fill out parts form by EOD today!

18: More testing and coverage

Coverage

Notion of how completely a piece of code has been tested with a particular set of tests, with respect to a specific metric

Examples:

- What % of requirements have been tested?
- What % of lines of code have been tested?

100% coverage does **not** mean 100% tested, but it's a start to assess testing thoroughness

White box testing guided by coverage

Branch (aka decision) - *for every branch (e.g. if-statement), is there at least one test case that evaluates that branch to true and one that evaluates it to false?*

Condition - *like branch coverage, but looking at conditions* within *branches (e.g. looking at x > 0 and y == 2 separately rather than just x > 0 || y == 2)*

Path - *is there a test case that exercises every unique* path *through the code (as opposed to considering each branch independently)*

Branch coverage

```
if (x == 3 && y < 0 ) {
 // do something;
} else {
  // do something else
}

q = x + z;

if (q < y) {
  if (x == z) {
    // do another thing
  }
  // do a fourth thing
}
```

(x, y, z)	x==3 && y < 0	x + z < y	x == z
(0, 0, 0)	false	false	n/a
(1, 1, 3)	false	false	n/a
(3, -1, 0)	true	false	n/a
(3, -1, -5)	true	true	false
(3, 5000, 3)	false	true	true

Condition coverage

```
if (x == 3 && y < 0 ) {
 // do something;
} else {
  // do something else
}

q = x + z;

if (q < y) {
  if (x == z) {
    // do another thing
  }
}
```

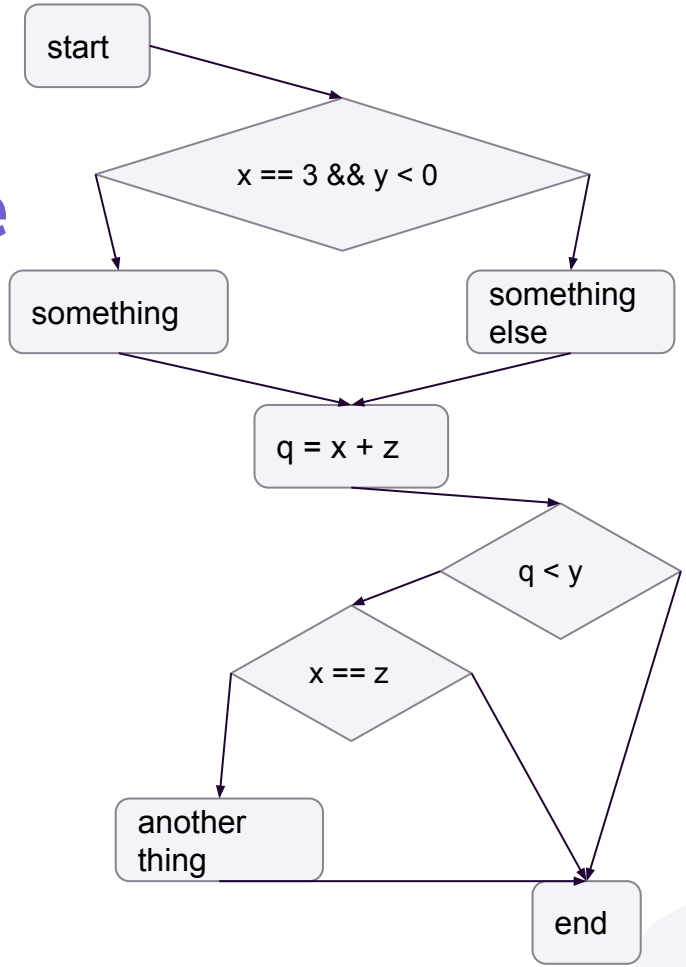
(x, y, z)	x==3	y < 0	x + z < y	x == z
(3, -1, 0)	true	true	false	n/a
(-1, 2, 1)	false	false	true	false
(3, 5000, 3)	true	false	true	true

