters before testing. Therefore, it will make a high cost of each testing. However, pair-wise testing in the testing process has an importance for Software Development Life Cycle (SDLC) and Software Testing Life Cycle (STLC) in Thailand.

***IndexTerms*—Pair-wise testing, combination, input parameters, Orthogonal Array**

I. Introduction

Pair-wise testing or2-way interaction was invented the algorithm to reduce the size of a number of test cases. There are 2 algorithm; IPO In-parameter-order [1] and the AETG [2].which recognized and be the most reference. Principles of test suite are to find errors or defects that hard to find and input parameters without Combinatorial Software Testing [3].Software testing in the modern world has 3 objects: reducing amount of time used in experiment. Find the maximum number of defects by using a minimum number of test case [4]. Especially to test the accuracy of the system. With collaboration in various input parameters such as browser, database systems. That has to work on different operating systems. These may be cause of problems as the number of test case to cover every possibility (not included, impossible case) all resources of computer system cannot compile all the possibility in all tests. Moreover, the risk of software testing cannot cover every possibility. Although some of test cases decreased with Equivalence Partitioning and Boundary-Value Analysis can reduce the number of tests at a certain level [5] However, there are a lot of test case for testing every possibility are exists.

II.Applied Software Testing with Pair-wise testing.

Pair-wise testing is a criteria requirement testing which requires each pair of input parameters in the system, besides, it also combine all of the possible input parameters at two parameters to cover at least one case2-way interaction [1].to test the Pair-wise efficiently, therefore, using a concept "*Orthogonal Array"* to reduce amount of test case process[6]. This is a way of statistical methods that applied to test the pair-wise. Orthogonal Array is ideally suited to test every possibility [2].Orthogonal Array is resulted from the interaction of the input parameters to the system by pairing the values of any two columns within the array [7]. All possible values of the pair column will be used to test interoperability between the input parameters defined. In case of test the interaction with the input parameters in the form of components.

2-way interaction plan the input parameters include the following parameters.

A = "A_1", B = "B_1", C = "C_1"

That in the generated set of **2-way** tests, at least one test that covers **every** possible interaction

involving values from 2 different parameters. One example **2-way** test is:

A = "A_1" Together with **B = "B_1"**

So **every** valid **2-way** combination is covered, but it is possible that a valid **3-way** combination is **not** covered. For example, there might **not** be a single test that includes all 3 of the following:

A = "A_1" together with **B = "B_1"** together with **C = "C_1"**

Thus, all of the 2-way interaction are required to cover all the input parameters. But it is possible that the 3-way Interaction correct input parameters may not cover all of the following:
A = "A_1" with B = "B_1" with C = "C_1"

Compared to 2-way interaction test plan with 3-wayinteraction is more thorough and better able to detect defects that are hard to find, but it will take longer to execute the test plan.

Table 1: shows the Orthogonal Array of equation $L_4(2^3)$.

Factor			
Row	A	B	C
1	A1	B1	C1
2	A1	B2	C2
3	A2	B1	C3
4	A2	B2	C1

$$L_4(2)^3$$

L_4= Number of rows

(2)= Maximum value=2, 3…,n

^{3}=Number of columns

Figure 1: shows the equations used to Orthogonal Array.

III. Generate all the possible input parameters.

Suppose that A and B are parameters. Then input parameters A (A_1, A_2) and input parameters B (B_1, B_2) Matching A and B and result is {(A_1, B_1), (A_1, B_2), (A_2, B_1), (A_2, B_2)} When the parameter extended and input parameters C (C_1, C_2, C_3) respectively. So when the combination between the three parameters are input parameters (A_1, B_1, C_1), (A_1, B_2, C_2), (A_2, B_1, C_3) but not cover the matching of input parameters. {(A_2, C_1), (B_2, C_1), (A_2, C_2), (B_1, C_2), (A_1, C_3), (B_2, C_3)} Thus, (A_2, B_2) on the match with the input parameters (C_1, C_2, C_3) input parameters C_1 will get (A_2, B_2, C_1) covers only match (A_2, C_1) And (B_2, C_1), but if input parameters C_2 will get (A_2, B_2, C_2) covers matching only (A_2, C_2), but if input parameter C_3 will get (A_2, B_2, C_3) covering the match specific (B_2, C_3) the result shown that the three input parameters (C_1, C_2, C_3) and C_1 have the more Comprehensive tests than C_2, C_3 therefore select Test set of (A_2, B_2, C_1) and will earn 4 the capture pair that not cover (A_2, C_2), (A_1, C_3), (B_1,. C_2), (B_2, C_3) to take generate new test to cover all four pairs, the next step is to test the Pair-Wise Testing by algorithm of IPO (Input Parameter Order), is divided by two al algorithm is IPO. (In-Order-Parameter).

