CHAPTER 6

PERFORMANCE ANALYSIS

6.1 INTRODUCTION

Usually the goal of any development process is to satisfy requirements and to start performance analysis as early as possible on the available software artifacts, possibly supported by suitable models or metrics. The aim of this chapter is to analyze the performance of the developed integrated test case prioritization system.

In the performance analysis process of this integrated system for test case prioritization, the metrics APFD, APSC, APBC APPC and APFC and APFD are used.

Using these metrics

- The effectiveness of the system before and after prioritization is analyzed.
- The effectiveness of different levels of project is analyzed.
- And also the integrated system’s effectiveness is compared with existing approaches.
6.2 PRIORITIZED TEST SUITE EFFECTIVENESS

The designed integrated multi objective test case prioritization system is tested with 10 different projects. The results obtained from two projects are used for performance analysis and is discussed here.

The benefits of an integrated multi objective test case prioritization system are measured using a weighted average of the percentage of faults detected (APFD) (Elbaum et al 2001), average of the percentage of statement coverage (APSC) (Zheng Li et al 2007), average of the percentage of branch or decision coverage (APBC) (Zheng Li et al 2007), average of the percentage of path coverage (APPC) (Zheng Li et al 2007), average of the percentage of function coverage (APFC) (Zheng Li et al 2007) and average percentage of faults detected based on cost metric (Elbaum et al 2001) for a given test suite.

The APFD value is a measure that shows how quickly faults are identified for a given test suite, the APSC is a measure of rate of coverage of statements for a given test suite, the APBC is a measure of rate of coverage of branches or decision for a given test suite, the APPC is a measure of rate of coverage of paths for a given test suite, the APFC is a measure of rate of coverage of functions for a given test suite and APFDc is a measure of average percentage of faults detected based on cost metric for a given test suite.

6.2.1 Average Percentage of Faults Detected (APFD)

In APFD metric (Elbaum et al 2001), the test cases are ordered based on the test optimization indices and the test suite’s performance are analyzed. The efficacy of the ordering of the test suite is evaluated in order to measure the performance of the optimization method used in this research.
Efficacy is measured by the rate of faults detected. The following metrics is utilized to compute the level of efficacy.

By using the weighted average of the number of faults identified during the execution of the test suite, the APFD is computed. Let \( T \) be the test suite under evaluation, \( F \) is the number of faults contained in the program under test \( P \), \( n \) is the total number of test cases, and \( \text{reveal}(i, T) \) is the position of the first test in \( T \), which reveals the fault \( i \). The formula for calculating the APFD metric is given below in Equation (6.1).

\[
\text{APFD}(T, P) = \left[ 1 - \frac{\sum_{i=1}^{F} \text{reveal}(i, T)}{nF} \right] + \frac{1}{2n}
\]  

(6.1)

**Sample Calculation:** The APFD metric is calculated as follows. It is explained with the results obtained from a simple mobile project with the number of test cases \( n=10 \) and the number of faults \( F=6 \). This can be represented in the following Table 6.1.

<table>
<thead>
<tr>
<th>Faults</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>x</td>
<td>X</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>X</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using Table 6.1, the fault detection performance of each test case for Project 1 is illustrated as Figure 6.1.

**Figure 6.1  Fault detection performance of each test case for a simple project**

Here the number of test cases is 10, i.e., \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\}, and the number of faults occur during the regression testing is 6, i.e., \{F1,F2,F3,F4,F5,F6\}. The prioritized test suites with test sequence \{T10,T5,T9,T2,T1,31,T4,T6,T7,T8\},

Then, the APFD metric After Prioritization (AP) is

\[
Apfd(T, P) = 1 - \frac{(3+1+1+1+1+1)}{10^{*}6} \cdot \frac{1}{2^{*}10} = 0.9
\]

The APFD metric Before Prioritization (BP) is

\[
Apfd(T, P) = 1 - \frac{(1+2+4+5+5+5)}{10^{*}6} \cdot \frac{1}{2^{*}10} = 0.68
\]
Figure 6.2  APFD metric for a simple project

In Table 6.1, the fault identified during each test cases is listed. From the Table 6.1 and Figure 6.1, it is observed that the prioritized test cases identify the faults at an early stage. The APFD measure of prioritized test cases are higher than the non prioritized order for the simple project as shown in Figure 6.2. In Table 6.1, test case 10 can identify more number of faults when compared to others. Thus Test case 10 will be first executed.

From Table 6.1, before prioritization the order of test suite is \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\}. When this test suite is executed test case T1 uncovers single fault thereby executing 16.66% of the faults in the system after the execution of the first test case in the test suite. One additional fault is uncovered by T2 thereby executing 33.33% of the faults in the system after second test case of the test suite has been executed. No additional fault is uncovered by T3 and one additional fault by T4 thereby executing 50% of the faults in the system after executing the fourth test case of the test suite. Three additional faults were uncovered by T5 thereby executing 100% of the system after the execution of the fifth test case of the test suite. Figure 6.3 shows the
relationship between percent of fault detected per test case of test suite executed.

