ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

An Exhaustive Test Case Diminution Using Proficient Constraint Handling Techniques

Dr. V. Sangeetha[1], M.Bharathi[2]

Assistant Professor, Department of Computer Science, Periyar University College, Pappiredipatti, Tamil Nadu, India[1]

Research Scholar, Department of Computer Science, Periyar University College, Pennagaram, Tamil Nadu, India[2]

ABSTRACT: Pairwise testing is a most familiar test planning technique. Due to resource constraints, it is nearly always impossible to exhaustively test all of these combinations of parameter values. In this paper we propose an approach to generate test cases for pairwise testing by applying and prioritizing input parameter and their corresponding values to be selected and also providing solution for the constraint handling between parameters and values and removing invalid test cases and evaluate the results of 100% coverage with specified constraints.

KEYWORDS: combinatorial testing, pairwise testing, constraint handling, test case generation

I. INTRODUCTION

Many software systems are highly configurable. It is usually infeasible to test all the possible configurations. To overcome this problem, Combinatorial Testing (CT) Techniques have been developed specifically for such systems.
CT is concentrated on pinpointing errors and faults arise owing to the interaction of altered parameters of software. Combinatorial methods aid to decrease the cost and increase the efficiency of software testing in numerous applications. CT is more efficient and effective compare with random testing. Also CT is an effective failure detection method for many types of software systems. CT which generates test groups by selecting values for each individual input parameters and by combining these parameter values.
For a system with p parameters, each of which has v values, the number of all possible combinations of values of these parameters is v^p[6]. Pairwise testing is a most familiar test planning technique. But investigations of actual failures in a number of software and systems show that pairwise testing may not be sufficient so high strength CT (t-way for t>2) may be desired. Due to resource constraints, it is nearly always impossible to exhaustively test all of these combinations of parameter values. Thus, a strategy is needed to select a subset of these combinations. One such strategy, called t-way testing, requires every combination of any t parameter values to be covered by at least one test, where t is referred to as the strength of coverage and usually takes a small value. The notation of t-way testing can reduce the number of tests.[6] For example, a system of 3 parameters that have 7 values each requires 7^3 tests for exhaustive testing. But it requires 49 for 2-way in pairwise testing. We can consider each combination of parameter values to represent one possible interaction among these parameters. However, testing has constraints on the combination of parameter values. As a result some combinations of parameter values are invalid. In this paper we propose an approach to generate test cases for pairwise testing by applying and prioritizing input parameter and their corresponding values to be selected and also providing solution for the constraint handling between parameters and values.
The paper is structured as follows: Section II describes the background of pairwise testing and constraint handling. Section III describes our proposed approach of generation of test cases with constraints and removes invalid test cases and executes the test cases. Finally, Section IV presents conclusion and a sketch for future work.

II. BACK GROUND

2.1 Pairwise Testing

Pairwise testing is a combinatorial testing technique which tests all possible combinations of set of input parameters. All pairs of combinations have been combined together at least once during the testing process. Instead of testing each and every combination, all individual pairs of interactions are tested. Investigations show that software defects are triggered by a single input parameters or a combination of two input parameters [8]. To demonstrate the concept of

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

pairwise testing, consider a system with four parameters A, B, C and D. Each of the first three parameters consist two values {A1, A2}, {B1, B2}, {C1, C2} and the fourth parameter consist of three values {D1, D2, D3} respectively. In order to test all combinations would require 2 X 2 X 2 X 3 = 24 test cases. But a generated pairwise test case set includes 6 test cases covered in all parameter interactions for this circumstance.

2.2 Constraint handling

In testing, some combination of parameters and values are frequently invalid and untestable because they do not exist for the system under test. The pairwise testing technique does not handle the constraints between input parameters and values. We must review the results obtained from pairwise testing and manually delete bad pair test cases themselves. Most existing test generation methods cannot deal with any constraints and finally ignore them. Ignoring constraints generate invalid test sets. When the generated test set contains many invalid test cases, this will cause a loss of combination coverage. Hence, proper constraint handling is a key issue for test set generation. We must specify the constraints before test set generation [5], [8], [14].

