CESL

MC/DC による現実的な網羅のススメ

松本 充広†

Modified condition/decision coverage（MC/DC）は，ソフトウェアのテスト網羅度として非常に有効な指標である．テスト網羅度は，ソフトウェアテストにおいて，充分にテストした，という品質基準を定義する指標の一つであり，テスト網羅度を用いることで，ソフトウェア品質を保持しながら，コストや時間が無制限に増大するのを避けることができる．MC/DC は，故障が人命に係わる航空機ソフトウェアのテスト時に使用するテスト網羅度として，米国の航空機産業の業界団体である RTCA が策定した DO-178B の中で開発された．MC/DC 開発以前に存在したテスト網羅度は，様々な問題を持っていたが，これらの問題が MC/DC で解決された．近年，信頼性向上を求める動きが高まり，車載ソフトウェアなどの航空機ソフトウェア以外の組込みソフトウェアの分野でも，MC/DC が注目されている．本稿は，テストの現場でどこまで網羅すればよいのか困っている方を対象に，MC/DC の使い方を紹介する．

A recommendation of MC/DC

MICHIHIRO MATSUMOTO†

Modified condition/decision coverage (MC/DC) is one of the most useful test coverage. Test coverage is a measure that is used for a quality standard that determines when we can stop software testing. By using test coverage, we can avoid that software testing is too expensive and time-consuming. MC/DC is the test coverage that was developed by RTCA in part of DO-178B. MC/DC solved the preexisting problems about test coverage. Recently, there is a movement to improve software quality. MC/DC is watched with keen interest in the field of embedded software, for example, automotive software. In this article, we introduce how to use MC/DC.

1. はじめに

Modified condition/decision coverage（MC/DC）は，ソフトウェアのテスト網羅度として非常に有効な指標である．

テスト網羅度は，ソフトウェアテストにおいて，充分にテストした，という品質基準を定義する指標の一つであり，テスト網羅度を用いることで，ソフトウェア品質を保持しながら，コストや時間が無制限に増大するのを避けることができる．

MC/DC は，MC/DC 開発以前に存在したテスト網羅度が持つ，様々な問題を解決したテスト網羅度である．

本稿は，テストの現場でどこまで網羅すればよいのか困っている方を対象に，MC/DC の使い方を紹介することを目的とする．

†キャッツ組込みソフトウェア研究所, CATS Embedded Software Laboratory

2. 背景

MC/DC は，故障が人命に係わる航空機ソフトウェアのテスト時に使用するテスト網羅度として，米国の航空機産業の業界団体である RTCA が策定した DO-178B の中で開発された．MC/DC 開発以前に存在した全てのテスト網羅度は，以下のどれかの問題を持っていたため，RTCA は，これらのテスト網羅度は，故障が人命に係わる航空機ソフトウェアのテスト時に使用するテスト網羅度として不十分，と判断したからである[1].

1. 全ての分岐を確認していない可能性がある．
2. 全ての条件を確認していない可能性がある．
3. ソースコードを記述する際，しばしば間違いが起こる A or B と A and B の記述ミスを検出できない可能性がある．
4. 条件の数を n 個とすると，テストケースの数が 2 の n 乗個になる．

これらの問題を全て解決したテスト網羅度が MC/DC

である.

DO-178B は, 航空機ソフトウェア開発のためのガイドラインであり, 米国の民間航空機やその装備品に搭載される航空機ソフトウェアは, 事実上, DO-178B に従う必要がある. DO-178B は, 航空機ソフトウェアをレベル A からレベル E に分類しているが, 故障が人命に係わるレベル A のソフトウェアに対し, MC/DC100%を満たすテストをすることを求めている. 近年, 信頼性向上を求める動きが高まり, 車載ソフトウェアなどの航空機ソフトウェア以外の組込みソフトウェアでも, MC/DC が注目されている.

3. MC/DC 開発以前に存在したテスト網羅度

MC/DC は, MC/DC 開発以前に存在した様々なテスト網羅度を検討し, その問題を解決するテスト網羅度として開発された. 3節では, まず, 3.1節において, テスト網羅度の定義に必要な用語を説明する. 次に, 3.2節において, MC/DC 開発以前に存在した様々なテスト網羅度の定義と問題について説明する. 最後に, 3.3節において, MC/DC で解決した他のテスト網羅度の問題を説明する.