After prioritization the order of test suite $T$ is given as $T_{10}, T_5, T_9, T_2, T_1, T_3, T_4, T_6, T_7, T_8$. When this test suite is executed test case $T_{10}$ uncovers five faults thereby executing 83.33% of the faults in the system after the execution of the first test case of test suite. No additional fault is uncovered by $T_5$ followed by one additional fault is uncovered by $T_9$ thereby executing 100% of the faults in the system after the execution of third test case itself.

Figure 6.3 shows the relationship between percent of fault detected per test case executed.

![Figure 6.3 APFD for non prioritized and prioritized test suite for a simple project](image)

**Figure 6.3** APFD for non prioritized and prioritized test suite for a simple project

### 6.2.1.1 Faults detected by project1 and project2

For projects 1 and 2, the APFD metric is calculated. Since the number of statements and test cases are more, the calculations are not shown
here. Based on the sample calculations for a simple project example the APFD metric is calculated and is given as:

For Project 1, number of test cases $n=214$ and the number of faults $F=58$.

- APFD after prioritization = 0.92
- APFD before prioritization = 0.78

For Project 2, number of test cases $n=228$ and the number of faults $F=89$.

- APFD after prioritization = 0.93
- PFD before prioritization = 0.69

![Figure 6.4 APFD metric for project1 and project2](image)

From Figure 6.4, it is clearly shown that the APFD measure of prioritized test cases are higher than the non prioritized order for both projects. Figure 6.5 shows the relationship between percent of faults covered per unit of test suite executed Project 1.
Figure 6.5  APFD for non prioritized and prioritized test suite for project1

Figure 6.6 shows the relationship between percent of fault covered per unit of test suite executed Project 2.

Figure 6.6  APFD for non prioritized and prioritized test suite for project2
From Figures 6.5 and 6.6, it is clearly seen that the prioritized test suite results in the early detection of most of the faults in the system.

6.2.2 Coverage metrics

The coverage metrics include

- Statement coverage
- Branch coverage
- Path coverage
- Function coverage

6.2.2.1 Average percentage of statements covered

Statement coverage (Zheng Li et al 2007) measures the rate at which a prioritized test suite covers the statements. Let T be a test suite containing n test cases, and let S be a set of m statements in the code. Let TS_i be the first test case in ordering of T which covers the statement i.

\[
\text{APSC} = \frac{\sum_{i=1}^{m} TS_i}{nm} + \frac{1}{2n}
\]  

(6.2)

Sample Calculation: The APSC metric is calculated using Equation (6.2) as follows. It is explained with the results obtained from a simple project with the number of test cases \( n = 10 \) and the number of statements \( m = 12 \). This can be represented in following Table 6.2.
Table 6.2 The statement coverage by the test cases in the simple project

<table>
<thead>
<tr>
<th>Statements</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
<th>S11</th>
<th>S12</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Using Table 6.2, the statement coverage performance of each test case is illustrated as Figure 6.7.
Figure 6.7 Statement coverage performance of each test case for a simple project

Here the number of test cases is 10, i.e., \( \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\} \), and the number of statements in the code is 12 i.e., \( \{S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12\} \). The prioritized test suite with test sequence \{T3,T6,T10,T2,T5,T4,T7,T9,T1,T8\} then the APSC metric after prioritization is

\[
Apsc(T,P) = \left[ 1 - \frac{(1+1+2+1+1+1+2+1+1+1+1)}{10*12} \right] + \frac{1}{2*10} = .933
\]

The APSC metric before prioritization is

\[
Apsc(T,P) = \left[ 1 - \frac{(3+2+3+6+1+3+3+2+3+3+3)}{10*12} \right] + \frac{1}{2*10} = .759
\]
Figure 6.8 APSC metric for the simple project

In Table 6.2, the statements covered during each test case execution is listed. From the Table 6.2 and Figure 6.7, it is observed that the prioritized test cases cover more number of statements at an early stage. The APSC measure of prioritized test cases are higher than the non prioritized order for the simple project as shown in Figure 6.8. In Table 6.2, test case 3 covers more number of statements when compared to others. Thus test case 3 will be first executed.

From Table 6.2, before prioritization the order of test suite is \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\}. When this test suite is executed, test case T1 covers two statements thereby executing 16.66% of the statements in the system after the execution of the 1st test case from the test suite. Two additional statements are covered by T2 thereby executing 33.33% of the statements in the system after the execution of the 2nd test case from the test suite. Seven additional statements are covered by T3 thereby executing 91.66% of the system after the 3rd test suite has executed. No additional statement is covered by T4 and T5. One additional statements were uncovered by T6 thereby executing 100% of the system after the 6th test suite has
executed. Figure 6.9 shows the relationship between percent of statement covered per unit of test suite executed.