III. PROPOSED APPROACH

The proposed technique focuses on generation of test sets without and with constraints and compares the results. The whole process consists of four steps:

1) Select parameters and values
2) Define parameters and constraint values
3) Generate test cases
4) Execute test sets

3.1 Select parameter and values

The parameters and values are collected from the database or environments. To illustrate the concept of pairwise testing, consider a transport system with four parameters and their respective values and their symbolic representations are shown in Table1.

Parameters and their values	Symbolic Representation
Vehicle Type = { 2wheel , 4wheel }	V = { v1 , v2 }
Fuel type = { petrol , diesel , gas }	F = { f1, f2, f3 }
Accommodation = { single , double, suite }	A = { a1, a2, a3 }
Specialization = { ac , non-ac }	S = { s1 , s2 }

Table 1 : Parameters and Value representation

There are two vehicle types, three fuel types, three accommodation options and two specialization options. Different end users may use different combinations of these parameter values.

3.2 Define parameters and constraint values

Some combination of parameters value is invalid from Table 1 which will generate worthless test cases and must be eliminated from the result of the test set. Table 1 show the possible input parameters and values could be executed for Vehicle types (V), Fuel types (F), Accommodation (A), and Specialization (S). Defining parameters and values with ACTS tool are shown in Figure 1.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

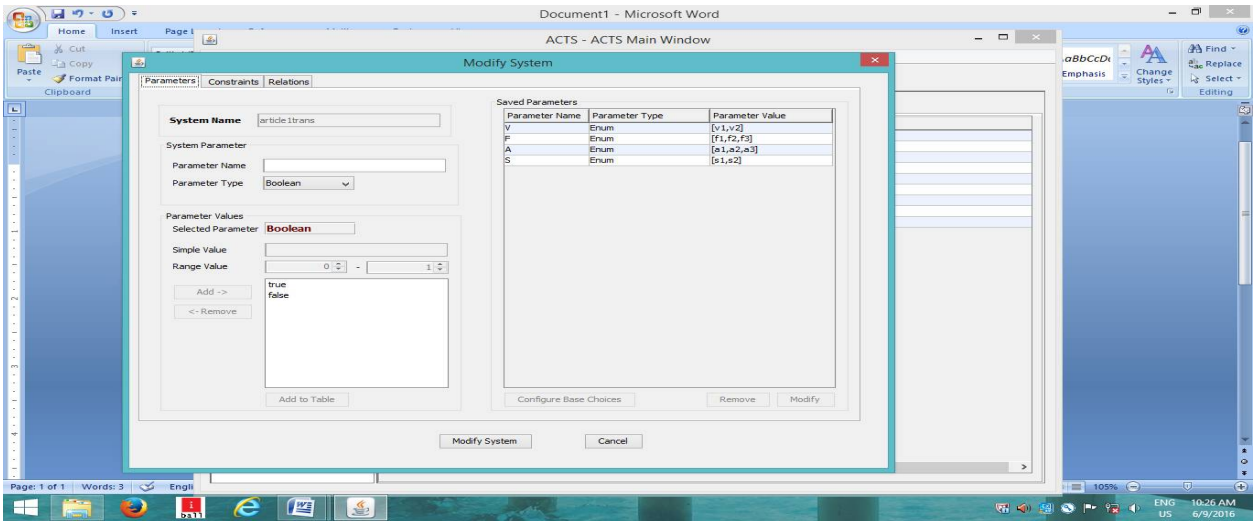

Figure 1 : Example for defining input parameter and their values

Generation of test set without constraints will cover invalid pair test cases. Invalid pair test cases combine 2wheel with AC, or 2wheel with diesel or 2wheel with gas. In Table 2, we give a formal syntax that can be used to specify the constraints for parameters in Table 1.

