CHAPTER 5

COMPONENT BASED SOFTWARE TESTING TECHNIQUES AND

METHOD FOR TEST CASE GENERATION

5.1 Introduction

Testing is one of the core activities of software development process. Since Component-Based Software engineering relies on the concept of “use of pre-built and pre-tested components”, focus of developers is on Black-box testing (functional testing) as well as White-box testing (structural testing). Black-box emphasizes the behavioural attributes of the components when they interact with each other. White-box testing techniques are used to address the testing of the structural design and internal code of the software. In this work, functional testing and structural testing strategy and test case generation techniques for CBSE are suggested. When two components are integrated then they generate some specific effect. This strategy is named Integration-Effect graph. This strategy is a Black-Box technique as it covers the input and output domains only. Proposed method is compared only with Boundary-Value Analysis method since other Black-Box techniques require specific input conditions and number of test cases depend on those conditions. Further White-box testing technique named Cyclomatic complexity has been described.

5.2 Testing Techniques

For the testing purpose, software constructs are divided into two broad categories: Input/Output constructs and Process/Logic/Code constructs. Input/Output constructs refer to inputs provided to the software and outputs produced by the software. Normally input/output is presented in the form of data, information and specific values. Process/Logic/Code constructs refer to the actual processing of software. It involves the coding and internal structure of the software. On the basis of type of construct, testing techniques are classified into two major classes: Black-Box testing and White-Box testing.

5.2.1 Black-Box Testing

Black-Box testing techniques consider only inputs and outputs of the software. These techniques are applicable to that software whose code is not available or accessible. They
divide inputs and output domains in various classes or partitions and test them separately. In the literature various categories of Black-Box testing techniques are defined including Boundary-Value Analysis, Equivalence Class partitioning, Decision Table-Based and Cause-Effect Graphing. The detailed description has been done of this technique in chapter 2.

5.2.2 White-Box Testing
White-Box testing techniques consider the internal logic of the software. They are applicable to the structural code of the software. In this technique the program statements are checked and errors are fixed. The detailed description has been done of this technique in chapter 2.

5.3 Proposed Test Case Generation Technique for Black-Box Components
In this work, a test case generation technique for Black-Box components is developed. The fundamental behind this technique is that when two components are integrated then they generate some specific effect, and hence it is named ‘Integration-Effect graph’. This technique considers the interacting arguments of the components during their integration. Using Integration-Effect graph, we develop an ‘Integration-Effect matrix’ to count the number of test cases in the context of CBSE.

5.3.1 Flow Graph Notations
To draw the flow graph diagram we use graph theory notations [149]. A component can make a request for some service to another component, which is shown as request edge, and the component can respond through the response edge as shown in Figure 5.1. In coding, these interactions take place through some arguments or operands. The calling component passes some argument(s) to the called component and in response the called component will return some argument(s) to the calling one. This scenario is shown in the Figure 5.1.

![Figure 5.1 Interaction between Two Components](image-url)
Figure 5.1 is similar as the Figure 4.1 in Chapter 4 having only the difference of Requesting and Responding arguments. This figure is used as the base case of proposed technique.

5.3.2 Components Integration-Effect Graph

Components Integration-Effect Graph strategy is suggested for software where more than one components are involved to provide and access their services as per the specification and architecture document [39]. When two or more than two components are assembled they must share some sort of information to each other through well defined interfaces [40], [41].

In the CBSE, we have some third party components for which the code is not available. So it is difficult to apply rigorous testing, structural testing or exhaustive testing on such software. Therefore, Black-Box strategy is quite suitable for such software. To represent the integration of components, at least two components are required. Figure 5.1 shows the base case of the proposed technique. Components are interacting with each other through request and response edges. Request and response edges correspond to interacting and returning arguments.

5.3.3 Method of Generating Test Cases

Usually on the basis of the architectural design of the software, we draw a control flow graph to show the interaction among the components, as shown in Figure 5.1. But in Integration-Effect graph, we introduced the effect of their integration via an interface. Integration-Effect graph based on the control flow graph is shown in Figure 5.2.

With the help of this Integration-Effect graph we draw an Integration-Effect Matrix as discussed in Table 5.1. This matrix contains two types of values:

- Integration values among components, and
- Effect values, generated due to the integration of different components.

Integration-Effect values among components can be computed as:

\[ \text{Effects of Individual Components} \land \text{Effects generated due to the Integration of Components} \]

where, these Effects are the Boolean values, either 0 or 1.

On the basis of these values and the above defined formula, we can draw an Integration-Effect matrix. From the Integration-Effect matrix, we can define the number of
test cases in two steps:

**Step1.** Number of ‘1s’ under ‘Effect’ column in each row will specify the number of test cases for corresponding component.