Path coverage

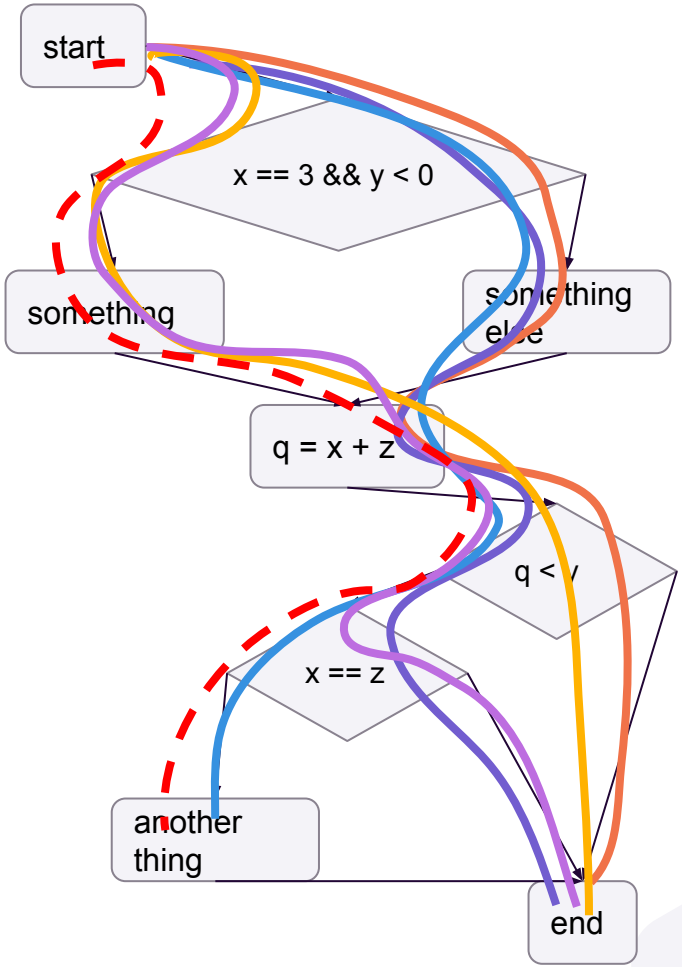
```
if (x == 3 && y < 0 ) {
 // do something;
} else {
  // do something else
}

q = x + z;

if (q < y) {
  if (x == z) {
    // do another thing
  }
}
```



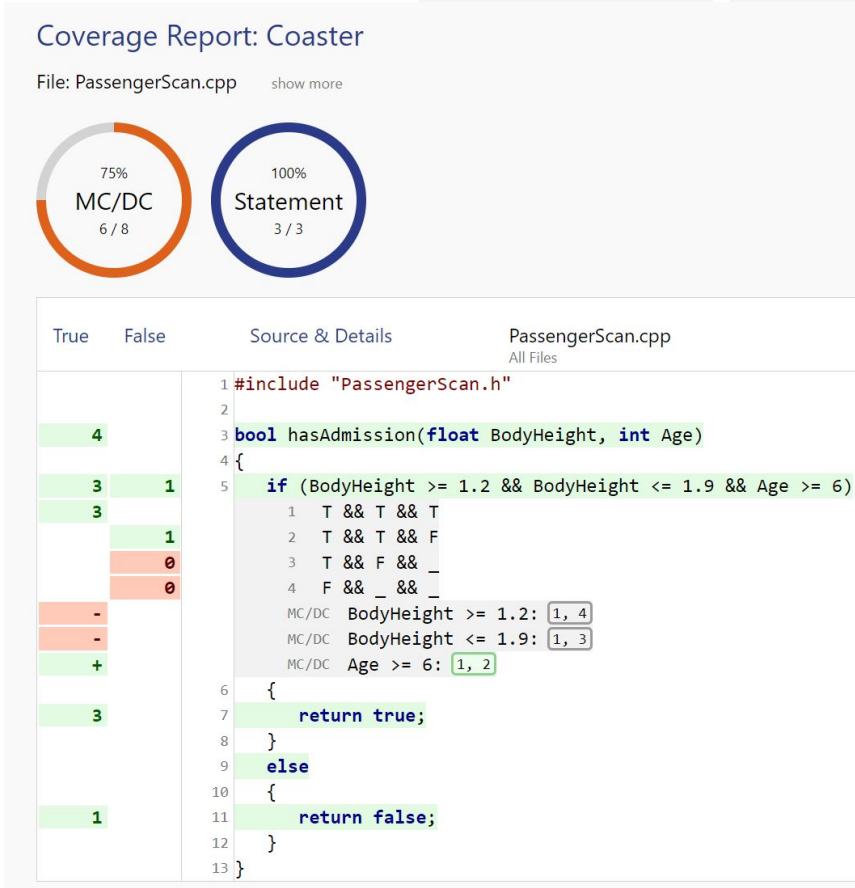
Six paths through the flowchart, but one is impossible according to the data



