International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Survey on Test Case Reduction Techniques

Vaibhav Chaurasia[1], Yogita Chauhan[2], Thirunavukkarasu K.[3]

[1, 2] M. Tech.-Student, School of Computing Science and Engineering, Galgotias University, Uttar Pradesh, India

[3]Assistant Professor, School of Computing Science and Engineering, Galgotias University, Uttar Pradesh, India

Abstract: ***Test case reduction is traditional in nature. It is very old field on which engineers work from the beginning of the third generation. Reduction is not an easy task, several techniques in this field is developed. There are several challenges faced by the engineers, for instance, cost and time. The reduction is the nightmare of the developer, as, the level of testing is increases, techniques of reduction also becomes more complex. An old technique teaches us the basic of the reduction of test cases without any complexity and through simple algorithms. So, this paper gives the overview of the existing techniques and gives the literature review of test case reduction techniques in software testing field.***

Keywords: Software Testing, Test Case Reduction, Exception Handling, Dynamic Domain Reduction, Independent Path, Basis Path Testing

1. Introduction

Testing is the name or an activity which is compulsory in the field of software engineering without which the life cycle of the software is not been completed. Many authors give their own definition of software testing but in the simple words, testing is an activity which validate the behaviour of the program that whether it is working properly or not.

Software testing provides many challenges to the developers and tester as well. Testing has a different technique like a white-box, black-box, multi condition, LCSAJ [3], condition coverage [3], integration testing, and coverage based testing [7] and list goes on. Every different testing has its own unique goals but the overall goal of the testing is to reduce the cost and time of the software development, since it is second important stage after the development.

Studies conclude us that software testing takes fifty-percent or more of cost development [1] [2]. Software Testing is the field which always attract the attention of the users, developers, managers as it takes nearly same cost as of development. To minimize the test cases, hundreds of techniques will be discovered, but still research continues in this field. Testing is a constructive part not a destructive one, as, it is considered as an art of finding errors whose aim to evaluate the attribute and capabilities of software [1]. Beizer also states: More than the act of testing, the act of designing tests is one of the best bug preventers known ([2], pg. 3). We should always focus on reduction of the test cases but more important part that we should know where and when to stop testing.

When and where part of testing is interesting, knowledge plays vital role. Here knowledge refers to the personal experience and context. Level of knowledge of everyone is different which affects the overall testing. In general, knowledge can be used as information to guide and to recognize failure in the software [4].

In this paper we are discussing about the old techniques used in the basis path testing to reduce the test cases. Dynamic Domain Reduction [5][8], Basis path testing with exception handling [6], Test Case Reduction [9][10]. This paper gives a brief introduction and algorithms proposed by the authors. In last, a comparison has been done and shows how the following test case reduces.

All over testing is not easy to handle. So, test case have taken because, "Test case is a set of conditions or variables under which a tester will determine whether an application, software or one of its features is working as it was originally established for it to do" [16]. On base on test case we decide whether, requirement fail or pass and all test cases is collect in the Test Suite. Test case is the basic part or atomic part or smallest part of the software testing. Every aspect of testing based on the test cases and selection of test cases. Even small change in test case lead to large change in test suite and at last, requirement is affected.

In this paper, we are discussing about the Survey on Test case Reduction and compare the techniques in paper [5][6][9]. All papers are about, reduction of the test cases by using different methods, but, Control Flow Graph in all the methods. In section 2, we discuss the problem statement. In section 3, we discuss the algorithm used in different techniques. Then, in section 4, we evaluate all the techniques and compare them by different parameters (for instance cost, time) and then conclude all the techniques,. in section 5, in respect of their advantages and disadvantages. At last, in section 6, we discuss about the future aspect of test case reduction.

2. Test Case Reduction Problem Statement

In general, developers and testers thought, testing is only to find a defect from the given code. It is true but not completely, incomplete information tends to increase the number of test cases instead to reduce it. In this paper, test case reduction basic techniques can be explained. The techniques which are discussed in this paper based on three problems [9][10]

- **Reduce number of test cases** - The reduction technique reduces the cost of executing and validating tests. Therefore, it is of great practical advantage to reduce the number of test cases.

- **Generate the test cases automatically** - One of the most important components in a testing environment is an automatic test data generation.
- **Minimum test case runs**– Use less time is spent on test runs.