**Step2.** Calculate the total number of test cases achieved in step 1. This specifies the total number of test cases for the given software.

In this chapter, we have considered four scenarios along with various case studies i.e., scenario 1 has case study 1, scenario 2 has case study 2, scenario 3 has case study 3 and scenario 4 has case study 4, to analyse Integration-Effect graph.

**Scenario 1: When we have two components**

This scenario is shown in Figure 5.1; here we have two components C1 and C2. These components can communicate through some interaction and returning arguments. This is shown as an edge from C1 to C2 and C2 to C1. To integrate C1 and C2, we need an interface. This interface will be compatible with C1 and C2.

**a) Integration-Effect Graph for two components**

When we integrate these components, they will generate some effect. If the effect is in specified way, we say that components are working as per the requirement. But if the effect is not as per our intention, we need to test these components. We draw an Integration-Effect graph to show the integration among components and the effects of these integrations as shown in Figure 5.2.

![Figure 5.2 Integration-Effect Graph for Two Components](image-url)

We represent the component Integration-Effect graph as:
The output of integration of component C1 and C2 is 1 (without any error) if, the effect of component C1 (denoted as Eff(C1)) is 1, effect of component C2 (Eff(C2)) is 1 and the integration effect of C1 and C2 is error free. To generate the true Integration-Effect of component C1 and C2, we have to take into account the individual as well as combined effects generated due to component C1 and component C2 and the integrated effects of C1 and C2. That is:

$$\text{Int}(C1 \land C2) = \text{Eff}(C1) \land \text{Eff}(C2) \land \text{Eff}(C1 \land C2)$$

(5.1)

where, $\text{Int}(C1 \land C2)$ represents integration of components C1 and C2, $\text{Eff}(C1)$ represents the effect generated due to component C1, $\text{Eff}(C2)$ represents the effect generated due to component C2, $\text{Eff}(C1 \land C2)$ represents the effect generated due to the integration of the components C1 and C2, and $\land$ denotes the AND operation and $\lor$ denotes the OR operation. If the effect is true that is without any error it is denoted as 1, and if it is negative that is, it having error, it is denoted as 0. For $\text{Int}(C1 \land C2)$ to be true, $\text{Eff}(C1)$, $\text{Eff}(C2)$, and $\text{Eff}(C1 \land C2)$ all must be true, that is, 1. That is, component C1, C2 and integration of C1 and C2 must be error free.

b) Integration-Effect Matrix for two components

Integration-Effect matrix is a row-column matrix. If two components are connected through an edge, then they are represented as 1, otherwise 0. Component C1 is connected with component C2, so the value in the matrix C1 to C2 is 1. Each component is connected to itself by the property of cohesion, so we have used 1 to show their connectivity. Component C1 is connected to itself; therefore C1 to C1 has value 1. If the component C1 is error free, the value of Eff(C1) is 1, otherwise it is 0. If the component C2 is error free the value of Eff(C2) is 1, otherwise it is 0. If the integration of component C1 and C2 is error free the value of Eff(C1 $\land$ C2) is 1, otherwise it is 0.

Possible values of Integration-Effect matrix for two components are given in Table 5.1:

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>Eff(C1): 0/1</td>
<td>0/1</td>
<td>Eff(C1$\land$C2): 0/1</td>
</tr>
<tr>
<td>C2</td>
<td>0/1</td>
<td>Eff(C2$\land$C1): 0/1</td>
<td>1</td>
<td>Eff(C2): 0/1</td>
</tr>
</tbody>
</table>
i. Integration-Effect matrix when no error in integration

Table 5.2 shows the Integration-Effect matrix for the case where all the integrated components are error free and their Integration-Effects are also error free.

Table 5.2 Actual values of Integration-Effect Matrix for Two Components with no error

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Row C1 has all the values as 1 which represents that C1 is connected with every other component of the graph and the integration effect of C1 with every other component is error free.
Row C2 has all the values as 1 which represents that C2 is connected with every other component of the graph and the integration effect of C1 with every other component is error free.

ii. Integration-Effect matrix when there is an error in integration

Table 5.3 shows the Integration-Effect matrix for the case where the integrated components have errors and their Integration-Effects also contain errors.