After prioritization, the order of test suite is \{T3,T6,T10,T2,T5,T4,T7,T9,T1,T8\}. When this test suite is executed, test case T3 uncovers ten statements thereby executing 83.33\% of the statements in the system after 1st of test suite has executed. Two additional statement is uncovered by T6 thereby executing 100\% of the statements in the system after 2\textsuperscript{nd} test suite has executed.

Figure 6.9 shows the relationship between percent of statement covered per unit of test suite executed.

![Figure 6.9: APSC for non prioritized and prioritized test suite for simple project](image)

**Figure 6.9** APSC for non prioritized and prioritized test suite for simple project

### 6.2.2.1.1 Statement coverage for project1 and project2

For projects 1 and 2, the APSC metric is calculated. Since the number of statements and test cases are more, the calculations are not shown here. Based on the calculation for a simple project, the APSC metric is calculated for project1 and project2 and is given as:
For Project1, number of test cases $n=214$ and the number of statements $m=342$.

APSC after prioritization $= .912$
APSC before prioritization $= .718$

For Project2, number of test cases $n=228$ and the number of statements $m=406$.

APSC after prioritization $= .933$
APSC before prioritization $= .699$

Figure 6.10 APSC metric for project1 and project2

From Figure 6.10, it is clearly shown that the APSC measure of prioritized test cases are higher than the non prioritized order for both projects. Figure 6.11 shows the relationship between percent of statement covered per unit of test suite executed Project 1.
Figure 6.11 APSC for non prioritized and prioritized test suite for project1

Figure 6.12 shows the relationship between percent of statement covered per unit of test suite executed Project 2.

Figure 6.12 APSC for non prioritized and prioritized test suite for project2
From Figures 6.11 and 6.12, it is clearly seen that the prioritized test suite results in the early coverage of most of the statements in the system.

### 6.2.2.2 Average percentage of branches covered

APBC (Zheng Li et al. 2007) measures the rate at which a prioritized test suite covers the branches. Let $T$ be a test suite containing $n$ test cases, and let $B$ be a set of $m$ branches in the code. Let $T_{Bi}$ be the first test case in ordering of $T$ which covers the branches $i$.

$$\text{APBC} = \frac{T_{B_1} + T_{B_2} + \ldots + T_{B_m}}{nm} + \frac{1}{2n} \quad (6.3)$$

**Table 6.3 The branches covered by the test cases in the simple project**

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T8</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
Sample Calculation: The APBC metric is calculated using Equation (6.3) as follows. It is explained with the results from a simple project with number of test cases \( n=10 \) and the number of branches \( m=10 \). This can be represented in following Table 6.3. Using Table 6.3, the branches coverage performance of each test case for the simple project is illustrated as Figure 6.13.

![Simple Project](image)

**Figure 6.13 Branch coverage performance of each test case for a simple project**

Here, the number of test cases is 10, i.e., \( \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\} \), and the number of branches in the code is 10 i.e., \( \{B1,B2,B3,B4,B5,B6,B7,B8,B9,B10\} \). The prioritized test suite with test sequence \( \{T5,T8,T10,T6,T3,T1,T9,T2,T4,T7\} \) then the APBC metric after prioritization is

\[
Apbc(T, P) = \left[ 1 - \frac{1}{10^8} \left( \frac{1+1+1+1+1+1+2+2}{10^8} \right) \right] + \frac{1}{2^{*10}} = .93
\]
The APBC metric before prioritization is

\[
Apbc(T, P) = \left[ 1 - \left( \frac{1+5+3+5+2+5+1+3+4}{10*10} \right) \right] + \frac{1}{2*10} = .71
\]

**Figure 6.14 APBC metric for the simple project**

In Table 6.3, the branches covered during each test case execution are listed. From the Tables 6.3 and Figure 6.13, it is observed that the prioritized test cases cover more number of branches at an early stage. The APBC measure of prioritized test cases are higher than the non prioritized order for the simple project as shown in Figure 6.14. In Table 6.3, test case 5 covers more number of branches when compared to others. Thus test case 5 will be first executed.

From Table 6.3, it is seen that, before prioritization, the order of test suite is \( \{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\} \). When this test suite is executed test case T1 covers two branches thereby executing 20% of the branches in the system after the 1\textsuperscript{st} test suite has executed. One additional branch is covered by T2 thereby executing 30% of the branches in the system after the 2\textsuperscript{nd} test suite has executed. Two additional branch is covered by T3
thereby executing 50% of the branches in the system after the 3\textsuperscript{rd} test suite has executed and one additional branch by T4 thereby executing 60% of the branches in the system after the 4\textsuperscript{th} test suite has executed. Four additional branches were covered by T5 thereby executing 100% of the system after the 5\textsuperscript{th} test suite has executed. Figure 6.15 shows the relationship between percent of branch covered per unit of test suite executed.