Constraint	Description
If(Vehicle=”2wheel”) then (Fuel=”Petrol”)	If Vehicle is 2wheeler, then fuel must be Petrol
If(Vehicle=”2wheel”&& Fuel=”Petrol”)then (Specialization=”non-ac”)	If Vehicle is 2wheeler and fuel is Petrol then Specialization must be non-ac
If (Vehicle=”4wheel”) then (Specialization=”ac”)	If Vehicle is 4wheeler, then Specialization must be ac
If(Vehicle=”2wheel”) then (Specialization=”non-ac”)	If Vehicle is 2wheeler, then Specialization must be non-ac

Table 2: Defined Constraints

Constraints setting with ACTS tool as shown in Figure 2.

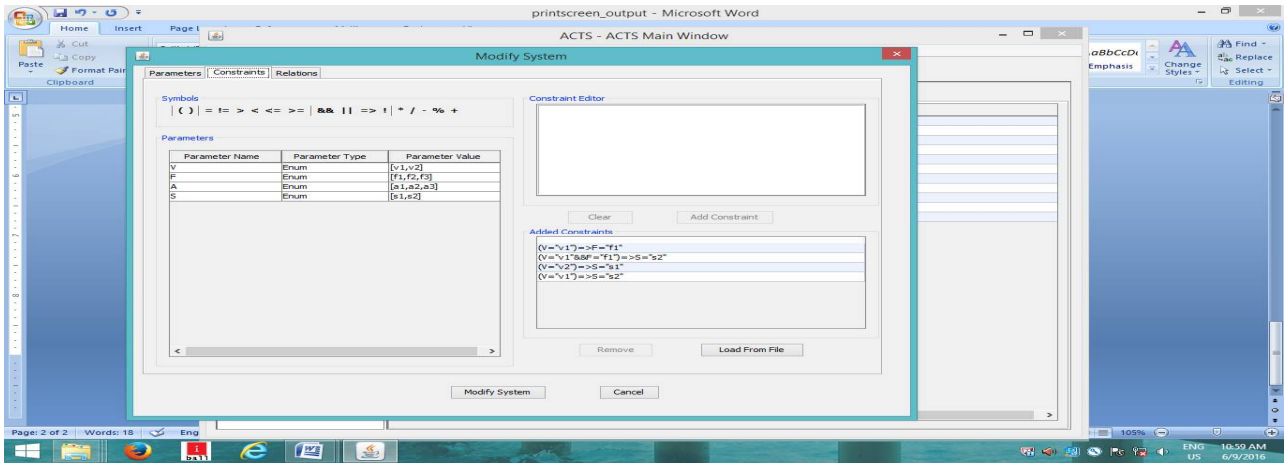

Figure 2: Sample constraint settings with ACTS tool

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

3.3 Generate Test cases

First we generate a pairwise test case for first two parameters until cover all values and then extend it to another parameter. Repeat the same process until all the parameters and values are covered from Table. In order to test all combinations required 2 x 3 x 3 x 2 = 36 test cases shown in Table 3.

Test case	V	F	A	S
1	v1	f1	a1	s1
2	v1	f1	a1	s2
3	v1	f1	a2	s1
4	v1	f1	a2	s2
5	v1	f1	a3	s1
6	v1	f1	a3	s2
7	v1	f2	a1	s1
8	v1	f2	a1	s2
9	v1	f2	a2	s1
10	v1	f2	a2	s2
11	v1	f2	a3	s1
12	v1	f2	a3	s2
13	v1	f3	a1	s1
14	v1	f3	a1	s2
15	v1	f3	a2	s1
16	v1	f3	a2	s2
17	v1	f3	a3	s1
18	v1	f3	a3	s2
19	v2	f1	a1	s1
20	v2	f1	a1	s2
21	v2	f1	a2	s1
22	v2	f1	a2	s2
23	v2	f1	a3	s1
24	v2	f1	a3	s2
25	v2	f2	a1	s1
26	v2	f2	a1	s2
27	v2	f2	a2	s1
28	v2	f2	a2	s2
29	v2	f2	a3	s1
30	v2	f2	a3	s2
31	v2	f3	a1	s1
32	v2	f3	a1	s2
33	v2	f3	a2	s1
34	v2	f3	a2	s2
35	v2	f3	a3	s1
36	v2	f3	a3	s2

