Coverage Metrics
SFWR ENG 3S03: Software Testing

Alicia Marinache

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Acknowledgments: Slides adapted from Dr. R. Khedri and Dr. R.Paige.

Objectives

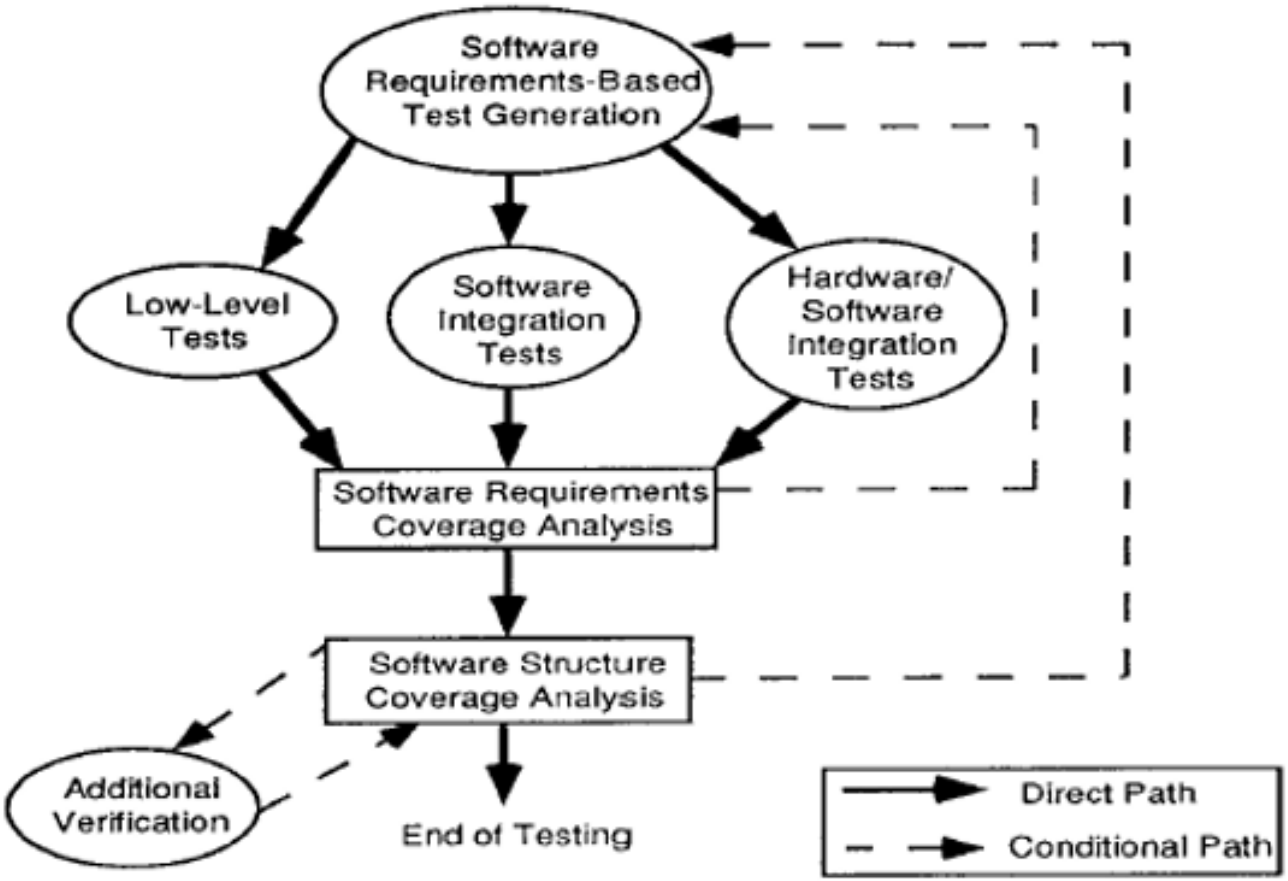
- Understand why and what to measure when testing
- Learn to build control flow graphs
- Examine coverage criteria from the measurement POV

What goals we might have when testing?