Table 2: Input parameters of Equivalence Partition into Orthogonal Array.

A	B	C
A_1	B_1	C_1
A_2	B_2	C_2
		C_3

Table 3: all possible input parameter in Orthogonal Array.

Run	A	B	C
1	A_1	B_1	C_1
2	A_1	B_1	C_2
3	A_1	B_1	C_3
4	A_2	B_1	C_1
5	A_2	B_1	C_2
6	A_2	B_1	C_3
7	A_1	B_2	C_1
8	A_1	B_2	C_2
9	A_1	B_2	C_3
10	A_2	B_2	C_1
11	A_2	B_2	C_2
12	A_2	B_2	C_3

For example, to test the interaction with input parameters in the form of components with Collaboration such as online registration page FillRegisterProfile input parameters needed to create Test Case 2x10x100 = 2000 Test Case. (does not include terms Negative) If the input parameters will affect the test was not successful in a short time.

Applying with others techniques in such as Equivalence Partition or covering the specific

techniques may choose to use Boundary Value Analysis. The configurations on the input parameters in Table 2 and Level to the table. Next step, to generate all of the possibilities to create a test case of each input parameter is 12 case shown in Table 3.

IPO_H: definition of al algorithm assumes that T {p_1, p_2, ... p_{i-1}} horizontal grow the T = p_i is extended in each test in T by inserting into the p_i.

IPO_V: definition of al algorithm assumes that T {p_1, p_2,...p_{i-1}} horizontal grow of T = p_i for is a set of pairs not covered by T when | π |> 0 let generate new test sets in π. π and take the T to the "-" Unspecified Value of the parameter as if p_i is the last parameter, each value "-" set as $pk_1 \leq k \leq I$ replace every value of p_k, otherwise "-" These are replaced by the parameters of Horizontal Growth.

Step 1: algorithm applied by Mapping with principle of Orthogonal Array (Textbox, Radio button, List box) Mapping Parameter A, B, C and set capabilities to support the value of each input parameter in Orthogonal Array equation. L_4 (3^3) shown in Table 2.

Step 2: Define values the possibility of the input parameters in table {(On, 0, Valid Int), (On, other, Invalid Int), (Off, AlphaSpecial, Char), (Off, Other, Valid Int}).

Step 3: Table 5 shows that the input parameters in the table does not cover the test. To pairing the other 4 pairs follows {(Off, Invalid Int), (On, AlphaSpecial Char), (0, Invalid Int), (other, AlphaSpecial Char)} These can performed using the algorithm Input Parameter Oder (IPO) apply without having to input parameter is missing (Y.Lie and KC Tai 2002), but we can use to generate new test can take input parameters disappear into the Run 5The Radio Button (On) and putting - in the ListBox by algorithm IPO in the TextBox values AlphaSpecial Char (On, AlphaSpecial Char) Run 6 the Radio Button (Off) and put - in the Listbox based algorithm in the Text Box input Invalid Int (Off, Invalid Int).

Step 4: Put the first input parameters (Off, Invalid Int), (On, AlphaSpecial Char) are found in the input parametersListBox marked with required to out the remaining input parameter (0, Invalid Int), (other, AlphaSpecial Char) Run 5 rows by the values 0 ListBox and TextBox input values the rows Invalid Int Run 6 the ListBox and other Text Box values Invalid Int values by generating a new input parameter. (On, 0 , Invalid Int) and in the Run (Off, Other, AlphaSpecial Char).

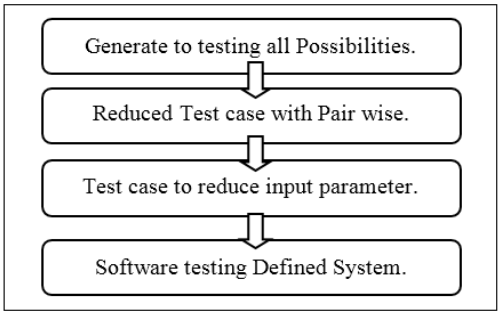

Figure 2: shows the process of testing the system with Pair-wise.

First, the example in Pairwise Testing of registration online. When taken the test case with FillRegisterProfile page by Equivalence Partition includes of the following parameter.