Table 3: All Possible combinations

In this example, exhaustive testing requires 36 test cases, but pair-wise combinatorial testing requires only 9 test cases without constraints shown in Table 4 and 11 test cases with constraints shown in table 5. The entire test suite covers all possible pairwise combinations between transport system parameters.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Test case	V	F	A	S
1	v2	f1	a1	s2
2	v1	f1	a2	s1
3	v2	f1	a3	s1
4	v1	f2	a1	s2
5	v2	f2	a2	s1
6	v1	f2	a3	s2
7	v2	f3	a1	s1
8	v1	f3	a2	s2
9	v2	f3	a3	s2

Table 4 : Generated test cases without constraints

The highlighted test cases in Table 4 represent invalid test cases for specified constraints. In this situation, we must perform validity check with replace method of constraint handling used to determine whether all the constraints are satisfied by a test case with the specified constraints from Table 4. If invalid test cases are detected, invalid parameter values of that test case should be changed by another valid parameter value and removes the invalid test case. The resulting final test set satisfies all combinations with specified constraints shown in Table 5.

Test case	V	F	A	S
1	v2	f1	a1	s1
2	v1	f1	a2	s2
3	v2	f1	a3	s1
4	v2	f2	a1	s1
5	v2	f2	a2	s1
6	v2	f2	a3	s1
7	v2	f3	a1	s1
8	v2	f3	a2	s1
9	v2	f3	a3	s1
10	v1	f1	a1	s2
11	v1	f1	a3	s2

Table 5 : A Generated test cases with constraints

The acts tool implements the IPOG algorithm to generate combinatorial test sets. Consider the input from Table 1 which has four parameters. The acts tool can generate a combinatorial test set for this input shown in Figure 3. When we add parameter and their corresponding values and acts tool then automatically generates a t-way test set, where t is the strength of coverage. The generated test cases can be saved in multiple formats for testing purposes. Figure 4 shows sample of test case in XML format.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