After prioritization, the order of test suite is \{T5,T8,T10,T6,T3,T1,T9,T2,T4,T7\}. When this test suite is executed, test case T5 uncovers eight branches thereby executing 80% of the branches in the system after the 1\textsuperscript{st} test suite has executed. Two additional branches are uncovered by T8 thereby executing 100% of the branches in the system after the 2\textsuperscript{nd} test suite has executed.

Figure 6.15 shows the relationship between percent of branch covered per unit of test suite executed.

![Simple Project](image)

**Figure 6.15** APBC for non prioritized and prioritized test suite in the simple project
6.2.2.2.1 Branch coverage for project1 and project 2

For projects 1 and 2, the APBC metric is calculated. Since the number of branches and test cases are more, the calculations are not shown here. Based on the example the APBC metric is calculated and is given as:

For Project1, number of test cases \( n=214 \) and the number of branches \( m=25 \).

APBC after prioritization = .878

APBC before prioritization = .684

For Project2, number of test cases \( n=228 \) and the number of branches \( m=31 \).

APBC after prioritization = .891

APBC before prioritization = .653

Figure 6.16 APBC metric for project1 and project2
From Figure 6.16, it is clearly seen that the APBC measure of prioritized test cases are higher than the non prioritized order for both projects.

Figure 6.17 shows the relationship between percent of branch covered per unit of test suite executed Project 1.

![Figure 6.17 APBC for non prioritized and prioritized test suite for project1](image)

Figure 6.17 APBC for non prioritized and prioritized test suite for project1

Figure 6.18 shows the relationship between percent of branch covered per unit of test suite executed Project 2.
Figures 6.17 and 6.18, clearly shows that the prioritized test suite results in the early coverage of most of the branches in the system.

### Average percentage of paths covered

Path coverage (Zheng Li et al 2007) measures the rate at which a prioritized test suite covers the paths. Let $T$ be a test suite containing $n$ test cases, and let $P$ be a set of $m$ paths in the code. Let $T_P_i$ be the first test case in ordering of $T$ which covers the path $i$.

\[
APP C = \frac{T_{P_1} + T_{P_2} + \ldots + T_{P_m}}{nm} + \frac{1}{2n} \quad (6.4)
\]

**Sample Calculation:** the APPC metric is calculated using Equation (6.4) as follows with the results from a simple project with number of test cases $n=10$ and the number of paths $m=10$. This can be represented in following Table 6.4.
Table 6.4  The path coverage by the test cases in the simple project

<table>
<thead>
<tr>
<th>Paths</th>
<th>Test Cases</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T7</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T9</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using Table 6.4, the paths coverage performance of each test case for Project1 is illustrated as Figure 6.19.

Figure 6.19  Path coverage performance of each test case for a simple project
Here the number of test cases is 10, i.e., \( \{ T1, T2, T3, T4, T5, T6, T7, T8, T9, T10 \} \), and the number of paths in the code is 10 i.e., \( \{ P1, P2, P3, P4, P5, P6, P7, P8, P9, P10 \} \). The prioritized test suite with test sequence \( \{ T2, T9, T4, T5, T1, T3, T6, T10, T8, T7 \} \) then the APPC metric after prioritization is

\[
\text{Appc}(T, P) = \left[ 1 - \frac{(1+1+1+2+1+1+1+3+1)}{10^4} \right] + \frac{1}{2^8} = .92
\]

The APPC metric before prioritization is

\[
\text{Appc}(T, P) = \left[ 1 - \frac{(2+1+1+2+1+2+2+4+2)}{10^4} \right] + \frac{1}{2^8} = .86
\]

![Simple Project](chart.png)

**Figure 6.20** APPC metric for the simple project

In Table 6.4, the paths covered during each test case execution are listed. From the Tables 6.4 and Figure 6.19, it is observed that the prioritized test cases cover more number of paths at an early stage. The APPC measure of prioritized test cases are higher than the non prioritized order for the simple
project as shown in Figure 6.20. In Table 6.4, test case 2 covers more number of paths when compared to others. Thus test case 2 will be first executed.

From Table 6.4, it is seen that, before prioritization, the order of test suite is \{T1, T2, T3, T4, T5, T6, T7, T8, T9, T10\}. When this test suite is executed test case T1 uncovers three paths thereby executing 30% of the paths in the system after the 1\textsuperscript{st} test suite has executed. Six additional paths is uncovered by T2 thereby executing 90% of the paths in the system after the 2\textsuperscript{nd} test suite has executed. No additional path is uncovered by T3 and one additional path is uncovered by T4 thereby executing 100% of the paths in the system after the 4\textsuperscript{th} test suite has executed. Figure 6.21 shows the relationship between percent of path covered per unit of test suite executed.

