ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

Orthogonal Array Approach for Test Case Optimization

Shubhra Banerji

Project Manager, Software Testing Research & Development Unit, IBM India Pvt. Ltd., Bangalore, India

ABSTRACT: **In Software Testing, we try to cover all test scenarios and test cases to test an application/product to achieve 100% test coverage. However, most of the times we do not have the leisure of time to test all the test cases [1]. In such cases, we have to balance between the specified time and quality so that we achieve maximum test coverage. This becomes a very big challenge for the Software Test Managers.**

According to Gupta [6], managers adopt multiple strategies to optimize the testing effort and achieve the right balance between Cost, Schedule and Quality. Striking a right balance decides the success or failure of any software solution.
In this paper, we have proposed a technique called Orthogonal Array Approach for reducing the number of test cases that needs to be tested for any given software and hence improve the efficiency of testing. With this technique, reduced numbers of test cases are generated automatically. Fewer test cases would reduce time consumption of the testing as a whole and hence the cost of testing will also reduce. The details of the technique are described along with two case studies for illustration. The advantages of the method are clearly brought out with the help of analysis graphs.

Keywords: Software Testing, Test Case Generation, Test Case Reduction, Orthogonal Array Approach.

I. INTRODUCTION

Software testing has now become a very complex and challenging task. To achieve this challenging task, we need to build a proper strategy. The strategy provides a road map that describes the steps to be undertaken as part of testing. It also lays down the effort, time, and resources required to achieve the same.
Thus, testing strategy consists of test planning, test case design, test case execution and test result data collection and evaluation **[5].**

Software testing activity is not just the identification and specification of defects. It covers reporting and also offers suggestions and recommendations for appropriate actions to be taken for improving the software product/solution. **[2]**

For any given software application, we have a huge number of test cases. We require to identify only those test cases that would lead us to expose maximum number of undetected errors. Despite the importance of techniques in identifying these test cases, developing the techniques remains one the most difficult aspects of software testing. **[3]**

The paper has been divided into several sections. Section 2, provides an insight into the need for Test Optimization. Section 3 describes the "Orthogonal Array Approach" for generating optimal number of test cases. In section 4, two case studies have been described. Section 5 states the results and gives the analysis graphs demonstrating the benefits of using this approach. The concluding remarks and scope for further work has been given in section 6.

II. WHY OPTIMIZATION?

Testing process, being the last stage before the release of the product to the customer, holds the key to success of the product in the market.
With the day to day increasing competition in the market, there is a need to reduce the testing lifecycle so that the desired testing can be performed with high quality and less cost. Considering the fact that time is always at a premium, the need for having an optimized testing process is therefore very essential. **[7]**

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

This requires the Testing Team and Test Manager to optimize the Testing Process **[4].** We have explored the usage of the Orthogonal Array Approach described in the next section to reduce the number of test cases significantly.

III. ORTHOGONAL ARRAY APPROACH

The following terminologies have been used in this approach

1) Factor (f): Those parameters that the tester intentionally changes during testing to study its effect on the output.
2) Levels (p): The different values of factors used in testing.

During testing, we basically observe the average change in the response when a factor is changed from one level to another level. Hence to achieve the entire test coverage, we should have (f*p) number of test cases.

As mentioned above, all these test cases can be executed in an ideal situation with infinite time and budget available. We are using orthogonal array approach to reduce the number of test cases.

The following steps are followed to construct the orthogonal array for testing a program with f factors, each factor having p levels:

1. Check if p is a prime number.
2. In case where each factor has different levels, Check whether the highest level is a prime number.
3. If the highest level not a prime, identify the next highest prime number.
4. Check if f <=p+1.If not, check if f is a prime number, else identify the next highest prime number.
5. There exists an OA (Orthogonal Array) with p^2 rows and (p+1) columns.
6. When p=3, we have an OA with 9 rows and 4 columns.
7. We construct "p tuples" (e1, e2, ….ep) as follows:

   e1 = (0,1,2,….,p-1) = (0,1,2)

   e2 = (1,2,…….,p) = (1,2,3)

   ei=(ei-1 + e1) mod p, for i= 3 to p

   e3 = (e2 + e1) mod 3 = (1,3,2)

8. Let us now construct the 9 rows and 4 columns.

   The 9 rows represent the 9 test cases.

