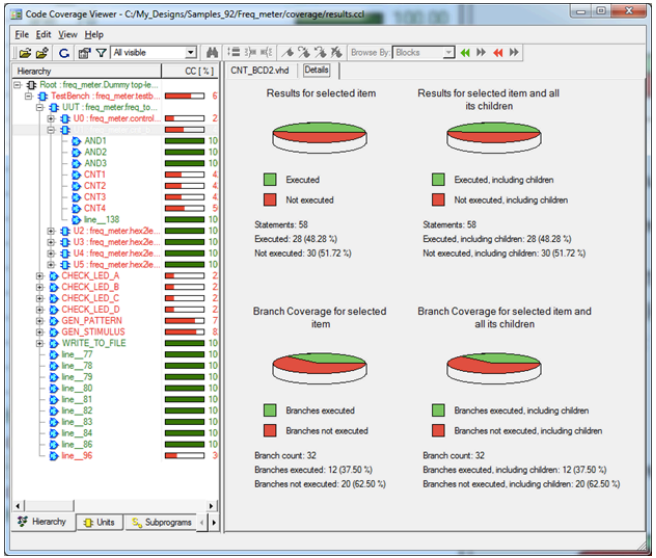
Exploring Code Coverage

CS585 Software Verification and Validation
http://cs585.yusun.io
January 21, 2015

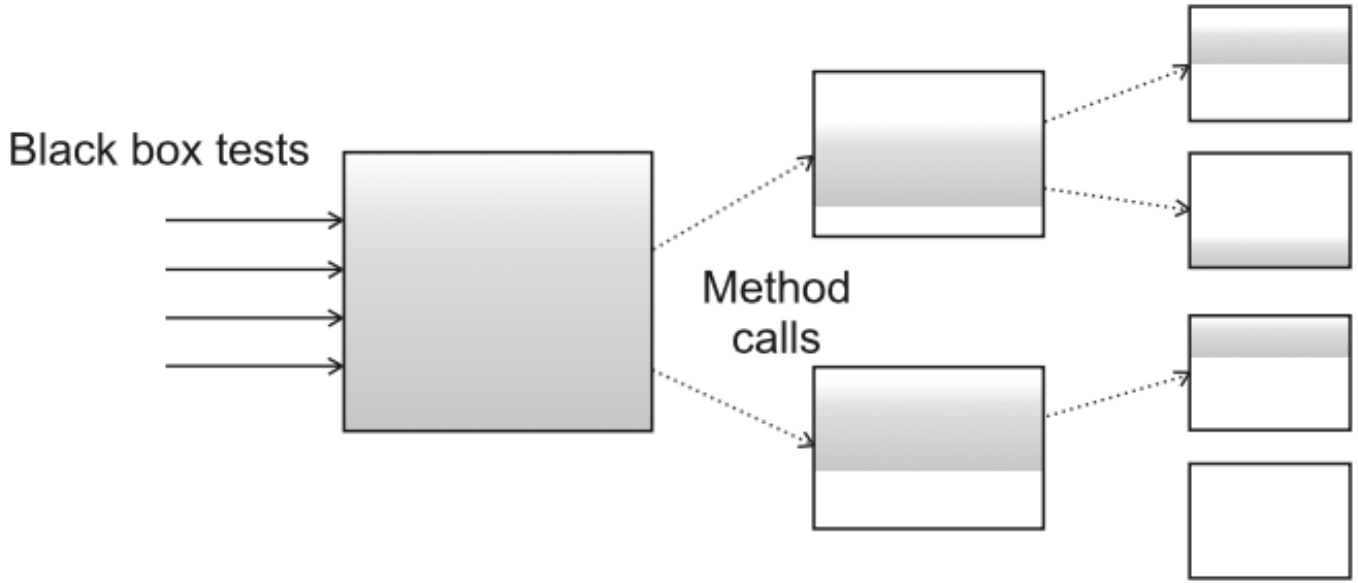
Yu Sun, Ph.D.
http://yusun.io
yusun@cpp.edu