Table 5.3 Actual values of Integration-Effect Matrix for Two Components with errors

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Row C1 has all the values as 1 which represents that C1 is connected with every other component of the graph.
In row C2, the corresponding value of C1 is 1, means there is an interaction between C2 and C1, but their Integration-Effect value is 0, means there is an error in their interaction. And this integration needs retesting (Table 5.3).
Table 5.4 Integration-Effect Matrix for Two Components

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>![1]</td>
<td>1</td>
<td>![1]</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>![1]</td>
<td>1</td>
<td>![1]</td>
</tr>
</tbody>
</table>

**Case Study 1:**

This case study is based on scenario 1, which signifies the number of test cases derived from the Integration-Effect Matrix shown in Table 5.4:

**Case 1:** Number of test cases where C1 is involved

\[
\text{Number of test cases where C1 is involved} = (\text{Number of 1s under “Effect” column in row C1}) - 1
\]

\[
= 2 - 1 = 1
\]

**Case 2:** Number of test cases where C2 is involved

\[
\text{Number of test cases where C2 is involved} = (\text{Number of 1s under “Effect” column in row C2}) - 1
\]

\[
= 2 - 1 = 1
\]

Total number of test cases derived from the proposed method:

\[
\text{Total number of test cases} = \text{Number of test cases for C1} + \text{Number of Test cases for C2}
\]

\[
= 1 + 1 = 2.
\]

**Number of test cases through Boundary Value Analysis method:**

Assuming that ‘n’ is the number of components then, the minimum number of test cases are \(4n+1\).

If we have 2 components, then \(n = 2\).

Therefore, Number of test cases = \(4n + 1\)

\[
= 4 \times 2 + 1 = 9
\]

**Scenario 2: When we have Three Components**

This case is shown in Figure 5.3, where we have three components C1, C2 and C3. These components can communicate to each other to provide and access services. Component C1 is communicating with component C2 and component C3. This is shown as an edge from C1 to C2 and C1 to C3. There is no interaction between components C2 and C3. To integrate C1 and C2, we need an interface; similarly we need interface to integrate C1 and C3.
Now to show the effects of their integration we draw the Integration-Effect graph. When we integrate these components, they will generate some effect. If the effect is true it is denoted as 1, otherwise 0. This is shown in Figure 5.4.

From the Figure 5.4, we note that atmost two components interact at a time through some interface. So during integration testing we need to focus on atmost two components at a time.
The output of integration of component C1, C2 and C3 is 1 (i.e., true or without any error) if the effect of component C1 (Eff(C1)) is 1 (without any error), effect of component C2 (Eff(C2)) is 1 (without any error), effect of component C3 (Eff(C3)) is 1 (without any error), and the integration effect of C1, C2 and C3 is error free.

To generate the true Integration-Effect of component C1, C2 and C3, we have to take into account the individual as well as combined effects generated due to component C1, C2 and component C3. That is:

\[ \text{Int}(C1 \land C2 \land C3) = \text{Eff}(C1) \land \text{Eff}(C2) \land \text{Eff}(C3) \land \text{Eff}(C1 \land C2) \land \text{Eff}(C2 \land C3) \land \text{Eff}(C1 \land C3) \]  \hspace{1cm} (5.2)

where, \( \text{Int}(C1 \land C2 \land C3) \) represents integration of components C1, C2 and C3,
\( \text{Eff}(C1) \) represents the effect generated due to component C1, effect is 1 if C1 is error free and 0 if C1 have an error. \( \text{Eff}(C2) \) represents the effect generated due to component C2, effect is 1 if C2 is error free and 0 if C2 have an error. \( \text{Eff}(C3) \) represents the effect generated due to component C3, effect is 1 if C3 is error free and 0 if C3 is having error. 
\( \text{Eff}(C1 \land C2) \) represents the effect generated due to the components C1 and C2. The integrated effect of C1 and C2 is 1 if the integration is error free, otherwise the effect is 0. 
\( \text{Eff}(C2 \land C3) \) represents the effect generated due to the components C2 and C3. The integrated effect of C2 and C3 is 1 if the integration is error free, otherwise the effect is 0. 
\( \text{Eff}(C1 \land C3) \) represents the effect generated due to the components C1 and C3.

The integrated effect of C1 and C3 is 1 if the integration is error free, otherwise the effect is 0. \( \land \) denotes the AND operation and \( \lor \) denotes the OR operation. If the effect is positive that is without any error it is denoted as 1, and if it is negative that is, the having error, it is denoted as 0.

For \( \text{Int}(C1 \land C2 \land C3) \) to be true, \( \text{Eff}(C1) \), \( \text{Eff}(C2) \), \( \text{Eff}(C3) \), and \( \text{Eff}(C1 \land C2 \land C3) \) all must be true, that is, 1.

b) Integration-Effect Matrix for three components

Possible values of Integration-Effect matrix for three components are shown in Table 5.5.
Table 5.5 Possible values of Integration-Effect Matrix for Three Components