The 1st column is constructed as follows:

Rows	1	2	3	4
1	1			
2	1			
3	1			
4	2			
5	2			
6	2			
7	3			
8	3			
9	3			

9. The 1st p (p=3) rows are constructed as follows:

Rows	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2			
5	2			
6	2			
7	3			
8	3			
9	3			

Now we will make use of the tuples constructed in Step 7.

10. Insert ei's in ((i-1)*p) +1 th rows, i = 2, 3..., p

In our case, for p =3,

e2 in ((2-1)*3) + 1 th row = 4th row; e2 = (1,2,3)

e3 in ((3-1)*3) + 1 th row = 7th row; e3 = (1,3,2)

Rows	1	2	3	4
1	1	1	1	1

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2			
6	2			
7	3	1	3	2
8	3			
9	3			

11. Fill (i*p) + 2 th row as (e(i+1) + 1) mod p, i = 1,2,……,p

In our case, for p =3,

5^{th} row = (e2 + 1) mod 3; ((2, 3, 4) mod 3) i.e. (2, 3, 1)

8^{th} row = (e3 + 1) mod 3; ((2, 4, 3) mod 3) i.e. (2, 1, 3)

Rows	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2			
7	3	1	3	2
8	3	2	1	3
9	3			

12. Fill (i*p) + 3 rd row as (e (i+1) + 2) mod p, i = 1, 2,….., p

In our case, for p =3,

6^{th} row = (e2 + 2) mod 3; ((3, 4, 5) mod 3) i.e. (3, 1, 2)

9^{th} row = (e3 + 2) mod 3; ((3, 5, 4) mod 3) i.e. (3, 2, 1)

Rows	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

13. When p=5, we have an OA with 25 rows and 6 columns.

Let us construct the OA with p=5.

14. We construct 5 tuples as follows:

e1 = (0,1,2,….,p-1) = (0,1,2,3,4)

e2 = (1,2,……,p) = (1,2,3,4,5)

ei=(ei-1 + e1) mod p, for i= 3 to p

e3 = (e2 + e1) mod 5 = (1,3,5,7,9) mod5 = (1,3,5,2,4)

e4 = (e3 + e1) mod 5 = (1,4,7,5,8) mod 5 = (1,4,2,5,3)

e5 = (e4 + e1) mod 5 = (1,5,4,8,7) mod 5 = (1,5,4,3,2)

15. The 1^{st} column is constructed as follows:

Rows	1	2	3	4	5	6
1	1					
2	1					
3	1					
4	1					
5	1					
6	2					
7	2					
8	2					
9	2					
10	2					
11	3					
12	3					
13	3					
14	3					
15	3					
16	4					
17	4					
18	4					

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

19	4					
20	4					
21	5					
22	5					
23	5					
24	5					
25	5					

16. THE 1ST P (P=5) ROWS ARE CONSTRUCTED AS FOLLOWS:

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2					
7	2					
8	2					
9	2					
10	2					
11	3					
12	3					
13	3					
14	3					
15	3					
16	4					
17	4					
18	4					
19	4					
20	4					
21	5					
22	5					
23	5					
24	5					
25	5					

17. Insert ei's in ((i-1)*p) +1 th rows, i = 2, 3,….., p

In our case, for p =5,

e2 in ((2-1)*5) + 1 th row = 6th row; e2 = (1,2,3,4,5)
e3 in ((3-1)*5) + 1 th row = 11th row; e3 = (1,3,5,2,4)
e4 in ((4-1)*5) + 1 th row = 16th row; e4 = (1,4,2,5,3)
e5 in ((5-1)*5) + 1 th row = 21st row; e5 = (1,5,4,3,2)

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2					
8	2					
9	2					
10	2					
11	3	1	3	5	2	4
12	3					
13	3					
14	3					
15	3					
16	4	1	4	2	5	3
17	4					
18	4					
19	4					
20	4					
21	5	1	5	4	3	2
22	5					
23	5					
24	5					
25	5					