These problem statements directly affect the cost and time of the testing, so, it is better to be dividing the code into the small parts. The steps are as follows:

2.1 Control Flow Testing

It refers to the order in which each statement is executed or evaluated [11]. Its aim to check validity of control flow without executing or testing every path otherwise it is impractical to check every path [9].

2.2 Independent program paths

It is the path introduces at least new condition for a statement. In terms of flow graph, it moves along one path at least before it is defined [9].

2.3 Cyclomatic Complexity

It gives the quantitative measure of the logical complexity and measure complexity. It 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 executes at least once [2].

- Number of regions in flow graph
- Edges-nodes + 2
- Predicate node + 1

3. Test Case Reduction Techniques

The techniques are discussed in this section are fundamental and effective:

3.1 Dynamic Domain Reduction

It is an automatic test generation method uses constraints derived from test program controls the execution path in CFG to reduce domains until test data satisfies constraints found for test program [5].In this, split algorithm is the main, it divides the variables. The algorithm is as follows [5]:

1. Compute current search point
2. I is a value from a set or search pt = (½, ¼, ¾…..)
3. Try to equally split leftexpr's and rightexpr's domain

```
IF ( ldomain.Bot >= rdomain.Bot AND ldomain.Top <= rdomain.Top )
    Split = ( ldomain.Top – ldomain.Bot )*search pt + ldomain.Bot

ELSE IF ( ldomain.Bot <= rdomain.Bot AND ldomain.Top >= rdomain.Top )
    Split = ( ldomain.Top – ldomain.Bot )*search pt + rdomain.Bot

ELSE IF ( ldomain.Bot >= rdomain.Bot AND ldomain.Top >= rdomain.Top )
    Split = ( ldomain.Top – rdomain.Bot )*search pt + rdomain.Bot
ELSE
    Split = ( rdomain.Top – ldomain.Bot )*search pt + ldomain.Bot
END IF
RETURN split
END GetSplit
```

3.2 Test Case Reduction using Common Test Generator

It is the technique used with cyclomatic complexity, works to find the common test cases at the time of generation and it is a technique working on parallel execution and it improves the efficiency of the test cases.

The following steps to generate the test cases [9]:

1. Write the source code
2. Using code, draw the respective flow graph
3. Determine the cyclomatic complexity of the flow graph
4. Prepare test cases from the following flow graph
5. Find all possible constraints from start to end nodes of CFG
6. Identify the variable with maximum and minimum values in CFG, if any
7. Finding constants values in CFG, if any
8. Using the above data, draw a Table of all possible test cases

3.3 Basis Path for Programs with Exception Handling Constructs

Software testing is mainly divide into black-box and white-box. But it doesn't matter with type of testing or technique used in testing, it is incomplete without exceptions. In every program code, exception occurs and it takes 10% of total function [6]. Sometimes program behaves anonymously at that time exceptions are appeared in the program and protect program from exceptions and garbage values. The following steps to generate the test cases for Exception Control Flow Graph (ECFG) [6]:

1. Write the source code
2. Construct an uncompleted CFG in which there is not outgoing edge matching to the statement of throw
3. Determine the exception type of the statements that may thrown
4. Add the outgoing edges of the statement of throw and necessary exception nodes.

4. Evaluation

Now, we have to evaluate all the algorithms discussed till now, with the help of the example. We consider it following parameters for evaluation:

1. Cost
2. Time
3. Path Coverage

The following steps for evaluation of algorithm:

4.1 Source Code

```
int value (m1, m2, m3)
{ int tot;
 char e1;
 float e2;
```

```
tot = 0;
if(m1 < m2)
try{ m3 = m3 + 7;
If(m1 < m3)
tot = m1 + 11;
else
tot = m1+ 5;
else
{ m3 = m3 + 11;
tot = m1 + m2 + m3;
}
Catch(char e1)
{ print("Garbage Value");
}
Catch(float e2)
{ print("Problem occurred");
}
return(tot);
}
```