Overhead of coverage tools

article by Klaus Lambertz at embedded.com :

[Measuring code coverage for embedded software]



[image source]

Modified Condition/Decision Coverage (MC/DC)

A more comprehensive coverage metric required by some software safety standards

- Each entry and exit point is invoked
- Each decision takes every possible outcome <- branch coverage
- Each condition in a decision takes every possible outcome <- condition coverage
- **Each condition in a decision is shown to independently affect the outcome of the decision**

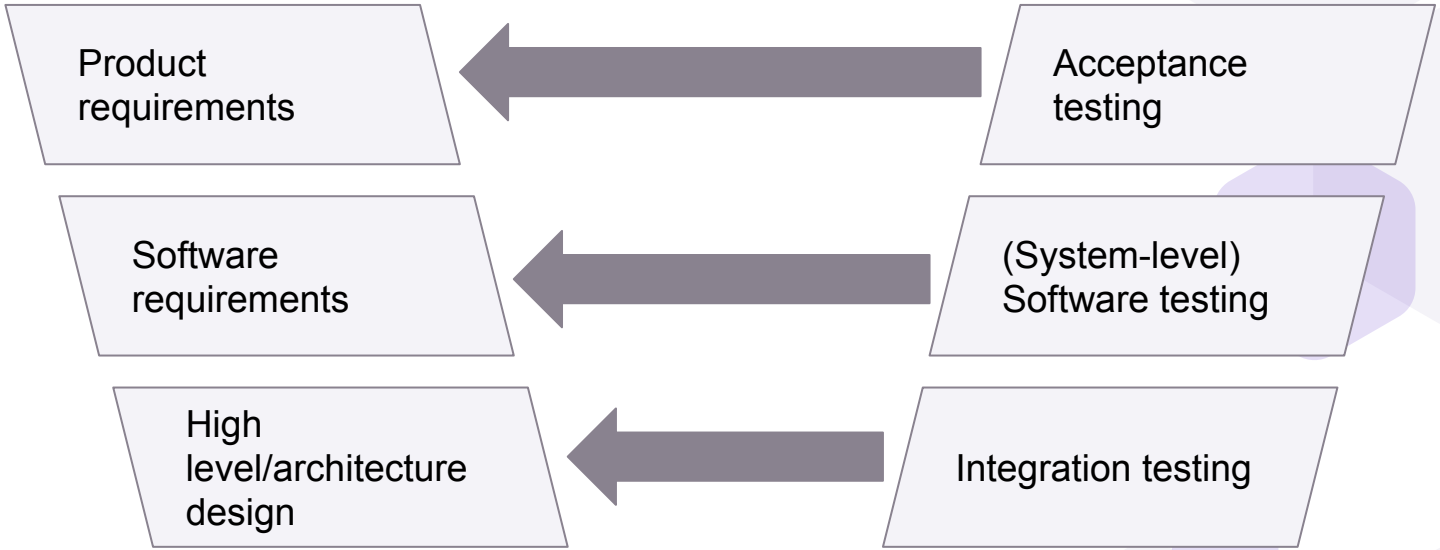
Each condition in a decision is shown to independently affect the outcome of the decision

Hold all but one condition constant. Does changing that condition affect the outcome of the decision?

```
(x + y) == 3 && (y < 0 || x == 2)
```

x	y	x + y == 3	y < 0	x == 2	decision
2	1	true	false	true	true
1	2	true	false	false	false
4	-1	true	true	false	true
3	-1	false	true	false	false

Rest of the V


Product requirements
Acceptance testing
Software requirements
(System-level) Software testing
High level/architecture design
Integration testing

Integration testing

Use high level design (architecture diagram and sequence diagrams) to test interfaces between modules/components

- Test every interface (message format, correctness of values)
- Test timing and sequence of messages sent
- Test that unexpected messages are handled

Assume modules are performing individual duties correctly (**why?**) and just test the *communication* between them

Sequence diagram test example

Scenario: check available funds at ATM