18. Fill (i*p) + 2 th row as (e(i+1) + 1) mod p, i = 1,2,…….,p

In our case, for p =5,

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

7th row = (e2 + 1) mod 5; ((2, 3, 4, 5, 6) mod 5) i.e. (2, 3, 4, 5, 1)
12th row = (e3 + 1) mod 5; ((2, 4, 6, 3, 5) mod 5) i.e. (2, 4, 1, 3, 5)
17th row = (e4 + 1) mod 5; ((2, 5, 3, 6,4) mod 5) i.e. (2,5,3,1,4)
22nd row = (e5 + 1) mod 5; ((2, 6, 5, 4, 3) mod 5) i.e. (2, 1, 5, 4, 3)

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2					
9	2					
10	2					
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3					
14	3					
15	3					
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4					
19	4					
20	4					
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5					
24	5					
25	5					

8th row = (e2 + 2) mod 5; ((3, 4, 5, 6, 7) mod 5) i.e. (3, 4, 5, 1, 2)
13th row = (e3 + 2) mod 5; ((3, 5, 7, 4, 6) mod 5) i.e. (3, 5, 2, 4, 1)
18th row = (e4 + 2) mod 5; ((3, 6, 4, 7, 5) mod 5) i.e. (3, 1, 4, 2, 5)
23rd row = (e5 + 2) mod 5; ((3, 7, 6, 5, 4) mod 5) i.e. (3, 2, 1, 5, 4)

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2					
10	2					
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3					
15	3					
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4					
20	4					
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5					
25	5					

19. Fill (i*p) + 3 rd row as (e (i+1) + 2) mod p, i = 1, 2, ….., p

In our case, for p =5,

20. Fill (i*p) + 4 th row as (e (i+1) + 3) mod p, i = 1, 2, ….., p

In our case, for p =5,

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

9th row = (e2 + 3) mod 5; ((4, 5, 6, 7, 8) mod 5) i.e. (4, 5, 1, 2, 3)
14th row = (e3 + 3) mod 5; ((4, 6, 8, 5, 7) mod 5) i.e. (4, 1, 3, 5, 2)
19th row = (e4 + 3) mod 5; ((4, 7, 5, 8, 6) mod 5) i.e. (4, 2, 5, 3, 1)
24th row = (e5 + 3) mod 5; ((4, 8, 7, 6, 5) mod 5) i.e. (4, 3, 2, 1, 5)

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2	4	5	1	2	3
10	2					
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3	4	1	3	5	2
15	3					
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4	4	2	5	3	1
20	4					
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5	4	3	2	1	5
25	5					

10th row = (e2 + 4) mod 5; ((5, 6, 7, 8, 9) mod 5) i.e. (5, 1, 2, 3, 4)
15th row = (e3 + 4) mod 5; ((5, 7, 9, 6, 8) mod 5) i.e. (5, 2, 4, 1, 3)
20th row = (e4 + 4) mod 5; ((5, 8, 6, 9, 7) mod 5) i.e. (5, 3, 1, 4, 2)
25th row = (e5 + 4) mod 5; ((5, 9, 8, 7, 6) mod 5) i.e. (5, 4, 3, 2, 1)

Rows	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	2	2	2	2
3	1	3	3	3	3	3
4	1	4	4	4	4	4
5	1	5	5	5	5	5
6	2	1	2	3	4	5
7	2	2	3	4	5	1
8	2	3	4	5	1	2
9	2	4	5	1	2	3
10	2	5	1	2	3	4
11	3	1	3	5	2	4
12	3	2	4	1	3	5
13	3	3	5	2	4	1
14	3	4	1	3	5	2
15	3	5	2	4	1	3
16	4	1	4	2	5	3
17	4	2	5	3	1	4
18	4	3	1	4	2	5
19	4	4	2	5	3	1
20	4	5	3	1	4	2
21	5	1	5	4	3	2
22	5	2	1	5	4	3
23	5	3	2	1	5	4
24	5	4	3	2	1	5
25	5	5	4	3	2	1

21. Fill (i*p) + 5 th row as (e (i+1) + 4) mod p, i = 1, 2, ….., p

In our case, for p =5,

IV. CASE STUDIES

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

Case Study 1: The above approach was used for a project where Compatibility Testing had to be performed for various Browser-OS-Database combinations.