3.1. 用語説明

まず, テスト網羅度を説明するために必要な用語について説明する.

【テストケース】

プログラムに現れる各入力変数の入力値の組のこと.

例1)x, y を入力変数とする. また, プログラムを下記とする.

```
if    (x > 2) and (y == 5)    then    z = z -2;
```

このとき, (x = 3, y = 5)は, テストケースである.

【テスト】

複数のテストケースの集まりのこと.

例 2)x, y を入力変数とする. また, プログラムを下記とする.

```
if    (x > 2) and (y == 5)    then    z = z -2;
```

このとき, (x = 3, y = 5), (x = 1, y = 5)はテストである.

【プログラムの入口／出口】

プログラムの実行開始行をプログラムの入口, プログラムの実行終了行をプログラムの出口と呼ぶ.

例 3)プログラムを下記とする.

```
if    (x > 2) and (y == 5)    then    return;
if    (z == 3) or (y > 2)     then    z = 2 * z;
```

このとき, プログラムの入口は1行目で, プログラムの出口は1行目と2行目である. 1行目は return により, プログラムが終了する可能性があるため, プログラムの出口になる.

【条件(Condition)】

ブール値を取り, and, or, not などの論理演算子を含まない式のことを条件と呼ぶ.

例 4)プログラムを下記とする.

```
if    (x > 2) and (y == 5)    then    z = z -2;
```

このとき, (x > 2) と (y == 5)は条件である.

【判定(Decision)】

0 個以上の条件を and, or, not などの論理演算子で結合した式のことを判定と呼ぶ.

例 5)プログラムを下記とする.

```
if    (x > 2) and (y == 5)    then    z = z -2;
```

このとき, (x > 2) and (y == 5)は, 判定である.

3.2. MC/DC 開発以前に存在したテスト網羅度の定義と問題

3.2節では, MC/DC 開発以前に存在した命令網羅度(SC), 判定網羅度(DC), 条件網羅度(CC), 条件判定網羅度(CDC), 複合条件網羅度(MCC)の定義と問題について説明する.

表 1のように，①命令網羅度（SC），②判定網羅度（DC）と条件網羅度（CC），③条件判定網羅度（CDC）の順で，網羅する範囲（長所）が増加しているが，これらの中で最も網羅する範囲が広い条件判定網羅度（CDC）でも，A or B と A and B の記述ミスを検出できない可能性がある，という問題（短所）がある．

一方，複合条件網羅度（MCC）は，条件／分岐の確認はもちろん，A or B と A and B の記述ミスも検出できるが，条件の数を n 個とすると，テストケースの数が 2 の n 乗個になる，という問題（短所）がある．

以降，順番に，各テスト網羅度の定義と問題について説明する．

表 1 各テスト網羅度の長所と短所

テスト網羅度	長所（網羅する範囲）	短所（問題）
命令網羅度（SC）	全ての行の実行を確認できる	確認できない分岐が存在する可能性がある
判定網羅度（DC）	全ての分岐を確認できる	確認できない条件が存在する可能性がある
条件網羅度（CC）	全ての条件を確認できる	確認できない分岐が存在する可能性がある
条件判定網羅度（CDC）	全ての条件／分岐を確認できる	A or BとA and Bの記述ミスを検出できない可能性がある
複合条件網羅度（MCC）	全ての条件／分岐を確認でき、A or BとA and Bの記述ミスを検出できる	条件の数をn個とすると，テストケースの数が2のn乗個になる

3.2.1. 命令網羅度（Statement Coverage / SC）の定義と問題

与えられたテストが，命令網羅度（SC）100%である，の定義は，以下を満たすことである．

① プログラムの全行を少なくとも一回はテストすること

命令網羅度（SC）100%テストにより，全ての行が実行できることを確認できる．

しかし，命令網羅度（SC）100%テストには，確認できない分岐が存在する可能性がある，という問題がある．

例 6）x, y, z を入力変数とする．また，プログラムは下記とする．

```
if    (x > 2) and (y == 5)    then    z = z -2;
if    (z == 3) or (y > 2)    then    z = 2 * z;
```

