Certification Authorities Software Team (CAST)

Position Paper CAST-10

What is a "Decision" in Application of Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC)?

Completed June 2002

What is a “Decision” in Application of Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC)?

Abstract:

The purpose of this paper is to present the certification authorities’ position of what a decision is when applying Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC).

Key words:

Decision, decision coverage (DC), modified condition/decision coverage (MC/DC), structural coverage, branch, branch point, code, code-control construct, condition, Boolean expression

1 Purpose

The purpose of this paper is to present the certification authorities’ position of what a decision is when applying Modified Condition/Decision Coverage (MC/DC) and Decision Coverage (DC).

2 Background

There has been confusion regarding the definition of “decision” in DO-178B/ED-12B [1]. The definition affects the application of decision coverage (DC) for Levels B and A software and modified condition/decision coverage (MC/DC) for Level A software.

DO-178B/ED-12B includes the following definitions [1]:

> *Condition - A Boolean expression containing no Boolean operators.*
>
> *Decision - A Boolean expression composed of conditions and zero or more Boolean operators. A decision without a Boolean operator is a condition. If a condition appears more than once in a decision, each occurrence is a distinct condition.*
>
> *Decision Coverage - Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.*
>
> *Modified Condition/Decision Coverage - Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect*

that decision's outcome. A condition is shown to independently affect a decision's outcome by varying just that condition while holding fixed all other possible conditions.

The confusion centers around the "traditional" industry understanding of "branch point" (which is typically an if-then-else, do-while, or case statement) versus the DO-178B/ED-12B literal definition of "decision."

IEEE Std 100-1992, Standard Dictionary of Electrical and Electronic Terms, includes the following relevant definitions, which illustrate the "traditional" industry terminology [2]:

- ***branch (1) (software).*** *(A) A computer program construct in which one of two or more alternative sets of programs statements is selected for execution. See also:* ***case; jump; go to; if-then-else.*** *(B) A point in a computer program at which one of two or more alternative sets of program statements is selected for execution. Syn:* ***branchpoint.*** *(C) Any of the alternative sets of program statements in (A). (D) To perform the selection in (A)." …* ***(6) (electronic computer).*** *(A) A set of instructions that are executed between two successive decision instructions. (B) To select a branch as in (A). (C) Loosely, a conditional jump. See* ***conditional jump.***
- ***branch testing.*** *Testing designed to execute each outcome of each decision point in a computer program. Contrast with:* ***path testing; statement testing.***
- ***path testing.*** *Testing designed to execute all or selected paths through a computer program. Contrast with:* ***branch testing; statement testing.***
- ***control statement (software).*** *A program statement that selects among alternative sets of programs statements or affects the order in which operations are performed. For example, if-then-else, case. Contrast with:* ***assignment statement, declaration.***

In the creation of the MC/DC tutorial (a joint effort between NASA, FAA, and several industry participants), this particular issue was discussed at a high level. The MC/DC tutorial focuses specifically on MC/DC and states: *"a decision is not synonymous with a branch point. MC/DC applies to all decisions – not just those within a branch point"* ([3], page 13). The tutorial was reviewed by several industry participants and SC-190/WG-52 members and found to be satisfactory for MC/DC application.

Basically, for MC/DC, the tutorial indicates that a decision includes the traditional branch points plus Boolean operations that appear in assignment statements, actual parameters, indexers, aggregates, etc. For example, both assignment statements below contain decisions, even though neither of the statements are branch points such as an "if" statement (reference page 13 of the MC/DC tutorial):

```
A := B or C;

E := A and D;
```

The MC/DC tutorial supports the "literal" definition of decision in DO-178B/ED-12B. It should also be noted that DO-178B/ED-12B also asks for entry and exit point coverage, which is also not part of the "traditional" branch coverage.

The following concepts illustrate the "literal" definition of decision [4].

1. Structural coverage guidelines are:
   a) Every statement in the program has been invoked at least once;
   b) Every point of entry and exit in the program has been invoked at least once;
   c) Every control statement (i.e., branchpoint) in the program has taken all possible outcomes (i.e., branches) at least once;
   d) Every non-constant Boolean expression in the program has evaluated to both a True and a False result;
   e) Every non-constant condition in a Boolean expression in the program has evaluated to both a True and a False result;
   f) Every non-constant condition in a Boolean expression in the program has been shown to independently affect that expression's outcome.

