UDC 004.891.3

S.A. VILKOMIR

Software Quality Research Laboratory, University of Limerick, Ireland

USING MC/DC AND RC/DC CRITERIA FOR SPECIFICATION-BASED TESTING OF SAFETY-CRITICAL SOFTWARE

Software testing coverage criteria for logical expressions are considered including a new Reinforced Condition/Decision Coverage (RC/DC) criterion. This new criterion has been developed from the well-known Modified Condition/Decision Coverage (MC/DC) criterion and is more suitable for the testing of safety-critical software where MC/DC may not provide adequate assurance. Specific examples of using these criteria for specification-based testing are addressed.

software testing criteria; MC/DC; RC/DC; Specification-based testing

Introduction

The methods and criteria of software testing are traditionally divided into structural (or white-box) and functional (or black-box) aspects [1, 2]. Structural testing criteria (i.e., criteria that take into account an internal structure of the program) are in turn divided into data-flow and control-flow criteria.

Control-flow criteria, in particular, examine logical expressions, which determine the branch and loop structure of the program. When logical expressions are used for software specification, the same control-flow criteria could be used for specification-based testing. This group of criteria is considered in the paper.

This paper is based on the author's previous results [3, 4] and is structured as follows. Section 'MC/DC' presents the use of control-flow criteria for specification-based testing and considers the definition of the Modified Condition/Decision Coverage (MC/DC) criterion [5]. This criterion is used mainly for testing of safety-critical avionics software and is the most complicated and controversial control-flow criterion.

In the next section, we analyze a major shortcoming of the MC/DC criterion, namely the deficiency of requirements for the testing of the 'false operation' type of failures. Examples of failures of this type are considered to illustrate the problem. These have an especially vital importance for safety-critical applications.

Section 'RC/DC' presents the definition of a new Reinforced Condition/Decision Coverage (RC/DC) criterion, which eliminates the shortcoming of MC/DC. The central point is the requirement that each condition in a decision is shown to be varied without changing the outcome of the decision.

MC/DC

Using control-flow criteria for specification-based testing. The aim of control-flow criteria is to help in testing *decisions* (the program points at which the control flow can divide into various paths) and *conditions* (atomic predicates which form component parts of decisions) in a program. The simplest control-flow criteria were formulated in the 1960s and 1970s. The following are based on the well-known book by G. Myers [1]:

– statement coverage: every statement in the program has been executed at least once;

– decision coverage: every statement in the program has been executed at least once, and every decision in the program has taken all possible outcomes at least once;

– condition coverage: every statement in the program has been executed at least once, and every condition in each decision has taken all possible outcomes at least once;

– multiple condition coverage (MCC): every statement in the program has been executed at least once, and all possible combinations of condition outcomes in each decision have been invoked at least once.

The MCC criterion is the strongest and requires full searching of various combinations of conditions values that is not normally possible in practice. The other criteria mentioned above are weaker and require considerably less test patterns that is not sufficient for safety-critical software [6]. As a compromise, the MC/DC criterion has been proposed [7, 5].

Definition of MC/DC. The definition of the MC/DC criterion, according to [5], is the following:

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

The main part of the MC/DC definition is '*each condition has been shown to affect the decision's outcome independently'*. The key word in this definition is 'independently'; i.e., the aim of MC/DC is the elimination during testing of the mutual influence of the individual conditions and the testing of the correctness of each condition separately.

Investigation of MC/DC has initially been considered in [7, 8]. Detailed consideration of the different aspects of this criterion was carried out more recently in [9, 10, 11, 12, 13]. Different forms of this criterion, e.g. *Masking MC/DC* [14] were proposed. As an example illustrated the definition of MC/DC, consider decision $d = A \wedge B \wedge C \wedge D$ where A, B, C and D are conditions. Two test cases are required to test condition A: ($A = 1, B = 1, C = 1, D = 1$) when $d = 1$ and ($A = 0, B = 1, C = 1, D = 1$) when $d = 0$. Similar test cases are required to test conditions B, C and D. Totally, five test cases are required according the MC/DC definition.

