MC/DC coverage criterion

Modified Condition/Decision Coverage (MC/DC) is a good answer.

The MC/DC criterion looks at **combinations** of conditions, as path coverage does.

MC/DC instead of testing *all* possible combinations, identifies the *important* combinations that need to be tested.

If conditions have only **binary** outcomes (that is, *true* or *false*), the number of tests required to achieve 100% MC/DC coverage is N + 1, where N is the number of conditions in the decision.

Note that $(N+1) << 2^N$

MC/DC coverage criterion: an example

Conditions a, b, c are evaluated to Boolean values:

```
if(a && (b || c))
```

How many test for MC/DC? And for Path Coverage?

If we apply the MC/DC criterion only 3+1 = 4 tests cases will be enough, instead of 8 (path coverage criterion)

For each condition (e.g. ‘a’) we should look for:

- A case where the condition ‘a’ is `true` (T1)
- A case where the condition ‘a’ is `false` (T2)
- T1 and T2 must have different outcome (one true and one false)
- Variable ’b’ and ‘c’ in T1 must have the same value in T2

T1 and T2 are called independence pairs, because the variable ‘a’ independently influences the outcome (decision).

MC/DC coverage criterion: an example

For each condition (e.g. 'a') we should look for:
- A case where the condition 'a' is `true` (T1)
- A case where the condition 'a' is `false` (T2)
- T1 and T2 must have different outcome (one true and one false)
- Variable 'b' and 'c' in T1 must have the same value in T2

Test case	isLetter (a)	last == s (b)	last == r (c)	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
			false	false
			true	false
			false	false

T1 and T5 are a pair of test (an independence pair) where `isLetter` is the only parameter that is different and the outcome (decision) changes.

MC/DC coverage criterion: an example

For each condition (e.g. ‘a’) we should look for:
- A case where the condition ‘a’ is `true` (T1)
- A case where the condition ‘a’ is `false` (T2)
- T1 and T2 must have different outcome (one true and one false)
- Variable ’b’ and ‘c’ in T1 must have the same value in T2

Test case	isLetter	last == s	last == r	decision
T1	true	true	true	true
T2	true	true	false	true
T3	true	false	true	true
T4	true	false	false	false
T5	false	true	true	false
T6	false	true	false	false
T7	false	false	true	false
T8	false	false	false	false

MC/DC coverage criterion: an example

We found the following pairs:

- `isLetter`: {1, 5}, {2, 6}, {3, 7}
- `last` == s: {2,4}
- `last` == r: {3,4}

What pair to choose for each variable?

For the last two conditions is straightforward because we have just one pair, so we surely need tests: T2, T3, T4

What pair to pick for `isLetter`?

MC/DC coverage criterion: an example

We found the following pairs:

- `isLetter`: {1, 5}, {2, 6}, {3, 7}
- `last` == s: {2,4}
- `last` == r: {3,4}

For the last two conditions is straightforward because we have just one pair, so we surely need tests: T2, T3, T4

What pair to pick for `isLetter`?

If we pick {1, 5} we will have at the end 5 tests (T1, T2, T3, T4, T5)!

While with {2, 6}, {3, 7} makes no difference, in both cases we have only 4 tests (as expected) which is better than 8 (path coverage)!:

- {2, 6} -> T2, T3, T4, T6
- {3, 7} -> T2, T3, T4, T7

MC/DC coverage criterion: exercises

Practice yourself with MC/DC!

Exercise 1 (10 mins):

- Consider the expression (A & B) | C
  What the test suite to achieve 100% MC/DC coverage?

Test case	A	B	C	(A & B) | C
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	T
8	F	F	F	F

MC/DC coverage criterion: exercises

Solution of Exercise 1:

- Consider the expression (A & B) | C
  What the test suite to achieve 100% MC/DC coverage?

{2, 3, 4, 6}
{2, 4, 5, 6}

MC/DC coverage criterion: exercises

Exercise 2 (10 mins):

- Consider the expression A && (A || B)
  What the test suite to achieve 100% MC/DC coverage?

Test	A	B	Decison
1	F	F	F
2	F	T	F
3	T	F	T
4	T	T	T

MC/DC coverage criterion: exercises

Solution of Exercise 2:

- Consider the expression A && (A || B)
  What the test suite to achieve 100% MC/DC coverage?

