Test Case Reduction based upon Path Coverage Criteria

Monika Grover
Research Scholar
DCSE, GJUS&T, Hisar

Pradeep Kumar Bhatia
Professor
DCSE, GJUS&T, Hisar

ABSTRACT
The cost of commercial software systems is usually over budget and within limited time duration. Most critical and time consuming phase of a software development process is effort estimation and 50% of this phase is often devoted to testing effort estimation. If we can find a way to reduce effort estimation, then it will greatly deduct total cost and effort needed to be spent in software development. Researchers have been continuously trying to find new ways to reduce effort estimation and improving methods which have been devised already. This paper proposes to use path-oriented test case generation and their reduction with reference to different priorities such as complexity, impact, path coverage etc. We are using different examples to reduce the test cases using various factors i.e. Test case Complexity, its impact and the path coverage.

Keywords
Test case reduction, complexity of test case, impact factor, path coverage

1. INTRODUCTION
Redundancy of software largely depends upon how efficiently and skilfully it was tested during its development process. It can also be stated as the process of validating and verifying that software meets the business and technical requirements that guided its design and development, so that it works as expected [1]. Many ways have been devised through which we can reduce the number of test cases to be implemented that can greatly reduce testing effort of a software developing process [2, 14]. There are many artificial intelligent concepts, such as neutral network, fuzzy logic, learning algorithms and case based reasoning (CBR) [15, 16] to resolve issue of testing effort.

Case-based reasoning (CBR) is an artificial intelligence method that solves problems on the basis of previous similar cases and past experiences [17].CBR has an uncontrollable costs issue to test the system. The maintaining CBR is known as CBM. David C. Wilson presented the overall concepts of CBR and case based maintenance. It can be categorized into two types: traditional-based and ontology-based. "CBM was defined as the process of refining a CBR system's case-base to improve the system's performance. It implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives."

Aamodt and Plaza [18] provided scheme of the CBR working cycle comprising of four phases: RETRIEVE, REUSE, REVISE and RETAIN. All these four phases rely on the knowledge available in the form of previous similar cases. Both cases and knowledge are important factors and basis of a CBR system. Any system lacking either of these two cannot make use of full efficiency of CBR which will ultimately result in unsatisfactory working performance [20, 21]. Applications of CBR are being used in different fields including medical and non-medical [21-25].

1.1 Definitions
Before we proceed further, there are some terms which are necessary to be introduced to understand working of CBR as follows:

a) **Test suite**: A test suite can be defined as a group of test cases and is used to test software for a specific property or behaviour. It is mainly a part of software developing process. A single test case can be added to a number of test suites.

b) **Case Base** is a collection of cases in CBR, which can be defined as the following:

Given a case - base C = {c_1... c_n}, for c ε C
where, C = CBR, c = case

c) **Auxiliary Case** is a case that does not have a direct effect on the competence of a system when it is deleted. Auxiliary cases do not affect competence at all. When the test case is deleted, it can only reduce the efficiency of the system. A case can be called as an auxiliary case if the coverage it provides is subsumed by the coverage of one of its reachable cases.

d) **Pivotal Case** is the case that directly affects competence of a system when deleted.

A case is a pivotal case if its deletion directly reduces the competence of a system (irrespective of the other cases in the case-base). A case can be called as a pivotal case if it is reachable by no other case but itself, there are different studies has been done to reduce the test case using case base reasoning.

e) **Coverage Set** : Given a case- base C = {c_1,c_2……c_n}
Coverage (c) = {c'∈C (adaptable (c', c))}

f) **Reachability Set:**

Given a case – base C = {c_1, c_2……c_n}
Reachability(c) = {c`∈C (adaptable (c, c`)}

Whenever we need to create a test suite, we need to identify the number of the states existed in the given program. Fig. 1 depicts control flow graph of a source code. The states presents in the fig.1are s_1, s_2, s_3, s_4 and s_5. We will need to find all possible test suites that can cover all given states existed and its notation is given below:

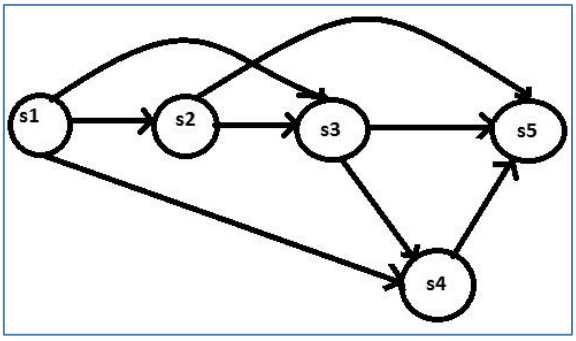

Fig 1: Control Flow Graph

$$TC_n = \{s_1, \ldots, s_n\} \quad \text{.....(Eq 1)}$$

where, TC is the test case and s_n is the stage in the control flow graph that will be tested.

To find all possible test suites, we depend on different stages that involve in the source code.

