ijergs
Open Access
International Journal of Engineering Research and Generic Science (IJERGS)
Available Online at www.ijergs.in
ISSN: 2455 - 1597
Volume -3, Issue - 6, November – December - 2017, Page No. 01 - 10

An Efficient Software Testing By Test Case Reduction, Prioritization and Parallelization

MR.PRADEEP UDUPA, MTECH (CSE), MCA, MPHIL, (Ph.D)
PRIST UNIVERSITY THANJAVUR VALLAM
E-Mail: pradeepumlr@gmail.com
Dr. S. NITHYANANDAM, M.Tech., Ph.D
PRIST UNIVERSITY THANJAVUR VALLAM

Abstract

Software Testing is the process of verifying and validating system with goal of detecting and eliminating errors or it involves validating an attribute or complete program or system and determining that it generate expected results and required outputs. Software is not unlike other physical processes where inputs are received and outputs are produced. In this study APFD, prioritization technique, test case rank, test case reduction used and algorithm is developed to optimize the testing efficiency &reduce the execution time by reducing no of test cases, prioritization, fault detection, and further parallelization concept is used to increase the execution speed, decrease the effort and increase testing efficiency.

Keywords: Test Optimization, Test Case Prioritization, Testing Strategy

Chapter 1: Introduction

1.1 Testing Overview

Software Testing is the process of detecting, exploring and correcting errors. It is used ensure software quality and completeness. Here the main objective is:

To reduce overall numbers of test runs. Generally, the larger the input domain, larger the Testing would be. To avoid this problem, a minimum set of test cases needs to be created using an algorithm to select a subset that covers the entire input domain. In addition, every time inputting data to .exe file the testing would take longer to run, in case of regression testing where every modification demands repeat testing. Therefore, decreasing number of the test Cases do have benefit in efficiency.

1.2. Problem Description

A. Issue of Interest

As the size of program increases and no of instructions increases, it's very difficult to generate appropriate test cases; modal test case should identify and explore undetected errors. An important property of software testing is that the Number of the test cases that have a direct impact on the cost of testing, specifically that of Regression Testing. When we should execute test repeatedly for every Modification, it is better to less no of test cases. Notwithstanding the importance of techniques is in identifying these test cases and developing the methodology which remains one of the most difficult properties of software testing.

B. Problems Interested

1. Diminish no of possible test cases. This reduces effort, complexity, and validity
2. Prioritize test cases to increase speed
3. Calculate apfd to improve performance & quality

4. Diminish no of runs

By reducing no of possible test cases by proposed

Algorithm

5. Diminish total and average time required to execute by adopting parallelizing techniques

Steps :=>

1) Take a code segment

2) Generate flow graph

3) Compute cyclometic complexity

4) Find independent paths

5) Use algorithm to reduce test cases

6) Generate test cases for each variable

7) Analyze all independent paths

8) Prioritize test cases

9) Parallelize to reduce time

1.3 Motivation

Testing is a major phase in developing software product. After completing a software product, software developers invest more effort in testing it. It includes designing test case plan, producing test inputs for exploring program behaviors, etc. According to a 2002 NIST report, it is estimated that over $22 billion of the costs of software errors could be removed by incorporating better software testing methods. Current inefficient testing methods often still take up half or more of a software project's budget. Out of all those testing costs, generating test inputs for running a program takes a huge amount of time. Those include, but are not limited to, generating inputs for exploring every possible program behavior, valid inputs, invalid inputs, and generating performance testing data. To generate these data, software firms need to hire professionals who know to produce inputs which are very expensive.

1.4 Contributions

The main goal of this dissertation project is to investigate how we can improve the Efficiency of test case execution. It is done by

1) Diminish no of test case

2) Ordering test cases based on priority.

3) Simultaneously running and early exploitation of errors.

Given a large number of existing test cases, our proposed method reduce and rank them such that

Test cases can explore more no of faults and program behaviors in a given time, and reduce overall execution time

1.5 Organization

This dissertation is organized as follows. Chapter2 introduces some software testing concepts and techniques used in the rest of dissertation and summarize some related work. Chapter3 describes our methodology and experimental results for test case reduction. Chapter4 presents test case prioritization and selection methods and experimental results for them. I conclude this dissertation in Chapter5.