このとき，テストケース(x = 3, y = 5, z = 5)からなるテストを考える．以下の理由により，このテストは命令網羅度（SC）100%テストである．

1. 表 2のように，プログラム1行目の判定(x > 2) and (y == 5)，2 行目の判定(z == 3) or (y > 2)の値が共に true であり，then 部分が実行される．従って，プログラムの全行が実行され，①を満たす．

しかし，表 2のように，プログラム1行目の判定(x > 2) and (y == 5)，2 行目の判定(z == 3) or (y > 2)では，共に，false になる分岐が確認できていない．つまり，命令網羅度（SC）100%テストでは，確認できない分岐が存在する可能性がある．

表 2 例 6 のテストケースと判定値

x	y	z	(x > 2) and (y == 5)	(z == 3) or (y > 2)
3	5	5	true	true

3.2.2. 判定網羅度（Decision Coverage / DC）の定義と問題

与えられたテストが，判定網羅度（DC）100%である，の定義は，以下を満たすことである．

① プログラムの全入口／出口を少なくとも一回はテストすること
② プログラムの全判定は可能な値を少なくも一回はテストすること

判定網羅度（DC）100%テストにより，全ての分岐を確認できる．

しかし，判定網羅度（DC）100%テストには，確認できない条件が存在する可能性がある，という問題がある．

例 7）x, y を入力変数とする．また，プログラムは下記とする．

```
if    (x  == 3) or (y > 2)    then    z = 2 * z;
```

このとき，テストケース(x = 3, y = 1), (x = 1, y = 1)からなるテストを考える．以下の理由により，このテストは判定網羅度（DC）100%テストである．

1. 表 3のように，テストケース(x = 3, y = 1)に対して，プログラム1行目の判定(x == 3) or (y > 2)が true になるため，プログラム1行目，すなわち，プログラムの入口／出口が実行され①を満たす．
2. テストケース(x = 3, y = 1), (x = 1, y = 1)は，表 3のように1行目の判定(x == 3) or (y > 2)を true / false にするので②を満たす．

しかし，表 3のように，条件(y > 2)は，true になる場合が確認できていない．つまり，判定網羅度（DC）

100%テストでは，確認できない条件が存在する可能性がある．

表 3 例 7 のテストケースと判定値／条件値

x	y	(x == 3) or (y > 2)	x == 3	y > 2
3	1	true	true	false
1	1	false	false	false

3.2.3. 条件網羅度（Condition Coverage / CC）の定義と問題

与えられたテストが，条件網羅度（CC）100%である，の定義は，以下を満たすことである．

① プログラムの全入口／出口を少なくとも一回はテストすること
② プログラムの判定に含まれる全条件は可能な値を少なくも一回はテストすること

条件網羅度（CC）100%テストにより，全ての条件を確認できる．

しかし，条件網羅度（CC）100%テストには，確認できない分岐が存在する可能性がある，という問題がある．

例 8）x, y を入力変数とする．また，プログラムは下記とする．

```
if   (x  == 3) or (y > 2)   then   z = 2 * z;
```

このとき，テストケース(x = 3, y = 1), (x = 1, y = 3)からなるテストを考える．以下の理由により，このテストは条件網羅度（CC）100%テストである．

1. 表 4のように，テストケース(x = 3, y = 1)に対して，プログラム1行目の判定(x == 3) or (y > 2)が true になるため，プログラム1行目，すなわち，プログラムの入口／出口が実行され①を満たす．
2. テストケース(x = 3, y = 1), (x = 1, y = 3)は，表 4 のように(x == 3), (y > 2)の条件を共に true / false にするので②を満たす．

しかし，表 4のように，判定(x == 3) or (y > 2)は false になる分岐を確認できていない．つまり，条件網羅度（CC）100%テストでは，確認できない分岐が存在する可能性がある．

表 4 例 8 のテストケースと判定値／条件値

x	y	(x == 3) or (y > 2)	x == 3	y > 2
3	1	true	true	false
1	3	true	false	true

3.2.4. 条件判定網羅度（Condition Decision Coverage / CDC）の定義と問題

与えられたテストが，条件判定網羅度（CDC）100%である，の定義は，以下を満たすことである．

① プログラムの全入口／出口を少なくとも一回はテストすること
② プログラムの判定に含まれる全条件は可能な値を少なくも一回はテストすること
③ プログラムの全判定は可能な値を少なくも一回はテストすること