A case study of using MC/DC for specification-based testing. A case study of the MC/DC use for specification–based testing of a nuclear reactor protection system has been considered in [4]. Here we consider a new example for the following specification of the same system: the system should shut down a reactor when two from four circulation pumps are out of operation or in the case of decrease of the water level more than 650 mm in any one steam generator provided that the corresponding circulation pump operates normally.

Let L_i and C_i ($i = 1 \ldots 4$) be conditions to describe correspondently the level of water and the operation of circulation pumps:

$L_i = 0$ – the level is normal;

$L_i = 1$ – the level is decreased;

$C_i = 0$ – the pump is in normal operation;

$C_i = 1$ – the pump is out of operation.

The decision that is responsible for this specification of the actuation is:

$$d = (C_1 \wedge C_2) \vee (C_1 \wedge C_3) \vee (C_1 \wedge C_4) \vee$$
$$\vee (C_2 \wedge C_3) \vee (C_2 \wedge C_4) \vee (C_3 \wedge C_4) \vee$$
$$\vee (L_1 \wedge \neg C_1) \vee (L_2 \wedge \neg C_2) \vee (L_3 \wedge \neg C_3) \vee (L_4 \wedge \neg C_4).$$

A general number of all possible combinations of values of the 8 conditions equals $2^8 = 256$. According to the definition of MC/DC, the number of required test cases is considerably lower. One of the possible sets of test cases is shown in Table 1 (10 test cases). Pairs of test cases for every specific condition are marked '*'. For example, test case 1 and test case 4 are marked for the condition L_3 because they provide variation of the decision d ($d = 0$ for test case number 1 and $d = 1$ for test case number 4) during variation of the condition L_3 ($L_3 = 0$ for test case number 1 and $L_3 = 1$ for test case number 4) while the values of all other conditions are fixed.

Table 1

Test data satisfying the MC/DC criterion

num	**Values**									**Variations**							
	L_1	L_2	L_3	L_4	C_1	C_2	C_3	C_4	d	L_1	L_2	L_3	L_4	C_1	C_2	C_3	C_4
1	0	0	0	0	0	0	0	0	0	*	*	*	*				
2	1	0	0	0	0	0	0	0	1	*							
3	0	1	0	0	0	0	0	0	1		*						
4	0	0	1	0	0	0	0	0	1			*					
5	0	0	0	1	0	0	0	0	1				*				
6	0	0	0	0	1	0	0	0	0						*	*	*
7	0	0	0	0	1	1	0	0	1					*	*		
8	0	0	0	0	1	0	1	0	1							*	
9	0	0	0	0	1	0	0	1	1								*
10	0	0	0	0	0	1	0	0	0					*			

Application for safety-critical software

The shortcoming of MC/DC. The main aim of MC/DC is testing situations when changing a condition implies a change in a decision. Often a decision can be associated with some safety-critical operation of a system. In such cases, MC/DC requires the testing of situations when changing one condition has some consequence on the operation of the system. A software error in such situations could involve *non-operation* (inability to operate on demand) type of failures. Such situations are extremely important and the MC/DC requirements are entirely reasonable.

However, as we show below, this criterion has one substantial shortcoming, namely deficiency of requirements for testing of the *false actuation* (operation without demand) type of failures. This could make this criterion insufficient for many safety-critical applications. The false actuation of a system could be invoked by a software error in situations when changing a condition should not imply changing a decision. Below we consider two examples from the specification-based point of view.

Railway points. Consider a railway computer control system and a decision that is responsible for switching over the points by which trains can be routed in one direction to another. Let there be two tracks (main and reserved); the condition determines track states (which may be either occupied or clear) and the decision determines changing the route from the main track to the reserved track and vice versa. Consider two situations for the non-operation and false actuation types of failures.

The first situation is when the main track becomes occupied (varying the condition) and, therefore, it is necessary to switch over the points to the reserve track (varying the decision). The failure in this situation involves keeping the value of the decision instead of varying it; this means non-operation of the system and could result in a possible crash.

The second situation is when the reserved track becomes occupied (varying the condition) and, therefore, it is necessary to keep the main track as a route (keeping the decision). The failure in this situation involves varying the value of the decision instead of keeping it fixed that means false operation of the system and a possible crash.

