Software Quality Assurance

Code Coverage Analysis

SQA – IT4100

Learning Outcomes

Code Coverage Methods

- Statement Coverage
- Decision Coverage
- Path Coverage
- Condition Coverage
- Multiple Condition Coverage

Code Coverage

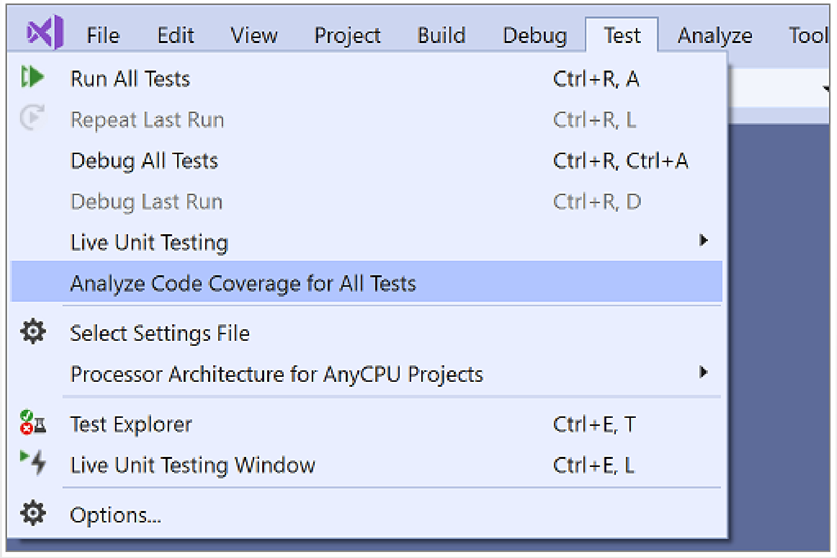
- Code coverage is a term used in software testing to describe how much program source code is covered by a testing plan.
- Developers look at the number of program subroutines and lines of code that are covered by a set of testing resources and techniques.
- Code coverage is also known as test coverage.

Code Coverage Analysis

- Code coverage analysis can provide reassurance that programs are broadly tested for bugs and relatively error-free.
- Code coverage analysis is done mostly to find the precise areas that are not covered by testing strategies.
- Code coverage analysis and other testing aspects in beta or other development rounds exposes bugs to a small test audience, rather than to millions of users when products finally go live.

Code Coverage Analysis (Cont.)

- Microsoft Visual Studio have specific menu tools for doing code coverage analysis.


File
Edit
View
Project
Build
Debug
Test
Analyze
Tool
Run All Tests Ctrl+R, A
Repeat Last Run Ctrl+R, L
Debug All Tests Ctrl+R, Ctrl+A
Debug Last Run Ctrl+R, D
Live Unit Testing
Analyze Code Coverage for All Tests
Select Settings File
Processor Architecture for AnyCPU Projects
Test Explorer Ctrl+E, T
Live Unit Testing Window Ctrl+E, L
Options...

Reference: https://docs.microsoft.com/en-us/visualstudio/test/using-code-coverage-to-determine-how-much-code-is-being-tested?view=vs-2019

Code Coverage Analysis (Cont.)

- Developers may use relatively manual methods that involve mapping out the software source code and determining where testing applies.
- Third-party vendors also provide specific code coverage tools for different programming languages.

pitest.org

JACOCO
Java Code Coverage

nunit

Cobertura
A code coverage utility for Java.

CodeCover

Uses of Code Coverage Analysis

- Helps to measure the efficiency of test implementation
- Offers a quantitative measurement.
- Defines the degree to which the source code has been tested.

Code Coverage Methods

- Statement Coverage
- Decision Coverage
- Path Coverage
- Condition Coverage
- Multiple Condition Coverage

Statement Coverage

- A metric which ensures that each statement of the code is executed at least once.
- Measures the number of lines executed.
- Verifies what the written code is expected to do and not to do.
- This method can be considered as white box testing, as it intends to evaluate the internal structure of the code.
- A programmer is the one who can perform this task efficiently.

