MC/DC Software Testing

Some Context & Definitions

What is a unit? This whole thing is a "unit"

Is this unit safety-critical? Yes

When should units be tested?

- Early in development, as fu
- During "unit testing" phase

What is a decision? This is a Decision

What is a condition? These are Conditions ☺

- This is when MC/DC is done

How are they tested?

- Run each unit with differing input-> verify result
- MC/DC: Exercise all meaningful paths for "coverage"

```
1  boolean function check_abort(boolean   off_course,
                                boolean     abort_commanded,
                                boolean valid_abort_command)
2   {
3     boolean result = false;   // true if aborting;
4     if (off_course OR
5         (abort_commanded AND valid_abort_command))
6     then
7        result = true;   // initiate abort sequence
8     else
9        result = false;  // don't abort, keep flying
10    return result;
11  }
```

Sample Code

Four MC/DC Rules

Why? To make sure it works right and all paths re covered

1. Each entry and exit point is invoked (lines 1 and 10 are tested)
2. Each decision takes every possible outcome (lines 7 and 9 are tested, abort (true) or don't abort (false))

3. Each condition in a decision takes every possible outcome
   - Conditions need to be tested for both TRUE and FALSE

4. Each condition in a decision is shown to independently affect the outcome
   - Why? This tests each condition ***only when it matters*** – when it effects outcome
   - Two primary types of MCDC[1] -- may choose either of these, or a combination of both
     - Unique-Cause
     - Masking

Examples follow

“Unique Cause” MC/DC Example

Truth Table

Test Case	Conditions (Input Variables)			Result of Decision (abort=1 not=0)
	Off-course	Abort Cmd	Valid Cmd	
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

4 Green Test Cases for MC/DC

All 8 Test Cases for MC/DC

```
 1 boolean function check_abort(boolean   off_course,
                                boolean     abort_commanded,
                                boolean valid_abort_command)
…
 4     if (off_course OR
 5         (abort_commanded AND valid_abort_command))
…
```

Sample Code

Call the function 4 times with these values: Unit Test Code

```
r = check_abort(1,0,1);  // test case 6, r = abort
r = check_abort(0,0,1);  // test case 2, r = no abort
r = check_abort(0,1,1);  // test case 4, r = abort
r = check_abort(0,1,0);  // test case 3, r = no abort
```

- “Unique Cause” MCDC
  - Only one condition in a decision is changed at a time, ***holding all others constant,*** while verifying change in result
- Example:
  - The function “check_abort” needs to be run with **pairs of test cases (6,2,4,3), in that order**
    - *Start with Cases 6 and 2, changing only “off_course” yields a different result, verifying independence of “off course”*
    - *Next use test 2 and 4, changing only “abort” yields a different result, verifying “abort”*
    - *Last, run cases 4 and 3, which only changes “valid” for a different result*
- N+1 tests are needed (N is the number of conditions in the decision), vs. 2^N for all permutations
  - This is the *minimum set of meaningful test cases*

“Masking” MC/DC Example

Truth Table

Test Case	Conditions (Input Variables)			Result of Decision (abort=1 not=0)
	Off-course	Abort Cmd	Valid Cmd	
1	0	0	X	0
2	0	0	X	0
3	0	1	0	0
4	0	1	1	1
5	1	X	X	1
6	1	X	X	1
7	1	X	X	1
8	1	X	X	1

X’s are “don’t care” values since left-to-right evaluation is stopped

```
 1 boolean function check_abort(boolean   off_course,
                                boolean     abort_commanded,
                                boolean valid_abort_command)
…
 4     if (off_course OR
 5        (abort_commanded AND valid_abort_command))
…
```

Sample Code

Unit Test Code

```
Call the function 4 times with these values:
r = check_abort(0,1,0);   // test case 3, r = no_abort
r = check_abort(0,1,1);   // test case 4, r = abort
r = check_abort(0,0,X);   // cases 1 or 2,r = no_abort
r = check_abort(1,X,X);   // cases 5,6,7,or 8, r=abort
```

- “Masking” MCDC – relies on computer “short circuiting” – evaluation stopping as soon as a decision can be made
  - The condition under test changes, ***but others can change*** as long as they will be “short circuited” (marked ”don’t care” in table)
  - Any tests including both (3 and 4), either (1 or 2), or any one of (5,6,7,8) meet the criteria
    - (3,4,1,5), (3,4,2,5), (3,4,1,6), (3,4,2,6), (3,4,1,7), (3,4,2,7), (3,4,1,8), (3,4,2,8)
- Example:
  - *Start with cases 3 and 4 which both must be run since there are no short circuits to verify “valid command”*
  - *Next run either (1 or 2) and 4 since the result changes in both of those cases, to verify “abort cmd”*
  - *Last, run one case from (5,6,7,8) which changes “valid” for a different result*
- This will require $2*\sqrt{N}$ tests, less than in the “unique cause” form of MC/DC

- Further Reading & References:

  - [1] AJohn J. Chilenski. “An Investigation of Three Forms of the Modified Condition/Decision Coverage (MCDC) Criterion. Technical Report”, DOT/FAA/AR-01/18, US. Department of Transportation, Federal Aviation Administration, April 2001.