Thus, from the safety point of view, these situations are symmetrical and can lead to a crash. Therefore, both types of failures should be considered and both situations should be tested with the same accuracy.

Protection system for a nuclear reactor. Consider a decision that is responsible for actuating a reactor protection system at a nuclear power plant (i.e., the reactor shutdown) and a condition that describes some criterion for the actuation (e.g., excessive pressure over some specified level). Varying this decision because of variation of the condition should be tested since failure in this situation means the non-operation of the system in case of emergency conditions and can lead to the nuclear accident.

Nevertheless, keeping the value of the decision is also important. The failure in this situation means the false actuation of the system during normal operating and can lead to non-forced reactor shutdown, the deterioration of the physical equipment, and the underproduction of electricity.

The typical architecture of nuclear reactor protection systems (three channels with *2 from 3* logical voting) takes into account this particular problem. The use of three identical channels decreases the probability of the system not operating correctly.

However, if it is only required to consider this factor, the *1 from 3* logic is more reliable. The aim of using *2 from 3* voting is to provide protection against false actuation of a system as in this case the false signal from one channel does not lead to system actuation.

Thus, during software testing for the reactor protection system, it is necessary to include test cases for both varying and keeping a decision's outcomes.

The examples considered above demonstrate that for many cases testing only varying a decision when varying a condition (i.e., using MC/DC) is insufficient from the safety point of view.

To eliminate this shortcoming, a new RC/DC criterion in critical applications has been proposed by Vilkomir and Bowen [4]. We consider it below from the specification-based point of view.

RC/DC

Definition of RC/DC. As we have shown in the previous section, MC/DC does not require testing some situations, which can be important for safety. The main idea of RC/DC is for future development of MC/DC with the purpose of making it more effective.

Testing according RC/DC should include test cases according MC/DC and additional test cases for testing important situations when a false actuation of a system is possible. In that way, all requirements of MC/DC are valid and a new requirement for keeping the value of a decision when varying a condition is added to the testing regime.

With the objective of ensuring compatibility and continuity with the MC/DC definition, we define RC/DC as follows:

Every point of entry and exit in the program has been invoked at least once, each condition in a decision has been shown to affect the decision's outcome independently, and each condition in a decision has been shown to keep the decision's outcome independently. A condition is shown to affect and keep a decision's outcome independently by varying just that condition while holding fixed (if it is possible) all other conditions.

The reservation 'if it is possible' is used because it is far from always being possible to affect or keep the value of a decision independently.

A case study of using RC/DC for specification-based testing. Continue the consideration of the case study of specifications for the nuclear reactor protection system. Hence the RC/DC criterion includes MC/DC, the test cases for MC/DC (table 1) should be supplemented by additional test cases according RC/DC requirements.

RC/DC requires that a condition should 'keep the decision's outcome'. When the decision has outcome 1, it means that the protection system has already actuated.

In this case, the further behavior of the system has no practical interest.

The situation when the decision has outcome 0 is more important. A failure 'to keep 0' means a false actuation of the reactor protection system that entails significant economic loss. In Table 2, we consider an example (between many others) of test cases only for this situation.

Similar to table 1, pairs of test cases for every specific condition are marked '*'. For example, test case 3 and test case 9 are marked for the condition C_2 because they keep the value of the decision d ($d = 0$ for both test cases) during variation of the condition C_2 ($C_2 = 1$ for test case number 3 and $C_2 = 0$ for test case number 9) while the values of all other conditions are fixed.

Table 2

Test data satisfying the RC/DC criterion

num	**Values**									**Variations**							
	L_1	L_2	L_3	L_4	C_1	C_2	C_3	C_4	d	L_1	L_2	L_3	L_4	C_1	C_2	C_3	C_4
1	0	0	0	0	1	0	0	0	0	*				*			
2	1	0	0	0	1	0	0	0	0	*							
3	0	0	0	0	0	1	0	0	0		*				*		
4	0	1	0	0	0	1	0	0	0		*						
5	0	0	0	0	0	0	1	0	0			*				*	
6	0	0	1	0	0	0	1	0	0			*					
7	0	0	0	0	0	0	0	1	0				*				*
8	0	0	0	1	0	0	0	1	0				*				
9	0	0	0	0	0	0	0	0	0					*	*	*	*

As it is shown in Table 2, we need 9 test cases to test situations when variations of every single condition should keep the decision outcome 0. Together with 10 test cases from Table 1, we use 19 test cases (from 256 possible combinations) to test decision d according the RC/DC requirements.