Statement Coverage Calculation

$$\text{Statement Coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} \times 100$$

- *Note: All statements including the statement with a function name and statements with only braces ("{" "}") are counted in statement coverage.*

Statement Coverage - Example

```
read a;
read b;
if (a>b)
        print "A is greater than B";
else
        print "B is greater than A";
```

Test Conditions:

1. a = 5, b = 1
2. a = 1, b = 5

Statement Coverage – Example (Cont.)

Total number of Statements = 6

Condition 1 :

If a=5 and b=1, then the first print statement is executed.

Number of distinct statements executed = 4

Condition 2 :

If a=1 and b=5, then the second print statement is executed.

Number of distinct statements executed = 2

Statement Coverage – Example (Cont.)

$$\textbf{Statement coverage} = \frac{\textbf{Number of executed statements}}{\textbf{Total number of Statements}} \times 100\%$$

Statement Coverage = [(4+2) / 6] x 100% = (6 / 6) x 100%

= 100%

Statement Coverage – Exercise

Calculate the statement coverage for the given code in following conditions.

When;

1. a = 5 and b = 7
2. a = 4 and b = 4

```
Read a;
read b;
i=0
if a>b
  while(i<a)
     print(i)
     i++;
  end while
else
  while(i<b)
    print(i)
    i++;
  end while
end if;
```

Decision Coverage

- Decision coverage reports the true or false outcomes of each Boolean expression.
- There are many different methods of reporting this metric. All these methods focus on covering the most important combinations.

Decision Coverage Calculation

$$\textit{Decision Coverage} = \frac{\textit{Number of decision outcomes exercised}}{\textit{Total number of decision outcomes}} \times 100\%$$

Decision Coverage - Example

```
Read a;
If (a> 5)
        a=a*3
print (a)
```

Test Conditions:

1. a = 2
2. a = 7

Decision Coverage – Example (Cont.)

Total number of decision outcomes = 2

Condition 1 :

If a=2, then the statement inside is not executed.

Number of distinct decision outcomes exercised = 1

Here the “FALSE" outcome of the decision If (a>5) is checked.

Condition 2 :

If a=7, then the statement inside is executed.

Number of distinct decision outcomes exercised = 1

Here the “TRUE" outcome of the decision If (a>5) is checked.

Decision Coverage – Example (Cont.)

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

Statement Coverage = [(1+1) / 2] x 100% = (2 / 2) x 100%

= 100%

Decision Coverage – Exercise

Calculate the decision coverage for the given code in following conditions.

When;

1. a = 5 and b = 7
2. a = 3 and b = 1

```
read a;
read b;
if a>b
    while (a>b)
        print("B is still small")
        b++
        if(b>a)
            print("B is now greater than A")
else
    if b>7
       print("B is greater than A")
```

Path Coverage

- A path is a unique sequence of branches from the function entry to the exit.
- Path coverage covers a function from its entry till its exit point.
- Path coverage refers to designing test cases such that all linearly independent paths in the program (method) are executed at least once.
- A control flow graph describes how the control flows through the application.
- A linearly independent path can be defined in terms of what's called a control flow graph of an application.
- A linearly independent path is a path with at least one new edge in the control flow graph.
- Cyclomatic complexity metric can be used to identify the number of independent paths in a program.

Path Coverage Calculation

$$\textbf{\textit{Path Coverage}} = \frac{\textbf{\textit{Number of linearly independent paths executed}}}{\textbf{\textit{Total number of linearly independent paths}}} \times 100\%$$

Path Coverage

- **Steps:**

1. Draw the control flow graph
2. Identify linearly independent paths and total number of linearly independent paths in the program.
3. Identify linearly independent paths executed by each test condition.
4. Identify the number of linearly independent paths covered by all test conditions.
5. Calculate the path coverage.