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
<th>C3</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>Eff(C1): 0/1</td>
<td>1</td>
<td>Eff(C1 Λ C2): 0/1</td>
<td>1</td>
<td>Eff(C1 Λ C3): 0/1</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>Eff(C2 Λ C1): 0/1</td>
<td>1</td>
<td>Eff(C2): 0/1</td>
<td>0</td>
<td>Eff(C2 Λ C3): 0/1</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>Eff(C3 Λ C1): 0/1</td>
<td>0</td>
<td>Eff(C3 Λ C2): 0/1</td>
<td>1</td>
<td>Eff(C3): 0/1</td>
</tr>
</tbody>
</table>

Table 5.6 shows the Integration-Effect matrix for the above scenario (Figure 5.4):

Table 5.6 Actual values of Integration-Effect Matrix for Three Components

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
<th>C3</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Case Study 2:

This case study is based on scenario 2, which signifies the number of test cases derived from the Integration-Effect Matrix shown in Table 5.6:

Case 1: Number of test cases where C1 is involved

= (Number of 1s under “Effect” column in row C1) – 1.
= 3 - 1 = 2

Case 2: Number of test cases where C2 is involved

= (Number of 1s under “Effect” column in row C2) – 1.
= 2 – 1 = 1

Case 3: Number of test cases where C3 is involved

= (Number of 1s under “Effect” column in row C3) – 1
= 2 – 1 = 1

Total number of test cases derived from the proposed method:
Number of test cases:

\[= \text{Number of test cases for } C1 + \text{Number of test cases for } C2 + \text{Number of test cases for } C3 \]
\[= 2 + 1 + 1 = 4.\]

Number of test cases through Boundary Value Analysis method:

Assuming that ‘n’ is the number of components then, the minimum number of test cases are 4n+1. If we have 3 components, then n = 3.

Therefore, Number of test cases = 4 * n + 1 = 4 * 3 + 1 = 13

Scenario 3: When we have Four Components

This is the case when we have four components C1, C2, C3, and C4. This case is shown in Figure 5.5. Component C1 is communicating with component C2, C3 and component C4. Component C2 is integrated with C1 and C4 and C3 in integrated with C1 and C4. And finally C4 is interacting with C1, C2, and C3. Every requesting component gets a response. This is shown as an edge from C1 to C2 and C2 to C1. Similarly an edge from C1 to C3 and C3 to C1, and so on. Here outgoing edges from the components are used to show the request by the component to other components and the incoming edges to the component are used to show the response of requests by some other components.

Figure 5.5 Interaction among Four Components

a) Integration-Effect Graph for four components

Integration of these components will produce some effect. As the previous cases when these components are integrated they will generate some effect. Figure 5.6 shows the
Integration-Effect graph of four components. The output of integration of component C1, C2, C3 and C4 is 1 (i.e., true or without any error) if the effect of component C1 (Eff(C1)) is 1 (without any error), effect of component C2 (Eff(C2)) is 1 (without any error), effect of component C3 (Eff(C3)) is 1 (without any error), effect of component C4 (Eff(C4)) is 1 (without any error) and the integration effect of C1, C2, C3 and C4 is error free.

\[
\text{Int}(C1 \Lambda C2 \Lambda C3 \Lambda C4) = \text{Eff}(C1) \Lambda \text{Eff}(C2) \Lambda \text{Eff}(C3) \Lambda \text{Eff}(C4) \\
\Lambda \text{Eff}(C1 \Lambda C2) \Lambda \text{Eff}(C1 \Lambda C3) \\
\Lambda \text{Eff}(C1 \Lambda C4) \Lambda \text{Eff}(C2 \Lambda C1) \\
\Lambda \text{Eff}(C2 \Lambda C3) \Lambda \text{Eff}(C2 \Lambda C4) \\
\Lambda \text{Eff}(C3 \Lambda C1) \Lambda \text{Eff}(C3 \Lambda C2) \\
\Lambda \text{Eff}(C3 \Lambda C4) \Lambda \text{Eff}(C4 \Lambda C1) \\
\Lambda \text{Eff}(C4 \Lambda C2) \Lambda \text{Eff}(C4 \Lambda C3) \\
\left(5.3\right)
\]

where, \(\text{Int}(C1 \Lambda C2 \Lambda C3 \Lambda C4)\) represents integration of components C1, C2, C3 and C4, \(\text{Eff}(C1)\) represents the effect generated due to component C1, effect is 1 if C1 is error free.
and 0 if C1 is having error.

$Eff(C2)$ represents the effect generated due to component C2, effect is 1 if C2 is error free and 0 if C2 is having error. $Eff(C3)$ represents the effect generated due to component C3, effect is 1 if C3 is error free and 0 if C3 is having error.

$Eff(C4)$ represents the effect generated due to component C4, effect is 1 if C4 is error free and 0 if C4 is having error. $Eff(C1 \Lambda C2)$ represents the effect generated due to the components C1 and C2.

The integrated effect of C1 and C2 is 1 if the integration is error free, otherwise the effect is 0.