Table 1. Set of Test Cases [Test Suit]

TC_1 = $\{s_1, s_2\}$	**TC_{12}** = $\{s_1, s_2, s_3, s_4, s_5\}$
TC_2 = $\{s_1, s_3\}$	**TC_{13}**= $\{s_2, s_3\}$
TC_3 = $\{s_1, s_4\}$	**TC_{14}**= $\{s_2, s_5\}$
TC_4 = $\{s_1, s_2, s_3\}$	**TC_{15}**= $\{s_2, s_3, s_4\}$
TC_5 = $\{s_1, s_2, s_5\}$	**TC_{16}**= $\{s_2, s_3, s_5\}$
TC_6 = $\{s_1, s_3, s_4\}$	**TC_{17}**= $\{s_2, s_3, s_4, s_5\}$
TC_7 = $\{s_1, s_3, s_5\}$	**TC_{18}**= $\{s_3, s_4\}$
TC_8 = $\{s_1, s_4, s_5\}$	**TC_{19}**= $\{s_3, s_5\}$
TC_9 = $\{s_1, s_2, s_3, s_4\}$	**TC_{20}**= $\{s_3, s_4, s_5\}$
TC_{10} = $\{s_1, s_2, s_3, s_5\}$	**TC_{21}**= $\{s_4, s_5\}$
TC_{11} = $\{s_1, s_3, s_4, s_5\}$	

The test suite provides us the set of all possible test cases. Now, to determine, how many test cases are covered under one test suite. We will further check the coverage value of each test case. Lower the coverage value, less chance to find software bugs or errors during testing process and vice-versa.

According to the above definition of coverage set, the coverage set for each test cases is given below:

Table 2. Coverage Set

Coverage (1)	$\{TC_1, TC_4, TC_5, TC_9, TC_{10}, TC_{12}\}$
Coverage (2)	$\{TC_2, TC_6, TC_7, TC_{11}\}$
Coverage (3)	$\{TC_3, TC_8\}$
Coverage (4)	$\{TC_4, TC_5, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{15}, TC_{16}, TC_{17}\}$
Coverage (5)	$\{TC_1, TC_4, TC_5, TC_{10}, TC_{12}\}$
Coverage (6)	$\{TC_2, TC_6, TC_9, TC_{11}, TC_{12}, TC_{15}, TC_{17}, TC_{18}, TC_{20}\}$
Coverage (7)	$\{TC_2, TC_7, TC_{10}, TC_{11}, TC_{16}, TC_{19}\}$
Coverage (8)	$\{TC_3, TC_8, TC_{11}, TC_{12}, TC_{17}, TC_{20}TC_{21}\}$
Coverage (9)	$\{TC_1, TC_2, TC_4, TC_6, TC_9, TC_{10}, TC_{13}, TC_{15}, TC_{17}, TC_{18}, TC_{20}\}$
Coverage (10)	$\{TC_1, TC_4, TC_7, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{16}, TC_{19}\}$
Coverage (11)	$\{TC_2, TC_6, TC_8, TC_{11}, TC_{12}, TC_{17}, TC_{20}, TC_{21}\}$
Coverage (12)	$\{TC_1, TC_4, TC_5, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{15}, TC_{17}, TC_{18}, TC_{20}, TC_{21}\}$
Coverage (13)	$\{TC_4, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{15}, TC_{16}, TC_{17}\}$
Coverage (14)	$\{TC_5, TC_{14}\}$
Coverage (15)	$\{TC_4, TC_6, TC_9, TC_{10}, TC_{11}, TC_{12}, TC_{13}, TC_{15}, TC_{17}, TC_{18}, TC_{20}\}$
Coverage (16)	$\{TC_4, TC_7, TC_9, TC_{10}, TC_{17}, TC_{21}\}$
Coverage (17)	$\{TC_4, TC_5, TC_6, TC_9, TC_{11}, TC_{12}, TC_{13}, TC_{15}, TC_{17}, TC_{18}, TC_{20}, TC_{21}\}$
Coverage (18)	$\{TC_6, TC_9, TC_{11}, TC_{12}, TC_{15}, TC_{17}, TC_{18}, TC_{20}\}$
Coverage (19)	$\{TC_7, TC_{10}, TC_{16}, TC_{19}\}$
Coverage (20)	$\{TC_6, TC_9, TC_{11}, TC_{12}, TC_{15}, TC_{17}, TC_{20}, TC_{21}\}$
Coverage (21)	$\{TC_8, TC_{11}, TC_{12}, TC_{17}, TC_{20}, TC_{21}\}$

Considering coverage values calculated above, a reachability set is made for each test case.

Table 3. Reachability Set

Reachability (1)	{1, 4, 5, 9, 10, 12}
Reachability (2)	{2, 6, 7, 11}
Reachability (3)	{ 3, 8}
Reachability (4)	{4, 5, 9, 10, 12, 13, 15, 16, 17}
Reachability (5)	{1, 4, 5, 10, 12}
Reachability (6)	{2, 6, 9, 11, 12, 15, 17, 18, 20}
Reachability (7)	{1, 7, 10, 11, 17, 16, 19}
Reachability (8)	{3, 8, 11, 12, 17, 20, 21}
Reachability(9)	{1, 2, 4, 6, 9, 10, 13, 15, 17, 18, 20}
Reachability (10)	{2, 6, 8, 11, 12, 17, 20, 21}
Reachability (11)	{1, 4, 5, 9, 10, 12, 13, 15, 17, 18, 20, 21}
Reachability (12)	{1, 4, 5, 9, 10, 12, 13, 15, 17, 18, 20, 21}
Reachability (13)	{4, 9, 10, 12, 13, 15, 16, 17}
Reachability (14)	{5, 14}
Reachability (15)	{4, 6, 9, 10, 12, 13, 15, 17, 18,20}
Reachability (16)	{4, 7, 9, 10, 17, 21}
Reachability (17)	{4, 5, 6, 9, 11, 12, 13, 15, 17, 18, 20}
Reachability (18)	{6, 9, 11, 12, 13, 15, 17, 18, 20}
Reachability (19)	{7, 10, 16, 19}
Reachability (20)	{6, 9, 11, 12, 13, 15, 17, 20, 21}
Reachability (21)	{8, 11, 12, 17, 20, 21}

Auxiliary set = $\{TC_1, TC_2, TC_3, TC_4, TC_5, TC_6, TC_7, TC_8, TC_9, TC_{10}, TC_{11}, TC_{12}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}\}$ ..... (Eq 2)

2. PROPOSED METRICS

Redundant test cases increase testing effort and further increase the software cost. Our main goal is to reduce the testing time by removing the duplicate test cases. We can also prioritize the test cases considering the regression testing. Prior techniques for test case prioritization are based on the total number of coverage requirements [7]

In this section, we have proposed five metrics that helps to reduce redundancy test cases on the basis of path coverage criteria. Further, we will use five metrics to compare the reduction of the test suite.