Code Coverage

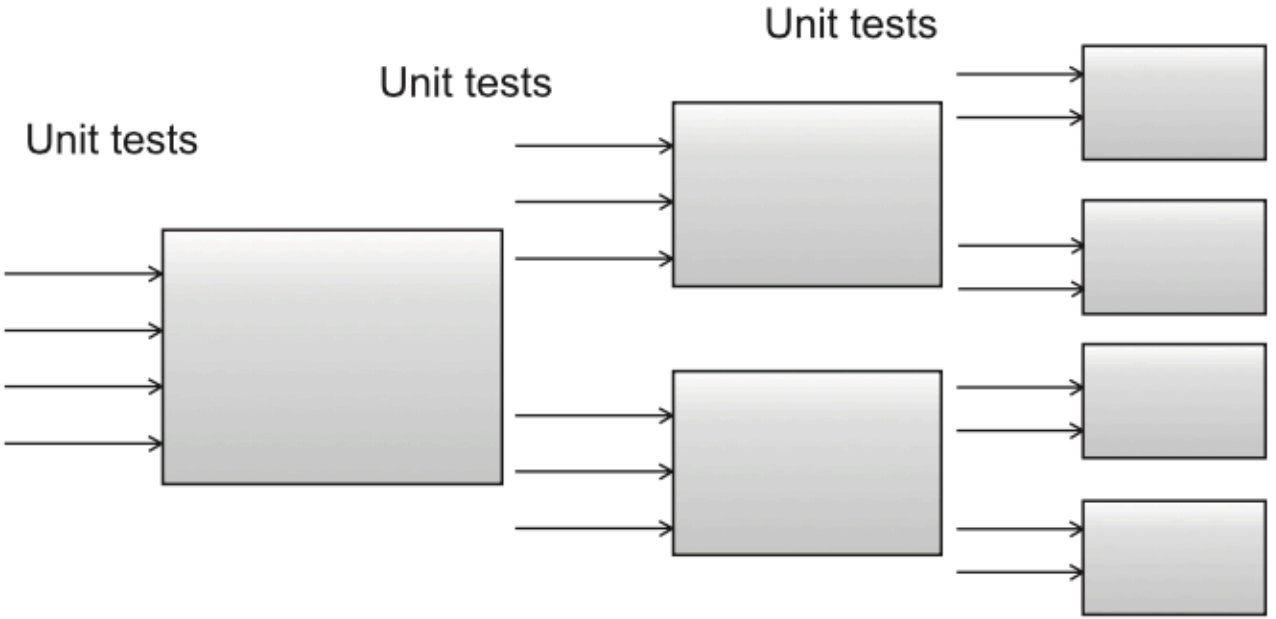
- *Code coverage* is a measurement of percentage of instructions of code being executed while the automated tests are running



Partial Test Coverage with Black Box Tests



Complete Coverage using White Box tests



Code Coverage

- High code coverage implies that the code has been thoroughly unit tested and has a lower chance of containing bugs than code with a low code coverage
- Focus on writing meaningful (business logic) unit tests and not on having 100 percent coverage. It's easy to cheat and have 100 percent coverage with completely useless tests

Types of Code Coverage

- *Statement or line coverage*
  - Measures the statements or lines being covered
- *Branch coverage*
  - Measures the percentage of each branch of each control structure, such as the if else and switch case statements
- *Function or method coverage*
  - Measures the function execution

Statement Coverage

- Statement coverage:
  - Series of test cases to check every statement at least once
- Weakness
  - Branch statements
- Consider ABS function

```
if ( y >= 0) then
   y = 0;
abs = y;
```

test case

input: y = 0

expected result: 0

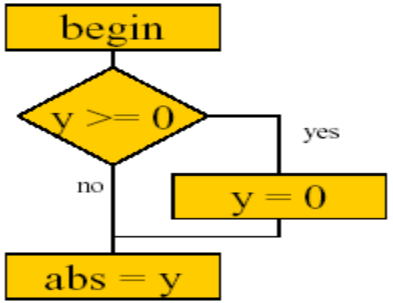
actual result: 0

This case gives the false assumption that the code is correct

Branch Coverage

- Series of tests to check all branches (solves problem on previous slide); edge coverage

```
if ( y >= 0) then
   y = 0;
abs = y;
```

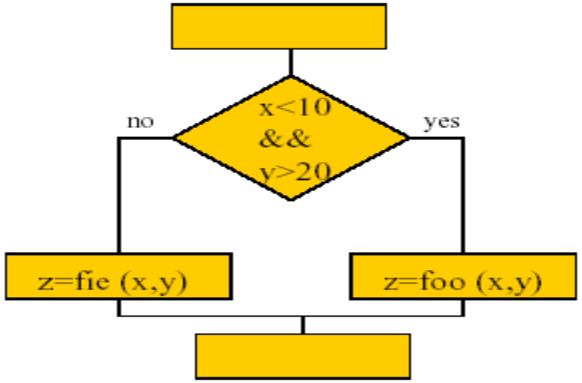


Branch=>statement not true other way…

test case-1:
input: y = 0
expected result: 0
actual result: ?

test case-2:
input: y = -5
expected result: 5
actual result: ?