Conclusion

The paper considers the use of the MC/DC and RC/DC criteria for specification-based software testing. It is argued that MC/DC criterion does not include requirements for testing of 'false operation' type failures. Such failures, as we have shown in several examples, can be highly important in safety-critical computer systems.

The RC/DC criterion aims to eliminate this shortcoming and requires the consideration of situations when varying a condition keeps the value of a decision constant.

Using RC/DC gives an advantage for specification-based testing since it requires testing safety-important situations when a false actuation of a system is possible.

Although the number of required test cases rises, the growth remains linear compared to the number of conditions in a decision, making the approach practicable.

We have illustrated application of the RC/DC criterion in the specification-based testing of nuclear reactor protection system software.

References

1. Myers G. The Art of Software Testing // Wiley-Interscience, 1979.

2. Roper M. Software Testing // McGraw-Hill, 1994.

3. Vilkomir S.A., Bowen J.P. Formalization of Software Testing Criteria Using the Z Notation // Proceedings of 25th IEEE Annual International Computer Software and Applications Conference (COMPSAC), Chicago, Illinois, USA, 8-12 October 2001. – IEEE Computer Society Press. – P. 351-356.

4. Vilkomir S.A., Bowen J.P. Reinforced Condition/Decision Coverage (RC/DC): A New Criterion for Software Testing // Proceedings of 2nd International Conference of Z and B Users (ZB2002), Grenoble, France, 23-25 January 2002. Springer-Verlag, Lecture Notes in Computer Science. – Vol. 2272. – P. 295-313.

5. RTCA. Software Considerations in Airborne Systems and Equipment Certification // DO-178B, RTCA, Washington DC, USA, 1992.

6. Dupuy A., Leveson N. An Empirical Evaluation of the MC/DC Coverage Criterion on the HETE-2 Satellite Software // Proceedings of the Digital Aviation Systems Conference (DASC), Philadelphia, USA, October 2000.

7. Chilenski, J., Miller, S. Applicability of Modified Condition/Decision Coverage to Software Testing // Software Engineering Journal. – September 1994. – P. 193-200.

8. Chilenski J., Newcomb P.H. Formal Specification Tool for Test Coverage Analysis // Proceedings of the Ninth Knowledge-Based Software Engineering Conference. – 20-23 September 1994. – P. 59-68.

9. Bishop P.G. MC/DC based estimation and detection of residual faults in PLC logic networks // Supplementary Proceedings 14th International Symposium on Software Reliability Engineering (ISSRE '03), Fast Abstracts. – Denver, Colorado, USA. – 17-20 November, 2003. – P. 297-298.

10. Hayhurst K.J., Veerhusen D.S. A Practical Approach to Modified Condition/Decision Coverage // 20th Digital Avionics Systems Conference (DASC), Daytona Beach, Florida, USA. – 14-18 October 2001. – Vol. 1. – P. 1B2/1--1B2/10.

11. Pretschner A. Compositional Generation of MC/DC Integration Test Suites // Proc. TACoS'03, Warsaw, March 2003. Electronic Notes in Theoretical Computer Science 82(6). – 2003. – P. 1-11.

12. Vilkomir S.A., Kapoor K., Bowen J.P. Tolerance of Control-Flow Testing Criteria // Proceedings of 27th IEEE Annual International Computer Software and Applications Conference (COMPSAC), Dallas, Texas, USA, 3-6 November 2003. IEEE Computer Society Press. – 2003. – P. 182-187.

13. White A.L. Comments on Modified Condition/Decision Coverage for Software Testing // 2001 IEEE Aerospace Conference Proceedings, 10-17 March 2001, Big Sky, Montana, USA. – Vol. 6. – P. 2821-2828.

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

Поступила в редакцию 24.02.2006

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.