Chapter 2: Back ground Study

2.1 Check Lists or Test Case

A test case is designed to test whether system works properly or not basically it is a one step, or it is a no of steps, it is used to test the correct behavior/operations and features of an application. An expected result or expected outcome.

2.2 Validation Suite

A validation suite is a group of test cases and it has some specified set of behaviors. A validation suite often contains detailed instructions or goals for each set of test cases and information on the system configuration to be used during testing. It contains details about the test.

2.3 Testing Control Flow

Used to test every possible path,. The test can be used when the number of all Available paths is so large and testing the entire individual path becomes very difficult.

2.4 Independent Paths

It is any path resides in the program that exposes at least one new set of processing statements or new criteria. It must move along at least one Edge that has not been traversed before the path is defined.

2.5 Cyclomatic Complexity

It gives a measure of the logical complexity. This value gives the number of independent paths in the basis set and an upper bound for the number of tests to ensure that each statement is executed at least once. An independent path is any path through program that introduces at least one new set of processing statements or a new condition

Cyclomatic complexity =no of Edges-nodes+2

2.6. Why to Reduce No Of Test Cases…?

1. Large the test cases more the complexity
2. Large the test cases more probable no of errors
3. Error tracing is to be performed
4. Huge no of testers are needed
5. It will take long time

2.7. Need & Scope of the Study

1) To explore maximum no of errors.

2) To prioritize, reduce test cases, and run time.

2.8. Why to Use Proposed Technique

Existing techniques such as DDR, basis path test leads to more number of test cases and inefficient so we

1. Diminish no of test cases

Proposed technique reduces overall test cases, effort of testing, executing and validating test.

2. Diminish total no of test runs
3. Decrease time required to run test cases

In proposed research we try to reduce no of test cases by finding (how)

Min, max, and constant values in the entire test cases though finding no test paths.

• Here an efficient algorithm is written to reduce total no of test cases an analysis is made with other algorithm to prove proposed algorithm has greater efficiency and takes less no of test cases to execute and time required to execute and cost required to execute will be less.

• Here first we design a flow graph for algorithm then find all independent path in program next for each independent path we design range of test cases and no of test cases.

2.9. Proposed Algorithm (How)

In this we use following steps to decrease no of test cases

1. Detect criteria's from begin to end nodes. A criteria can be (>, >=, <, <=, ==! =)

2) Detect the variables with max and min Values in the path, if any. To reduce the Test cases, the max variable would be set at the largest Value within its range, while assigning the min variable at the least possible value of its range

3) Detect fixed values in the path, if any. When fixed Values can be found for any variable in the path,

These values are then assigned to the given variables at each node, and then use parallelization technique to run test cases parallel.

4) Using all of the above values to design a table to provide all possible test cases.

Next we reduce test case by above algorithm then prioritize test case by giving rankings using test case ranking

Chapter 3: Literature Review

In last few years there were many publications which discussed the concept of test case reduction. In this section different test case reduction methods and related works are discussed.

(A) **Constraint-based test data generation** by Richard A. DeMillo and A. Jefferson Offutt presented an approach to test data generation that uses control- flow analysis, symbolic evaluation It includes

• Constraint generation

Extract a constraint system from the program and a Testing objective will be satisfied

• Constraint solving:

Solve the constraint system to generate test data

(B) dynamic domain reduction (DDR) by a.jefferson Offutt zhenyi jin uses get split algorithm which is used to split a domain by determining the split point to obtain new domains that have two variables with constraints. The two

• Constraints on inputs

If inputs satisfy constraints, then testing objective new domains must satisfy the constraints, and the size of the two new domains should be equal. There are two cases for two variables that have been modified for new domains. These cases are defined depending on the relationships between the two variables' domains

1) Non-intersecting sets of values defined by two domains, in which case the constraint is either satisfied, or is infeasible.

2) Intersecting sets of values defined by two domains, in which case the constraint may or may not be satisfied.