Metric I: Test Case Complexity Factor(TCCF)

A complexity of test case is the significant criteria; the complexity of test case measures the number of states that covers the test case.

Notation **Comp(TC)**
Where Comp: Complexity Factor
TC: Test case

Comp(TC) = { High, Medium, Low}

High: When the number of states in test case are greater than average number of states of all test cases. **i.e.** $\boldsymbol{n(TC)} > n(TCavg)$
Medium : When the number of states in test case are equal to average number of states of all test cases. **i.e.** $\boldsymbol{n(TC)} = n(TCavg)$
Low: When the number of states in test case are less than the average number of states of all test cases. **i.e.** $\boldsymbol{n(TC)} < n(TCavg)$

The process to remove the test case with minimal complexity is described below:

Step 1: Determine the coverage set

Step 2: Determine the reachability set.

Step 3: Define the auxiliary set.

Step 4: Compute the average complexity of the test cases and as per the rule of the Comp(TC), categorise the test cases into low, medium and high.

Step 5: Remove test cases with minimum complexity.

Metric II: Test Case Impact Factor(TCIF)

The impact recognizes the test case that finds the errors within the states at numerous times.

Notation **Imp(TC)**
Where Imp: Impact factor
TC: Test case

Imp(TC) = {High, Medium, Low}

High: The test case has revealed at least one fault for many times.
Medium: The test case has revealed faults for only one time.
Low: when the test case has never revealed faults.

The steps to apply the test case impact factor are given below:

Step 1: Determine the coverage set

Step 2: Determine the reachability set.

Step 3: Define the auxiliary set.

Step 4: Compute the impact value for all test cases in the auxiliary set as per the rule of Imp(TC).

Step 5: Remove all test cases that have minimum Impact value.

Metric III: Path Coverage Factor(PCF)

The coverage value can specify how many nodes the test case can cover. In other words, the coverage value is an indicator to measure that each test case covers. It means that the higher coverage value, the more nodes can be contained and covered in the test case.

Notation

Cov(n) = value

Where, Cov is a coverage value, value is a number of test cases in each coverage group and n is a coverage relationship.

The procedure of this method is described below:

Step 1: Determine the coverage set

Step 2: Determine the coverage value

Step 3: Remove the test cases with minimum coverage value.

Metric IV: **Test Suite Minimization Effectiveness and Its Impact**

Effectiveness of Test Case Minimization=

$$\left(1 - \frac{\textit{Number of the test cases in reduced test suite}}{\textit{Number of the test cases in original test suite}}\right) * 100$$

Impact of Test Case Minimization=

$$\left(1 - \frac{\text{Number of fault detected in reduced test suite}}{\text{Number of fault detected in original test suite}}\right) * 100$$

Metric V: **Average percentage of Fault Detection**

$$= 1 - \frac{TF1 + TF2 + \ldots + TFM}{NM} + \frac{1 * N}{2}$$

Where as T is the test suite under test.

- M is the number of faults in the program under test P.
- n is the total number of test cases.
- TFi is the position of the first test in T that reveals fault i.

3. RESULTS AND DISCUSSION

In this section, we have applied the five metrics on two different control flow graphs derived from two source codes or programs.

In the metrics i.e. test case complexity factor, test case impact factor and path coverage; three steps need to perform. That is identifying the number of states; all possible test cases; coverage set; the reachability set and auxiliary set. So, we will implement these steps and then further apply metrics.

3.1 Example 1

We will consider fig 1 in order to apply five metrics.

Number of states, possible test cases, coverage set, and reachability set corresponding to Figure 1 already described in subsection 1.1 with equation (Eq 1), Table 1, Table 2, Table 3 (Eq 2) respectively.

a) Implementation of Metric I - Test case complexity Factor on fig.1

As the total number of test cases are 21. So, the average number of the stages is 3.

Ultimately, in the last step, we remove test cases having minimum complexity values from auxiliary set.

Comp (Complexity values)		
Low (8)	**Medium** (8)	**Large** (4)
TC_1, TC_2, TC_3, TC_{13}, TC_{14}, TC_{18}, TC_{19}, TC_{21}	TC_4, TC_5, TC_6, TC_7, TC_8, TC_{15}, TC_{16}, TC_{20}	TC_9, TC_{10}, TC_{11}, TC_{17}

According to the definition of the Test Case Complexity, the test cases with the minimum complexity value will be eliminated. So, TC_1, TC_2, TC_3, TC_{13}, TC_{14}, TC_{18}, TC_{19}, TC_{21}test cases will be removed from the auxiliary set.

It is difficult to define and measure the software quality. The inadequate testing leads the software towards poor quality, expensive and vast time-to-deliver. In conclusion, software testing engineers require identifying the impact of each test case in order to acknowledge and understand clearly the impact of ignoring some test cases. In this paper, an impact value is considered as the impact of test cases in term of the ability to detect faults if those test cases are removed and not be tested.

b) Implementation of Metric II: Test case impact Factor on fig. 1

In the step 5 of the metric 2, the test case with the low impact value will be removed from the auxiliary set.