- *TextBox* Constraint assigned a value from 1 to 100
- *RadioButton* comprising Status on or off
- *ListBox* contains a value from 0 to 9

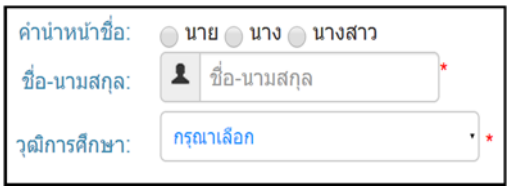

Figure 3: FillRegisterProfilePag

Exam: System requirement of Name and surname (Text Box)

This test plan using a standard format IEEE 829-1998 [9, 10,11]for plan, analysis, collecting and report results from testing system process of Training Registration Systems which is a website registration online for Software testing training in QuickTest Professional For Beginner course for general guest in 27-28 March 2013 at SIPA Training Room Chiangmai.

Table 4: Show System requirement of Name and surname.

Valid EC	Invalid EC	Invalid EC
*VEc1- must be Thai alphabet [ก-ฮ].	*IVc1.0 : Other languages (not Thai)	
VEc2- must not	IVc2.0 : any	

have any number in the name or surname.	number in name / surname	
VEc3- must have space between name and surname at least 1 space.	IVc3.0 : space less between name and surname	
VEc4- minimum 5 alphabets but not over 30 alphabets.	IVc4.0 : there are alphabet less than 5 in name / surname	IVc4.1 : there are alphabet more than 30 in name / surname
VEc5- Not null	IVc5.0 :Null	

*VEc: Valid Equivalence Case
*IVc: Invalid Equivalence Case

Conditions

1. In case of invalid name or surname, system must alert "Invalid" and show "Please fill valid name / surname".

2. In case of name / surname are not filled, system must alert "Invalid" and show "Please fill valid name / surname". When fill valid data, system will allow to click submit button and save data in database. Show in Table 5.

Table 5: Test Case for testing field name.

Test Case	Class	Input Condition	Expected Result
1	VEc1- must be Thai alphabet [ก-ฮ].	วสันต์ อุทัยเลี้ยง	System show Fill Register Profile Page
2	IVc1.0 : Other languages (not Thai)	Mariya Kajana	System must alert "Invalid" and show "Please fill valid name / surname".
3	IVc2.0 : any number in name / surname	จักรพงษ์ หงส์ต่ำ111	System must alert "Invalid" and show "Please fill valid name / surname".
4	IVc3.0 : space less between name and surname	ธีระยุทธอุทัยเลี้ยง	System must alert "Invalid" and show "Please fill valid name / surname".
5	IVc4.0 : there are alphabet less than 5 in name / surname	กาน	System must alert "Invalid" and show "Please fill valid name / surname".
6	IVc4.1 : there are alphabet more than 30 in name / surname	วิสุทธ์ แก้วเกริก เกียรติกาญกุล ประเสริฐเลิศรุจีศรี ไพรวัลย์	System must alert "Invalid" and show "Please fill valid name / surname".
7	IVc5.0 : Null	""	System must alert "Invalid" and show "Please fill valid name / surname".

Remark: not need to test, covered in test case number 1

Table 6: shows the reduction in the number of Level Input parameters used to operation together.

Radio Button(2)	ListBox(2)	TextBox(3)
ON	0	Valid Int
OFF	Other	Invalid Int
		AlphaSpecial

Table 7: shows the possibilities of Radio Button, List Box, Text Box.

Run	Radio	ListBox(2)	TextBox(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial
4	Off	Other	Valid Int
5			
6			

Table 8: shows input parameters to the table with the algorithm IPO.

Run	Radio Button(2)	ListBox(2)	TextBox(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5	On	-	AlphaSpecial Char
6	Off	-	Invalid Int

Table 7: shows the possibilities of Radio Button, List Box, Text Box.

Ru	Radio	ListBox(2	TextBox(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial
4	Off	Other	Valid Int
5			
6			

Table 8: shows input parameters to the table with the algorithm IPO.

Run	Radio Button(2)	ListBox(2)	TextBox(3)
1	On	0	Valid Int
2	On	other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5	On	-	AlphaSpecial Char
6	Off	-	Invalid Int

Table 9: shows input parameters to table and generating a new input parameter.

Run	Radio Button(2)	ListBox(2)	TextBox(3)
1	On	0	Valid Int
2	On	Other	Invalid Int
3	Off	0	AlphaSpecial Char
4	Off	Other	Valid Int
5	On	0	Invalid Int
6	Off	other	AlphaSpecial Char

When Table 9 is arranged into fillRegisterProfile Test case. It found that of Run 1-6 a input parameters in the Radio button compared to the Title of the table fillRegisterProfile Test case the List Box comparable to Position and TextBox compared with Name and Surename, When the input parameters in the similarity of both tables will have values in the table fillRegisterProfile Test case which can be put to the test in Manual test and Automated test.