- Find (and fix!) maximum number of bugs
- Know if we have undiscovered bugs
- Comply with regulator-set standards
- Have a compelling defence in a court case
- Do the testing with minimum time and cost
- ...

- Industrial perspective: DO178C Software Considerations in Airborne Systems and Equipment Certification

- Complementary goals for testing
  - Demonstrate the software satisfies its requirements
  - Demonstrate with a high degree of confidence that errors that could lead to unacceptable failure conditions have been removed



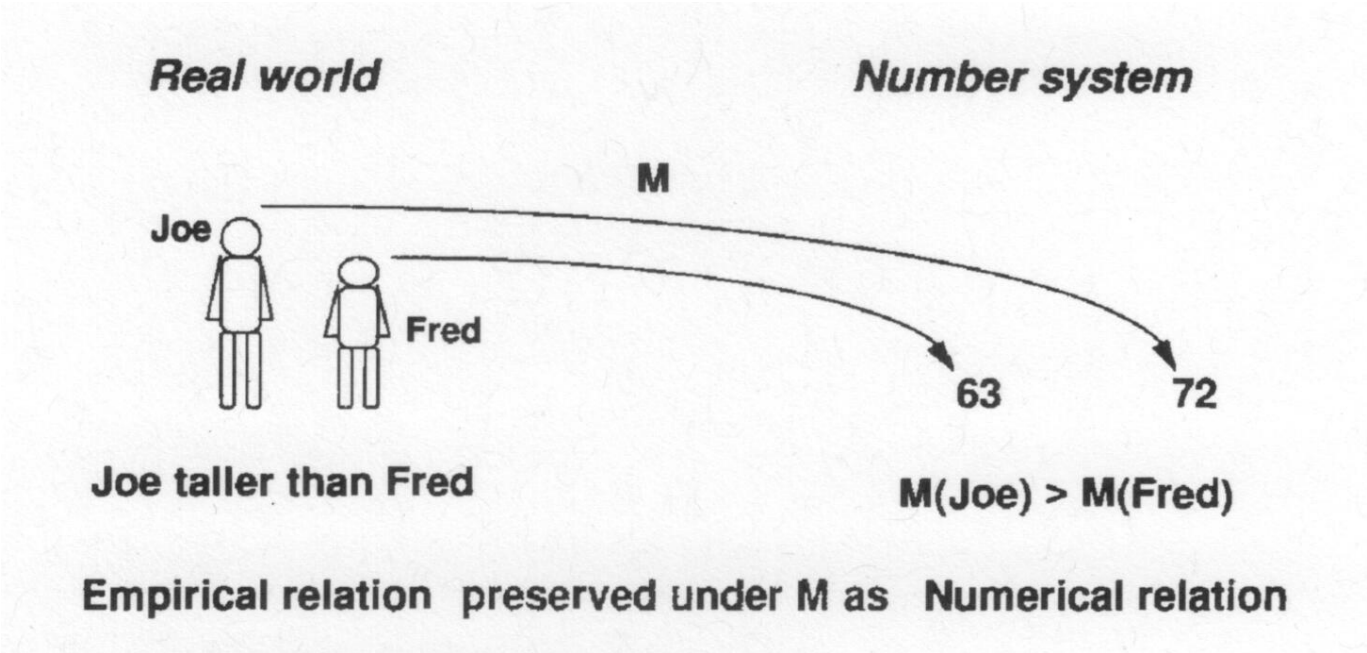
- When is testing enough?
- Reliance on structural coverage as a metric
- Coverage is a measure

Definition (Measurement)

A measurement is a mapping from the empirical world to the formal, relational world. Consequently, a measure is a value or a symbol assigned to an entity by this mapping in order to characterize an attribute.

Definition (Software Metrics)

A system or standard of measurement of software that gives a numerical (or symbolic) value to some attribute of interest.