条件判定網羅度（CDC）は，条件網羅度（CC）と判定網羅度（DC）とを合わせたテスト網羅度である．条件判定網羅度（CDC）100%テストにより，全ての条件と，全ての分岐を確認できる．

しかし，条件判定網羅度（CDC）100%テストには，ソースコードを記述する際，しばしば間違いが起こる A or B と A and B の記述ミスを検出できない可能性がある，という問題がある．

例 9）x, y を入力変数とする．また，プログラムは下記とする．

```
if   (x  == 3) or (y > 2)   then   z = 2 * z;
```

このとき，テストケース(x = 3, y = 3), (x = 1, y = 1)からなるテストを考える．以下の理由により，このテストは，は，条件判定網羅度（CDC）100%テストである．

1. 表 5のように，テストケース(x = 3, y = 3)に対して，プログラム1行目の判定(x == 3) or (y > 2)が true になるため，プログラム1行目，すなわち，プログラムの入口／出口が実行され①を満たす．
2. テストケース(x = 3, y = 3), (x = 1, y = 1)は，表 5 のように，(x == 3), (y > 2)の条件を共に true / false にするので②を満たす．
3. テストケース(x = 3, y = 3), (x = 1, y = 1)は，表 5 のように，プログラムの判定(x == 3) or (y > 2)を true / false にするので③を満たす．

表 5 例 9 のテストケースと判定値／条件値

x	y	(x == 3) or (y > 2)	x == 3	y > 2
3	3	true	true	true
1	1	false	false	false

ここで, プログラムの判定を(x == 3) and (y > 2)に変更し, それぞれに対し, このテストを再度実行すると, 表6のように, 条件判定網羅度(CDC)100%になる. つまり, A or B を間違いて A and B と記述しても, 条件判定網羅度(CDC)100%テストでは検出できない可能性がある.

表 6 例 9 のテストケースと判定値

x	y	(x == 3) or (y > 2)	(x == 3) and (y > 2)
3	3	true	true
1	1	false	false

3.2.5. 複合条件網羅度(Multiple Condition Coverage / MCC)の定義と問題

与えられたテストが, 複合条件網羅度(MCC)100%であるとは, 以下を満たすことである.

① プログラムの全入口／出口を少なくとも一回はテストすること
② プログラムの判定に含まれる条件値の全組合せを少なくとも一回はテストすること

複合条件網羅度(MCC)100%のテストは, 判定に含まれる条件値の全組合せを確認するという, 余すところのない, 徹底的なテストなので, 理論的には最も理想的なテストである. 条件／分岐の確認はもちろん, A or B と A and B の記述ミスも検出できるテストである.

しかし, 条件の数を n 個とすると, テストケースの数が 2 の n 乗個になり, 条件が多いプログラムに対しては, 非実用的である.

例 10) x, y を入力変数とする. また, プログラムは下記とする.

```
if   (x  == 3) or (y > 2)    then    z = 2 * z;
```

このとき, テストケース(x = 3, y = 3), (x = 1, y = 3), (x = 3, y = 1), (x = 1, y = 1)からなるテストを考える. 以下の理由により, このテストは複合条件網羅度(MCC)100%テストである.

1. 表 7のように, テストケース(x = 3, y = 3)に対して, プログラム1行目の判定(x == 3) or (y > 2)が true になるため, プログラム1行目, すなわち, プログラムの入口／出口が実行され①を満たす.
2. 表 7のように, 条件(x == 3), (y > 2)の条件値の全組合せを確認しているので②を満たす.

表 7のように, テストケース(x = 3, y = 3), (x = 1, y = 3), (x = 3, y = 1), (x = 1, y = 1)からなるテストは, 全条件, 判定を true／false にして確認を行なっている. また, 表 8のように, (x == 3) or (y > 2)と(x == 3) and (y > 2)の記述ミスも検出できる. しかし, 条件の数は 2 個なので, テストケース数は 2^2=4 個になる.

