

Introduction to Software Testing
Chapter 8.1
Building Testing Tools –Instrumentation

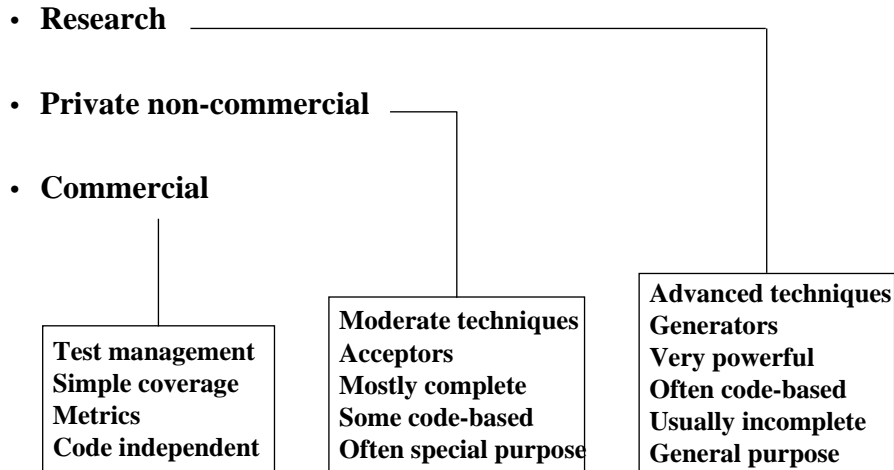
Paul Ammann & Jeff Offutt

www.introsoftwaretesting.com

Chapter 8 Outline

- 1. Instrumentation for Graph and Logical Expression Criteria**
- 2. Building Mutation Testing Tools**

Categories of Test Tools



Types of Test Tools

- 1. Coverage analyzers (Acceptors)**
 - What coverage ? (most common is node)
 - What level? (sometimes object code)
- 2. Flow graph generators**
- 3. Metrics**
- 4. Instrumentation support**
- 5. Test execution automation**
 - Runs scripts and reports results
 - JUnit, HttpUnit, ...
- 6. Capture / replay**
 - Useful for GUIs
- 7. Stubs and Drivers**
- 8. Test data generators**
 - Graph path – based
 - Data flow
 - Logic based
 - Functional
 - Mutation
 - Random
 - Only random test data generators exist

Tools Instrumentation

The key technology behind many test tools is “instrumentation”

Tools Instrumentation

- Coverage analysis is measured with instrumentation
- **Instrument** : One or more statements inserted into the program to monitor some aspect of the program
 - Must not affect the behavior
 - May affect timing
 - Source level or object code level

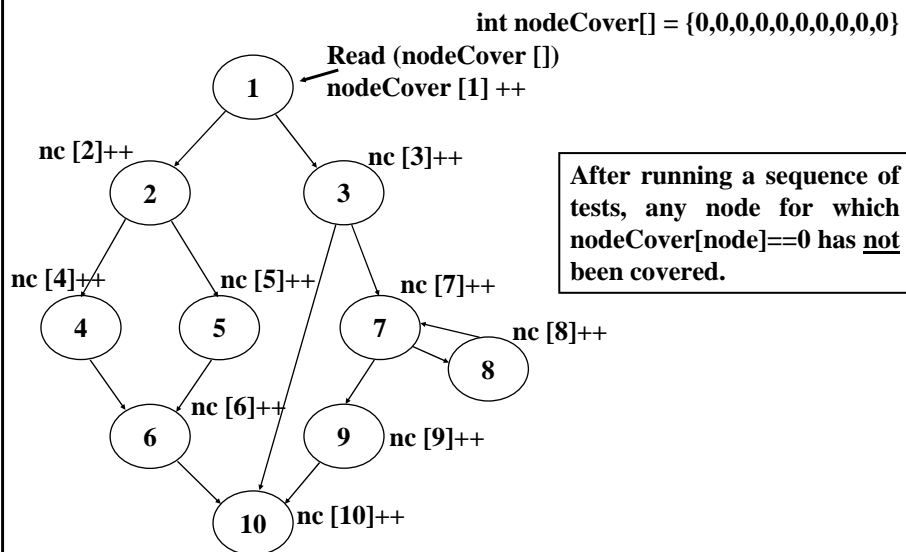
```
public int min (int A, B)
{
    int m = A;
    if (A > B)
    {
        m = B;
    }
    return (m);
}
```

Mark: “if body is reached”

Instrumenting for Statement Coverage

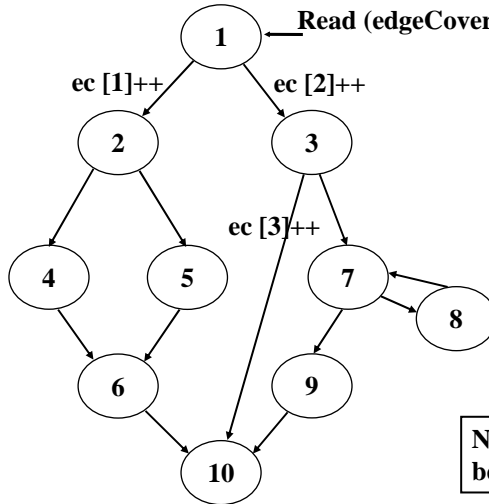
1. Each node is given a unique id #
 - Node # or statement #
2. Create an array indexed by id #s – `nodeCover []`
3. Insert an *instrument* at each node
 - `nodeCover [i] ++;`
4. Save `nodeCover []` after each execution
 - Must accumulate results across multiple test cases