Table 10: Compared input parameter values from table 5 with input parameter values from table 9.

Run	Radio & Title		List Box(2)		Text Box(3)&Name	
1	On	นาย	0	0	Valid Int	วสันต์
2	On	นางสา	Other	1	Invalid Int	Mariya
3	Off	“”	0	0	AlphaSpecial	mol@
4	Off	“”	Other	3	Valid Int	รัตนวดี
5	On	นาย	0	0	Invalid Int	ธีระยุทธ
	ff		ther		lphaSpecial Char	ee$$

Remark: List Box number 0 = Not Select 1=Tester 2=Administrator 3=Programmer 4=System analysis

Table 11: fillRegisterProfile Test case example.

TC	Title	Name&Surna	Educat	Expected
1	นาย	วสันต์ อุทัยเลี้ยง	0	Pass/Fail
2	นางสาว	Mariya Kajana	1	Pass/Fail
3	“”	mol@	0	Pass/Fail
4	“”	รัตนวดี เศรษฐจิตร	3	Pass/Fail
5	นาย	ธีระยุทธอุทัยเลี้ยง	0	Pass/Fail
6	“”	Tee$$	2	Pass/Fail

IV.Comparison reduced Test Case between Manual Testing and Pair-wise.

Testing input parameters of the from the table with program Hexawise Website, found that, if you test all possibilities of input parameters [8], all cases may be built Test case 2x10x100 = 2000 cases, When the Equivalence Partition principle applied to the test may be reduced to 12 test case and

when the principle of pair-wise testing the apply with the specified input parameters reduced to only 6 test case

From the table, if input parameters generated by Manual testing, must be take the time to test all the possible input parameters approximately 15 minutes. However, when the input parameters are generated with Pair-wise remaining input parameters, only 1 in 4 of all input parameters. That corresponds to the time spent in each test set to test the possibility of the input parameters. In table 8 show that taking time less compared to the input parameters were generated using Manual Testing. However, a covered test suite was generated with Pair-wise Testing is no different from the Manual Testing for the input parameters have been set by the algorithm IPO, makes percentage coverage is still in an acceptable range as shown in Fig. 4

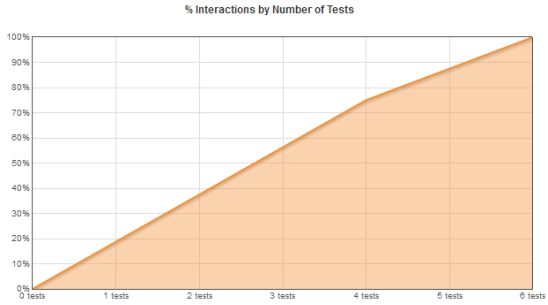

Figure 4: Test Analysis for FillRegisterProfile (2-way interaction) by www.hexawise.com

V.Conclusion

Pairwise testing also has several of Interest Algorithm. Because many people have invented. And improved development for effectively. Moreover, to finding bug and reduce the number of input parameters to a minimum. This paper presents the application of Pairwise reduction in the number of web pages online registration in Thai language shown in Table 10. By using Hexawise website in testing Input parameters are generated with Pair-wise. Found that the set of input parameters are very similar. Percent coverage compared to Manual testing and easy to do and not excessive complexity shown in Figure 5, 6, 7. In the future, if you can take up a highly complex algorithm, applied to test with other software. Reducing the number of tests, find errors, and reduce time and effectiveness.

Table 10: compares the performance of test case for All-pair testing and Pair-wise testing.

Result: Analyzing result shows testing between

No.	Type of Testing	No. of Case	Times	Percent Coverage
1	All pair testing	2000	≥15 minutes	100
2	All pair testing +Equivalence Partition	12	≥1 minutes	100
3	Pairwise testing +Equivalence Partition	6	≥0.25 minutes	≥80

types of testing 3 types

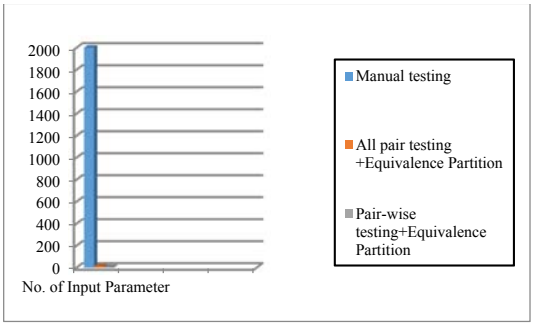

Figure 5: This picture refer data from table 10, shows Number of Cases that comparing 3 types: Manual testing, manual testing+Equivalence Partition, and Pair-wise testing.