The factors and various levels for each of the factors are listed below in Table 1:

Table 1: Factors and Levels listed for the Compatibility Testing Scenario

Factors	Level 1	Level 2	Level 3
A. Web Browser	IE 5.0	Netscape 4.7	Mozilla 1.3.1
B. Web Server OS	Sun OS 2.8	HP-UX 11	Windows NT Server 4.0
C. Type of Operation	Retrieval of Data	Saving of Data	Data Deletion
D. Database	Oracle 8i	SQL Server 7.0	Sybase ASE 12.5

Here the highest level =3, which is a prime number.

Hence the OA with p=3 is constructed as follows in Table 2:

Table 2: Orthogonal Array constructed for the Compatibility Testing Scenario

Test Number	Web Browser	Web Server OS	Type of Operation	Database
1	IE 5.0	Sun OS 2.8	Retrieval of Data	Oracle 8i
2	IE 5.0	HP-UX 11	Saving of Data	Sybase ASE 12.5
3	IE 5.0	Windows NT Server 4.0	Data Deletion	SQL Server 7.0
4	Netscape 4.7	Sun OS 2.8	Saving of Data	SQL Server 7.0
5	Netscape 4.7	HP-UX 11	Data Deletion	Oracle 8i
6	Netscape 4.7	Windows NT Server 4.0	Retrieval of Data	Sybase ASE 12.5
7	Mozilla 1.3.1	Sun OS 2.8	Data Deletion	Sybase ASE 12.5
8	Mozilla 1.3.1	HP-UX 11	Retrieval of Data	SQL Server 7.0
9	Mozilla 1.3.1	Windows NT Server 4.0	Saving of Data	Oracle 8i

Benefit shown to the customer:

For 4 Factors, each with 3 levels, the total no. of test cases = $3^4 = 81$.
With the Orthogonal Array Approach, we have been able to reduce it to 9 Test Cases.
Thus we have been able to reduce the testing effort to (1/9) i.e. 11.11% of the total effort.
Hence, Effort Saved = 88.89%.

Case Study 2: Customer was delighted with our approach and suggested us to add 1 more browser and 1 more OS to this combination.

Hence, the factors and various levels for each of the factors are now as follows in Table 3:

Table 3: Factors and Levels listed for the Compatibility Testing Scenario with 1 more browser and 1 more OS added

Factors	Level 1	Level 2	Level 3	Level 4
A. Web Browser	IE 5.0	Netscape 4.7	Mozilla 1.3.1	Safari
B. Web Server OS	Sun OS 2.8	HP-UX 11	Windows NT Server 4.0	Windows Vista
C. Type of Operation	Retrieval of Data	Saving of Data	Data Deletion	
D. Database	Oracle 8i	SQL Server 7.0	Sybase ASE 12.5	

Here the highest level =4, which is not a prime number.
The second highest level =3, which is a prime number.
Hence we can construct the Orthogonal Array with 4*3= 12 rows and 4 (3+1) columns.

The OA is constructed as follows in Table 4:

Table 4: Orthogonal Array constructed for the Compatibility Testing Scenario with 1 more browser and 1 more OS added

Test Number	Web Browser	Web Server OS	Type of Operation	Database
1	IE 5.0	Sun OS 2.8	Retrieval of Data	Oracle 8i
2	IE 5.0	HP-UX 11	Saving of Data	Sybase ASE 12.5
3	IE 5.0	Windows NT Server 4.0	Data Deletion	SQL Server 7.0
4	Netscape 4.7	Windows Vista	Saving of Data	SQL Server 7.0
5	Netscape 4.7	Sun OS 2.8	Data Deletion	Oracle 8i
6	Netscape 4.7	HP-UX 11	Retrieval of Data	Sybase ASE 12.5
7	Mozilla 1.3.1	Windows NT Server 4.0	Data Deletion	Sybase ASE 12.5
8	Mozilla 1.3.1	Windows Vista	Retrieval of Data	SQL Server 7.0
9	Mozilla 1.3.1	Sun OS 2.8	Saving of Data	Oracle 8i