	Entity	Attribute	Measure
1	Completed project	Duration	Months from start to finish
2	Completed project	Duration	Days from start to finish
3	Program code	Length	Number of lines of code (LOC)
4	Program code	Length	Number of executable statements
5	Integration testing process	Duration	Hours from start to finish
6	Integration testing process	Rate at which faults are found	Number of faults found per KLOC (thousand LOC)
7	Tester	Efficiency	Number of faults found per KLOC (thousand LOC)
8	Program code	Quality	Number of faults found per KLOC (thousand LOC)
9	Program code	Reliability	Mean time to failure (MTTF) in CPU hours
10	Program code	Reliability	Rate of occurrence of failures (ROCOF) in CPU hours

- Can we do exhaustive testing?
- How many tests are required for a decision with n inputs? 2^n
- If you could run 100 tests per second (automated) and you were to exhaustively test a decision with 36 inputs, how long it will take to execute all tests?
- Answer: $\approx$22 YEARS

Definition (Coverage)

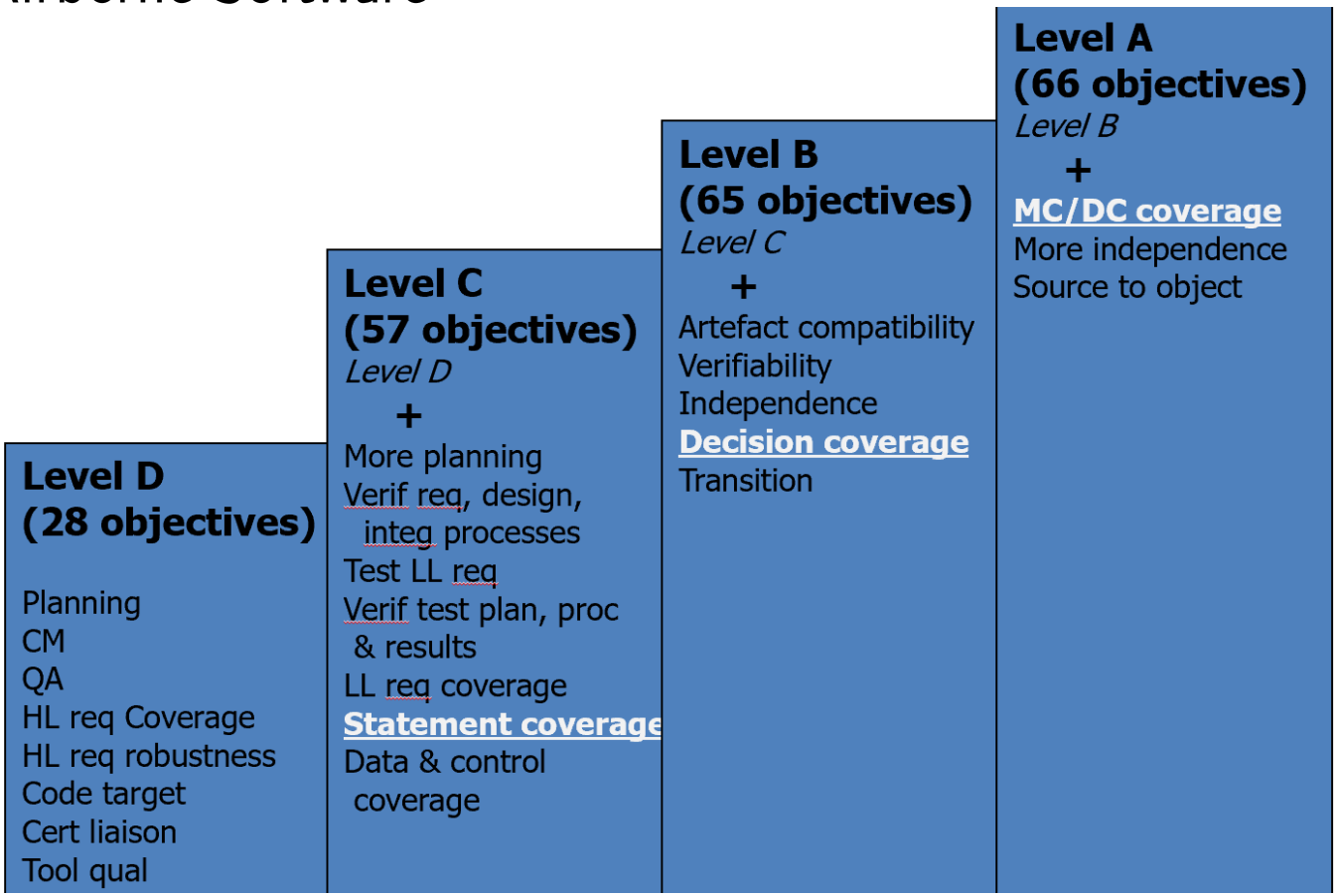
The proportion of the program structure that is exercised by a test case (or a test suite)