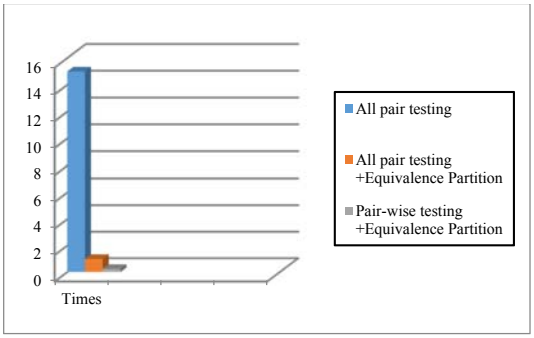

Figure 6: Figure 5: This picture refer data from table 10, shows Number of Cases that comparing 3 types: Manual testing, Manual testing Equivalence Partition, and Pair-wise testing.

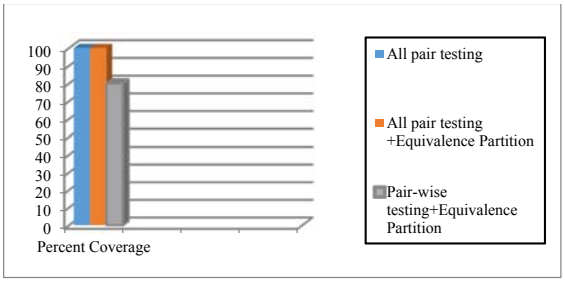

Figure 7: Figure 5: This picture refer data from table 10, shows Number of Cases that comparing 3 types: Manual testing, Manual testing Equivalence Partition, and Pair-wise testing.

Table 11. Comparison between the IPO and AETG Algorithm.

N	Subject	Number 0f Pair-wise est case (IPO)		Pair-wise test case AETG)	
		Before	After	Before	After
1	4^3	9	9	9	9
2	4^4	10	10	10	10
3	5^4	11	11	10	10
4	5^5	14	13	12	11
5	6^4	12	11	11	10

ACKNOWLEDGMET

The authors acknowledge Assoc. Prof. Rangsit Sirirangsi Associate Professor in Information Technology, Faculty of Science, Maejo University, and Expertise in software testing and automation test.

References

1] Y. Lei and K. Tai. In-Parameter-Order: A test generation Strategy for pairwise testing. In *Proceedings of 3rd IEEE Intl. Symp. on High Assurance Systems Engineering*, pages254–261, 1998.

[2] D. Cohen, S. Dalal, M. Fredman, and G. Patton. The AETGsystem: An approach to testing based on combinatorial design.*IEEE Trans. Software Eng.*, 23(7):437–443, 1997.

[3] Rick Kuhn and Raghu Kacker, Y.Lie,JustinHunter.CombinatorialTesting.IEEE Computor Society 2009

[4]Shiba T., Tsuchiya T., &Kikuno T. (2004). Using artificial life techniques to generate test cases for combinatorial

Testing. *Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC'2004)*,

Hong Kong, China, 72-78.

[5] RangsitSirirangsi. (2013) Software testing ISBN : 978-974-8445-29-8 page 149-163,2013

[6] Mandl R.(1985) Orthogonal Latin squares: An application of experimental design to compiler testing.Communications of the ACM,28(10),1054-1058.

[7] Shubhra Banerji (2012) Orthogonal Array Approach for Test Case Optimization. Inter National Journal of Advanced Research and Communication Engineering Vol.1,Issue9, November 2012.

[8]TestTools, Justin Hunter: www.hewawise.com

[9] Relevant standard-IEEE Std.839-1998 for Software test plan.

[10]Templates for testing: http://www.tuffy.aust.com/tcs20003.htm

[11] Functional and System Test plan Standard: Version 2.1 Copyright© 1992 and 1993 by Alan R. Weiss and Steven Menyhurt.

BIOGRAPHY

NAME	Mr. Wasan Uthaileang
DATE OF BIRTH	25 October 1980
PLACE OF BIRTH	Chonburi, Thailand
INSTITUTIONS ATTENDED	Maejo University, 2009-2011 Bachelor of Science (Information Technology**)** Mahidol University, 2012-2014 Master of Science (Technology of Information System Management**)**
HOME ADDRESS	848/575 Prachachuen Rd,Wongsawang, Bangsue,Bangkok 10800 Tel 089-496-3998 Email: wasan.uth@mahidol.ac.th
PUBLICATION / PRESENTATION	**Uthaileang W.**,Kiattisin S., Leelasantitham A., Pairwise testing Applied With Online Registration, The Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering (JICTEE), Thailand, 5-8 March. 2014