# Code Coverage → Modified Condition/Decision Coverage



## Example of Modified condition/decision coverage (MC/DC - MCDC)

### Simple example

Assume we want to test the following code extract:

```
if ( (A || B) && C )
{
    /* instructions */
}
else
{
    /* instructions */
}
```

where A, B and C represent atomic boolean expressions (i.e. not divisible in other boolean sub-expressions).

In order to ensure **Condition coverage** criteria for this example, A, B and C should be evaluated at least one time "true" and one time "false" during tests, which would be the case with the 2 following tests:

1. A = true / B = true / C = true
2. A = false / B = false / C = false

In order to ensure **Decision coverage** criteria, the condition ( (A ou B) et C ) should also be evaluated at least one time to "true" and one time to "false". Indeed, in our previous test cases:

1. A = true / B = true / C = true ---> la décision est évaluée à "true"
2. A = false / B = false / C = false ---> la décision est évaluée à "false"

and Decision coverage is also realized.

However, these two tests do not ensure a **Modified condition/decision coverage** which implies that each boolean variable should be evaluated one time to "true" and one time to "false", and this with *affecting the decision's outcome*. It means that from a test case to another, *changing the value of only one atomic condition will also change the decision's outcome*; although with only the two previous tests, it is impossible to know which condition influences the decision's evaluation...

In practice, for a decision with *n* atomic boolean conditions, we have to find at least *n+1* tests in order to be able to ensure Modified condition/decision coverage. As there are 3 atomic boolean conditions (A, B et C) in our example, we can (for instance) choose the following set of tests:

1. A = false / B = false / C = true ---> decision is evaluated to "false"
2. A = false / B = true / C = true ---> decision is evaluated to "true"
3. A = false / B = true / C = false ---> decision is evaluated to "false"
4. A = true / B = false / C = true ---> decision is evaluated to "true"

Indeed, in this case:

- between the 1st and 4th test scenarios, only A changed of value, which also made the decision's outcome change its value ("false" in the 1st case, "true" in the 2nd);
- in the same way, between 1st and 2nd, only B changed of value, which also made the decision's outcome change its value (passing from "false" to "true");
- eventually, between 2nd and 3rd, only C changed of value and decision's outcome's value also changed (passing from "true" to "false").

Besides, Decision and Condition coverage criteria are still respected (each boolean variable and the decision's outcome itself take at least one time the "true" and "false" values). The **Modified condition/decision coverage** is then ensured.

### More complex example

Assume we replace the condition: ( (A || B) && C )
by: ( ((u == 0) || (x>5)) && ((y<6) || (z == 0)) )

A full Test Coverage would consist into building the following truth table and testing each combination:

| Test case n° | A: (u == 0) | B: (x>5) | C: (y<6) | D: (z == 0) | ( (A \|\| B) && (C \|\| D) ) |
|---|---|---|---|---|---|
| 1 | F | F | F | F | F |

| | | | | | |
|---|---|---|---|---|---|
| 2 | F | F | F | T | F |
| 3 | F | F | T | _ | F |
| 4 | F | T | F | F | F |
| 5 | F | T | F | T | T |
| 6 | F | T | T | _ | T |
| 7 | T | _ | F | F | F |
| 8 | T | _ | F | T | T |
| 9 | T | _ | T | _ | T |

On the other hand, to ensure Modified condition/decision coverage, we should test (for instance) only the 5 combinations here-before underlined in yellow.

Go back to Test Coverage Levels

***Glossary***:

**Condition**: a logical indivisible (atomic) expression. It is often called boolean variable, represented by a capital letter (A, B, C, etc.), can only be equal to "true" or "false", but *can not be divided in other simpler "sub-conditions"*.

**Decision** : a logical expression which can be composed of several conditions separated by logical operators like "or", "and", "xor".

The Test Coverage Analyzer Testwell CTC++ can do measurements for all kinds of test coverage criteria (particularly for MC/DC and MCC) requested for "critical software development" in order to get certifications, such as DO-178B of the Federal Aviation Administration (FAA).

last updated: 11 February 2015


Author: Christophe Sourisse, Verifysoft Technology GmbH


*mailto:info*