2. Based upon these definitions:
   - Statement Coverage requires (a) only
   - DC requires (b, c, d)
   - MC/DC requires (b, c, d, e, f)

The issue is that at least some industry participants are not applying this "literal" definition of decision. I.e., some industry participants are equating branch coverage and decision coverage, leading to inconsistency in the interpretation and application of DC and MC/DC in the industry. Tool manufacturers, in particular, tend to be inconsistent in the approach, since many of them come from a "traditional" testing background (using the IEEE definitions), rather than an aviation background.

There are a number of potential reasons for this inconsistency:

- Lack of clarification and training materials on DO-178B/ED-12B
- The difference from the aviation community and the traditional software testing community
- Lack of agreement among the authors of DO-178B/ED-12B themselves
- Use of commercial verification tools that are developed outside the aviation industry

There seems to be a variety of opinions about the definition of decision and coverage requirements. Three common opinions prevail:

1) Some support the "literal" definition of DO-178B/ED-12B definition for decision coverage (i.e., a decision is more than a branch point). The DO-178B "literal" definition of decision is: *A Boolean expression composed of conditions and zero or more Boolean operators. A decision without a*

Boolean operator is a condition. If a condition appears more than once in a decision, each occurrence is a distinct condition.

2) Some believe that the "intended" definition of DO-178B/ED-12B is different than the "literal" definition (i.e., branch coverage is equal to decision coverage).
3) Some apply the "literal" definition of decision for MC/DC and the "relaxed" definition for DC. See the following section for examples of this "relaxed" decision coverage interpretation.

There is variance in the interpretation of the meaning of decision, even among those who were on the joint RTCA/EUROCAE committee that produced DO-178B/ED-12B.

3 An Example To Illustrate the Dilemma

Consider the following code in order to illustrate the difference between "branch point" and DO-178B/ED-12B's definition of "decision":

```
A := B or C;
E := A and D;
if E then …
```

3.1 For the "literal" definition of decision for MC/DC, all of the following must be shown:

- A has been assigned both True and False (item "d" from section 2 guidelines above);
- B has the correct independent effect on A (items "e" and "f" from section 2 guidelines above);
- C has the correct independent effect on A (items "e" and "f" from section 2 guidelines above);
- E has been assigned both True and False (item "e" from section 2 guidelines above);
- A has the correct independent effect on E (item "f" from section 2 guidelines above);
- D has the correct independent effect on E (items "e" and "f" from section 2 guidelines above); and
- E has the correct independent effect on the if-then statement (i.e., E has been both True and False, and the if-then statement has branched True and False accordingly) (item "c" from section 2 guidelines above).

This applies whether the assignments (Boolean expressions) are contained in the same code component as the if-then statement, or in a different code component. This illustrates one of the major benefits of the “literal” definition for MC/DC: **no matter how the logic is distributed across the system’s computer program and modules, MC/DC will address it**.

3.2 If “branch point” equals “decision” for MC/DC, it must be shown that:

- E has the correct independent effect on the if-statement (i.e., E has been both True and False, and the if-statement has branched True and False accordingly) (item “c” from section 2 guidelines above).

This interpretation allows significantly weaker verification to be performed on logic. In fact, one can now write code using temporaries for logic so that there would be no difference between DC and MC/DC.

3.3 For the “literal” definition of decision for DC, it must be shown that:

- A has been assigned both True and False (item “d” from section 2 guidelines above);
- E has been assigned both True and False (item “d” from section 2 guidelines above); and
- E has the correct independent effect on the if-statement (i.e., E has been both True and False, and the if-statement has branched True and False accordingly) (item “c” from section 2 guidelines above).

The third coverage guideline subsumes the second; therefore, there are two essential coverage guidelines for DC with the literal definition. Note that if the three statements are in the same component, the third coverage guideline ensures that A has been assigned a True value, satisfying half of the first coverage requirement. If the three statements are distributed across different components, then this may no longer be true unless the coverage analysis is performed with integrated components, rather than stand-alone components.