4.2 Control Flow Graph

For above source code, respective control flow graph given in Fig. 1

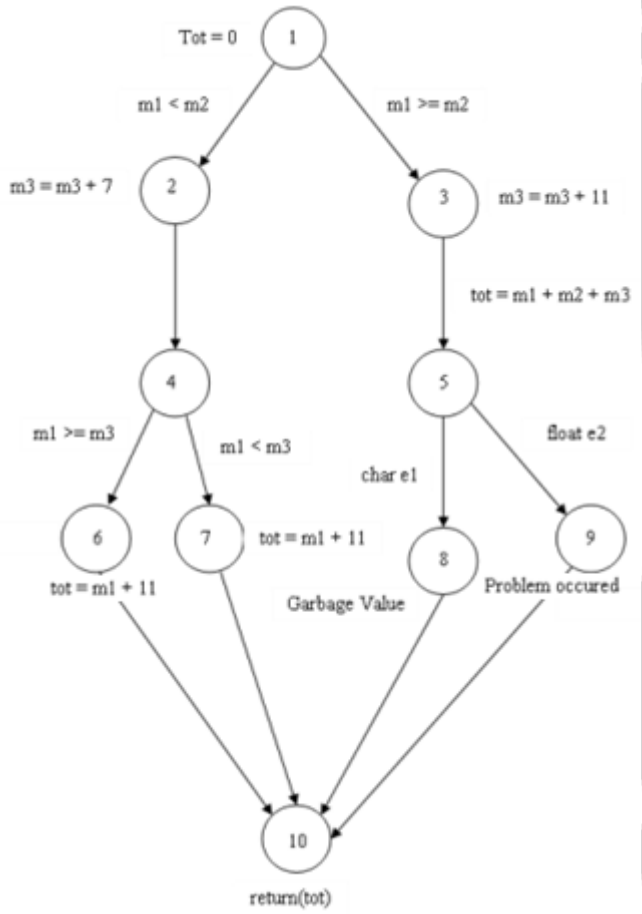

Figure 1: Control Flow Graph of Source Code

4.3 Comparison of the techniques

Control flow graph is been made from the source code. Now, with the help of Tables we must understand the detail of every technique. The Table 1 explains the steps in DDR algorithm [5]:

Table 1: Dynamic Domain Reduction

S. No.	Dynamic Domain Reduction	
	Steps	Result
1	Cyclomatic Complexity	Number of Region + 1 = 4
2	Independent Paths	Path 1: 1,2,4,6,10 Path 2: 1,2,4,7,8 Path 3: 1,3,5,10 Path 4: 1,3,5,10
3	Initials or Range taken	<0 to 30>, <10 to 50>, <0 to 40>
4	Constraints	m1 < m2, m1>= m3, m3 = 10
5	Split Values (for m1 < m2)	8, 10, 11, 13, 15
6	Total test cases calculated	51 (after applying split algorithm [5])

This shows that DDR algorithm reduce the overall test cases. Now, the next Table shows the result of the Test Case Reduction Technique without exception. The Table 2 explains the step in Common Test Case Generation [10]

Table 2: Common Test Case Generation

S. No.	*Common Test Case Generation*	
	Steps	*Result*
1	Cyclomatic Complexity	Number of Region + 1 = 4
2	Independent Paths	Path 1: 1,2,4,6,10 Path 2: 1,2,4,7,8 Path 3: 1,3,5,10 Path 4: 1,3,5,10
3	Initials or Range taken	<0 to 30>, <10 to 50>, <0 to 40>
4	Constraints	m1 < m2, m1>= m3, m3 = 10
5	All test Cases	<10..30> , <50>, <10>
6	Total test cases calculated	21 (after calculating the range of the test cases [10])

Now, as we compare to the DDR algorithm much more test case is reduced as we are taking common test cases which is occurred in the control flow graph.
In the next algorithm, Test case reduction with Exception Handling can be compared. The Table 3 explains the same [6]:

Table 3: Test Case Generation with Exception Handling Constructs

S. No.	*Common Test Case Generation*	
	Steps	*Result*
1	Cyclomatic Complexity	Number of Region + 1 = 4
2	Independent Path (CFG)	Path 1: 1,2,4,6,10 Path 2: 1,2,4,7,8 Path 3: 1,3,5,10 Path 4: 1,3,5,10
3	Independent Paths (ECFG)	Path 1: 1,2,4,6,10 Path 2: 1,2,4,7,8 Path 3: 1,3,5,8,10 Path 4: 1,3,5,9,10