表 7 例 10 のテストケースと判定値／条件値

x	y	(x == 3) or (y > 2)	x == 3	y > 2
3	3	true	true	true
1	3	true	false	true
3	1	true	true	false
1	1	false	false	false

表 8 例 10 のテストケースと判定値

x	y	(x == 3) or (y > 2)	(x == 3) and (y > 2)
3	3	true	true
1	3	true	false
3	1	true	false
1	1	false	false

3.3. MC/DC で解決した他のテスト網羅度の問題

3.2節で述べた各テスト網羅度の問題の中で, MC/DC 導入以前に解決されていなかった問題は, 以下の2つである.

1. ソースコードを記述する際, しばしば間違いが起こる A or B と A and B の記述ミスを検出できない可能性がある(条件判定網羅度(CDC)の問題).
2. 条件の数を n 個とすると, テストケースの数が 2 の n 乗個になる(複合条件網羅度(MCC)の問題)

4節で説明するように, MC/DC は, この 2 つの問題を解決したテスト網羅度である.

4. MC/DC の概要

MC/DC は, Modified condition/decision coverage の略で, ソフトウェアのテスト網羅度の一つである.

MC/DC は, 条件／分岐の確認はもちろん, ソースコードを記述する際, しばしば間違いが起きる A or B と A and B の記述ミスも検出でき, かつ, 条件の数を n 個とすると, テストケースの数が 2 の n 乗個にならない, 現実

的なテスト網羅度である.

与えられたテストが, MC/DC100%である, の定義は, 以下を満たすことである.

① プログラムの全入口／出口を少なくとも一回はテストすること
② プログラムの判定に含まれる全条件は可能な値を少なくも一回はテストすること
③ プログラムの全判定は可能な値を少なくも一回はテストすること
④ プログラムの判定の全条件は判定の出力に独立に影響することを示すこと

④の「判定の出力に独立に影響すること」の解釈として様々な解釈が存在するが, 本稿では, 最もきつい解釈である,「他の条件の値を固定し, 条件の値を変更することで, 判定の出力値を変更すること」を採用する.

MC/DCは, 条件判定網羅度(CDC)に④を追加したテスト網羅度である. ④により, 条件判定網羅度(CDC) 100%テストでは検出できない可能性があった A or B と A and B の記述ミスも, 例 11 のように, MC/DC100%テストでは検出することが出来る. これは, A を true(あるいは false)に固定して, B を true／false にするとき, A or B と A and B とでは判定値が異なるためである.

例 11)x, y を入力変数とする. また, プログラムは下記とする.

```
if   (x  == 3) or (y > 2)    then    z = 2 * z;
```

このとき, テストケース(x = 1, y = 1), (x = 1, y = 3), (x = 3, y = 1)からなるテストを考える. 以下の理由により, このテストは MC/DC100%テストである.

1. 表 9のように, テストケース(x = 1, y = 3)に対して, プログラム1行目の判定(x == 3) or (y > 2)が true になるため, プログラム1行目, すなわち, プログラムの入口／出口が実行され①を満たす.
2. テストケース(x = 1, y = 1), (x = 1, y = 3), (x = 3, y = 1)は, 表 9のように, (x == 3), (y > 2)の条件を共に true / false にするので②を満たす.
3. テストケース(x = 1, y = 1), (x = 1, y = 3), (x = 3, y = 1)は, 表 9のように, プログラムの判定(x == 3) or (y > 2)を true / false にするので③を満たす.
4. 条件(x == 3)に関しては, テストケース(x = 1, y = 1), (x = 3, y = 1)により, 条件(y > 2)を false に固定し, 条件(x == 3)の値を false／true にすることで, 判定(x == 3) or (y > 2)を false／true にしている. また, 条件(y > 2)に関しては, テストケース(x = 1, y = 1), (x = 1, y = 3)により, 条件(x == 3)を false に固定し, 条件(y > 2)の値を false／true にすることで, 判定(x == 3) or (y > 2)を false／true にしている. 従って④を満たす.

また, このテストでは, 表 10のように, (x == 3) or (y > 2)と(x == 3) and (y > 2)の記述ミスを検出できる.