  - Standards requiring MC/DC testing for Safety-Critical code:
    - Aircraft - DO-178B (Safety-critical Level A or B)
    - Automotive - ISO-26262 (ASIL D)
    - Nuclear - IEC-61508-3 (SIL 1-3)
    - Spacecraft - NASA NPR-7150.2 (Class A Safety-critical)

- Provide comments to where you’ve seen this or to lorraine.e.prokop@nasa.gov

- Related follow-on “software shorts”
  - “Using gcov for MC/DC Testing”

Backup for MCDC

MC/DC Software Testing One-Pager

MC/DC (Modified Condition/Decision Coverage) Rules:

1. Each entry and exit point is invoked (lines 1 and 9 are tested)
2. Each decision takes every possible outcome (lines 6 and 8 are tested, abort and null)
3. Each condition in a decision takes every possible outcome (off_course, abort, and valid are each tested for both TRUE and FALSE)
4. Each condition in a decision is shown to independently affect the outcome - Why? This tests each condition ***only when it matters*** – when it effects outcome

- Three (3) types of MCDC[1] -- Unique-Cause (N+1 tests), Masking (2*Sqrt(n) tests), and Unique-Cause+Masking (NPR7150.2 does not dictate which to use)
  - "Unique Cause" MCDC holds all variables constant and changes only one at a time testing for unique outcome
    - Example: Only rows 6,2,4,3 in table need be tested (*in that order*) and represent one possible *minimum set of meaningful test cases (i.e Starting with row 6, changing only "off_course" in Test 2 results in a different outcome, then test 4 changes only "abort" with a different outcome, then test #3 changes "valid" for a different outcome)*
  - "Masking" MCDC allows other conditions to change as long as only the condition of interest influences the outcome (See backup for performance)
    - Masking MCDC is implemented by "short-circuiting" in computer programs (i.e. stopping as soon as test criteria are met)
    - Example: Any 4 tests including one of (5,6,7,8), both (3 and 4), and either (1 or 2) meet the criteria. (i.e. (5,1,3,4), (6,2,3,4), (8,1,3,4), etc…)

Sample Code

```
 1  function check_abort()
 2  {
 3    if (off_course OR
 4       (abort_commanded AND valid_abort_command))
 5    then
 6       abort();    // initiate abort sequence
 7    else
 8       null;       // do nothing
 9    return;
10  }
```

Test Case	Off-course	Abort Cmd	Valid Cmd	Outcome
1	0	0	0 X	0
2	0	0	1 X	0
3	0	1	0	0
4	0	1	1	1
5	1	0 X	0 X	1
6	1	0 X	1 X	1
7	1	1 X	0 X	1
8	1	1 X	1 X	1

Possible Test Cases:

- 8 Cases for full Condition Combination Coverage (CCC)
- 4 Green Row Cases (6,4,2,3 in that order) for "unique cause" MC/DC
- **X's indicate "don't care"** for "masking" MC/DC

- Standards requiring MC/DC testing for Safety-Critical code
  - Aircraft - DO-178B (Safety-critical Level A or B)
  - Automotive - ISO-26262 (ASIL D)
  - Nuclear - IEC-61508-3 (SIL 1-3)
  - Spacecraft - NASA NPR-7150.2 (Class A Safety-critical)

[1] AJohn J. Chilenski. "An Investigation of Three Forms of the Modified Condition/Decision Coverage (MCDC) Criterion. Technical Report", DOT/FAA/AR-01/18, April 2001.

MC/DC Software Training

Introduction Notes:

- *Who I am*

- *First in a series of software shorts on topics in software*

- *What is MC/DC? Modified Decision/Coverage testing – sounds sorta "questionable"*

    - *Way of testing the minimum set of all meaningful logic paths through the code*
        - *Not all logical paths through the code, but only the paths that effect the outcome*

- *Why do it?*
    - *Aside from it being required for safety critical code in space and aircraft, automotive, nuclear industries – it just makes sense!*
    - *Not to kill someone*

    How to do it?
    - *I will go over -*
        - *What is a unit*
        - *What are the rules of MC/DC*
        - *Provide simple examples of both forms of MC/DC*
            - *And a warning – I will show some short code – do not fear! I will step through it!*

- Hope this was helpful
- Provide comments to where you've seen this or to lorraine.e.prokop@nasa.gov
- Related follow-on "software shorts"
  - " "Using gcov for MC/DC Testing"
- Have a nice day!

