CISC327 - Software Quality Assurance

Lecture 15

White Box Testing
White Box Testing

• Outline
  – 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 Testing

• 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

• **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
Example: Statement Coverage

```java
// 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);
    }
}
```
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

```java
// 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);
    }
}
```
Basic Block Coverage

- **Basic Block Analysis Method**
  - 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

```java
// 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);
  }
}
```
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

```java
// 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);
    }
```

<table>
<thead>
<tr>
<th>Block</th>
<th>x input</th>
<th>y input</th>
<th>Test</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>T1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>T2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>T3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

3 for (int i = 1; i <= x; i++)
    { if (i % y == 0)
        c.println(i);
    }
}
Example: Decision Coverage

- Decision Coverage Tests
  - We make one test for each side of each decision

```java
// 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
    c.println("x is zero");
else
{
    for (int i = 1; i <= x; i++)
    {
        if (i % y == 0)
            c.println(i);
    }
}
```

<table>
<thead>
<tr>
<th>Decision</th>
<th>x input</th>
<th>y input</th>
<th>Test</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: true</td>
<td>0</td>
<td>0</td>
<td>T1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1: false</td>
<td>0</td>
<td>1</td>
<td>T2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2: true</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: false</td>
<td>1</td>
<td>1</td>
<td>T3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3: true</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: false</td>
<td>2</td>
<td>3</td>
<td>T4</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Condition Coverage

• **Condition Coverage Method**
  – Like decision coverage, but causes every condition expression to be exercised both ways (true/false)
  – A condition is any true/false sub-expression in a decision
    • **Example**: if 
      \((x == 1 \text{ } || \text{ } y > 2) \text{ } & \text{ } z < 3\) 
    • 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

• Loop Coverage Method
  – 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)
Loop Coverage

- Loop Coverage Method
  - 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
// 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);
    }
}

<table>
<thead>
<tr>
<th>Loop Body</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero times</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>once</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>twice</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>many times</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>
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

• **Reminder**
  – Assign #2 due Thursday