(C). Ping-Pong Technique

This technique selects less number of test cases by reordering test cases, based on heuristic technique which doesn't promise best solution but give good solution in appropriate time, by comparing the set of values of goal state and set of

states of achieved values. Here minimal test case reduces the cost of testing also it runs test in different order With difference between the actual and succeeded ordering Are from end to beginning Different Procedures can be followed

A. Forward Procedure: suppose test case<t1,t2.....tN> than

Executing the test cases from starting t1 to Tn

B. Reverse Procedure: If test cases of size N is there than

Executing test cases in reverse order from tN to t1

C. Inside Out Procedure: Run test cases from middle to both

The ends

(D) Test case reduction using parallelization

There are several works in the literature on reducing the number of test cases but in this paper we propose test case reduction technique *and parallelization where min, max, constant variable in all path are found and later more than one test case are made to run in parallel fashion which has greatest percentage of reduction in terms of no test cases and execution time required for running parallelization, debugging.*

Table 1. A Comparison between Test Case Reduction Techniques

Chapter 4: Existing Methodology

DDR Technique

A step follows assume given domain is i(0..30),j(0..50),k(0..40)

1. Detecting all criteria's from start to finish nodes. From criteria ma1<ma2, ma1>=ma3,ma3=10

2. Calculate split value and splitting Intervals for constraint ma1<ma2

. We choose the split Value=15

Using mentioned values. Input domain divided into two intervals. Ma1=0 to 15 and 16 to 30 ma2 into 10 to 30 and 31 to 50

&final interval by using splitting is ma1 0 to 10 and 11 to 30 ma2

31 to 50 ma3 is 10

So total test cases=31*1+31*20=651

Chapter 5: Proposed Methodology

We first reduce test case by our given algorithm then prioritize test case by giving rankings for test cases then find APFD which will prove that ours technique over performed

Then existing techniques, then parallelize our test cases to reduce time and cost involved. Here first we find no test paths then from each path we find min, max, and constant values and derive our reduced test cases by using

S.NO	TEST REDUCTION TECHNIQUE NAME	ADVANTAGE	DISADVANTAGE
1	Constraint based testing	it uses control- flow analysis, symbolic evaluation and reduces no of test cases based on criteria	1.More no of test cases 2. Time consuming 3. More expensive 4. Less efficient 5. More effort 6. No parallelization
2	D.D.R	Achieved a more reduction percentage of the test cases	1. less test cases compare to constraint bases 2. comparatively Less Time consuming 3.less expensive 4.less effort 5.no parallelization
3	PING PONG TECHNIQUE	Assure domain Coverage and cost effective	Time consuming, expensive technique more memory ,time and efficiency is required in executing the test cases
4	Prioritize techniques no order, reverse order	Attempt to detect possible no of errors and contribute in performance	Less efficient compare to proposed technique in terms of performance

Steps given below then further execute them parallel.

Assume that the path 1-2-4-8 is selected and the initial domains of the input variables are i(0..30),j(10..50),k(0..40)

The algorithm steps follow:

1) Identify criteria's from begin to end nodes. i < j, j> = k

2) Identify min values in the path. From the above criteria's, it is possible to Detect 'i' as the variable with the min value and 'j' as the variable with max value. In order to find a value of zero, the least value within the domain of variable 'j', can then be allotted to 'i' while the value of 'j' can be set at 50, the highest value of the variable.

3) Determine constant values in the path. 'k' fixed value obtained on node2 of the path has been used to replace the decide value of k (10) at the node.

4) Using all of the above-stated values to design a table to provide all test cases. 'i' value is 0...30, 'j' as the variable

With max value = 50, 'i' as the variable with the min value. 'j' will have the value of 31...50 because of 'j' should be greater than 'i' as per criteria in the path. Similarly, according to above method we can calculate test case for path2, path3 and path4

- Path2: 1-2-5-8 i<j, j<k, k=10(constant) Final Test case Range i=0..9,j=10..50,k=10
- Path3: 1-3-6-8 i>=j, j<k, k=20(constant)
- Final Test case Range i=10...30, j=0...30, k=20
- Path4: 1-3-7-8 i>=j,j>=k,k=20
- So i=30,j=0..50,k=20

TEST TABLE			
VARIABLE I	VARIABLE J	VARIABLE K	TEST CASES /PATH
0 to 30	31 to 50	10	T1 /P1
0 to 9	10 to 50	10	T2/P2
10 to 30	0 to 30	20	T3/P3
30	0 to 50	20	T4/P4