After prioritization, the order of test suite is \{T2, T9, T4, T5, T1, T3, T6, T10, T8, T7\}. When this test suite is executed, test case T2 uncovers eight paths thereby executing 80% of the paths in the system after the 1\textsuperscript{st} test suite has executed. Two additional path is uncovered by T9 thereby executing 100% of the paths in the system after the 2\textsuperscript{nd} test suite has executed.

Figure 6.21 shows the relationship between percent of path covered per unit of test suite executed.
Figure 6.21  APPC for non prioritized and prioritized test suite for the simple project

6.2.2.3.1 Path coverage  for project1 and project2

For projects 1 and 2, the APPC metric is calculated. Since the number of paths and test cases are more, the calculations are not shown here. Based on the example the APPC metric is calculated and is given as:

For Project1, number of test cases \( n=214 \) and the number of paths \( m=55 \).

\[
\text{APPC after prioritization} = .893 \\
\text{APPC before prioritization} = .784
\]

For Project2, number of test cases \( n=228 \) and the number of paths \( m=67 \).

\[
\text{APPC after prioritization} = .878 \\
\text{APPC before prioritization} = .653
\]
Figure 6.22 APPC metric for project1 and project2

From Figure 6.22, it is clearly seen that the APPC measure of prioritized test cases are higher than the non prioritized order for both projects.

Figure 6.23 shows the relationship between percent of path covered per unit of test suite executed for Project1.

Figure 6.23 APPC for non prioritized and prioritized test suite for project1
Figure 6.24 shows the relationship between percent of path covered per unit of test suite executed Project2.

![Graph showing APPC for non prioritized and prioritized test suite for project2](image)

**Figure 6.24** APPC for non prioritized and prioritized test suite for project2

From Figures 6.23 and 6.24, it is clearly seen that the prioritized test suite results in the early coverage of most of the paths in the system.

### 6.2.2.4 Average percentage of function covered

Function coverage (Zheng Li et al 2007) measures the rate at which a prioritized test suite covers the functions. Let $T$ be a test suite containing $n$ test cases, and let $F$ be a set of $m$ functions in the code. Let $TFN_i$ be the first test case in ordering of $T$ which covers the function $i$.

\[
APFC = \frac{TFN_1 + TFN_2 + \ldots + TFN_m}{nm} + \frac{1}{2n}
\]  

(6.5)
Table 6.5 The function covered by the test cases in the simple project

<table>
<thead>
<tr>
<th>Functions</th>
<th>FN1</th>
<th>FN2</th>
<th>FN3</th>
<th>FN4</th>
<th>FN5</th>
<th>FN6</th>
<th>FN7</th>
<th>FN8</th>
<th>FN9</th>
<th>FN10</th>
<th>FN11</th>
<th>FN12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sample Calculation: The APFC metric is calculated using Equation (6.5) as follows for the results from a simple project. Here the number of test cases $n=10$ and the number of functions $m=12$. This can be represented in following Table 6.5.

Using Table 6.5, the functions coverage performance of each test case for the simple project is illustrated as Figure 6.25.
Figure 6.25 Function coverage performance of each test case for a simple project

Here the number of test cases is 10, i.e., \( \{T1, T2, T3, T4, T5, T6, T7, T8, T9, T10\} \), and the number of functions in the code is 12 i.e., \( \{F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12\} \). The prioritized test suite with test sequence \( \{T3, T4, T6, T8, T5, T10, T1, T7, T9, T2\} \) then the APFC metric after prioritization is

\[
\text{Apfc} (T, P) = \left[1 - \frac{(1+2+1+2+1+1+2+1+1+1)}{10^{*}12}\right] + \frac{1}{2^{*10}} = .925
\]

The APFC metric before prioritization is

\[
\text{Apfc} (T, P) = \left[1 - \frac{(1+4+2+3+4+3+1+3+1+3+3+3)}{10^{*}12}\right] + \frac{1}{2^{*10}} = .792
\]
In Table 6.5, the functions covered during each test case execution is listed. From the Table 6.5, and Figure 6.25, it is observed that the prioritized test cases cover more number of functions at an early stage. The APFC measure of prioritized test cases are higher than the non prioritized order for the simple project as shown in Figure 6.26. In Table 6.5, test case 3 covers more number of functions when compared to others. Thus test case 3 will be first executed.

From Table 6.5, it is seen that, before prioritization the order of test suite is $\{T1,T2,T3,T4,T5,T6,T7,T8,T9,T10\}$. When this test suite is executed, test case T1 uncovers three functions thereby executing 25% of the functions in the system after the 1st test suite has executed. One additional function is uncovered by T2 thereby executing 33% of the functions in the system after the 2nd test suite has executed. Six additional functions are uncovered by T3 thereby executing 83% of the functions in the system after the 3rd test suite has executed. Two additional functions are covered by T4 thereby executing 100% of the functions in the system after the 4th test suite has executed. Figure 6.27 shows the relationship between percent of function covered per unit of test suite executed.
After prioritization, the order of test suite is \{T3,T4,T6,T8,T5,T10,T1,T7,T9,T2\}. When this test suite is executed, test case T3 uncovers nine functions thereby executing 75% of the functions in the system after the 1\textsuperscript{st} test suite has executed. Three additional functions are uncovered by T4 thereby executing 100% of the functions in the system after the 2\textsuperscript{nd} test suite has executed.