Result of all the algorithms is been discussed in the Tables I, II, III. All the algorithms have its own significant values, advantages and disadvantages. Overview of the entire algorithm i.e. its features, overall results, saving time effort, cost effort is discussed in the Table 4. These algorithms worked on the basic methodology i.e. Basis Path Testing. It is the most general and first method worked on the low level of the code, conditions, exceptions, loops etc. An automation tool has been introduced for the execution of the algorithms. But, manual testing is always prefers first by the companies due the personal experience of the testers. Basis path testing

is one technique implemented in the company that is done through automated as well as manual. In testing there are limited amount of resources and the biggest challenge is to choose the correct path which is useful in analysis. All the challenge is resolved by these algorithms.
An overall result has been discussed in Table 4:

Table 4: Overall Results

Parameters	*Results*		
	Common Test Case Generator	*Dynamic Domain Reduction*	*Test Case Generation with Exception Handling Constructs*
All possible test Cases(From traditional algorithm)	52111	52111	52115
Reduced Test Cases	651	21	25
Saving (in percent)	99.95	98.75	99.95
Time of Compilation	5.25	162.75	6.25

Saving (%) = 100 – ((100 * Reduced Test Case)/ All Possible Test Case)

Time of Compilation = Assumption of time taken by each test case * Reduced Test Cases

There are some of the results shown in Table IV. From these results we get a clear idea that traditional testing creates many test cases which is not useful to us. To reduce all un useful test cases we apply algorithms to reduce the complexity of time and space.

4.4 Graph Analysis

The data in the table4 above shows the result numerically but numbers are not sufficient for us. Result should be both numerically as well as graphically. Generally, graph shows us the axis on which various analysesare been shown and easy to understand, even for the naïve user. Result graphically shown in Fig. 2

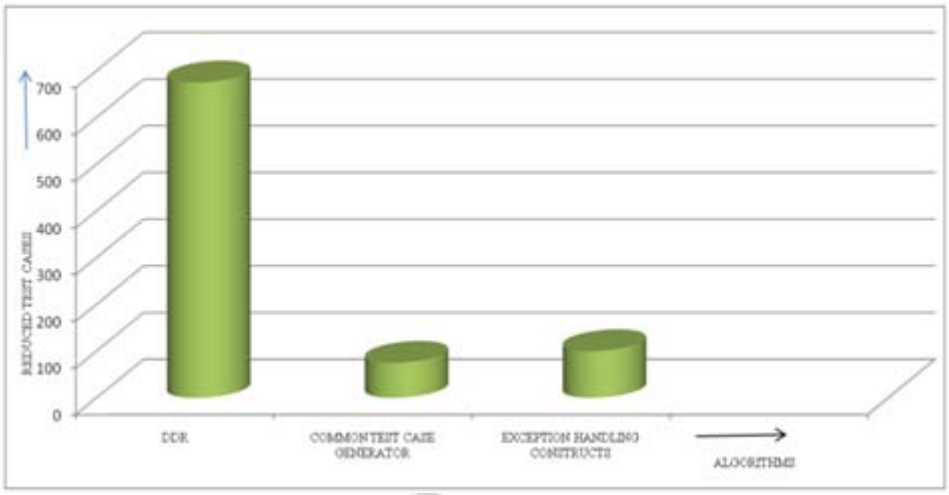

Figure 2: A bar graph shows relationship of algorithms with test case reduction

It shows the graph between the algorithms and test case reduction. The longest bar is of DDR algorithm and the shortest bar is of Common Test Generator algorithm. It shows that how nicely graph is been plot for the test case reduction. In this x-axis shows the various algorithms and y-axis shows the test case reduction. Plotting is been done on the basis of calculation done in Table IV (overall results).

5. Conclusion

So far different algorithms are discussed. Every algorithm has its own importance. Dynamic Domain Reduction works on the particular domain and its split points. Better the split point better the result came. It has some disadvantages, for instance, it is more expensive technique due to its dynamic nature but provides more information [5]. This technique only execute some paths and not every path but done symbolically. It handles loops and arrays simply. Takes large space and time to execute and applied only to numeric data software. It creates problem during aliasing, this is not resolved completely. The worst case in running time complexity of Dynamic Domain Reduction is**$\theta(P * K^D)$,** where K is split attempt at any decision, p is number of paths and k is constant [5]. Each time the value assigned, input space reduced to one dimension. Main problem of DDR is they cannot handle array completely, as, it considers array as a one variable.