**b) Integration-Effect Matrix for four components**

Possible values of integration-Effect matrix for four components (Table 5.7):

<table>
<thead>
<tr>
<th>Components</th>
<th>C1 Effect</th>
<th>C2 Effect</th>
<th>C3 Effect</th>
<th>C4 Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Eff(C1): 0/1</td>
<td>1</td>
<td>Eff(C1 Λ C2): 0/1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td>Eff(C2 Λ C1): 0/1</td>
<td>1</td>
<td>Eff(C2): 0/1</td>
</tr>
<tr>
<td>C3</td>
<td>0</td>
<td>Eff(C3 Λ C1): 0/1</td>
<td>0</td>
<td>Eff(C3 Λ C2): 0/1</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>Eff(C4 Λ C1): 0/1</td>
<td>0</td>
<td>Eff(C4 Λ C2): 0/1</td>
</tr>
</tbody>
</table>

Table 5.8 shows the Integration-Effect matrix for the above case (Figure 5.6):

<table>
<thead>
<tr>
<th>Components</th>
<th>C1 Effect</th>
<th>C2 Effect</th>
<th>C3 Effect</th>
<th>C4 Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$Eff(C1 Λ C3)$ represents the effect generated due to the components C1 and C3. The integrated effect of C1 and C3 is 1 if the integration is error free, otherwise the effect is 0.
$Eff(C1 \land C4)$ represents the effect generated due to the components $C1$ and $C4$. The integrated effect of $C1$ and $C4$ is 1 if the integration is error free, otherwise the effect is 0. $Eff(C2 \land C1)$ represents the effect generated due to the components $C2$ and $C1$. The integrated effect of $C2$ and $C1$ is 1 if the integration is error free, otherwise the effect is 0. In the same manner values of the other components can be derived. $\Lambda$ denotes the AND operation and $V$ denotes the OR operation. If the effect is positive that is without any error it is denoted as 1, and if it is negative that is, the having error, it is denoted as 0. For $Int(C1 \land C2 \land C3 \land C4)$ to be true, $Eff(C1), Eff(C2), Eff(C3), Eff(C4)$ and $Eff(C1 \land C2 \land C3 \land C4)$ all must be true, that is, 1.

**Case Study 3:**

This case study is based on scenario 3, which signifies the number of test cases derived from the Integration-Effect Matrix shown in Table 5.8:

**Case 1:** Number of test cases where $C1$ is involved

$$= (\text{Number of 1s under “Effect” column in row } C1) - 1$$

$$= 4 - 1 = 3$$

**Case 2:** Number of test cases where $C2$ is involved

$$= (\text{Number of 1s under “Effect” column in row } C2) - 1$$

$$= 3 - 1 = 2$$

**Case 3:** Number of test cases where $C3$ is involved

$$= (\text{Number of 1s under “Effect” column in row } C3) - 1$$

$$= 3 - 1 = 2$$

**Case 4:** Number of test cases where $C4$ is involved

$$= (\text{Number of 1s under “Effect” column in row } C4) - 1$$

$$= 4 - 1 = 3$$

*Total number of test cases derived from the proposed method:*

$$= \text{Number of test cases for } C1 + \text{Number of Test cases for } C2$$

$$+ \text{Number of Test cases for } C3 + \text{Number of Test cases for } C4$$
\[= 3 + 2 + 2 + 3 = 10.\]

*Number of test cases through Boundary Value Analysis method:*

Assuming that ‘\(n\)’ is the number of components then, the minimum number of test cases are \(4n+1\). If we have 2 components, then \(n = 2\).

Therefore, Number of test cases = \(4 \times n + 1 = 4 \times 4 + 1 = 17\)

**Scenario 4: When we have Five Components**

This is the case where we have five components C1, C2, C3, C4 and C5. This case is shown in Figure 5.7.

![Diagram of five components interacting](image)

**Figure 5.7 Interaction between Five Components**

Component C1 is communicating with component C2, C3 and component C4. Component C2 is integrated with C1, C3 and C4. Component C3 is communicating with C1, C2, C4 and C5. Component C4 is sharing information with C1, C2, C3 and C5. Component C5 is having interaction with C3 and C4. All the interacting components have request and response edges. This is shown as an edge from C1 to C2 and C2 to C1. Similarly, edges from C1 to C3 and C3 to C1 are drawn to show the communication. To show the interaction between C3, C4 and C5 we have used edges between them. There is a request edge and response edge from C4 to C1 and C1 to C4, and finally edges from C5 to C4 to show the interaction between them.
a) Integration-Effect Graph for five components

When we integrate these components, they will produce some effect. If the effect is in specified way we say that components are working as per the requirement. But if the effect is not as per our intention, we need to test these components. This is shown in Figure 5.8.