Redundant test cases increase testing effort and further increase the software cost. Our main goal is to reduce the testing time by removing the duplicate test cases. We can also prioritize the test cases considering the regression testing. Prior techniques for test case prioritization are based on the total number of coverage requirements [7]

Impact (Impact values)		
Low (14)	**Medium** (4)	**Large** (2)
TC_1, TC_4, TC_5, TC_9, TC_{10}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}	TC_2, TC_6, TC_7, TC_{11}	TC_3, TC_8

We remove test cases having minimum complexity values from auxiliary set. So, TC_1, TC_4, TC_5, TC_9, TC_{10}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20} and TC_{21} are removed.

Implementation of Metric III: Test case coverage on fig.1

Cov(n) Coverage value			
Test Case	**Cov value**	**Test Case**	**Cov value**
Cov (1)	6	Cov (12)	12
Cov (2)	4	Cov (13)	8
Cov (3)	2	Cov (14)	2
Cov (4)	9	Cov 15)	10
Cov (5)	5	Cov (16)	6
Cov (6)	9	Cov 17)	11
Cov (7)	7	Cov (18)	9
Cov (8)	7	Cov (19)	4
Cov (9)	11	Cov (20)	9
Cov (10)	8	Cov (21)	6
Cov (11)	12		

The classification of the test cases in low, medium and high

Cov (Coverage values)		
Low (4)	**Medium** (8)	**Large** (8)
TC_2, TC_3, TC_{14}, TC_{19}	TC_1, TC_5, TC_7, TC_8, TC_{10}, TC_{13}, TC_{16}, TC_{21}	TC_4, TC_6, TC_9, TC_{12}, TC_{15}, TC_{17}, TC_{18}, TC_{20}

Ultimately, in the last step, we remove test cases having minimum complexity values from auxiliary set. So, TC_2, TC_3, TC_{14}, TC_{19} are removed.

Test Cases Deleted For The Figure 1	
Filtering Method	**Removed Test Cases**
TCCF(8)	TC_1, TC_2, TC_3, TC_{13}, TC_{14}, TC_{18}, TC_{19}, TC_{21}
TCIF(14)	TC_1, TC_4, TC_5, TC_9, TC_{10}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}
PCF(4)	TC_2, TC_3, TC_{14}, TC_{19}

Implementation of Metric IV: Test Suite Minimization

	Effectiveness	**Impact**
TCCF	40	80
TCIF	70	90
PCF	20	60

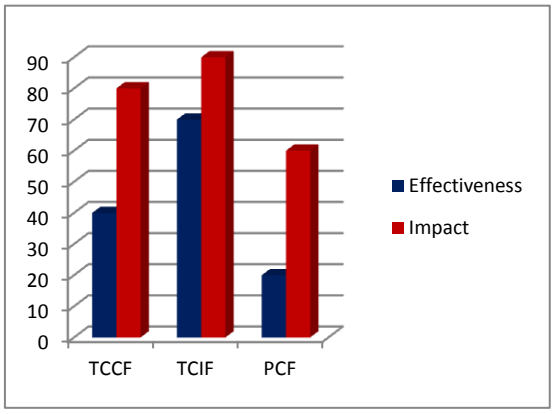

Fig. 2 - Test case reduction metric v/s Minimization

Implementation of Metric V: Average percentage of Fault Detection (APFD)

$$APFD=1-\frac{13+7+17}{21*3}+\frac{1*21}{2}$$

$$=1-0.59+12.5=12.91\%$$

3.2 Example 2

We will consider control flow graph derived from source code depicted in fig 3.

Number of states, possible test cases, coverage set, and reachability set corresponding to Figure 3 are shown in equation (Eq. 3), Table 4, Table 5, Table 6, and (Eq. 4) respectively.

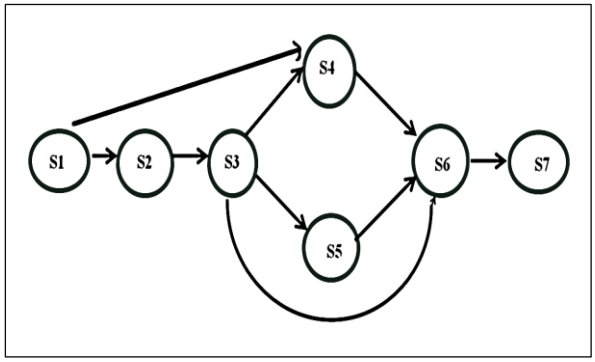

Fig.3 Control Flow Graph

The total number of the states needs to be identified.

$$TC_n=\{s_1, s_2, \ldots, s_n\} \quad \text{.....(Eq 3)}$$

Where, TC is the test case and s_n is the stage in the control flow graph that will be tested.

The test suite will be defined that depends on the used to test the entire case.

Table 4. Set of Test Cases

TC_1 = {S1, S2}	TC_{19} = {S2, S3, S5, S6}
TC_2 = {S1, S4}	TC_{20} = {S2, S3, S6, S7}
TC_3 = {S1, S2, S3}	TC_{21} = {S2, S3, S4, S6, S7}
TC_4 = {S1, S4, S6}	TC_{22} = {S2, S3, S5, S6, S7}
TC_5 = {S1, S2, S3, S4}	TC_{23} = {S3, S4}
TC_6 = {S1, S2, S3, S5}	TC_{24} = {S3, S5}
TC_7 = {S1, S2, S3, S6}	TC_{25} = {S3, S6}
TC_8 = {S1, S4, S6, S7}	TC_{26} = {S3, S4, S6}
TC_9 = {S1, S2, S3, S4, S6}	TC_{27} = {S3, S5, S6}
TC_{10} = {S1, S2, S3, S5, S6}	TC_{28} = {S3, S6, S7}
TC_{11} = {S1, S2, S3, S6, S7}	TC_{29} = {S3, S4, S6, S7}
TC_{12}= {S1, S2, S3, S4, S6, S7}	TC_{30} = {S3, S5, S6, S7}
TC_{13}= {S1, S2, S3, S5, S6, S7}	TC_{31} = {S4, S6}
TC_{14} = {S2, S3}	TC_{32} = {S4, S6, S7}
TC_{15}= {S2, S3, S4}	TC_{33} = {S5, S6}
TC_{16} = {S2, S3, S5}	TC_{34} = {S5, S6, S7}
TC_{17} = {S2, S3, S6}	TC_{35} = {S6, S7}
TC_{18} = {S2, S3, S4, S6}	

To find the coverage set for the figure 2.