3.4 If “branch point” equals “decision” for DC, it must be shown that:

- E has the correct effect on the if-statement (i.e., E has been both True and False, and the if-statement has branched True and False accordingly) (item “c” from section 2 guidelines above).

This interpretation allows weaker verification to be performed on logic. The weakening is proportional to the number of Boolean expressions flowing into the branch point (i.e., no weakening if a single expression of atomic values flows into the branch point), how many components the expressions are distributed across, and whether coverage analysis is performed with fully integrated components or stand-alone components.

4 Certification Authorities' Position on the "Literal" Definition of Decision

The certification authorities recommend the "literal" definition of decision for DC and MC/DC. The logic and control structure in Levels A and B software must be thoroughly exercised, whether it occurs at a branch point or not. To equate decision and branch point could allow Levels A and B software to be coded with all Boolean expressions outside of the components' code-control constructs (e.g., with all Boolean operators in assignment and declaration statements, possible in a data component or package, which is not considered to be a functional component, and therefore, may be considered exempt from the structural coverage criteria). This could significantly weaken the verification effort and test coverage achieved. This is clearly not acceptable to satisfy the objectives of DO-178B/ED-12B.

5 Certification Authorities' Position on Alternatives to The Literal Definition

DO-178B/ED-12B is recognized by Advisory Circular 20-115B and Temporary Guidance Leaflet #4 as "a means but not the only means" for compliance with the regulations, when software is used [5, 6]. DO-178B/ED-12B also addresses the use of alternate methods (section 12.3). Because of this, some manufacturers have proposed branch coverage as an alternative to decision coverage (not for MC/DC).

DO-178B/ED-12B (section 12.3), DO-248B/ED-94B (section 4.5) [7], and CAST-5 [8] position paper provide some insight on both alternative methods and alternative means. Each alternative method must be proposed, justified, and evaluated on a case-by-case basis and must obtain the concurrence of the approving certification authority.

Each application of an alternative means or method may have slightly different nuances to be evaluated. Some typical things to consider if branch coverage is proposed as an alternate means or method for decision coverage are listed below (Note: this is not an all-inclusive list):

- The developer should generate the test cases from the requirements.
- The developer should use normal range and robustness test cases to verify the correct variable usage and Boolean operators for all software requirements expressed by logical expressions (per 6.4.2.1d of DO-178B/ED-12B).
- The developer should verify correct loop operations and correct logic decisions (per 6.4.3 of DO-178B/ED-12B).
- The developer should establish and enforce standards and processes to make sure that the alternative method or means is not being abused (i.e., using Boolean expressions outside of the components' code-control constructs to reduce the structural coverage effort).

- Standards should be established and reviews should be performed to ensure that the designer and/or programmer (either human or machine) is not intentionally nor consistently using Boolean expressions outside of the components' code-control constructs to reduce the verification testing and structural coverage analysis efforts (i.e., abusing the relaxation).
- The developer should perform additional testing and structural coverage analysis, if the alternative method or means is not adequate (i.e., additional test cases and manual structural coverage analysis may be needed to address specific instances of violations discovered within the code).

6 References

[1] RTCA, Inc. document DO-178B and EUROCAE document ED-12B, "Software Considerations in Airborne Systems and Equipment Certification", dated December 1, 1992.

[2] IEEE Std 100-1992, "The New IEEE Standard Dictionary of Electrical and Electronics Terms," Fifth Edition

[3] "A Practical Tutorial on Modified Condition/Decision Coverage," published jointly be NASA and FAA in 2001.

[4] Information based on John Chilenski's (of The Boeing Company) Structural Coverage Course.

[5] AC 20-115B, *RTCA, Inc. Document RTCA/DO-178B*, dated January 11, 1993.

[6] Temporary Guidance Leaflet Number 4 (TGL No. 4), which calls out ED-12B for JAA.

[7] RTCA, Inc. document DO-248B and EUROCAE document ED-94B, "Final Report for Clarification of DO-178B (/ED-12B) 'Software Considerations In Airborne Systems and Equipment Certification'", 2001

[8] CAST-5 Paper, "Guidelines for Proposing Alternate Means of Compliance to DO-178B," Completed June, 2000