From Figure 5.8, it is noted that the output of integration of component C1, C2, C3, C4 and C5 is 1 (i.e., true or without any error) if the effect of component C1 (Eff(C1)) is 1 (without any error), effect of component C2 (Eff(C2)) is 1 (without any error), effect of component C3 (Eff(C3)) is 1 (without any error), effect of component C4 (Eff(C4)) is 1 (without any error), effect of component C5 (Eff(C5)) is 1 (without any error), and the integration effect of C1,
C2, C3, C4 and C5 is error free. To generate the true Integration-effect of component C1, C2, C3, C4 and C5, we have to take into account the individual as well as combined effects generated due to component C1, C2, C3, C4 and component C5.

We represent the component Integration-effect graph as:

\[
\text{Int}(C1 \land C2 \land C3 \land C4 \land C5) = \text{Eff}(C1) \land \text{Eff}(C2) \land \text{Eff}(C3) \land \text{Eff}(C4) \land \text{Eff}(C5)
\]

\[
\land \text{Eff}(C1 \land C2) \land \text{Eff}(C1 \land C3) \land \text{Eff}(C1 \land C4) \land \text{Eff}(C1 \land C5)
\land \text{Eff}(C2 \land C1) \land \text{Eff}(C2 \land C3) \land \text{Eff}(C2 \land C4) \land \text{Eff}(C2 \land C5)
\land \text{Eff}(C3 \land C1) \land \text{Eff}(C3 \land C2) \land \text{Eff}(C3 \land C4) \land \text{Eff}(C3 \land C5)
\land \text{Eff}(C4 \land C1) \land \text{Eff}(C4 \land C2) \land \text{Eff}(C4 \land C3) \land \text{Eff}(C4 \land C5)
\land \text{Eff}(C5 \land C1) \land \text{Eff}(C5 \land C2) \land \text{Eff}(C5 \land C3) \land \text{Eff}(C5 \land C4)
\]

Where, \(\text{Int}(C1 \land C2 \land C3 \land C4 \land C5)\) represents Integration of components C1, C2, C3, C4 and C5, \(\text{Eff}(C1)\) represents the effect generated due to component C1, effect is 1 if C1 is error free and 0 if C1 is having error. \(\text{Eff}(C2)\) represents the effect generated due to component C2, effect is 1 if C2 is error free and 0 if C2 is having error. \(\text{Eff}(C1 \land C3)\) represents the effect generated due to the components C1 and C3. The integrated effect of C1 and C3 is 1 if the integration is error free, otherwise the effect is 0. \(\text{Eff}(C1 \land C4)\) represents the effect generated due to the components C1 and C4. The integrated effect of C1 and C4 is 1 if the integration is error free, otherwise the effect is 0. \(\text{Eff}(C1 \land C5)\) represents the effect generated due to the components C1 and C5. The integrated effect of C1 and C5 is 1 if the integration is error free, otherwise the effect is 0. In the same manner other components value can be derived. \(\land\) denotes the AND operation and \(\lor\) denotes the OR operation. If the effect is positive that is without any error it is denoted as 1, and if it is negative that is, the having error, it is denoted as 0.

For \(\text{Int}(C1 \land C2 \land C3 \land C4 \land C5)\) to be true, \(\text{Eff}(C1), \text{Eff}(C2), \text{Eff}(C3), f(C4), \text{Eff}(C5)\) and \(\text{Eff}(C1 \land C2 \land C3 \land C4 \land C5)\) all must be true, that is, 1. \(\text{Eff}(C3)\) represents the effect generated due to component C3, effect is 1 if C3 is error free and 0 if C3 is having error. \(\text{Eff}(C4)\) represents the effect generated due to component C4, effect is 1 if C4 is error free and 0 if C4 is having error. \(\text{Eff}(C5)\) represents the effect generated due to component C5, effect is 1 if C5 is error free and 0 if C5 is having error. \(\text{Eff}(C1 \land C2)\) represents the effect generated due to the components C1 and C2. The integrated effect of
C1 and C2 is 1 if the integration is error free, otherwise the effect is 0.

b) Integration-Effect Matrix for five components

Possible values of Integration-Effect matrix for four components (Table 5.9):