$$\text{Coverage} = \left(\frac{\text{Executed Outcomes}}{\text{Total Outcomes}}\right) \times 100\%$$

Code test coverage used as a criterion for sufficiency of testing

- Statement Coverage
- Branch Coverage
- Condition Coverage
- Condition/Decision Coverage
- Modified Condition/Decision Coverage

Airborne Software



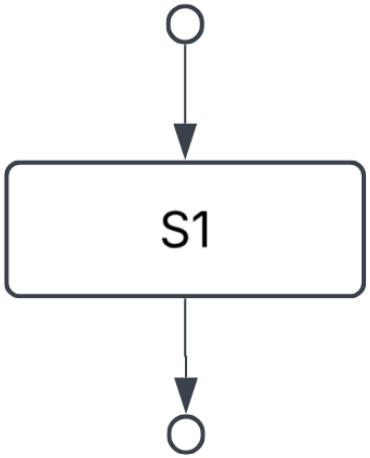
Automotive Software

Table 1 — Structural coverage metrics at the software unit level						
Methods			ASIL			
			A	B	C	D
1a	Statement coverage		++	++	+	+
1b	Branch coverage		+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)		+	+	+	++

Source: ISO 26262; ASIL (Automotive Safety Integrity Level)

A Control Flow Graph (CFG) can be built inductively

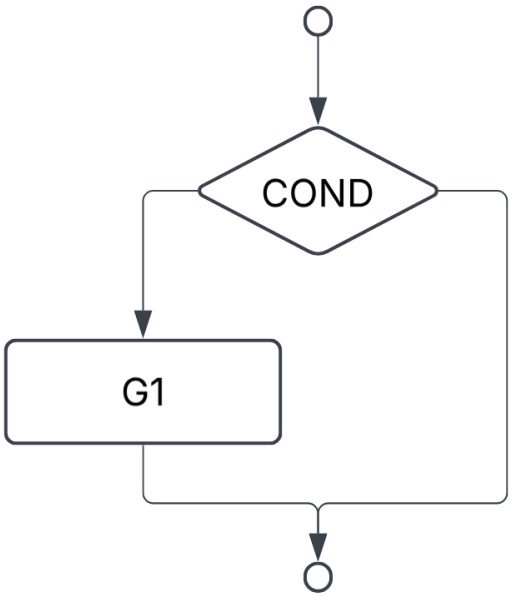
- The CFG of an elementary statement $S1$



- For any two statements, $S1, S2$, with corresponding CFGs $G1, G2$

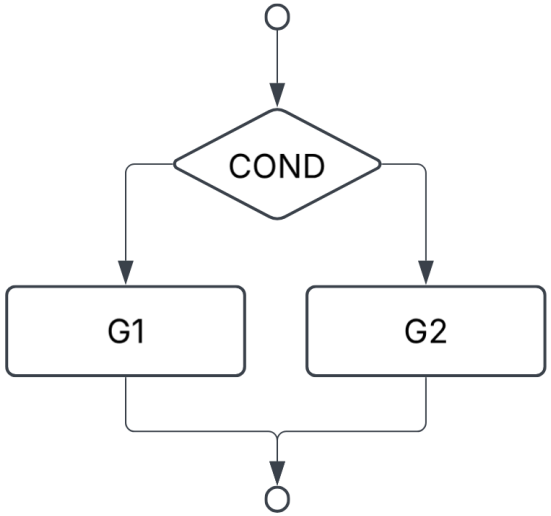
An *if – then* statement CFG

```
1  if COND then
     S1;
```