Table 5. Coverage Set

Coverage (1)	TC_1, TC_3, TC_5, TC_6, TC_7, TC_8, TC_9, TC_{10}, TC_{11}, TC_{12}
Coverage (2)	TC_2, TC_4, TC_8
Coverage (3)	TC_1, TC_3, TC_5, TC_6, TC_7, TC_9, TC_{10}, TC_{11}, TC_{12}
Coverage (4)	TC_2, TC_4, TC_8, TC_{31}, TC_{32}
Coverage (5)	TC_1, TC_3, TC_5, TC_9, TC_{12}, TC_{15}, TC_{18}, TC_{21}, TC_{23}, TC_{26}
Coverage (6)	TC_1, TC_3, TC_6, TC_7, TC_{10}, TC_{13}, TC_{16}, TC_{19}, TC_{22}, TC_{24}
Coverage (7)	TC_1, TC_3, TC_7, TC_{11}, TC_{17}, TC_{20}, TC_{25}
Coverage (8)	TC_2, TC_4, TC_{25}, TC_{31}, TC_{32}
Coverage(9)	TC_1, TC_3, TC_5, TC_9, TC_{12}, TC_{14}, TC_{15}, TC_{18}, TC_{24}, TC_{22}, TC_{26}, TC_{31}

Coverage (10)	{TC_1, TC_3, TC_6, TC_{10}, TC_{13}, TC_{14}, TC_{16}, TC_{19}, TC_{22}, TC_{24}, TC_{27}, TC_{30}, TC_{33}}
Coverage (11)	{TC_1, TC_2, TC_3, TC_5, TC_7, TC_{13}, TC_{16}, TC_{17}, TC_{20}, TC_{25}, $TC_{28,}$ TC_{30}}
Coverage (12)	{TC_1, TC_3, TC_5, TC_9, TC_{14}, TC_{15}, TC_{18}, TC_{21}, TC_{23}, TC_{26}, TC_{29}, TC_{31}, TC_{35}}
Coverage (13)	{TC_1, TC_3, TC_6, TC_{10}, TC_{13}, TC_{15}, TC_{16}, TC_{19}, TC_{22}, TC_{24}, $TC_{27,}$ TC_{30}, TC_{33}, TC_{35}}
Coverage (14)	{TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}, TC_{22}}
Coverage (15)	{TC5, TC_9, TC_{12}, TC_{14}, $TC_{15,}$ TC_{18}, TC_{21}, TC_{23}}
Coverage (16)	{TC_6, TC_{10}, TC_{13}, TC_{16}, TC_{19}, TC_{22}, TC_{24}}
Coverage (17)	{TC_7, TC_{11}, TC_{17}, TC_{20}, TC_{25}}
Coverage (18)	{TC_5, TC_9, TC_{12}, TC_{15}, TC_{18}, TC_{21}, TC_{23}, TC_{26}, TC_{31}}
Coverage (19)	{TC_6 , TC_{10}, TC_{13}, TC_{14}, $TC_{16,}$ TC_{19}, TC_{22}, TC_{24}, TC_{27}, TC_{30}}
Coverage (20)	{TC7, TC_{11}, TC_{14}, TC_{17}, TC_{25}, TC_{28}}
Coverage (21)	{TC_{14}, TC_{15}, TC_{18}, TC_{21}, TC_{23}, TC_{26} , TC_{31}, TC_{32}, TC_{35}}
Coverage (22)	{TC_6, TC_{10}, TC_{13}, TC_{16}, TC_{19}, TC_{22}, TC_{24}, TC_{27}, TC_{30}, $TC_{33,}$ TC_{35}}
Coverage (23)	{TC_5, TC_9, TC_{12}, TC_{15}, TC_{18}, TC_{21}, TC_{23}, TC_{26}}
Coverage (24)	{ TC_6, TC_{10}, TC_{13}, TC_{16}, TC_{19}, TC_{22}, TC_{24}, TC_{27}, TC_{30}}
Coverage (25)	{TC_7, TC_{10}, TC_{11}, TC_{17}, TC_{20}, TC_{25}, TC_{28}}
Coverage (26)	{TC_9, TC_{12}, TC_{18}, $_{TC21}$, TC_{23}, TC_{26}, TC_{29}, TC_{31} }
Coverage (27)	{ TC_{10}, TC_{13}, TC_{19}, TC_{22}, TC_{24}, TC_{27}, $TC_{30,}$ TC_{33}}
Coverage (28)	{TC_7, TC_{17}, TC_{20}, TC_{26}, TC_{35}}
Coverage (29)	{TC_9, TC_{12}, TC_{18}, TC_{21}, TC_{23}, TC_{26}, TC_{29}, TC_{32}}
Coverage (30)	{TC_{10}, TC_{13}, TC_{19}, TC_{22}, TC_{24}, TC_{27}, TC_{30}, TC_{33}}
Coverage (31)	{TC_4, TC_9, TC_{12}, TC_{18}, TC_{21}, TC_{26}, TC_{29}, TC_{31}}
Coverage (32)	{TC_4, TC_9, TC_{12}, TC_{18}, TC_{21}, TC_{26}, TC_{29}, TC_{31}, TC_{35}}
Coverage (33)	{TC_{10}, TC_{13}, TC_{19}, TC_{22}, TC_{27}, TC_{30}, $TC_3$3, TC_{34}}
Coverage (34)	{TC_{10}, TC_{13}, TC_{19}, TC_{22}, TC_{27}, TC_{30}, $TC_3$3, TC_{34}, TC_{35}}
Coverage (35)	{TC_8, TC_{11}, TC_{12}, TC_{13}, TC_{20}, TC_{21}, TC_{22}, TC_{28}, TC_{29}, TC_{30}, TC_{32}, TC_{34}, TC_{35})