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

10	Safari	HP-UX 11	Retrieval of Data	SQL Server 7.0
11	Safari	Windows NT Server 4.0	Saving of Data	Oracle 8i
12	Safari	Windows Vista	Data Deletion	Sybase ASE 12.5

Benefit shown to the customer:

For 4 Factors, 2 factors with 3 levels and the other 2 factors with 4 levels, the total no. of test cases = $3^2 * 4^{2 =}$ 144.

With the Orthogonal Array Approach, we have been able to reduce it to 12 Test Cases.

Thus we have been able to reduce the testing effort (1/12) i.e. 8.33% of the total effort.

Hence, Effort Saved = 91.67%.

V. RESULTS

Table 5 illustrates the results from the above 2 Case Studies.

Case Study 1 shows the reduction of Total Test Cases from 81 to 9 with the use of this algorithm.
88.89% efforts have been saved leading to a Time Saving of 144 hrs and Cost Saving of $3456 (Considering the customer billing rate of $ 24 /hr).

Case Study 1 shows the reduction of Total Test Cases from 144 to 12 with the use of this algorithm.
91.67% efforts have been saved leading to a Time Saving of 264 hrs and Cost Saving of $6336 (Considering the customer billing rate of $ 24 /hr).

Table 5 : Results for the above 2 Case Studies

Parameter	Case Study1	Case Study2
All possible Test cases	81	144
Reduced Test Cases	9	12
Effort Saving (%)	88.89	91.67
Time Saving(hrs)	144	264
Cost Saving ($)	3456	6336

Fig1 and Fig2 show the graphs analyzing the above results.

Fig 1 compares the Total TCs with the Reduced TCs for Case Study1 and Case Study2.
Fig 2 shows the Effort Saving (%) and Time Saving (Hrs) for Case Study1 and Case Study2.

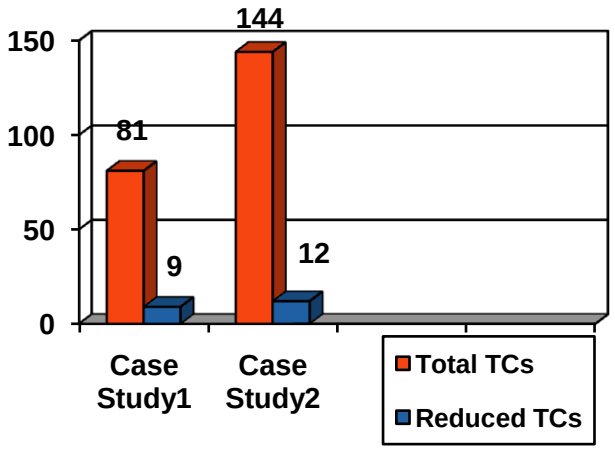


Fig 1: Total TCs and Reduced TCs for the above 2 Case Studies

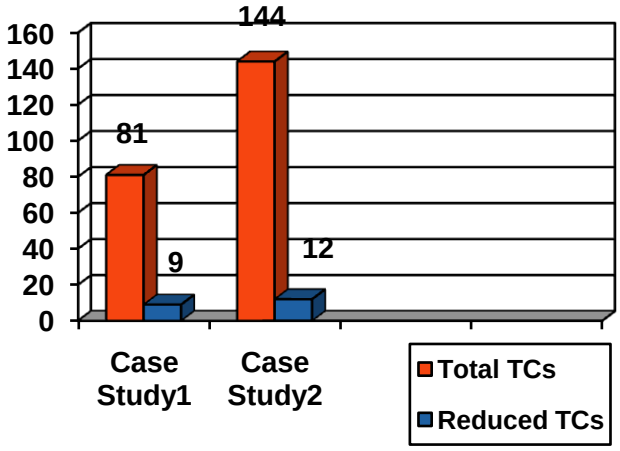
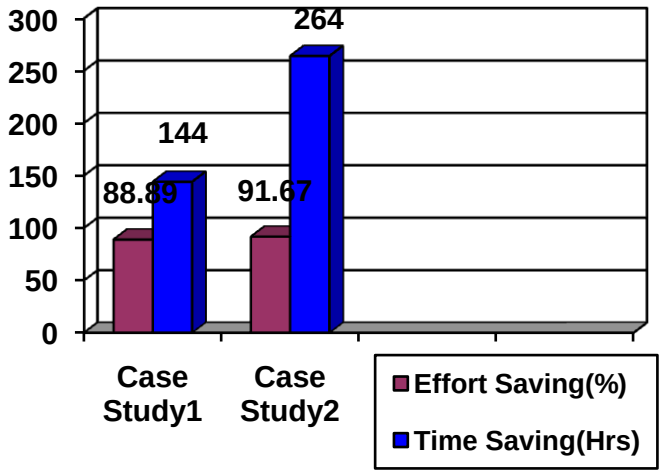