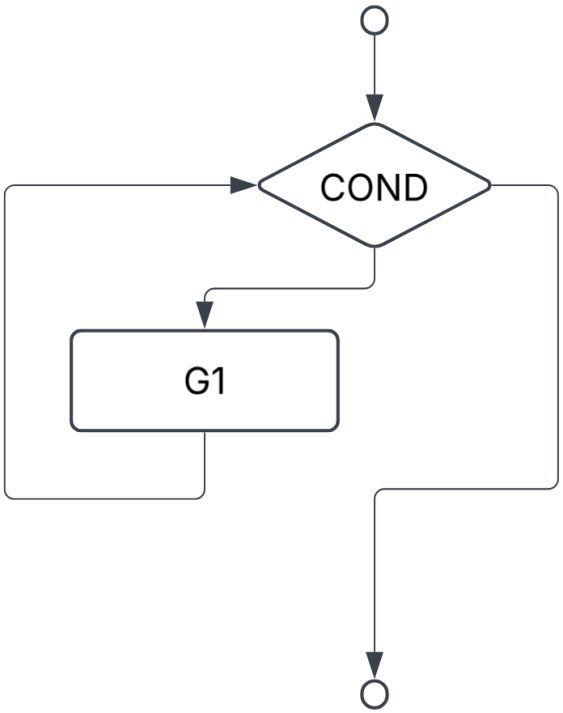
An *if – then – else* statement CFG

```
2  if COND then
     S1;
4  else
     S2;
```



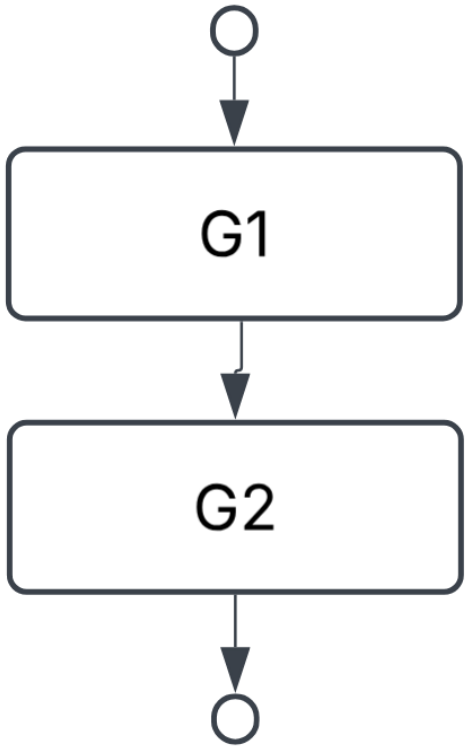
An *while* loop CFG

```
2          while COND loop
             S1;
           end loop;
```



A block statement CFG

```
1          S1;
           S2;
```



Definition (Path Coverage (Criterion))

The proportion of unique paths that are executed (reached) by the test set. A test set achieves path coverage if each path through the code is taken at least once.

$$\text{Path Coverage} = \left(\frac{\text{Executed Unique Paths}}{\text{Total Unique Paths}} \right) \times 100\%$$

Example

```
   if (A || B) {          // Decision 1
2     if (C && D) {        // Decision 2
        doW();            // Statement 1
4     }
      doX();              // Statement 2
6   } else {
      if (E) {            // Decision 3
8       doY();            // Statement 3
      }
10    if (F) {            // Decision 4
        doZ();            // Statement 4
12    }
   }
```

Exercise: Minimum number of tests to cover all paths

- Is Path Coverage same as Exhaustive testing?