There are many problems in the Dynamic Domain Reduction technique which is resolved by the technique called common test case generator. It seems to be, it takes the test cases common in the software program, it reduces the test cases much more efficiently and covers the less space and takes less time. This technique equally worked on the parallel execution of the software program instead of serial execution, which saves the resources of the hardware and software efficiently. Only limitation with this technique, it worked only with common variables and where more than two variables are there. It also worked with the fix values either it be constant or variables. But it reduces the time of compilation significantly. Serial execution takes more time to execute each test path and consumes more space for store the result. This technique work parallel execution, more advantageous for everyone.

In general, large program has an exception in it. The exceptions seems not affect program seriously but affect some or the other way. If there, exception exist in the program, program twice analyzed but with the algorithm of exception handling constructs, program is analyzed in one go. Exception leads some operation not worked properly and

program terminates unambiguously. Normally, difference between normal testing and exception testing is that program test twice but it takes more cost and time in execution. This makes algorithm simple and make test only once by construction exception control flow graph. It reduces the robustness of software by increasing cost of the program.

6. Future Work

We discussed, about the algorithms and its features, advantages, disadvantages, cost, time, test cases compare with traditional algorithm to all of these algorithms. Great work is done in the traditional algorithms, and in the three algorithms great work is been done. Time, cost and test cases are reduced significantly. Now, in the future perspective many type of testing are there, for instance, regression testing, acceptance testing, GUI testing and many more testing are there. Now, our scenario is changing and so the software development industry also changes. Basis path testing is the basic methodology for reducing test case but in current scenario, GUI based interface is been there and based on many languages which is being programmed on different platforms. Now, Web is been totally different from Standalone application and each has different types of testing performed. New type of testing come into scenario, for instance, unit testing, integration testing, alpha testing, beta testing, gorilla testing and many more techniques.

Instead, all these techniques continuous researches in this field are going on like testing is done with the fusion of mining and knowledge Engineering. For instance, Clustering Approach to Improving Test Case Prioritization [13] and using knowledge engineering test case is improving System Test Case Prioritization of New and Regression Test Cases. Testing is not all about the reduction of test cases but minimization, prioritization, selection, adequacy and also enhancement comes under the testing equally. These are some advanced techniques on which continuous work is going on.

Test case quality, equally affect the factor in software testing, Multi- Dimensional Measures for Test Case Quality and Research on New Techniques and Development Trend of Software Testing [14]. Testing now all done automatically, manual is the talk of past due to automated tools is been come [15].

Development in this field is very vast and it never ends, as, software industry grows simultaneously testing industry also increases. It is also possible that for instance, in history hardware and software industry splits and works independently, same testing and development industry also splits and works independently.

Now the tool has come for testing online also, no need to install tools in your desktop. These changes fulfil the needs of the naïve user, managers. This reduces the cost of the company and maximum features available within the small space. These tools are also compatible with the browsers, for standalone applications tools also available.

At last we conclude that there is a vast scope of testing in the future, as, testing industry grows very fast and steadily. In coming years both development and testing preferred or treated equally. Now days, advance techniques of software testing can leading the market and more than hundred type of testing are there in the market. Our techniques, is very basic one but, it has been emphasize on the coding part and not the interface or environment. These techniques can be a part of the advance level techniques by making them hybrid with other techniques. For instance, use Branch coverage with Regression testing. These, existing techniques can be hybrid with other techniques also. It is a future aspect.