Figure 6.27 shows the relationship between the percent of function covered per unit of test suite executed.

![Figure 6.27 APFC for non prioritized and prioritized test suite for the simple project](image)

6.2.2.4.1 Function coverage for project1 and project2

For projects 1 and 2, the APFC metric is calculated. Since the number of functions and test cases are more, the calculations are not shown here. Based on the example the APFC metric is calculated and is given as:

For Project1, number of test cases \( n=214 \) and the number of functions \( m=15 \).
APFC after prioritization $= 0.923$

APFC before prioritization $= 0.854$

For Project2, number of test cases $n=228$ and the number of functions $m=23$.

APFC after prioritization $= 0.917$

APFC before prioritization $= 0.755$

**Figure 6.28 APFC metric for project1 and project2**

From Figure 6.28, it is clearly shown that the APFC measure of prioritized test cases are higher than the non prioritized order for both projects.

Figure 6.29 shows the relationship between percent of function covered per unit of test suite executed project1.
Figure 6.29 APFC for non prioritized and prioritized test suite for project1

Figure 6.30 shows the relationship between percent of function covered per unit of test suite executed for Project2.

Figure 6.30 APFC for non prioritized and prioritized test suite for project2
From Figures 6.29 and 6.30, it is clearly shown that the prioritized test suite results in the early coverage of most of the functions in the system.

### 6.2.3 Average Percentage of Faults Detected based on Cost Metric (APFDc)

In **Average Percentage of Faults Detected based on cost**. The cost of a test case is related to the resources required to execute and validate it. Various measures are possible like:

- When the primary required resource is machine or human time, test cost can be measured in terms of the actual time needed to execute a test case.

- Another measure considers the monetary costs of test case execution and validation; this may reflect hardware cost, wages, cost of materials required for testing, earnings lost due to delays in failing to meet target release dates, and so on.

Here, the cost is estimated using execution time. For calculating the time two variables startTime and endTime are used. Before executing the test case, the current time is stored in a variable startTime. After execution the current time is stored in endTime. The execution time is calculated as the difference between endTime and startTime.

In APFDc metric, (Elbaum et al 2002) the test cases are ordered based on the execution time indices and a test suite’s performance are analyzed.
Let $e_1, e_2, \ldots, e_n$ be the execution time (cost) of $n$ test cases in a test suite $T$. Let $F$ be a set of $m$ faults revealed by $T$. $\text{APFD}_c$ is calculated using the Equation (6.6).

$$\text{APFD}_c (T, P) = \left[ 1 - \frac{\sum_{i=1}^{n} \text{reveal}(i, T)}{nf} \right] + \frac{1}{2n}$$  

(6.6)

Where $F = \{f_1, f_2, \ldots, f_m\}$.

$f_1, f_2, \ldots, f_m$ be the severities of $m$ faults.

reveal$(i,T)$ is the first test case that reveals the fault.

Hence the Equation (6.6) can be rewritten as Equation (6.7)

$$\text{APFD}_c (T, P) = \frac{\sum_{i=1}^{m} \left( f_i \left( \sum_{j=1}^{n} t_j - \frac{1}{2} t_{\text{reveal}(i, T)} \right) \right)}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} i = \text{mf}_i}$$  

(6.7)

**Sample Calculation**: $\text{APFD}_c$ metric is calculated using Equation (6.7) as follows with the results from a simple project with number of test cases $n=10$, the number of faults $m=10$ and $e_i$ denotes the execution time of test case $i$. This can be represented in following Tables 6.6 and 6.7.

Initially, the test cases are prioritized based on the execution time. The test suite before prioritization is given as \{T1, T2, T3, T4, T5, T6, T7, T8, T9, T10\} and the test suite $T'$ after prioritization is given as \{T7, T3, T4, T2, T1, T10, T5, T9, T6, T8\}. Using the result $T'$ the prioritization is done again based on the fault severities of the test cases which are represented in Table 6.7.
Table 6.6 The cost (execution time) of each test case

<table>
<thead>
<tr>
<th>Test cases</th>
<th>Cost (Execution time in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0.456</td>
</tr>
<tr>
<td>T2</td>
<td>0.345</td>
</tr>
<tr>
<td>T3</td>
<td>0.213</td>
</tr>
<tr>
<td>T4</td>
<td>0.267</td>
</tr>
<tr>
<td>T5</td>
<td>0.67</td>
</tr>
<tr>
<td>T6</td>
<td>0.87</td>
</tr>
<tr>
<td>T7</td>
<td>0.17</td>
</tr>
<tr>
<td>T8</td>
<td>0.99</td>
</tr>
<tr>
<td>T9</td>
<td>0.74</td>
</tr>
<tr>
<td>T10</td>
<td>0.631</td>
</tr>
</tbody>
</table>

Table 6.7 Fault severities of test cases

<table>
<thead>
<tr>
<th>Faults</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T9</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>T8</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Here, the faults F1, F2, F3, F7 and F9 which are represented in red colour are as considered as critical faults, the faults F5, F6 and F10 which are represented in blue colour are as considered as severe faults and the faults F4 and F8 which are represented in black colour are as considered as moderate faults.