  NO
- Path coverage ensures that each control flow path has been seen at least once
- But it doesn't ensure each condition takes all possible values
- Let's scale down

Definition (Statement Coverage)

A test set achieves statement coverage if each statement is executed at least once.

$$\text{Statement Coverage} = \left(\frac{\text{Executed Statements}}{\text{Total Statements}}\right) \times 100\%$$

- Exercise: Minimum tests to achieve 100% Statement Coverage

  (A=T, C=T, D=T), (A=F, B=F, E=T, F=T)
- How many unique test cases are?

  $2^6 = 64$
- Statement coverage is weak

Definition (Branch (or Decision) Coverage)

A test set achieves branch coverage if each branch (or CFG edge) is executed (reached) at least once

$$\text{Branch Coverage} = \left(\frac{\text{Executed Branches}}{\text{Total Branches}}\right) \times 100\%$$

- Exercise: Minimum tests to achieve 100% Branch Coverage (A=T, C=T, D=T), (A=T, C=F, D=T), (A=F, B=F, E=T, F=T), (A=F, B=F, E=F, F=F)
- Branch coverage is better that statement
- Is branch coverage same as path?
- Branch coverage ensures that each individual branch has been taken but there may be combinations that haven't been

- Coverage-based testing is structural (or syntactical)

```
1  if (averageMark >= 50 && hardFailCount == 0)
     grade = "pass";
3  else
     grade = "fail";
```

Test Cases:

- Input: (averageMark $\geqslant$ 50, hardFailCount == 0)
  Expected Output: grade == "pass"
- Input: (averageMark $<$ 50, hardFailCount == ANY)
  Expected Output: grade == "fail"
- Input: (averageMark == ANY, hardFailCount $>$ 0),
  Expected Output: grade == "fail"

```
    if (averageMark >= 50 && hardFailCount == 0)
2       grade = "pass";
    else
4       grade = "fail";
```

Branch coverage met by:

- (50, 0) – "pass"
- (49, 1) – "fail"

```
    if (averageMark >= 50 || hardFailCount == 0)
2     grade = "pass";
    else
4     grade = "fail";
```

Branch coverage met by:

- (50, 0) – "pass"
- (49, 1) – "fail"

Definition (Condition Coverage)

A test set achieves condition coverage if each individual condition in each decision evaluates to both 'true' and 'false' at least once.

$$\text{Condition Coverage} = \left(\frac{\text{Conditions evaluated T\& F}}{\text{Total conditions}} \right) \times 100\%$$

```
    if (averageMark >= 50 || hardFailCount == 0)
2     grade = "pass";
    else
4     grade = "fail";
```

Condition coverage met by:

- (50, 0) – "pass"
- (49, 1) – "fail"

Definition (Multiple Condition Coverage (MCC))

A test set achieves multiple condition coverage if each individual condition in each decision evaluates to both 'true' and 'false' at least once, in all combinations.

```
    if (averageMark >= 50 || hardFailCount == 0)
2     grade = "pass";
    else
4     grade = "fail";
```

MCC met by:

- (50, 0) – "pass"
- (49, 1) – "fail"
- (50, 1) – "pass" FAILURE! (Expected Output is different from Actual Output)
- (49, 0) – "pass" FAILURE!

How many tests do we need here?

```
    if (averageMark >= 50 ||
2       hardFailCount == 0 ||
        elephantCOunt > 3)
```

... here?

```
1   if (averageMark >= 50 ||
      hardFailCount == 0 ||
3     elephantCOunt > 3 &&
      specialString.equals('Treats'))
```

Definition (Modified Condition/Decision Coverage (MC/DC))

A test set achieves MC/DC if

- It achieves 100% branch (decision) coverage
- It achieves 100% condition coverage
- Each condition independently affects the decision outcome
- Each entry/exit point is executed

For n conditions

- MCC needs 2^n test cases
- MC/DC needs $n + 1$ test cases

How to find a minimal test set for MC/DC

```
    if ((A || B) && C)
2       ...
    else
4       ...
```