A: {(1, 3), (2, 4)}
B: { (empty) }

1. There is no independence pair for B. Thus, it is not possible to achieve MC/DC coverage for this expression.
2. Since there is no independence pair for B, this parameter does not affect the result. You should recommend that the developer restructure the expression without using B, making the code easier to maintain.
3. This example shows that software testers can contribute to code quality not only by spotting bugs but also by suggesting changes that result in better maintainability.

MC/DC coverage criterion: exercises

Question 3:

- Consider the expression `(A && B)||( A && C)`

What do you notice?

MC/DC coverage criterion: exercises

Question 3:

- Consider the expression `(A && B)||( A && C)`

What do you notice?

It is impossible to change the first A without changing the second A!

In such cases, we allow A to vary, but we fix all other variables (this is called masked MC/DC).

MC/DC coverage criterion: exercises

Question 4:

- Consider the expression `(A && B)||( A && not B)`

What do you notice?

MC/DC coverage criterion: exercises

Question 4:

- Consider the expression `(A && B)||( A && not B)`

What do you notice?

It is impossible to achieve MC/DC in such expression

There are no pairs that show the independence of B

Revisit the expression to simply A

Loops boundary adequacy criterion

Given that exhaustive testing is impossible, how to handle:

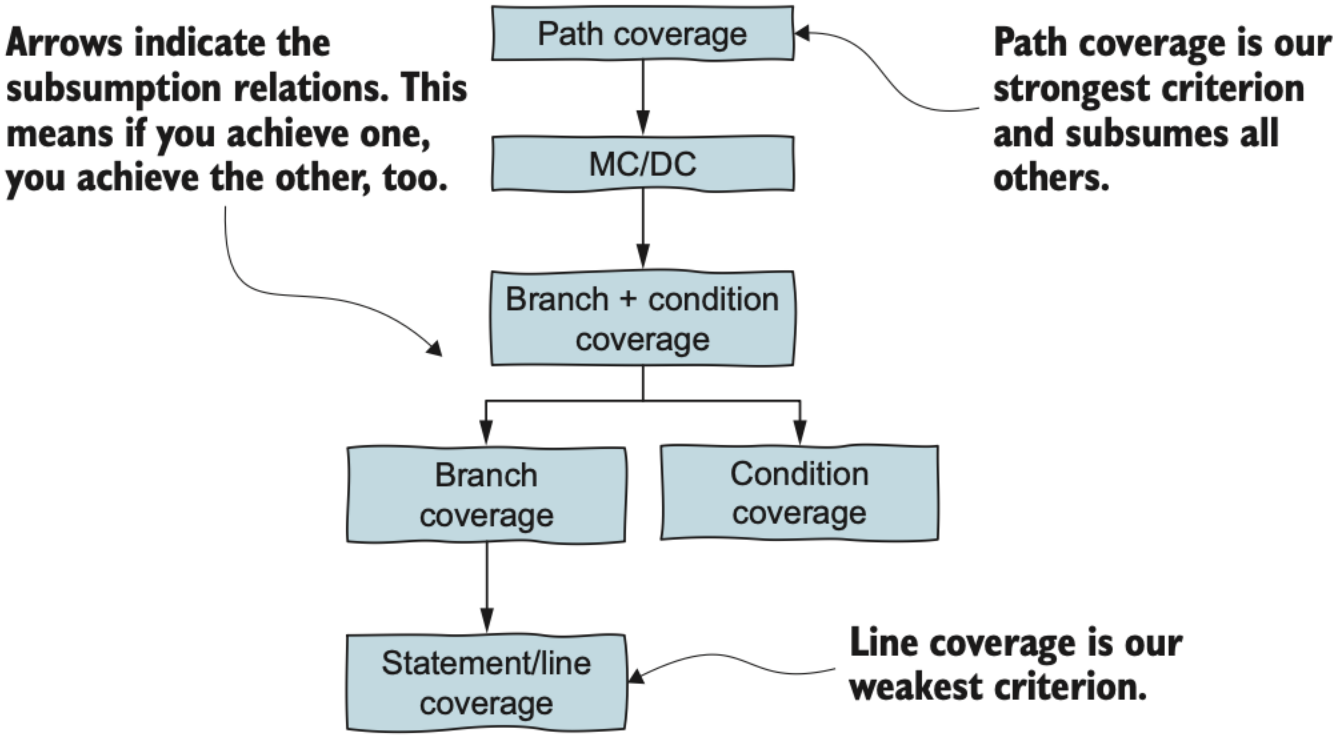
- A long-lasting loop (that runs for many iterations)
- An unbounded loop (that is executed an unknown number of times)