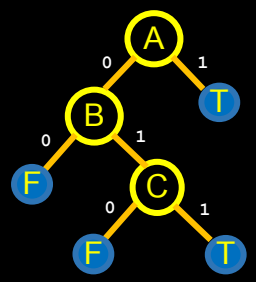
“Using gcov for MC/DC” One-Pager

- GCOV - Open-source tool used with gnu c compiler (gcc) to perform unit test code coverage analysis
  - Performs Object Branch Coverage (OBC) through generation of Binary Decision Diagrams (BDD) – see example
  - Object Branch Coverage had been proven[1] to be equivalent to MCDC if OBC contains all “tree-like” decisions
  - Does not identify *all* permutations to achieve Condition Combination Coverage (CCC) (which is a good thing, limiting tests to only those meaningful)
- Can gcov be used to meet MCDC criteria?
  - Yes, but some extra work must be done to ensure all decisions are “tree-like” (i.e. no graphs) – see example
    - All non-tree-like decisions can be converted to tree-like decisions[1]
      - Non tree-like decisions have been shown to be a VERY small percentage of overall decisions in several large code bases (<1% [4], “few” [2])
    - gcov performs “unique-cause+short-circuit” MC/DC, “short-circuit” being the implementation-equivalent to the “masking” form of MC/DC
    - The “taken at least once” statistic must be driven to 100% to achieve MC/DC coverage against all tree-like decisions
    - Open source tools are available to identify non-tree-like decisions
      - https://www.open-do.org/projects/couverture/, https://gitlab.com/gtd-gmbh/mcdc-checker/mcdc-checker

Sample BDD’s

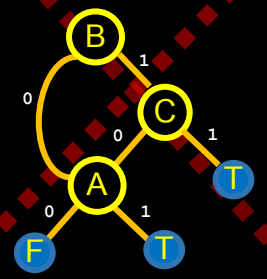
Tree-like ← rewrite — **Non-Tree-like**

`A or (B and C)`



- 6 branches/edges
- 4 tests needed to cover 4 leaves

`(B and C)or A`



- 6 branches/edges
- 3 tests needed to cover 3 leaves

```
GOV results running 3 tests: #3-010, 4=011 6=101; ABC)
Taken at least once:83.33% of 6   Taken at least once:100.00% of 6
```

- Guidelines
  - Must turn off all optimization; run gcc with the following switches
    - `gcc -O0 -fprofile-arcs -ftest-coverage`
  - Run “`gcov -b`” on output and drive “taken at least once” to 100%
  - Periodically or in CI
    - Run tool to identify non-tree-like branches
    - Rewrite all non-tree-like to be tree-like
    - Write additional unit tests to achieve 100% “taken at least once”

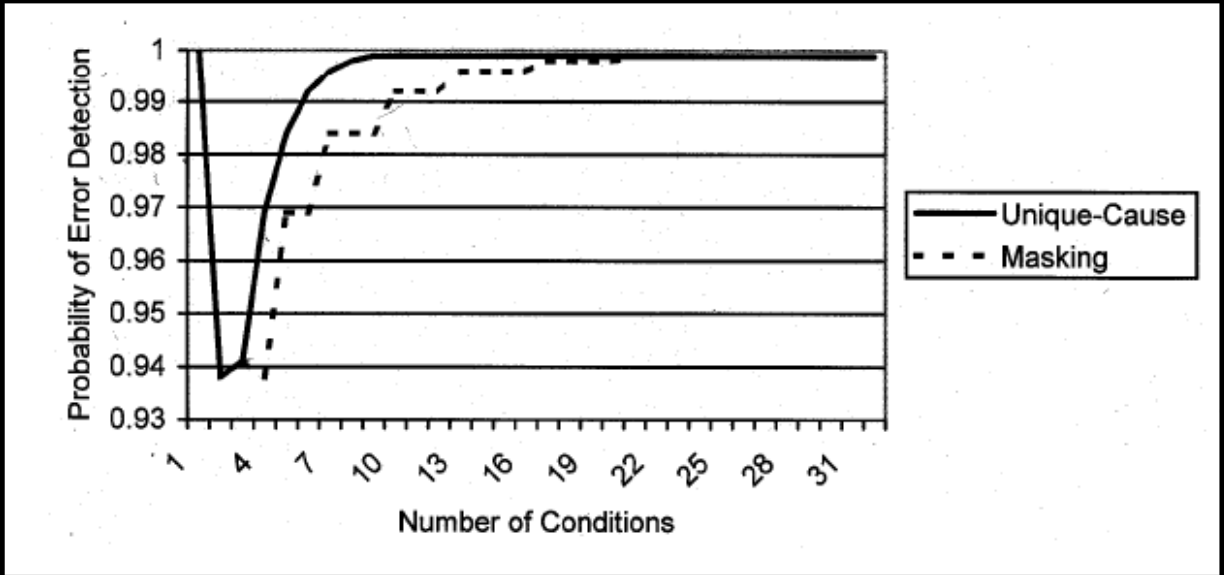
References:

- [1] Comar, Guitton, Hainque, and Quinot. “Formalization and Comparison of MCDC and Object Branch Coverage”, Embedded Real Time Software and Systems Conference, Feb 2012.
- [2] Thomas Wucher, Andoni Arregui. “MC/DC for Space: A new Approach to Ensure MC/DC Structural Coverage with Exclusively Open Source Tools”, ESA Software Product Assurance Workshop 2021.
- [3] AJohn J. Chilenski. “An Investigation of Three Forms of the Modified Condition/Decision Coverage (MCDC) Criterion. Technical Report”, DOT/FAA/AR-01/18, US. Department of Transportation, Federal Aviation Administration, April 2001.
- [4] Bordin, et.al. “Object and Source Coverage for Critical Applications with the COUVERTURE Open Analysis Framework”, Embedded Real Time Software and Systems Conference, May 2010.

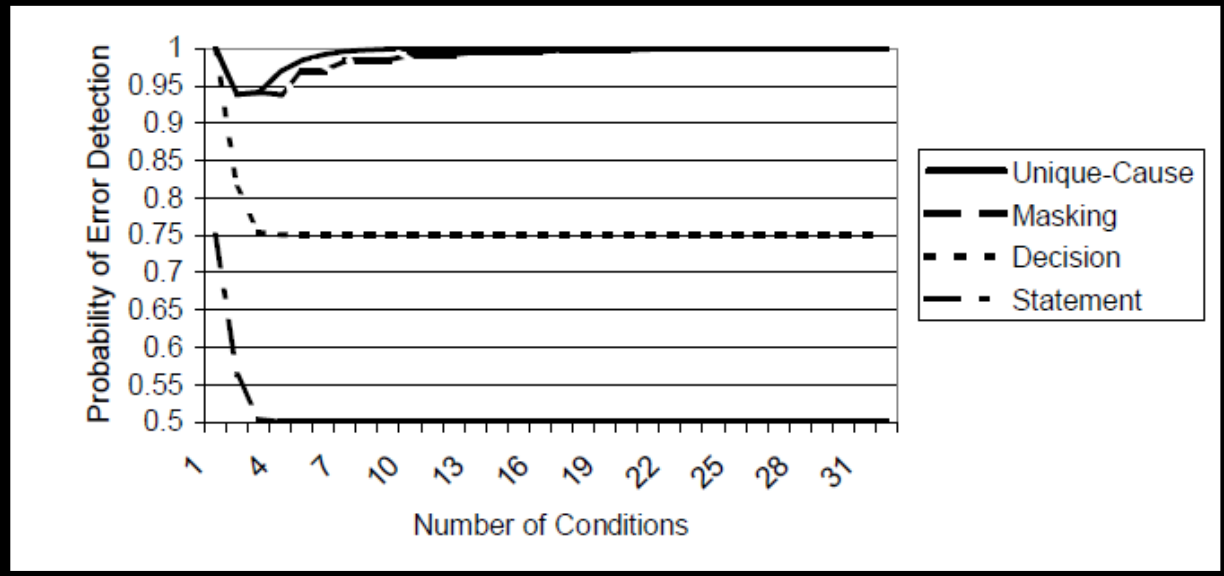
2023, Lorraine E. Prokop, Ph.D., NASA Technical Fellow for Software

Backup

Effectiveness of Masking vs. Unique-cause MCDC



Effectiveness of MCDC types vs. other coverages



Reference: AJohn J. Chilenski. "An Investigation of Three Forms of the Modified Condition/Decision Coverage (MCDC) Criterion. Technical Report", DOT/FAA/AR-01/18, US. Department of Transportation, Federal Aviation Administration, April 2001.