Path Coverage - Example

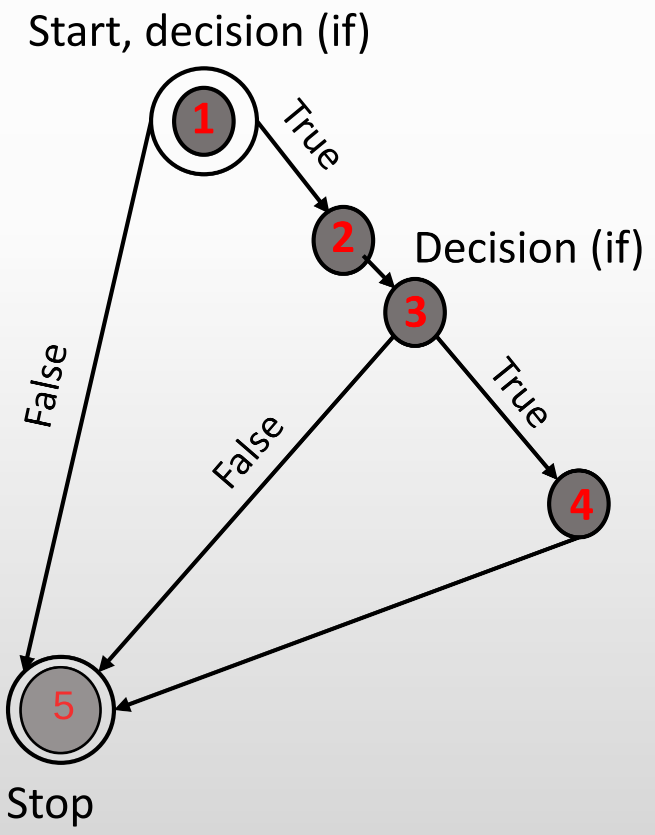
```
Demo(int a)
If (a> 5)
        a=a*3
        if (b > 8)
                b = b – 5
```

Test Conditions:

1. a = 4, b = 6
2. a = 6, b = 6

Path Coverage – Example (Cont.)

- Steps 1: Draw the Control Flow graph.



Path Coverage – Example (Cont.)

- Steps 2: Identify linearly independent paths and total number of linearly independent paths in the program.

Cyclomatic Complexity:

V(G) = e – n + 2 = 6 - 5 + 2 = **3**

Total number of linearly independent paths = 3

Path 1: 1→5

Path 2: 1→2→3→5

Path 3: 1→2→3→4 → 5

Path Coverage – Example (Cont.)

Steps 3: Identify linearly independent paths executed by the each test condition.

Condition 1:

If a=4, and b = 6

Paths executed = Path 1: 1→5

Condition 2:

If a=6, and b = 6

Paths executed = Path 2: 1→2→3 →5

Path Coverage – Example (Cont.)

Steps 4: Identify the number of linearly independent paths covered by all test conditions.

- Paths covered by all test conditions:

  Path 1: 1→5

  Path 2: 1→2→3 →5

Number of linearly independent paths covered by all test conditions = 2

Path Coverage – Example (Cont.)

Steps 5: Calculate the path coverage.

Number of linearly independent paths executed = 2

Total number of linearly independent paths = 3

Path Coverage = (2/3) * 100 = 67%

Path Coverage – Exercise 1

Calculate the path coverage for the given code in following conditions.

When;

1. a = 5 and b = 7
2. a = 3 and b = 1

```
read a;
read b;
if a>b
    while (a>b)
        print("B is still small")
        b++
        if(b>a)
            print("B is now greater than A")
else
    if b>7
    print("B is greater than A")
```

Multiple Condition Coverage

- Multiple Condition Coverage checks the coverage of all combinations of conditions in a program.
- Total test cases for a program with n number of conditions will be 2 to the power n.
  - If there are two conditions, then n=2.
  - Therefore, 4 (2^2) test cases are needed to get the full multiple condition coverage.

Code Coverage – Exercise

Calculate the statement coverage, decision coverage, and path coverage for the given code in following conditions.

When;

1. input = 5
2. input = 20

```
read input;
if input>10
    while (input!=0)
        print(“Valid”)
        if(input%2==0)
            print(“Even”)
        else
            print(“Odd”)
        input = input - 1
else
    if input>0
        print(“Please enter valid value!!”)
    else
        print(“Enter a value greater than 10”)
```