



CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019-fall

Part II-4 White Box Testing - Coverage

White Box Testing

- Today we continue our look at **white box** testing, with emphasis on **code coverage** methods
- We'll look at:
 - **Statement** coverage
 - **Basic block** coverage
 - **Decision** coverage
 - **Condition** coverage
 - **Branch** coverage
 - **Loop** coverage

Code Coverage Methods

- Two kinds:
 - **Statement** analysis (flow independent)
 - **Decision** analysis (flow dependent)
- Statement analysis methods
 - **Statement** coverage
 - **Basic block** coverage
- Decision analysis methods
 - **Decision** coverage
 - **Condition** coverage
 - **Loop** coverage
 - **Path** coverage

Statement Coverage Method

- Cause every statement in the program to be executed **at least once**, giving us confidence that every statement is at least **capable** of executing correctly
- **System**: Make a test case for each **statement** in the program, independent of the others
 - Test must simply cause the statement to be **run**, ignoring its actions and sub-statements (but still must check that result of test is correct)
- **Completion criterion**: A test case for every statement
 - Can be checked by **instrumentation injection** to track statement execution coverage

Previously EVIL PARROT

```
/* Puny mortal!
```

```
I scoff at your A1 black box testing! */
```

```
def createTicket (name, ...):
```

```
    if 'Kingston' in name:
```

```
        /* EVIL TIME */
```

```
        name = 'steven_is_drunk'
```

```
    [non-evil code to save ticket]
```



Example: Statement Coverage

```
    // calculate numbers less than x
    //    which are divisible by y
1  int x, y;
2  x = c.readInt();
3  y = c.readInt();
4  if (y == 0)
5      c.println("y is zero");
6  else if (x == 0)
7      c.println("x is zero");
8  else
9  {
10     for (int i = 1; i <= x; i++)
11     {
12         if (i % y == 0)
13             c.println(i);
14     }
15 }
```

Example: Statement Coverage

- **Statement Coverage Tests**
 - We blindly make one test for each statement, analyzing which **inputs** are needed to cause the statement to be executed
 - Create test case for each unique set of inputs

Example: Statement Coverage

```
// calculate numbers less than x
// which are divisible by y
1  int x, y;
2  x = c.readInt();
3  y = c.readInt();
4  if (y == 0)
5      c.println("y is zero");
6  else if (x == 0)
7      c.println("x is zero");
8  else
9  {
10     for (int i = 1; i <= x; i++)
11     {
12         if (i % y == 0)
13             c.println(i);
14     }
15 }
```

Stmt	x input	y input	Test	x	y
1	0	0	T1	0	0
2	0	0			
3	0	0			
4	0	0			
5	0	0			
6	0	1	T2	0	1
7	0	1			
8	1	1	T3	1	1
9	1	1			
10	1	1			
11	1	1			

Basic Block Coverage

- Cause every **basic block** (indivisible sequence of statements) to be executed at least once
 - Usually generates fewer tests
- **System**: Identify basic blocks by code analysis, design test case for each basic block
 - Sequence of statements in a row, ignoring sub-statements, such that if first is executed then following are all executed
- **Completion criterion**: A test case for every basic block
 - Can be checked by **instrumentation injection** to track statement execution coverage

Example: Basic Block Coverage

```
// calculate numbers less than x
//   which are divisible by y
int x, y;
1 x = c.readInt();
  y = c.readInt();
  if (y == 0)
2   c.println("y is zero");
  else
    if (x == 0)
3     4 c.println("x is zero");
    else
      {
5       for (int i = 1; i <= x; i++)
        {
6         if (i % y == 0)
7         c.println(i);
        }
      }
    }
```

The diagram illustrates basic block coverage for the provided code. Red arrows and numbers 1 through 7 indicate the flow of execution and the boundaries of basic blocks. Block 1 starts at the first `x = c.readInt();` and ends at the first `}` after the `for` loop. Block 2 starts at the `c.println("y is zero");` statement and ends at the `else` block. Block 3 starts at the `if (x == 0)` statement and ends at the `}` after the `for` loop. Block 4 starts at the `c.println("x is zero");` statement and ends at the `else` block. Block 5 starts at the `for (int i = 1; i <= x; i++)` statement and ends at the `}` after the `for` loop. Block 6 starts at the `if (i % y == 0)` statement and ends at the `}` after the `if` statement. Block 7 starts at the `c.println(i);` statement and ends at the `}` after the `if` statement.