Chapter 6: Result Evaluation

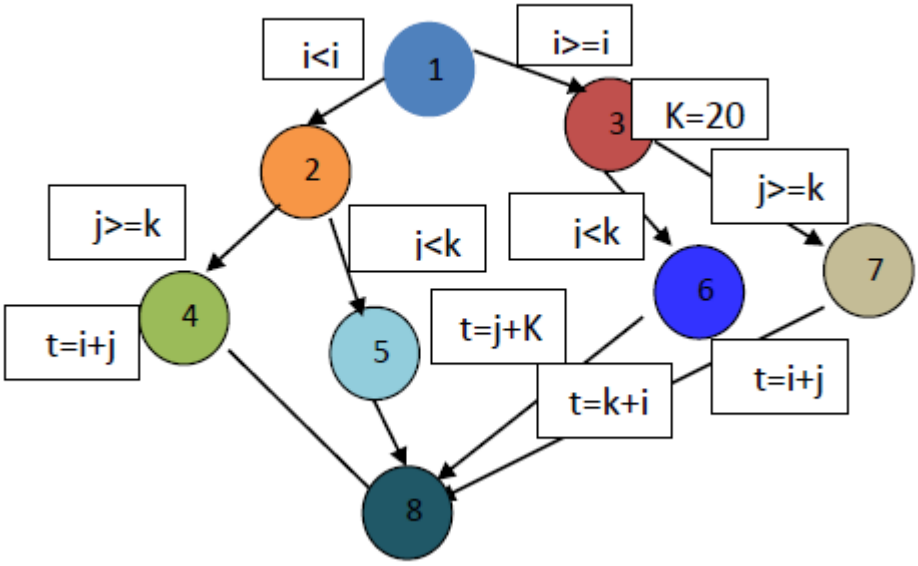
A comparative evaluation has been made between the Proposed Techniques, the Existing Technique (Get Split Algorithm technique). The following areas are used to compare with existing techniques:

1) Overall no of test cases

2) Percentage of reduction in test cases

3) Overall Compilation time

Control flow graph



To every fault a severity value has been allocated based on a 10 point scale is expressed as follows. Very High Severe: SV of 10 High Severe: SV of 8 Medium Severe: SV of 6 Less Severe: SV of 4 Least Severe: SV of 2.

RFT=N j/ TIMEj*10

PFD=NJ/total no of faults*10

RDA=NJ*SJ/TJ

Test cases/faults	T1	T2	T3	T4	norder	revord	prop
F1				*	4	1	1
F2				*	4	1	1
F3	*		*	*	1	4	1
F4	*	*	*	*	1	4	1
F5			*		3	2	3
F6		*			2	1	4
No of faults	2	2	3	4			
time	3	5	7	9			
severity	4	6	8	10	15	13	11

Test cases	TCR=RFD+PFD+RDA
T1	8.5
T2	10.6
T3	10.33
T4	10.5

Test cases	RFT	PFD	RDA	TCR
T1	6.66	1.81	2.66	11.13
T2	4.0	1.81	2.4	8.21
T3	4.28	2.72	3.42	10.42
T4	4.44	3.63	4.44	12.51

NO ORDER	REVERCE ORDER	PROSED ORDER
T1	T4	T4
T2	T3	T1
T3	T2	T3
T4	T1	T2

APFD=1-(TF1+TF2+…TFM)/M*N+1/2*N

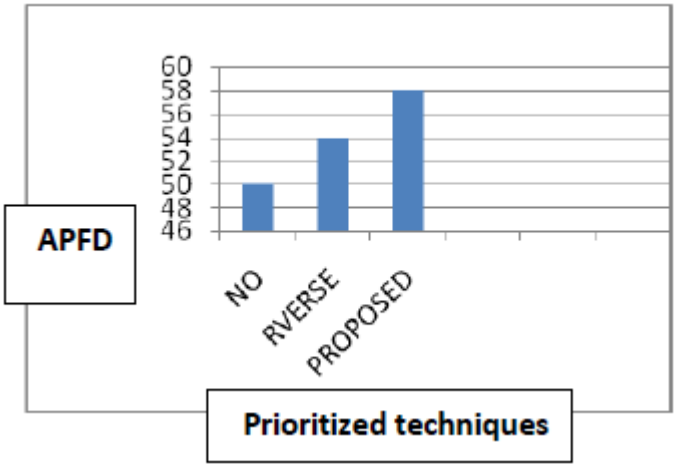
For no order APFD=1-(4+4+1+1+3+2)/6*4+1/2*4=1-.625+.125=50%