Fig 2 : Effort Saving and Time Saving for the above 2 Case Studies



ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 9, November 2012

VI. CONCLUSION

There are various different factors that determine the effectiveness of the testing. To keep the client happy, the managers have to constantly look for opportunities to decrease the overall cost of testing. Matured organizations are now looking at newer and long term solutions for defining the test effectiveness of their testing functions **[7].**

The above proposed technique has achieved huge reduction in the percentage of the test cases. Based on the analysis done, the proposed method can be considered a superior technique.

Limitation of the proposed technique lies in the fact that by selecting few optimum test cases, we are introducing a risk factor as only few test cases are executed selectively. Another limitation of this technique is the assumption that each of the factors is independent. This may not be the case always. The future work on the technique would try to address these problems.

REFERENCES

[1]B. Beizer. "Software Testing Techniques." Van Nostrand Reinhold, 2nd edition, 1990.

[2] B. Korel, "Automated Software Test Data Generation," Conference on Software Engineering, Vol 10, No. 8, pages 870-879, August 1990.

[3] L. A. Clarke, "A System to Generate Test Data and Symbolically Execute Programs," IEEE Transactions on Software Engineering, Vol. SE-2, No. 3, pages 215-222, September 1976.

[4] L. J. Morell. "A Theory of Error-Based Testing," PhD thesis, University of Maryland, College Park MD, 1984, Technical Report TR-1395.

[5] R. P. Mahapatra and Jitendra Singh, "Improving the Effectiveness of Software Testing through Test Case Reduction", PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 27 FEBRUARY 2008 ISSN 1307-6884.

[6] Shishank Gupta, "Parametric Test Optimization", Software Testing Conference, 2002.

[7] Shishank Gupta, "Testing Effectiveness – Zero defects or higher ROI?" Software Testing Conference, 2006.

BIOGRAPHY

Shubhra Banerji is with IBM India Pvt. Ltd. as Project Manager in their Software Testing Research & Development Unit.

She has a Masters Degree in Physics (Integrated M.Sc) from IIT Kharagpur. She is working for PhD at Jadavpur University In Software Testing.

She is the recipient of R.G. Chatterjee Memorial Gold Medal at IIT Kharagpur on being adjudged the best student in Grade Point Average, Project Work and Laboratory Practices.

She has 14 years of extensive software experience in Development and Testing in various Domains.

Earlier she was with Infosys Technologies Limited Bangalore where she was responsible for managing a group of 50 Test Engineers and Analysts to provide defect-free tested software to client and was involved in Estimation, Status Reporting, Metrics Analysis and Quality Management in her project.

Shubhra has presented a paper in "Quality Management in Software Testing "at the International Software Testing Conference organized by QAI in 2007.

Shubhra has presented a paper on "Digital Watermarking for Information Security in Applications" at Step-Auto Conference 2008 organized by ISQT.

Shubhra's paper "Project Client Relationship Management" has been selected for publication for PML, 2008 organized by QAI.

Shubhra has presented a Tutorial on "User Acceptance Testing: The Right Way" in SteP-IN 2009.

Shubhra can be contacted at: shubhra.banerji@gmail.com