Considering coverage values calculated above, a reachability set is made for each test case.

Table 6. Reachability Set

Reachability (1)	{1, 3, 5, 6, 7, 8, 9, 10, 11, 12}
Reachability (2)	{2, 4, 8}
Reachability (3)	{1, 3, 5, 6, 7, 9, 10, 11, 12}
Reachability (4)	{2, 4, 8, 31, 32}
Reachability (5)	{1, 2, 3, 5, 9, 12, 15, 18, 21, 23, 26}
Reachability (6)	{1, 3, 6, 7, 10, 13, 16, 19, 22, 24}
Reachability (7)	{1, 3, 7, 11, 17, 20, 25}
Reachability (8)	{2, 4, 25, 31, 32}
Reachability (9)	{1, 3, 5, 9, 12, 14, 15, 18, 22, 26, 31}
Reachability (10)	{1, 3, 6, 10, 13, 14, 16, 19, 22, 24, 27, 30, 33}
Reachability (11)	{1, 2, 3, 5, 7, 13, 16, 17, 20, 25, 28, 30}
Reachability (12)	{1, 2, 5, 9, 14, 15, 18, 21, 23, 26, 29, 31, 35}
Reachability (13)	{1, 3, 6, 10. 13, 15, 16, 19, 22,24, 27, 30, 33, 35}
Reachability (14)	{14, 15, 16, 17, 18, 19, 20, 21, 22}
Reachability (15)	{5, 9, 12, 14, 15, 18, 21, 23}
Reachability (16)	{6, 10, 13, 16, 19, 22, 24}
Reachability (17)	{7, 11, 17, 20, 25}
Reachability (18)	{5, 9, 12, 15, 18, 21, 23, 26, 31}
Reachability (19)	{6, 10, 13, 14, 16, 19, 22, 24, 27, 30}
Reachability (20)	{7, 11, 14, 17, 25, 28}
Reachability (21)	{14, 15, 18, 21, 23, 26, 31, 32, 35}
Reachability (22)	{6, 10, 13, 14, 16, 19, 22, 24, 27, 30, 33, 35}
Reachability (23)	{5, 9, 12, 15, 18, 21, 23, 26}

Reachability (24)	{6, 10, 13, 16, 19, 22, 24, 27, 30}
Reachability (25)	{7, 10, 11, 17, 20, 25, 28}
Reachability (26)	{9, 12, 18, 21, 23, 26, 29, 31}
Reachability (27)	{10, 13, 19, 22, 24, 27, 30, 33}
Reachability (28)	{7, 17, 20, 26, 35}
Reachability (29)	{9, 12, 18, 21, 23, 26, 29, 32}
Reachability (30)	{10, 13, 19, 22, 24, 27, 30, 33}
Reachability (31)	{4, 9, 12, 18, 21, 26, 29, 31}
Reachability (32)	{4, 9, 12, 18, 21, 26, 29, 31, 35}
Reachability (33)	{10, 13, 19, 22, 27, 30, 33, 34}
Reachability (34)	{10, 13, 19, 22, 27, 30, 33, 34, 35}
Reachability (35)	{8, 11, 12, 13, 20, 21, 22, 28, 29, 30, 32, 34, 35}

Auxiliary set: it is a set which does not directly affect the error finding ability of test cases.

Auxiliary set = { TC_1, TC_2, TC_3, TC_4, TC_5, TC_6, TC_7 , TC_8 , TC_9, TC_{10}, TC_{11}, TC_{12}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}, TC_{22}, TC_{23}, TC_{24}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{29}, TC_{30}, TC_{31}, TC_{32}, TC_{33}, TC_{34}, TC_{35}} ..... (Eq 4)

Cov(n) Coverage value			
Test Case	**Cov value**	**Test Case**	**Cov value**
Cov(1)	10	Cov (19)	10
Cov (2)	3	Cov (20)	6
Cov (3)	9	Cov (21)	9
Cov (4)	5	Cov (22)	12
Cov (5)	11	Cov (23)	8
Cov (6)	10	Cov (24)	9
Cov (7)	7	Cov (25)	7
Cov (8)	5	Cov (26)	8
Cov (9)	11	Cov (27)	8
Cov (10)	13	Cov (28)	5
Cov (11)	12	Cov (29)	8
Cov (12)	13	Cov (30)	8
Cov (13)	14	Cov (31)	8
Cov (14)	9	Cov (32)	8
Cov (15)	8	Cov (33)	9
Cov (16)	7	Cov (34)	8
Cov (17)	5	Cov (35)	13
Cov (18)	9		

Metric I: Test case complexity factor

Comp Complexity values		
Low (19)	**Medium** (9)	**High** (7)
TC_1, TC_2, TC_3, TC_4, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{23}, TC_{24}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{31}, TC_{32}, TC_{33}, TC_{34}, TC_{35}	TC_5, TC_6, TC_7, TC_8, TC_{18}, TC_{19}, TC_{20}, TC_{29}, TC_{30}	TC_9, TC_{10}, TC_{11}, TC_{12}, TC_{13}, TC_{21}, TC_{22}