Branch Coverage



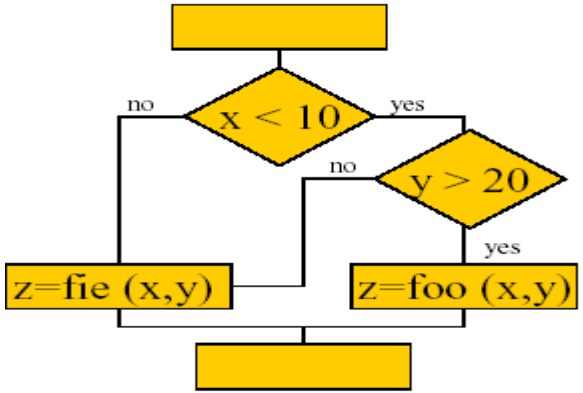

```
if ( x < 10 && y > 20) {
  z = foo (x,y);
else
  z = fie (x,y);
}
```

What is the *potential* problem here?

test case-1 (yes-branch):
input: x = -4, y = 30
expected result: ...
actual result: ?

test case-2 (no-branch):
input: x = 12, y = 12
expected result: ...
actual result: ?

Condition Coverage




```
if ( x < 10 && y > 20) {
  z = foo (x,y);
else
  z = fie (x,y);
}
```

	x<10	y>20
test-case-1:	t	t
test-case:2	t	f
test-case-3:	f	t
test-case-4	f	f

Code Instrumentation

- How to implement code coverage
  - In a copy of source code, each block of statement is instrumented with an accumulator flag
  - The tests run on the instrumented code and update the flags
  - Collects the accumulator flags and measures the ratio of the flags turned on versus the total number of flags

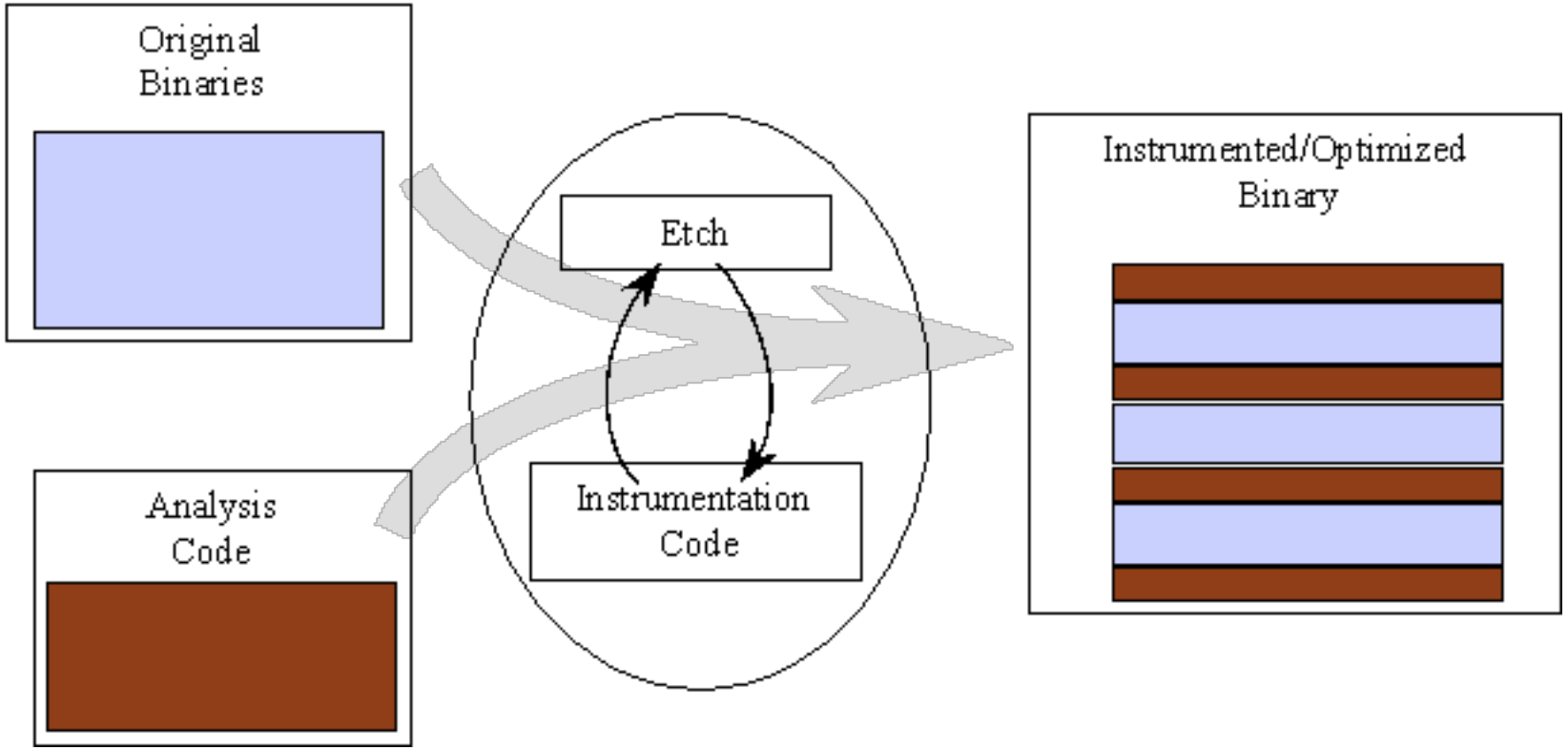
Source Code Instrumentation

```
int[] visitedLines = new int[14];
public int absSumModified(Integer op1 , Integer op2) {
  visitedLines[0] = 1;
  if(op1 == null) {
    visitedLines[1] = 1;
    if(op2 == null) {
      visitedLines[2] = 1;
      return 0;
    }else {
      visitedLines[3] = 1;
    }
  }else {
    visitedLines[4] = 1;
  }

  visitedLines[5] = 1;
  if(op1 == null) {
    visitedLines[6] = 1;
```