表 9 例 11 のテストケースと判定値／条件値

x	y	(x == 3) or (y > 2)	x == 3	y > 2
1	1	false	false	false
1	3	true	false	true
3	1	true	true	false

表 10 例 11 のテストケースと判定値

x	y	(x == 3) or (y > 2)	(x == 3) and (y > 2)
1	1	false	false
1	3	true	false
3	1	true	false

MC/DC100%テストでは, 他の条件を固定して条件の値を true／false にするテストケースを用意するので, 条件の数を n 個とすると, テストケースの数は, n + 1 個以上になる. 実際, 例 11 の場合(表 9)では, テストケースは 3 個(n = 2)である.

このように, MC/DC100%テストは, テストケースが 2 の n 乗個になる複合条件網羅度(MCC)100%テストと比べ, 実用的である.

5. MC/DC の実践例

MC/DC100%テストの構築方法として, 真理値表を用いる方法がある. この方法では, 原理的に, テストケース数が最小になる MC/DC100%テストを構築することが出来る.

この真理値表を用いる方法を, 例 12 を用いて説明する.

例 12)A, B, C を条件とする. また, プログラムは下記とする.

if (A or B) and C then z = 2 * z;

このプログラムに対し，MC/DC100%テストを構築する．

まず，真理値表（表 11）を以下のように，作成する．

1. A，B，C を true／false にする全組合せを表に記述する（白色の A，B，C の部分）．
2. 各行毎に，判定値を計算し，記述する．
3. MC/DC の④の独立に影響することを示すのに必要なテストケースのペアを求め，灰色の A，B，C の部分を求める．具体的には，次のように求める．

   A の列には，B，C を固定して A を true／false にしたとき，判定が true／false（または false／true）に変化するペアの相手の番号を記述する．同様に，B の列には，A，C を固定して B を true／false にしたとき，判定が true／false（または false／true）に変化するペアの相手の番号を記述し，C の列には，A，B を固定して C を true／false にしたとき，判定が true／false（または false／true）に変化するペアの相手の番号を記述する．

   例えば，A の1行目は，次のようになる．1行目は，(A = true, B = true, C = true)なので，B，C を固定して，A だけ値を変更したものは，(A = false, B = true, C = true)である．これは 5 行目である．ここで真理値表の1行目と5行目の判定をみると，共に true であるため，1行目と5行目はペアにはなれない．従って，A の1行目は，空白にする．

   また，A の 3 行目は，次のようになる．3 行目は，(A = true, B = false, C = true)なので，B，C を固定して，A だけ値を変更したものは，(A = false, B = false, C = true)である．これは 7 行目である．ここで真理値表の 3 行目と 7 行目の判定をみると，それぞれ true／false であり，3 行目と 7 行目はペアになる．従って，A の 3 行目には，7 を記述する．

このようにして作成した真理値表から，テストケース数が最小になる MC/DC100%テストを選択することが出来る．

例 12 の場合，3，5，6，7 行目からなるテスト，3，4，5，7 行目からなるテストが，テストケース数が最小になる MC/DC100%テストである．

表 11 例 12 の真理値表

	A	B	C	判定	A	B	C
1	true	true	true	true			2
2	true	true	false	false			1
3	true	false	true	true	7		4
4	true	false	false	false			3
5	false	true	true	true		7	6
6	false	true	false	false			5
7	false	false	true	false	3	5	
8	false	false	false	false			

6. 結論

4節，5節で説明したように，MC/DC は，条件／分岐の確認はもちろん，条件判定網羅度（CDC）では検出できなかった A or B と A and B の記述ミスを検出でき，また，条件の数を n 個とすると，複合条件網羅度（MCC）100%テストではテストケース数が 2 の n 乗個になるのに対し，MC/DC100%テストではテストケース数が n + 1 個以上と，n が大きくなった場合，MC/DC100%テストは複合条件網羅度（MCC）100%テストと比べて，テストケース数が圧倒的に少なくて済む．

このため，現実的なテスト網羅度として MC/DC をお勧めする．

7. 今後の課題

5節で説明した，真理値表を用いて MC/DC100%テストを求める方法は，テストケース数が最小になる MC/DC100%テストが求まるなど利点があるが，条件の数を n 個とすると，真理値表の行数が 2 の n 乗個になるため，n が大きくなると非実用的である．