Metric II: Test Case Impact factor

Impact values		
Low (28)	**Medium** (4)	**High** (3)
TC_5, TC_6, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}, TC_{22}, TC_{23}, TC_{24}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{29}, TC_{30},TC_{31}, TC_{32}, TC_{33}, TC_{34}, TC_{35}	TC_1, TC_3 , TC_7 , TC_{11}	TC_2, TC_4, TC_8

Metric III: Path Coverage factor

Cov (Coverage values)		
Low (19)	**Medium** (10)	**Large** (6)
TC_1, TC_2, TC_4, TC_7, TC_8, TC_{15}, TC_{16}, TC_{17}, TC_{20}, TC_{23}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{29}, TC_{30}, TC_{31}, TC_{32}, TC_{34}	TC_3, TC_5, TC_6, TC_9, TC_{14}, TC_{18}, TC_{19}, TC_{21}, TC_{24}, TC_{33}	TC_{10}, TC_{11}, TC_{12}, TC_{13}, TC_{22}, TC_{35}

Test Cases Deleted For The Figure 3	
Filtering Method	**Removed Test Cases**
TCCF(19)	TC_1, TC_2, TC_3, TC_4, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{23}, TC_{24}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{31}, TC_{32}, TC_{33}, TC_{34}, TC_{35}
TCIF(28)	TC_5, TC_6, TC_9, TC_{10}, TC_{12}, TC_{13}, TC_{14}, TC_{15}, TC_{16}, TC_{17}, TC_{18}, TC_{19}, TC_{20}, TC_{21}, TC_{22}, TC_{23}, TC_{24}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{29}, TC_{30}, TC_{31}, TC_{32}, TC_{33}, TC_{34}, TC_{35}
PCF(19)	TC_1, TC_2, TC_4, TC_7, TC_8, TC_{15}, TC_{16}, TC_{17}, TC_{20}, TC_{23}, TC_{25}, TC_{26}, TC_{27}, TC_{28}, TC_{29}, TC_{30}, TC_{31}, TC_{32}, TC_{34}

Metric IV: Test Suite Minimization

	Effectiveness	**Impact**
TCCF	54.29	80
TCIF	80	91.43
PCF	54.29	82.86

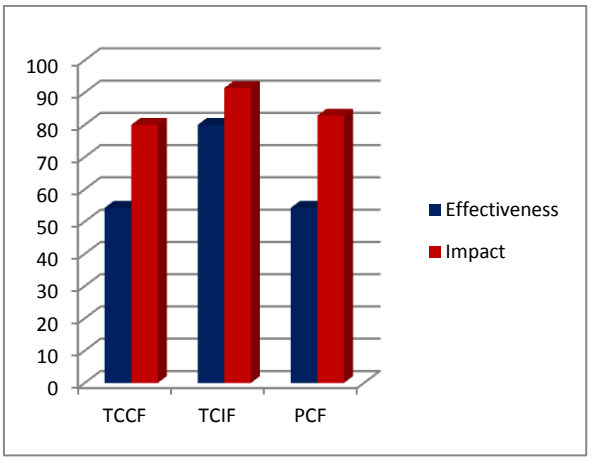

Fig 4 - Test case reduction metrics v/s Minimization

Metric *V: Average percentage of Fault Detection*

$$APFD=1-\frac{16+7+16}{35*3}+\frac{1*35}{2}$$

$$= 1-0.37+17.5=18.13\%$$

Table.7 summarizes the parameters Effectiveness, impact and APED corresponding to Example 1, and Example 2.

Table 7. Comparison of parameters

	Example 1	**Example 2**
Effectiveness	70	90
Impact	80	91.43
APFD	12.91	18.13

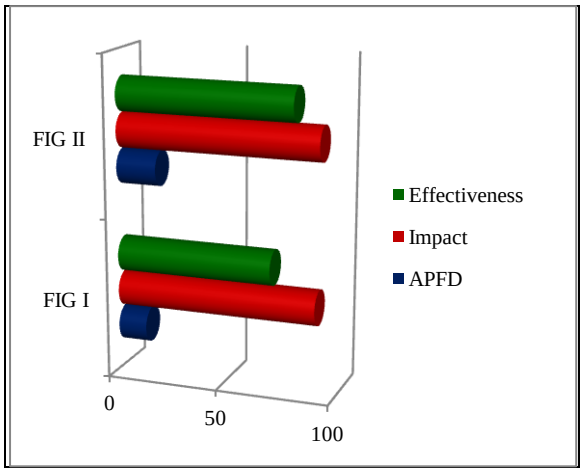

Fig. 5 Comparison of Parameters in Example 1 and Example 2

Excess test cases reduction from the test suite can produce a inefficient and unreliable software. While working on the project, that are still chances that can affect the testing process which can only be experience in the live project. The silver lining is that must define clearly to stop test case reduction. We can further study the optimal solution to limit the number of the test cases can be removed.

4. CONCLUSION

In this paper, we used different metrics to reduce the test suite hence to reduce the testing effort. From both examples 1 and 2 we conclude that the impact factor deletes a quite number of test cases as compare to the complexity factor and coverage factor. The size of the problem does not effect on the result. We have taken two problems; one as an example and other to experiment. In both cases; the impact metric provide us promising results.

We can see in our cases that the impact factor does remove the largest number of the test cases. The average percentage of complexity, impact and coverage is increasing with respect to the size of the problem. But, we also need to keep in mind that the quality of the software should not be downgraded along the way.