Statement Coverage Example



Edge Coverage Instrumentation

```
int edgeCover[] = {0,0,0,0,0,0,0,0,0,0,0}
```

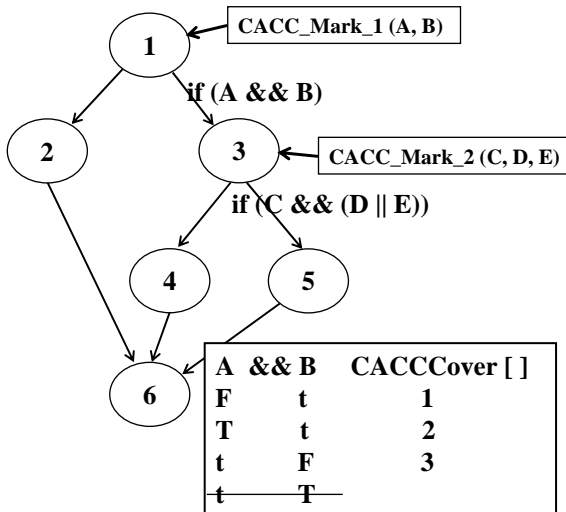


For each edge e, put `edgeCover[e]++` on the edge.

If `edgeCover[e] == 0`, e has not been covered.

Note that the arrays could be boolean

CACC Coverage Instrumentation



```
CACC_Mark_1 (A, B)
{
  if (!A)
  { // Don't allow short circuit
    if (B)
      CACC_Cover[1]++;
  }
  else if (A)
  {
    if (B)
      CACC_Cover [2]++;
    else
      CACC_Cover[3]++;
  }
}
```

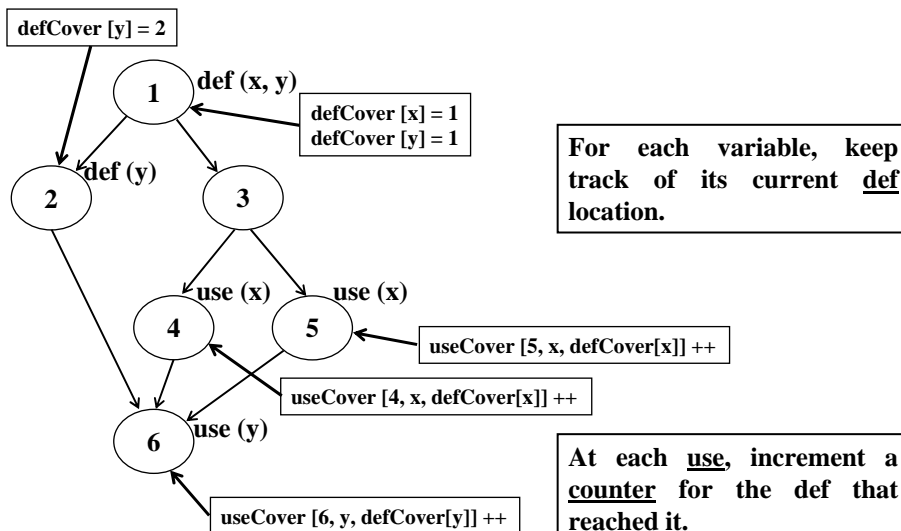
CACC Coverage Instrumentation (2)

C && (D E)	CC []
F { t t }	4
T { t f }	5
t F f	6
t T f	7
t f F	
t f T	8

```

CACC_Mark_2 (C, D, E)
{
  if (! C)
  {
    if (D || E)
      CACCCover [4]++;
  }
  else if (! D)
  {
    if (! E)
      CACCCover [6]++;
    else
    {
      CACCCover [5]++;
      CACCCover [8]++;
    }
  }
  else // C and D
  if (! E)
  {
    CACCCover [5]++;
    CACCCover [7]++;
  }
  else
    CACCCover [5]++;
}
    
```

All-Uses Coverage Instrumentation



Instrumentation Summary

- **Instrumentation can be added in multiple copies of the program**
 - Source code
 - Java byte code (or other intermediate code)
 - Executable
- **Instrumentation must not change or delete functionality**
 - Only add new functionality
- **Instrumentation may affect timing behavior**
- **Requires the program to be parsed**
 - Once parsed, inserting instruments is straightforward
- **Most challenging part is the user interface**
 - Present reports as dumps of instrument arrays ?
 - Present tables ?
 - Program source with colored annotations ?
 - Graph of program with colored annotations ?