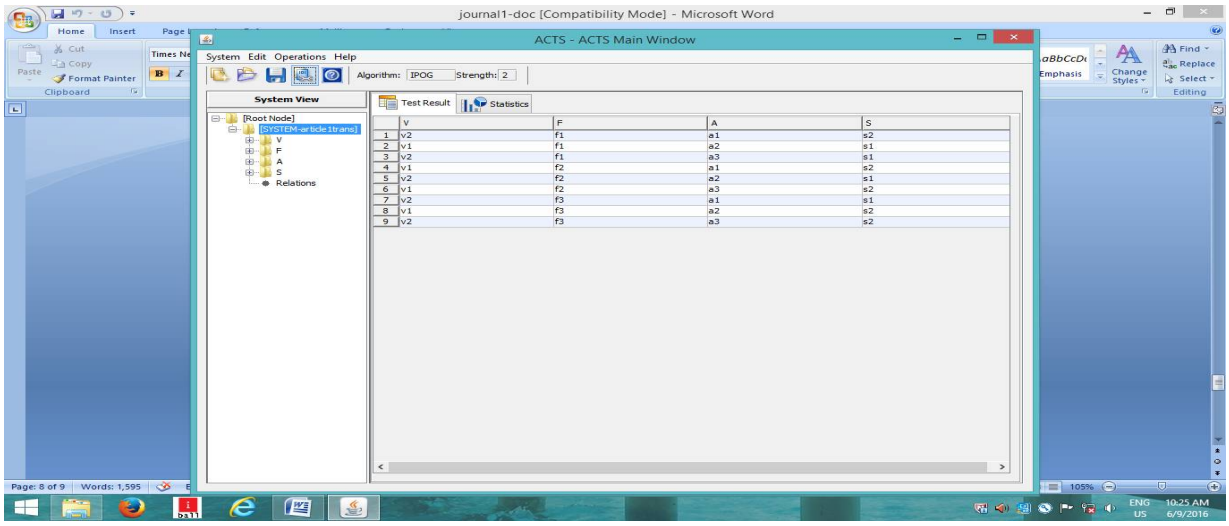

Figure 3 : Example Combinatorial test cases created with ACTS tool

As the size of the parameter and their value increase, the size of corresponding test sets increase. Managing this combinatorial growth with regard to both accuracy and execution time is still an open research issue. Greedy algorithm is recently used for t-way combinatorial testing (Bryce to appear; Lei 2008) [2]. However, the efficient generation of t-way combinatorial test suites remains an ongoing research topic.

XML format for parameters and values saved	Sample XML format for pairwise test set generation
<?xml version="1.0" encoding="UTF-8"?> -<System name="articletrans"> -<Parameters> -<Parameter name="V" type="1" id="0"> -<values> <value>v1</value> <value>v2</value> </values> <basechoices/> </Parameter> -<Parameter name="F" type="1" id="1"> -<values> <value>f1</value> <value>f2</value> <value>f3</value> </values> <basechoices/> </Parameter> -<Parameter name="A" type="1" id="2"> -<values> <value>a1</value> <value>a2</value> <value>a3</value> </values> <basechoices/> </Parameter> -<Parameter name="S" type="1" id="3"> -<values>	-<Testset doi="2"> -<Testcase TCNo="0"> <Value>1</Value> <Value>v2</Value> <Value>f1</Value> <Value>a1</Value> <Value>s2</Value> </Testcase> -<Testcase TCNo="1"> <Value>2</Value> <Value>v1</Value> <Value>f1</Value> <Value>a2</Value> <Value>s1</Value> </Testcase> -<Testcase TCNo="2"> <Value>3</Value> <Value>v2</Value> <Value>f1</Value> <Value>a3</Value> <Value>s1</Value> </Testcase> …… …… </Testset> -<Header> <Value/> <Value>V</Value>

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

<value>s1</value> <value>s2</value> </values> <basechoices/> </Parameter> </Parameters> <OutputParameters/> <Relations/> <Constraints/>	<Value>F</Value> <Value>A</Value> <Value>S</Value> </Header> -<Stat-Data> <ExecutionTime>0.0</ExecutionTime> <MaxDomainSize>3</MaxDomainSize> <MinDomainSize>2</MinDomainSize> <TotalNoOfCombination>37</TotalNoOfCombination> <TotalNoOfTests>9</TotalNoOfTests> <Algorithm>IPOG</Algorithm> </Stat-Data> </System>

Figure 4 : XML format for Parameters construction and pairwise test set generation

3.4 Execute test set

To evaluate the future approach, we consider the number of parameters and each has different parameter values and evaluates the test cases and compares the results generated with existing algorithms using existing combinatorial testing tools.

IV. CONCLUSION

The wide use of combinatorial testing will help to significantly reduce the cost of software testing while increasing software quality. It will also improve the productivity of software developers by reducing the time and effort they spend on testing. In this paper we have presented the proposed approach to generate test set for t-way testing and to handle constraints for avoiding invalid test cases. There are numerous directions to persist our work. First we execute a tool for higher level t-way strength coverage and carry out similar studies for real world applications to evaluate effectiveness of our approach.