Again the test cases are prioritized based on the fault severities. The test suite \( T \) before prioritization is given as \{T7, T3, T4, T2, T1, T10, T5, T9, T6, T8\} and the test suite after prioritization is given as \{T7, T3, T1, T4, T2, T10, T8, T5, T9, T6\}. Then using the Equation (6.7), \( \text{APFD}_c \) is calculated and is given as:

\[
\begin{align*}
\text{APFD}_c \text{ after prioritization} & = .83 \\
\text{APFD}_c \text{ before prioritization} & = .74
\end{align*}
\]

### 6.2.3.1 Faults detected based on cost metric by project1 and project2

For projects 1 and 2, the \( \text{APFD}_c \) metric is calculated. Since the number of faults and test cases are more, the calculations are not shown here. Based on the simple project, the \( \text{APFD}_c \) metric is calculated and is given as:

For Project1, number of test cases \( n=214 \) and the number of faults \( m=123 \) with 3 levels of severity.

\[
\begin{align*}
\text{APFD}_c \text{ after Prioritization} & = .70 \\
\text{APFD}_c \text{ before Prioritization} & = .67
\end{align*}
\]

For Project2, number of test cases \( n=228 \) and the number of functions \( m=187 \) with 3 levels of severity.
APFD\textsubscript{c} after Prioritization  $= .75$

APFD\textsubscript{c} before Prioritization  $= .65$

Figure 6.31 gives the APFD\textsubscript{c} metric for project 1 and project 2 and it is observed that the proposed reduces the computational cost.

6.3 PERFORMANCE ANALYSIS OF INTEGRATED SYSTEM USING SAMPLE PROJECTS

The performance of the integrated multi objective test case prioritization system is analyzed with 10 different projects. Just to prove the effectiveness of the developed system two case studies are discussed. For initial analysis and calculation of metrics, a simple mobile project in java was used. Other java projects include student evaluation system, reservation system, banking application, etc. Here the results from two sample projects are used for analysis.

Sample project refers to student evaluation system and banking application. The student evaluation system used in a college is taken. The
college has 943 students with their internal assessment marks and university examination results are considered. The banking application has 1023 customers with their transactions like deposit, withdrawal and balance statement is considered.

For both the projects, the test suite is initially given as input to feature selection module. The reduced test suite from feature selection is given to birds flocking and the optimized test suite from birds flocking is given to genetic algorithm for prioritization.

The input and output to each technique is compared and analyzed as shown in Figure 6.32 for Project1.

![Figure 6.32 Performance analysis of integrated system for project1](image)

From Figure 6.32, it is clearly seen that initially 943 test cases are given as input to the first technique feature selection. After reduction by feature selection, 617 test cases are given as output from feature selection which is given as input to next technique birds flocking. Then, after optimization 323 test cases are given as output from birds flocking which is given as the input to the third technique genetic algorithm. Then, after prioritization only 41 test cases are prioritized by genetic algorithm as the fittest test cases.
The input and output to each technique is compared and analyzed as shown in Figure 6.33 for Project2.

![Figure 6.33 Performance analysis of integrated system for project2](image)

From Figure 6.33, it is clearly seen initially 1023 test cases are given as input to the first technique feature selection. After reduction by feature selection, 816 test cases are given as output from feature selection which is given as input to next technique birds flocking. Then, after optimization 435 test cases are given as output from birds flocking which is given as the input to the third technique genetic algorithm. Then, after prioritization only 112 test cases are prioritized by genetic algorithm as the fittest test cases.

### 6.4 EFFECTIVENESS COMPARISON BETWEEN INTEGRATED SYSTEM AND EXISTING APPROACHES

The effectiveness of prioritization of the developed integrated system is compared with the existing approaches. Here the same projects Project1 and Project2 are given as input to existing approaches like Ant Colony algorithm, Genetic Algorithm and Hybrid method which combines...
PSO and Genetic Algorithm are compared with the developed integrated system.

The comparison is made by using 2 metrics APFD and $\text{APFD}_c$.