Source Code Instrumentation

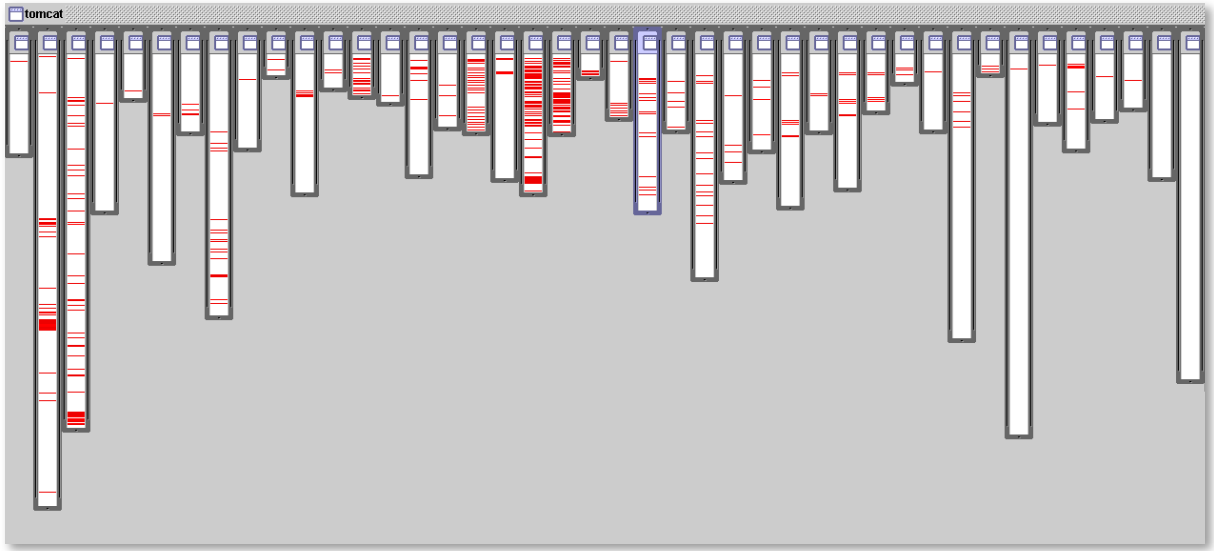
```
    if(op2 != null) {
      visitedLines[7] = 1;
      return Math.abs(op2);
    }else {
      visitedLines[8] = 1;
    }
  }else {
    visitedLines[9] = 1;
  }

  visitedLines[10] = 1;
  if(op2 == null) {
    visitedLines[11] = 1;
    return Math.abs(op1);
  }else {
    visitedLines[12] = 1;
  }
  visitedLines[13] = 1;
  return Math.abs(op1)+Math.abs(op2);
}}
```

Binary Code Instrumentation



Using Aspect-Oriented Programming

- OpenJava
- Javassist



Code Coverage Tool Support

- Cobertura
  - Instruments the bytecode offline and is a widely used coverage tool.
- EMMA
  - Instruments the bytecode offline or on the fly
- Clover
  - Instruments the source code
- JaCoCo
  - Instruments the bytecode on the fly while running the code

Use Cobertura in Maven

```
<reporting>
  <plugins>
            <!-- Normally, we take off the dependency report, saves time. -->
            <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-project-info-reports-plugin</artifactId>
                        <version>2.7</version>
                        <configuration>
                                    <dependencyLocationsEnabled>false</dependencyLocationsEnabled>
                        </configuration>
            </plugin>

            <plugin>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>cobertura-maven-plugin</artifactId>
                        <version>2.6</version>
                        <configuration>
                                    <formats>
                                                <format>html</format>
                                                <format>xml</format>
                                    </formats>
                        </configuration>
            </plugin>

   </plugins>
</reporting>
```

http://www.mkyong.com/qa/maven-cobertura-code-coverage-example/

Integrate Code Coverage with Build Tools

- Code coverage should be reported each time when a build is triggered