REFERENCES

1. M. N. Borazjany, L. S. Ghandehari, Y. Lei, R. Kacker and R. Kuhn, 'An Input Space Modeling Methodology for Combinatorial Testing', IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 372-381, 2013.
2. Renée C. Bryce , Yu Lei , D. Richard Kuhn and Raghu Kacker,' Combinatorial Testing', pp.1-13, 2010.
3. Q. Feng-an and J. Jian-hui, 'An Improved Test Case Generation Method of Pair-Wise Testing', 16th Asian Test Symposium (ATS 2007), pp. 149-154, 2007.
4. David Gilbert , 'A Nimble test plan : How to Reduce the Cost of Software Testing' , pp.179 -194 , 2011.
5. Mats Grindal, Jeff Offutt and Jonas Mellin, 'Handling Constraints in the Input Space when Using Combination Strategies for Software Testing', pp. 1-39, 2006.
6. Y. Lei, R. Kacker, D. R. Kuhn, V. Okun and J. Lawrence, 'IPOG: A General Strategy for T-Way Software Testing', 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), pp. 549-556, 2007.
7. Yu Lei, Raghu Kacker, D.Richard Kuhn, Vadim Okun and James Lawrence, 'IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing', Vol 18, pp.125-148, 2008.
8. Sompong Nakornburi and Taratip Suwannasart,'Constrained Pairwise Test Case Generation Approach Based on Statistical User Profile', Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol I , pp. 16-18, 2016.
9. Rick Kuhn, Raghu Kacker, Yu Lei and Justin Hunter, 'Combinatorial Software Testing', Computer, Vol.42, Issue 8, pp. 94-96, 2009.
10. Mrs. B. Vani and Dr. R. Deepalakshmi, 'Incremental Generation and Prioritization of t-way Strategy for Web Based Application', International journal of Engineering and Technology, Vol.7, Issue 2, pp.620-630, 2015.
11. Lixin Wang and Renxia Wan, 'A New Method of Reducing Pair-wise Combinatorial Test Suite', CIS Computer and Information Science, Vol.3, Issue 1, pp. 35-41, 2010.
12. Linbin Yu , Yu Lei , Raghu N. Kacker and D. Richard Kuhn,' ACTS: A combinatorial test generation tool', IEEE 6th International Conference on Software Testing, Verification and Validation(ICSTW), pp. 370-375, 2013.
13. Linbin Yu, Yu Lei, Mehra Nourozborazjany, Raghu N. Kacker, and D. Richard Kuhn,'An Efficient Algorithm for Constraint Handling in Combinatorial Test Generation', IEEE Sixth International Conference on Software Testing, Verification and Validation(ICSTW), pp. 242-251, 2013.
14. L. Yu, F. Duan, Y. Lei, R. N. Kacker and D. R. Kuhn, 'Constraint handling in combinatorial test generation using forbidden tuples', IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW) , pp. 1-9, 2015.
15. Jian Zhang, Zhiqiang Zhang and Feifei Ma, 'Introduction to Combinatorial Testing', Automatic Generation of Combinatorial Test Data SpringerBriefs in Computer Science, pp.1-16, 2014.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

BIOGRAPHY

Dr.V.Sangeetha is working as an Assistant professor in the Department of Computer Science, Periyar University college, Pappereddipatty, Dharmapuri,Tamilnadu, India. She has completed her Msc(cs) during the year of 2001 and M.Phil(Computer Science) during the year 2004 and completed her Ph.D in the year of 2014. She has been specialized in this area of software engineering, Data mining, Compiler design. She has attended many national and international conferences and presented several papers. She has fifteen years experience in the field of computer science.

M.Bharathi is a Research Scholar in the Department of Computer Science, Periyar University, Salem,Tamilnadu, India. She received her MCA during the year of 2005 and M. Phil (Computer Science) degree in 2008 and She has passed Tamilnadu State Eligibility Test (TNSET) 2012 of Bharathiyar University,Tamilnadu. Her research interests are software engineering and software testing. She has eleven years experience in the field of computer science.