- Build the truth table of the decision point
- Changing one condition at a time, find a pair of tuples where the outcome is flipped
- Pick one of the tuple, and repeat the process for the other conditions, one at a time

Check `https://github.com/Armin-Montigny/MCDC`

TID	A	B	C	Decision
T1	F	F	F	F
T2	F	F	T	F
T3	F	T	F	F
T4	F	T	T	T
T5	T	F	F	F
T6	T	F	T	T
T7	T	T	F	F
T8	T	T	T	T

- MC/DC is less expensive than MCC
- Still expensive
- Source vs Object Code Coverage
- Structural Coverage is NOT Structural Testing

Definition (Criteria Subsumption)

Criterion A subsumes criterion B if every test set that satisfies A also satisfies B.

- Faults discovered by a test set that satifies A may not be discovered by a test set that satisfies B
- Branch coverage subsumes statement coverage

```
int doX(int x) {
  if (x > 0)
    x++;
  return 10/x;
}
```

Test Cases (driven from requirements):

- TC1: (x=1, Expected: 5)
- TC2: (x=0, Expected: 10)

Test set 1

doX(1) = 5 passed

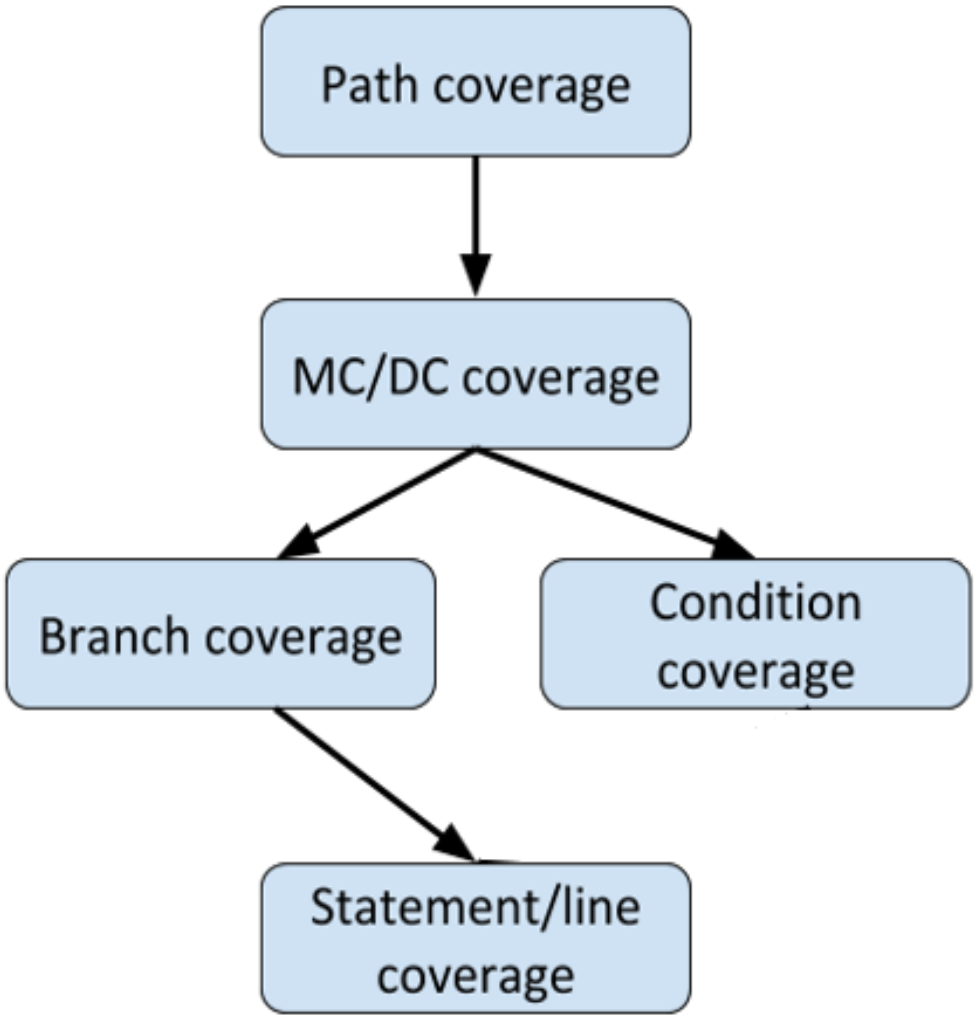
Achieves statement coverage

Test set 2

doX(1) = 5 passed

doX(0) throw exception! failed

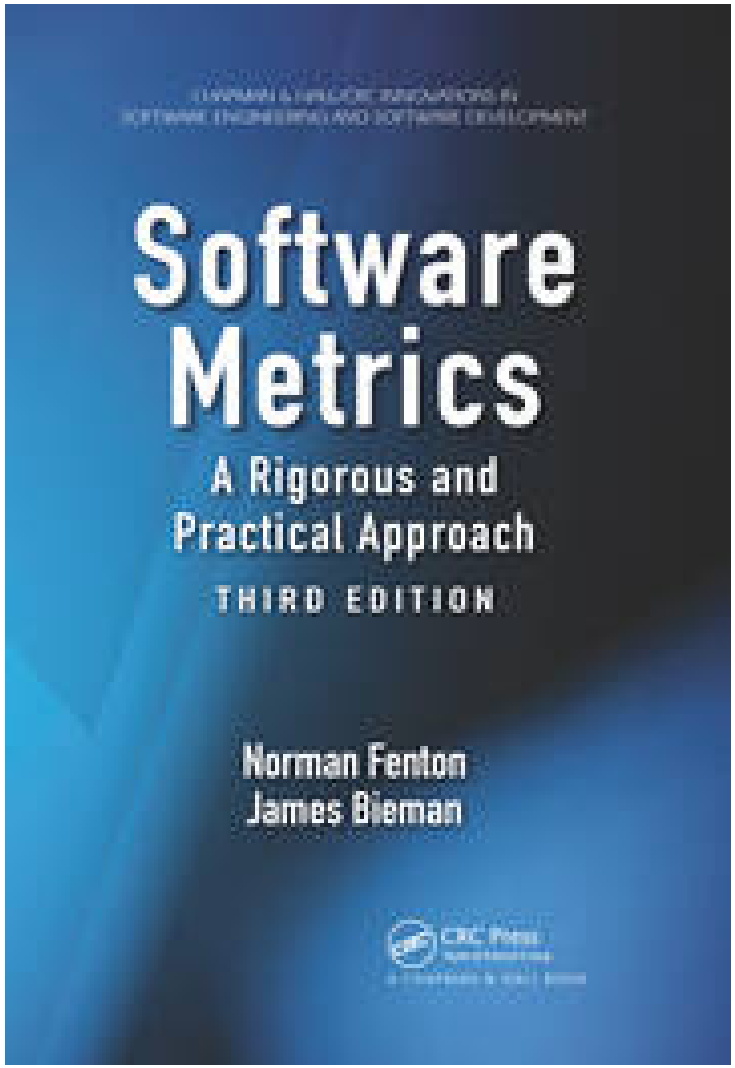
Achieves branch coverage



- Is the metric & evaluation method good for every project?
- What do we know what to focus our testing effort on?
  - Complexity metrics: criticality, # of states, coupling, coherence
  - Risk metrics

You can compute ...	Nominal	Ordinal	Interval	Ratio
Frequency distribution	Yes	Yes	Yes	Yes
Median and percentiles	**NO**	Yes	Yes	Yes
Add or Substract	**NO**	**NO**	Yes	Yes
Mean, standard deviation, standard error of the mean	**NO**	**NO**	Yes	Yes
Ratio, or coefficient of variation	**NO**	**NO**	**NO**	Yes

Table: List of what can be computed on a scale



References I

 Norman Fenton and James Bieman, *Software metrics: a rigorous and practical approach*, CRC press, 2014.