References

[1] Antonia Bertolino, “Software Testing Research: Achievements, Challenges, Dreams,” IEEE, 2007, Future of Software Engineering, Minneapolis.
[2] Boris Beizer, “Software Testing Techniques,” Dreamtech Press, 2003, 81-772-2260-0, 2nd ed.
[3] Aditya P. Mathur, “Foundation of Software Testing,” Pearson Education India, 2008, 81-317-0795-4, 4thed.
[4] Juha Itkonen, Nika V. Mantyla and Casper Lassenius, “The Role of the Tester’s Knowledge in Exploratory Software Testing,” IEEE Computer Society, vol. 39, no. 5, 2013, pp 707-724.
[5] Jefferson Offutt, Zhenyi Jin and Jie Pan, “The Dynamic Domain Reduction Procedure for Test Data Generation,” Software Practice and Experience, vol 29, no 2, 1999, pp. 167-193.
[6] Quingtan Wang, Shujuan Jiang and Yanmein Zhang, “,” In IACSIT Press, 2012, International Conference on Computer Science and Information Technology (ICCSIT), Singapore.
[7] Muhammad Shahid, Suhaimi Ibrahim and Mohd Naz’ri Mahrin, “A Study on Test Coverage in Software Testing,” In IACSIT, 2011, International Conference on Telecommunication Technology and Applications (CSIT), Singapore.
[8] Jefferson Offutt, Zhenyi Jin and Jie Pan, “The Dynamic Domain Reduction Procedure for Test Data Generation: Design and Algorithms*,” ISSE Tech. Rep. ISSE-TR-94-110, George Mason University, Washington D.C., USA, 1994.
[9] Dr. R.P. Mahapatra, M. Mohan and A. Kulothungan, “Effective Tool for Test Case Execution Time Reduction,” In IACSIT, 2011, International Symposium on Computing, Communication and Control (CSIT), Singapore.
[10] R.P. Mahapatra and Jitendra Singh, “Improving the Effectiveness of Software Testing through Test Case Reduction,” In World Academy of Science, Engineering and Technology, 2008.
[11] en.wikipedia.org/wiki/Control_flow, (Last Accessed Date: May. 12, 2014).
[12] Ryan Carlson, Hyunsook Do and Anne Denton, “A Clustering Approach to Improving Test Case Prioritzation: An Industrial Case Study,” 2011, Software Maintenance(ICSM), 2011, IEEE International Conference, Williamsburg
[13] Hema Srikanth, Laurie Williams and Jason Osborne, “System Test Case Prioritization of New and Regression Test Cases,” report NC 27695, North Carolina State University, Raleigh.
[14] Zhang Hongchun, “Research on New Techniques and Development Trend of Software Testing,” Atlantis

Press, 2013, International Conference on Computer Science and Electronics Engineering (ICCSEE), Paris.
[15] Zhi Quan Zhou, ShuJia Zhang, Markus Hagenbuchner, T.H. Tse, Fei-Ching Kuo and T.Y. Chen, "Automated functional testing of online search services," report TR-2010-06, University of Wollongong, University of Hong kong, Swinburne University of Technology, HKU, 2010.
[16] en.wikipedia.org/wiki/Test_Case, (Last Accessed Date: May. 8, 2014).

Author Profile

Vaibhav Chaurasia pursuing M.Tech. in Software Engineering from Galgotias University. He is completed B.Tech. in Computer Science from NIMS University in 2012 and pursuing M.Tech. in Software Engineering from Galgotias University currently in 2014 respectively. Area of interest is Software Testing.

Yogita Chauhan is pursuing M.Tech. in Software Engineering from Galgotias University Greater Noida, Delhi-NCR. She is completed B.Tech. in Information Technology from Gold Field Institute of Technology & Management, Faridabad affiliated from Maharshi Dayanand University, Rohtak.Area of interest are Software Testing, Data Mining.

Thirunavukkarasu K., is an Assistant Professor at Galgotias University, Greater Noida, Delhi-NCR. He is pursuing PhD in CSE in the research area of Spatial Database. He has been a student of Madras University, Bharathiar University, and Anna University, Chennai, India. He has more than 14 years of experience in Teaching and 3 years in software industry. He has taught for APIIT, (affiliated to Staffordshire University, UK) at Panipat, Vijaya College, Surana College and KKECS College, Bangalore University, Bangalore and worked as Software Engineer for I2 Technology, UK at Bangalore. He has involved in various academic activities like BoE member and Assistant Custodian for PG-Unit, Bangalore University, Bangalore. He has wide research interests that include Knowledge Engineering, Data Mining, and Databases Technology. He is a Member of IEEE, CSI and Life Member of ISTE. He has 9 certificates from IMS, IBM and trained 150 students on IBM DB2 certificates. He was the organizing Secretary for an International Conference ICACCT 2010. He has conducted various workshops, short term and summer courses. He has published 9 papers in international and 3 papers in national level.