Example: Basic Block Coverage

- Basic Block Coverage Tests
 - We make one test for each block, analyzing which **inputs** are needed to cause the block to be entered
 - Create test case for each unique set of inputs

```
// calculate numbers less than x
// which are divisible by y
1 int x, y;
  x = c.readInt();
  y = c.readInt();
  if (y == 0)
2    c.println("y is zero");
  else
3    if (x == 0)
4      c.println("x is zero");
    else
5    {
6      for (int i = 1; i <= x; i++)
7      {
6        if (i % y == 0)
7          c.println(i);
      }
    }
}
```

Block	x input	y input	Test	x	y
1	0	0	T1	0	0
2	0	0			
3	0	1	T2	0	1
4	0	1			
5	1	1	T3	1	1
6	1	1			
7	1	1			

Decision Coverage

- **Decision (Branch) Coverage Method**
 - Causes every **decision** (if, switch, while, etc.) in the program to be made both ways (or every possible way for switch)
 - **System**: Design a test case to exercise each decision in the program each way (true/false)
 - **Completion criterion**: A test case for each side of each decision
 - Can be checked by **instrumentation injection** to track branches taken in execution

Example: Decision Coverage

```
// calculate numbers less than x
//   which are divisible by y
int x, y;
x = c.readInt();
y = c.readInt();
1  if (y == 0)
    c.println("y is zero");
else
2  if (x == 0)
    c.println("x is zero");
    else
    {
        for (int i = 1; i <= x; i++)
        {
3          if (i % y == 0)
              c.println(i);
        }
    }
}
```

Example: Decision Coverage

- Decision Coverage Tests

- We make one test for each side of each decision

```
// calculate numbers less than x
// which are divisible by y
int x, y;
x = c.readInt();
y = c.readInt();
1 if (y == 0)
    c.println("y is zero");
else
2   if (x == 0)
      c.println("x is zero");
    else
3     {
        for (int i = 1; i <= x; i++)
        {
            if (i % y == 0)
                c.println(i);
        }
    }
}
```

Decision	x input	y input	Test	x	y
1: true	0	0	T1	0	0
1: false	0	1	T2	0	1
2: true	0	1			
2: false	1	1	T3	1	1
3: true	1	1			
3: false	2	3	T4	2	3

Condition Coverage

- Like decision coverage, but causes every **condition** to be exercised both ways (true/false)
- A condition is any true/false sub-expression in a decision
 - **Example**: if ((x == 1 || y > 2) && z < 3)
 - Requires separate condition coverage tests for each of:
 - **x** == 1 true / false
 - **y** > 2 true / false
 - **z** < 3 true / false
- More effective than simple decision coverage since exercises the different **entry preconditions** for each branch selected

Loop Coverage

- Most programs* do their real work in **do**, **while**, and **for** loops
- This method makes tests to exercise each **loop** in the program in four different states:
 - execute body **zero** times (do not enter loop)
 - execute body **once** (do not repeat)
 - execute body **twice** (repeat once)
 - execute body **many times**
(repeat more than once)

* in non-functional languages

Loop Coverage

- Usually used as an enhancement of a statement, block, decision, or condition coverage method
- **System**: Devise test cases to exercise each loop with zero, one, two, and many repetitions
- **Completion criterion**: A test for each of these cases for each loop
 - Can be verified using **instrumentation injection** in the code

Example: Loop Coverage

```
// calculate numbers less than x
//   which are divisible by y
int x, y;
x = c.readInt();
y = c.readInt();
if (y == 0)
    c.println("y is zero");
else if (x == 0)
    c.println("x is zero");
else
{
    for (int i = 1; i <= x; i++)
    {
        if (i % y == 0)
            c.println(i);
    }
}
```

Loop Body	x	y
zero times	-1	1
once	1	1
twice	2	1
many times	10	1

Summary

- White Box Testing
 - Code coverage methods
 - Statement analysis methods
(statement, basic block coverage)
 - Decision analysis methods
(decision, condition, loop coverage)
- Next time
 - More code coverage methods: path coverage
 - Data coverage methods