Table 5.9 Possible values of Integration-Effect Matrix for Five Components

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
<th>C3</th>
<th>Effect</th>
<th>C4</th>
<th>Effect</th>
<th>C5</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eff(C1): 0/1</td>
<td></td>
<td>Eff(C1 Λ C2): 0/1</td>
<td></td>
<td>Eff(C1 Λ C3): 0/1</td>
<td></td>
<td>Eff(C1 Λ C4): 0/1</td>
<td></td>
<td>Eff(C1 Λ C5): 0/1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eff(C2): 0/1</td>
<td></td>
<td>Eff(C2 Λ C1): 0/1</td>
<td></td>
<td>Eff(C2 Λ C3): 0/1</td>
<td></td>
<td>Eff(C2 Λ C4): 0/1</td>
<td></td>
<td>Eff(C2 Λ C5): 0/1</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eff(C3): 0/1</td>
<td></td>
<td>Eff(C3 Λ C1): 0/1</td>
<td></td>
<td>Eff(C3 Λ C2): 0/1</td>
<td></td>
<td>Eff(C3 Λ C4): 0/1</td>
<td></td>
<td>Eff(C3 Λ C5): 0/1</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eff(C4): 0/1</td>
<td></td>
<td>Eff(C4 Λ C1): 0/1</td>
<td></td>
<td>Eff(C4 Λ C2): 0/1</td>
<td></td>
<td>Eff(C4 Λ C3): 0/1</td>
<td></td>
<td>Eff(C4 Λ C5): 0/1</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eff(C5): 0/1</td>
<td></td>
<td>Eff(C5 Λ C1): 0/1</td>
<td></td>
<td>Eff(C5 Λ C2): 0/1</td>
<td></td>
<td>Eff(C5 Λ C3): 0/1</td>
<td></td>
<td>Eff(C5 Λ C5): 0/1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.10 presents the Integration-Effect matrix for the above case (Figure 5.8):

Table 5.10 Actual values of Integration-Effect Matrix for Five Components

<table>
<thead>
<tr>
<th>Components</th>
<th>C1</th>
<th>Effect</th>
<th>C2</th>
<th>Effect</th>
<th>C3</th>
<th>Effect</th>
<th>C4</th>
<th>Effect</th>
<th>C5</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Case Study 4:**

This case study is based on scenario 4 which signifies number of test cases derived from the Integration-Effect Matrix shown in Table 5.10:

**Case 1:** Number of test cases where C1 is involved

\[= \text{(Number of 1s under “Effect” column in row C1)} - 1 = 4 - 1 = 3\]

**Case 2:** Number of test cases where C2 is involved

\[= \text{(Number of 1s under “Effect” column in row C2)} - 1 = 4 - 1 = 3\]

**Case 3:** Number of test cases where C3 is involved

\[= \text{(Number of 1s under “Effect” column in row C3)} - 1 = 5 - 1 = 4\]

**Case 4:** Number of test cases where C4 is involved

\[= \text{(Number of 1s under “Effect” column in row C4)} - 1 = 5 - 1 = 4\]

**Case 5:** Number of test cases where C5 is involved

\[= \text{(Number of 1s under “Effect” column in row C5)} - 1 = 3 - 1 = 2\]

*Total number of test cases derived from the proposed method:*

\[= \text{Number of test cases for C1} + \text{Number of Test cases for C2} + \text{Number of Test cases for C3} + \text{Number of Test cases for C4} = 3 + 3 + 4 + 4 + 2 = 16.\]

*Number of test cases through Boundary Value Analysis method:*

Assuming that ‘n’ is the number of components then, the minimum number of test cases are 4n+1. If we have 2 components, then n = 2.

Therefore, Number of test cases = 4 * n + 1 = 4 * 5 + 1 = 21

### 5.4 Proposed Test Case Generation Technique for White-Box Components

This testing is applicable to those components whose source code is available and accessible. To count the number of test cases in conventional programs we have McCabe’s Cyclomatic complexity formula. In the context of Component-Based Software we do not have such technique. In this section we propose a method named ‘Cyclomatic complexity’ to count the number of test cases for White-Box components in Component-Based Software.

#### 5.4.1 Cyclomatic complexity for Component-Based Software

As we have discussed in chapter 4, the purpose of calculating the Cyclomatic complexity is to
find out the number of independent paths in the code segment and ultimately it defines the
number of test cases required to cover each and every independent logic path.
For a control flow graph $G$, the Cyclomatic complexity $V(G)$ for component based software
is suggested as:

$$V(G) = |E| - |V| + 2 + |P|$$  

(5.5)

where $|E|$ is the cardinality of the set of edges in a control flow graph, $|V|$ is the cardinality
of the vertex set and $|P|$ is the cardinality of interacting components. The constant 2 is used to
indicate that the node $V$ contributes to the complexity if its out-degree is 2.

To validate the complexity, we propose the metric:

$$V(G) = \sum_{i=1}^{n} (IC)_i + \sum_{j=1}^{m} (CR)_j + OR$$  

(5.6)

where, $(IC)_i = (IC_1, IC_2, IC_3, \ldots, IC_n)$ is the Cyclomatic complexity of ‘$n$’ interacting
components, $(CR)_j = (CR_1, CR_2, CR_3, \ldots, CR_m)$ is the number of closed regions, and $OR$ is
the open region.