5. REFERENCES

[1] Cem Kaner, 2006, Exploratory Testing, Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL.

[2] Gregg Rothermel, Roland H. Untch, Chengyun Chuand Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing", IEEE Transactions on Software Engineering, 2001.

[3] Gregg Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, 1999. Test case prioritization: An empirical study. In Proceedings of the IEEE International Conference on Software Maintenance, pages 179-188, Oxford, England, UK.

[4] Gregg Rothermel, Mary Jean Harrold, Jeffery Ostrinand, Christie Hong, 1998. An Empirical Study of the Effects of Minimization on the Fault Detection Capabilities of Test Suites. In Proceedings of IEEE International Test Conference on Software Maintenance (ITCSM'98), Washington D.C., pp. 34-43.

[5] Gregg Rothermel, Mary Jean Harrold, Jeffery von Ronne and Christie Hong, "Empirical Studies of Test-Suite Reduction", Journal of Software Testing, Verification, and Reliability, Vol. 12, No. 4, 2002.

[6] Gregg Rothermel and Mary Jean Harrold, "A Safe, Efficient Regression Test Selection Technique", ACM Transactions on Software Eng. And Methodology, 6(2): 173-210, 1997.

[7] Gregg Rothermel and Mary Jean Harrold, "Analyzing Regression Test Selection Techniques", IEEE Transactions on Software Engineering, 22(8):529-551, 1996.

[8] Sara Sprenkle, Sree devi Sampath and Amie Souter, "An Empirical Comparison of Test Suite Reduction Techniques for User-session-based Testing of Web Applications", Journal of Software. Testing, Verification, and Reliability, 4(2), 2002.

[9] Scott McMaster and Atif Memon, 2005. Call Stack Coverage for Test Suite Reduction. In proceedings of the 21st IEEE International Conference on Software

Maintenance (ICSM'05), pages 539-548, Budapest, Hungary.

[10] Scott McMaster and Atif Memon, 2006. Call Stack Coverage for GUI Test-Suite Reduction. In proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006), NC, USA.

[11] Scott McMaster and Atif Memon, Fault Detection Probability Analysis for Coverage-Based Test Suite Reduction, IEEE, 2007.

[12] Siripong Roongruangsuwan and Jirapun Daengdej, Test Case Reduction, Technical Report 25521. Assumption University, Thailand, 2009.

[13] Xiaofang Zhang, BaowenXu, Changhai Nie and LiangShi, "An Approach for Optimizing Test Suite Based on Testing Requirement Reduction", Journal of Software (in Chinese), 18(4): 821-831, 2007.

[14] Xiao fang Zhang, Baowen Xu, Changhai Nie and Liang Shi, "Test Suite Optimization Based on Testing Requirements Reduction", International Journal of Electronics & Computer Science, 7(1): 9-15, 2005.

[15] Barry W. Boehm, A Spiral Model of Software Development and Enhancement, TRW Defence Systems Group, 1998.

[16] Jirapun Daengdej, Ph.D. Thesis, Adaptable Case Base Reasoning Techniques for Dealing with Highly Noise Cases, The University of New England, Australia, 1998.

[17] Scott McMaster and Atif Memon, Fault Detection Probability Analysis for Coverage-Based Test Suite Reduction, IEEE, 2007.

[18] Zeina Chedrawy, Syed Sibte, Raza Abidi, 2005. An Intelligent Knowledge Sharing Strategy Featuring Item-Based Collaborative Filtering and Case Based Reasoning, 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), pp.67-72.

[19] A. Aamodt, E. Plaza, CBR: foundational issues, methodological variations and system approaches, AI Communications, vol. 7, no. 1, pp. 39-59, 1994.

[20] A. Cordier, B. Fuchs, J. Lieber, A. Mille, Acquisition interactive des connaissances d'adaptation intégrée aux sessions de raisonnement à partir de cas - Principes, architecture IAKA et prototype KAYAK, In: Actes du 15ème atelier de Raisonnement à Partir de Cas (RàPC 2007), pp. 71–84, 2007.

[21] F. Gavin, S. Zhaohao, 2003. R5 model for case-based reasoning. Knowledge-Based Systems, vol. 16, no. 1, pp. 59–65.

[22] A. Bouhana, A. Fekih, et al., An integrated case-based reasoning approach for personalized itinerary search in multimodal transportation systems, Transportation Research Part C, vol. 31, pp. 30–50, 2013.

[23] E. Armengol, A. Palaudàries, E. Plaza, Individual Prognosis of Diabetes Long-term Risks: A CBR Approach, Methods Inf Med, vol. 40, no. 1, pp. 46-51, 2001.

[24] I. Watson, Applying Case-Based Reasoning: Techniques for Enterprise Systems. San Francisco, CA: Morgan Kaufmann Inc. 1997.

[25] S. Nitsuwat, W. Paoin, Development of ICD-10-TM Ontology for a Semi-automated Morbidity Coding System in Thailand,‖ Methods Inf. Med, vol. 51, no. 6, pp. 519-528, 2012.

[26] S. Simon, P. Sankar, 2004. Foundations of Soft Case-Based Reasoning, 1st ed. Wiley-Interscience.

[27] Roongruangsuwan, S., Daengdej, J., 2010. Test case reduction methods by using CBR. International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010).

[28] Erum Ashraf, Tamim Ahmed Khan, Khurrum Mahmood and Shaftab Ahmed, "Value based PSO Test Case Prioritization Algorithm". International Journal of Advanced Computer Science and Applications, vol. 8, No. 1, 2017.

[29] Sonal Gandhi, Deepali Gupta, "Test Case Reduction & Prioritization". International Journal for Scientific Research & Development, vol. 2, issue 6, 2014.