For reverse order APFD=1-(1+1+4+4+2+1)/ 6*4+1/2*4=1-

.3125+.125=54%

For proposed order APFD=1-(1+1+1+1+3+4)/ 6*4+1/2*4=1-.4583+.125=58%

PRIORITIZATION TECHNIQUES	APFD%
NO ORDER	50
REVERCE ORDER	54
PROPOSED ORDER	58



NEXT we parallelize test cases by referring Test table given above

Now variable I used in 3 paths so range =total no of interval/3

Now variable j used in 4 paths so range =total no of interval/4

Since c is constant we not divide c

So range of I spitted into 3 parts i1) 0…..10 i2)11…..20 i3)21….30 similarly range of j spitted into 4 parts j1) 10…..20 j2)21…..30 j3)31….40 j4)41…50

Now when we execute them parallel fashion we have

Total no test cases=[31*51*41]*4=259284 Reduced test cases=[31*20*1+10*41*1+21*31*1+1*51*1]=[620+410+651+51]=1732 Assume each test case take .5 second then Without parallelization execution time is 1732*.5=866

So total no of test cases 259284, execution time129642.

Reduced test case for sequential execution no of test case1732, execution time 866. But with PRIORITIZED parallel execution test cases=1732 and execution time=433 and fault detection rate is more, time required to detect THE FAULTS WILL BE considerably decreases because test cases are exposed in prioritized order.

Chapter 7: Conclusion

Each algorithm has its own significance as well as drawbacks ddr works on specific domain & split points,

Ping pong and other existing techniques results in more no of test cases, compilation, time, effort, cost but proposed technique over performed by reducing no of test cases, prioritizing, revealing more faults by assigng test case rankings,

apfd calculation and finally prioritized reduced test cases by giving test case rankings to achieve optimized performance and parallelized and prioritized test cases to reduce overall running time and cost.

Chapter 8: Limitation

This executes serially and each and every path is to be examined so it will take more time and memory to store the result, it is effective when the variables are there with constant and fixed values it works well for parallel execution where we can save memory and increase speed of execution.

References

[1]. R. Wang, B. Qu, Y. Lu, "Empirical study of the effects of different profiles on regression test case reduction", IET Softw., Vol. 9, No. 2, 2015, pp.29–38.

[2]. B. Boehm and L. Huang, "Value -Based Software Engineering:A Case Study" ,IEEE Computer ,Vol.36,2003, pp.33-41

[3]. Arnicane, V., "Complexity of Equivalence Class And Boundary Value Testing Methods", International Journal of Computer Science and Information Technology, Vol. 751, 2009, pp.80-101.

[4]. Abhijit, A., Sawant1, P. H. Bari2 & P. M. Chawan3," Software Testing Techniques and Strategies",IJER Vol. 2, No 3, 2012, pp. 980-986.

[5]. Jeng B., Weyuker E. J., "A Simplified Domain -Testing Strategy". ACM Trans. Softw. Eng. Methodol. Vol. 3, No. 3, 1994, pp.254–270

[6].T.Gyim´othy, A. Besz´edes, and I. Forg´acs, An efficient relevant slicing´ method for debugging. SIGSOFT Softw. Eng. Notes, Vol. 24,No. 6, 1999, pp. 303–321.

[7].S. Biswas and R. Mall, Regression test selection techniques: A survey, Informatica 35. 2011, pp. 289–321.

Author's Bibliography

Mr.Pradeep Udupa completed his MTECH (CSE), MCA MPHIL & PERSUING PHD(CSE) his field of interest includes software engineering, testing, operating system,simulation,dbms,wireless communication,adbms,oops Currently working as assistant professor in mes eng kerala.