5.4.2 Case Studies

To compute the Cyclomatic complexity of Component-Based Software systems using the
proposed method, first we have to draw the control flow graph of the components. Notations
of control flow graph and base case of proposed method are similar as discussed in Chapter 4.

To analyse the proposed Cyclomatic complexity method, we have developed various case
studies in Chapter 4. In this chapter, a case study having 6 components is described. This
case study is depicted in Figure 5.9. In this case $C_1, C_2, C_3, C_4, C_5,$ and $C_6$ are six
interacting components. Here component $C_1$ is interacting with $C_2, C_3, C_4$ and $C_6$ and
derives closed regions $CR_1, CR_4, CR_5$, and $CR_8$ respectively. Component $C_2$ is interacting
with $C_3$ and forming closed region $CR_2$. Component $C_3$ is interacting with $C_5$ and $C_6$ and
deriving closed regions $CR_{10}$, and $CR_{15}$. Component $C_4$ is interacting with $C_5$ and $C_6$ and
forming closed regions $CR_9$, and $C_{13}$. Component $C_5$ is interacting with $C_6$ and forming
closed region $CR_{12}$. Integration of components $C_1, C_2$, and $C_3$ form closed region $CR_3$.
Integration of components $C_1, C_3$, and $C_5$ form closed region $CR_6$. Integration of
components $C_1, C_4$, and $C_5$ form closed region $CR_7$. Integration of components $C_4$, and $C_6$
form closed region $CR_{11}$. Integration of components $C_3, C_5$, and $C_6$ form closed region
$CR_{14}$. There is an open region $OR$. From Figure 5.9, it is observed that the Cyclomatic
complexities of $C_1$, $C_2$, $C_3$, $C_4$, $C_5$ and $C_6$ are 4, 2, 4, 4, 2 and 2 respectively, number of nodes ($V$) is 44, number of edges ($E$) is 70, and the number of interacting components ($P$) is 6.

- **Cyclomatic complexity using McCabe method:**
  
  \[ V(G) = e - n + 2p = 70 - 44 + 2 \times 6 = 38 \]

- **Using the proposed metric the Cyclomatic complexity is computed as:**
  
  \[ V(G) = |E| - |V| + 2 + |P| = 70 - 44 + 2 + 6 = 34 \]

**Verification is carried out as:**

The obtained result from the proposed metric is verified with the help of Equation (5.6) as:

\[ \text{i = 1 to 6, j = 1 to 15, IC}_1 = 4, \text{IC}_2 = 2, \text{IC}_3 = 4, \text{IC}_4 = 4, \text{IC}_5 = 2, \text{IC}_6 = 2, \text{CR} = 1, \text{then} \]

\[ V(G) = (4 + 2 + 4 + 4 + 2 + 2) + 15 + 1 = 34. \]

**Performance Results**

a. **Black-Box Testing:** Various case studies for various scenarios are analysed in the context of black-box testing, it is observed that by using the Integration-Effect matrix number of

![Figure 5.9 Interaction Scenario between Six Components](image-url)
test cases are reduced as compared to the boundary value analysis (black-box) method, and better results are achieved. The results of both the methods for varying number of components are shown in Table 5.11.

Table 5.11 Comparison of Black-Box Test Cases

<table>
<thead>
<tr>
<th>Components</th>
<th>Boundary Value Test Cases</th>
<th>Integration-Effect Metric Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>16</td>
</tr>
</tbody>
</table>

b. **White-Box Testing:** The performance of presented Cyclomatic complexity metric is better than the methods defined in literature. Comparisons are made with the McCabe’s method (White-Box) and achieved better results in terms of reduced number of test cases. Results are shown in Table 5.12.

Table 5.12 Comparison of White-Box Test Cases

<table>
<thead>
<tr>
<th>Number of Components</th>
<th>McCabe’s Complexity</th>
<th>Proposed Method Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (1 Closed Region)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3 (2 Closed Regions)</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>3 (4 Closed Regions)</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>4 (3 Closed Regions)</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>4 (7 Closed Regions)</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>5 (10 Closed Regions)</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td>6 (15 Closed Regions)</td>
<td>38</td>
<td>34</td>
</tr>
</tbody>
</table>

5.5 **Summary**

The proposed Integration-Effect matrix for Black-Box testing and the Cyclomatic complexity method for White-Box testing are inter-module techniques that are applicable to interaction among various components. The Integration-Effect matrix is helpful to test and record the effects of such components whose code is not accessible, and Cyclomatic complexity method is applicable to components for which code is available. We get a greater degree of predictability in terms of costs, effort, quality and risk if we can predict the testability of the software properly and early.