Loop boundary adequacy criterion:
a test suite satisfies this criterion if and only if for every loop

- There is a test case that exercises the loop zero times.
- There is a test case that exercises the loop once.
- There is a test case that exercises the loop multiple times.

NB: You can create also more than one test for the “multiple time” case.

Criteria subsumption



Weaker criterion: cheaper & faster, leave the code uncovered

Stronger criterion: higher cost, cover the code better

Branch vs Condition coverage

Example:

```
If(A||B)
```

T1= {true, false}
T2= {false, true}

What criterion are we satisfying?

Branch vs Condition coverage

Example:

```
If(A||B)
```

T1= {true, false}
T2= {false, true}

What criterion are we satisfying?

Condition, but not branch coverage!

Code Coverage (CC): summing up

- Not look at the coverage number blindly (do not take 100% as a goal, but try to understand the numbers)
- Not gaming the metric
- Structural testing & CC augment/complement specification-based testing:
  - Identify part of the code not covered yet by the test suite
  - Identify missed partitions
- Sometimes it is ok to leave some part of the code uncovered
- CC is not a *quality target* as it does not indicate a good test suite (100% coverage is not an indicator of how good a test suite is)
- If you have *low coverage* your system is NOT properly tested (but the vice versa is not true)

Code Coverage criteria

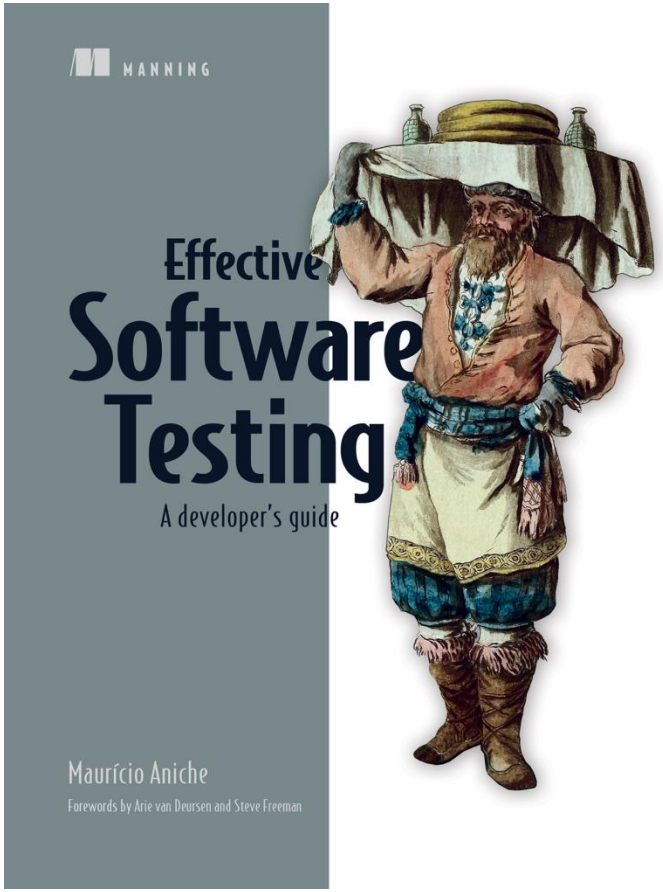
Rule of thumb:

- Start with branch coverage;
- If there are more complex expressions: evaluate if b+c or MC/DC is needed.

If expressions are too complicated, try to break them in smaller bits.

Reference book:

Effective Software Testing. A developer's guide. Mauricio Aniche. Ed. Manning. (**Chapter 3**)



References

- JetBrains IntelliJ IDEA Code Coverage documentation: https://www.jetbrains.com/help/idea/code-coverage.html

Azzurra Ragone

Department of Computer Science- Floor VI – Room 616
Email: azzurra.ragone@uniba.it