### 6.4.1 Comparison of Integrated System with Existing Approaches using APFD Metric

The Effectiveness of the prioritization of integrated system compared with existing approaches using APFD metric comparison is made for two projects project1 and project2. The Average Percentage of Fault-Detection (APFD) metric that is widely used today for evaluating the effectiveness of Test Case Prioritization Techniques. So, we use standard metrics like APFD, APSC, etc. to evaluate the performance of the other approaches like ant colony algorithm, genetic method and hybrid method which was also already evaluated by these metrics.

**Project1:** A Comparison of the integrated system with existing approaches for project1 is done using APFD metric. The results are shown in Table 6.8.

**Table 6.8 Comparison of the integrated system with existing approaches for project1 using APFD metric**

<table>
<thead>
<tr>
<th>Method</th>
<th>Effectiveness in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant Colony Algorithm</td>
<td>89</td>
</tr>
<tr>
<td>Genetic Method</td>
<td>88.2</td>
</tr>
<tr>
<td>Hybrid Method</td>
<td>90.1</td>
</tr>
<tr>
<td>Integrated System</td>
<td>92</td>
</tr>
</tbody>
</table>
From Table 6.8, it is clearly seen that the effectiveness of integrated multi objective test case prioritization is higher than the existing approaches that are taken for analysis.

**Project2** : A Comparison of the integrated system with existing approaches for project2 is done using APFD metric. The results are shown in Table 6.9.

**Table 6.9 Comparison of the integrated system with existing approaches for project2 using APFD metric**

<table>
<thead>
<tr>
<th>Method</th>
<th>Effectiveness in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant Colony Algorithm</td>
<td>86.9</td>
</tr>
<tr>
<td>Genetic Method</td>
<td>88</td>
</tr>
<tr>
<td>Hybrid Method</td>
<td>91.6</td>
</tr>
<tr>
<td>Integrated System</td>
<td>93</td>
</tr>
</tbody>
</table>

From Table 6.9, it is clearly seen that the effectiveness of integrated multi objective test case prioritization is higher than the existing approaches that are taken for analysis. The effectiveness of the projects are analyzed and compared with the existing system is shown in Figure 6.34.
6.4.2 Comparison of the Integrated System with Existing Approaches using APFD\(_e\) Metric

The Effectiveness of the prioritization of integrated system compared with existing approaches using APFD\(_e\) metric comparison is made for two projects project1 and project2.

**Project1:** A Comparison of existing approaches with the integrated system for project1 using APFD\(_e\) metric is done. The results are shown in Table 6.10.

Table 6.10 Comparison of the integrated system with existing approaches for project1 using APFD\(_e\) metric

<table>
<thead>
<tr>
<th>Method</th>
<th>Effectiveness in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant Colony Algorithm</td>
<td>69</td>
</tr>
<tr>
<td>Genetic Method</td>
<td>68.2</td>
</tr>
<tr>
<td>Hybrid Method</td>
<td>69.1</td>
</tr>
<tr>
<td>Integrated System</td>
<td>70</td>
</tr>
</tbody>
</table>
From Table 6.10, it is clearly seen that the effectiveness of integrated multi objective test case prioritization is higher than the existing approaches that are taken for analysis.

**Project2**: A Comparison of existing approaches with the integrated system for project2 using APFDₐ metric is done. The results are shown in Table 6.11.

**Table 6.11 Comparison of the integrated system with existing approaches for project2 using APFDₐ metric**

<table>
<thead>
<tr>
<th>Method</th>
<th>Effectiveness in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant Colony Algorithm</td>
<td>69</td>
</tr>
<tr>
<td>Genetic Method</td>
<td>68</td>
</tr>
<tr>
<td>Hybrid Method</td>
<td>73.9</td>
</tr>
<tr>
<td>Integrated System</td>
<td>75</td>
</tr>
</tbody>
</table>

From Table 6.11 it is clearly seen that the effectiveness of integrated multi objective test case prioritization is higher than the existing approaches that are taken for analysis.

The effectiveness of the projects are analyzed and compared with the existing system is shown in Figure 6.35.
The same projects are used to all approaches and the APFD metric and APFD$_c$ metric obtained are compared. And from the results obtained it is clearly seen that the integrated system has more effectiveness than the existing examined approaches.

6.5 SUMMARY

Since project 1 has 923 test cases and project 2 has 1023 test cases, the inference from the graphs are discussed for a simple project with 10 test cases. The main reason for higher performance is the integrated system. The current state of art case study mobile application is considered to evaluate the performance of the tool. The performance of the integrated multi objective test case prioritization is analyzed using APFD, APSC, APBC, APPC, APFC and APFD$_c$ metrics. The performance of the integrated multi objective test case prioritization system is also compared with some of already existing approaches. It is clearly seen from the results of the integrated system increases the effectiveness of the system as the APFD, APSC, APBC, APPC, APFC and APFD$_c$ metrics values shows an increase after prioritization.

Next chapter presents the highlights of the work done, summary, conclusion and the directions for the further research.