また，変数値を外部から直接与えられず，他の変数値から計算しなければならない変数が判定に含まれる場合，この判定のテストケースを実現するために，他の変数値を決定する方法は簡単ではない．

このため，MC/DC100%テストを求める方法として，モデル検査を応用した方法[3]や，モンテカルロ法を応用し 100%に出来るだけ近いパーセンテージのテストを生成する方法[4]など様々な方法が提案されている．弊社でも，モデルがデータフロー図の構造を持つソフトウェアを対象に，データフロー図の特徴を活かして MC/DC100%テストを求める方法を研究している．

8. むすび

本稿では，ソフトウェアのテスト網羅度の一つである

MC/DC とは何か？について説明した. MC/DC は, 条件／分岐の確認はもちろん, ソースコードを記述する際, しばしば間違いが起きる A or B と A and B の記述ミスも検出でき, かつ, 条件の数を n 個とすると, テストケースの数が 2 の n 乗個にならない, 現実的なテスト網羅度である.

テストの現場でどこまで網羅すればよいのか困っている方は, 是非, MC/DC を試してほしい.

参考文献

[1] Kelly J. Hayhurst, Dan S. Veerhusen, John J. Chilenski, and Leanna K. Rierson, “A Practical Tutorial on Modified Condition / Decision Coverage”, NASA/TM-2001-210876, 2001.

[2] Peter G Bishop, "MC/DC based estimation and detection of residual faults in PLC logic networks," ISSRE 2003, Fast Abstracts, Supplementary Proceedings, pp. 297-298, 2003.

[3] Sanjai Rayadurgam and Mats P.E. Heimdahl, “Generating MC/DC adequate test sequences through model checking,” Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard, pp.91-96, 2003.

[4] Reactive Systems, Inc., “Model-Based Testing and Validation with Reactis,” 2009.

[5] http://www.do178site.com/

[6] 進藤智則, “国産ジェット機が秘める効能”, 日経エレクトロニクス, 2008 年 7 月 28 日号, pp.41-74, 2008.

松本充広

1990 年 京都大学・理学部卒. 2002 年 北陸先端科学技術大学院大学情報科学研究科博士後期課程修了. 博士（情報科学）. 現在, キャッツ組込みソフトウェア研究所研究員.

付録 1. DO-178B における MC/DC の位置付け

MC/DC は, DO-178B の一部として策定されたテスト網羅度である. MC/DC をより良く理解していただくため, 付録 1 では, DO-178B における MC/DC の位置付けを説明する.

DO-178B は, 航空機ソフトウェア開発のためのガイドラインで, 米国の民間航空機やその装備品に搭載される航空機ソフトウェアは, 事実上, DO-178B に従う必要がある. DO-178B では, 航空機ソフトウェアをレベル A からレベル E に分類（表 12）している.

表 12 DO-178B での航空機ソフトウェアの分類

レベル	対象ソフトウェア
レベルA	予期しない振舞いにより、航空機の破滅的な故障（人命を完全に失う故障）に至る、システム機能の故障を引き起こすソフトウェア
レベルB	予期しない振舞いにより、航空機の重大で危険な故障（人命をある程度失う故障）に至る、システム機能の故障を引き起こすソフトウェア
レベルC	予期しない振舞いにより、航空機の重大な故障（重大な損害を与える故障）に至る、システム機能の故障を引き起こすソフトウェア
レベルD	予期しない振舞いにより、航空機の軽微な故障（軽微な損害を与える故障）に至る、システム機能の故障を引き起こすソフトウェア
レベルE	予期しない振舞いが、航空機の操作性／パイロットの作業量に影響しない、システム機能の故障を引き起こすソフトウェア

DO-178B では, 故障が航空機に重大な損害を与える可能性のあるレベル C 以上（レベル A／B／C）のソフトウェアに対し, 命令網羅度（SC）100%テストでテストすることを求めている. また, 故障が人命に影響する可能性のあるレベル B 以上のソフトウェアに対し, 判定網羅度（DC）100%テストでテストすることを求めている. そして, 故障が人命に係わる可能性のあるレベル A のソフトウェアに対し, MC/DC100%テストでテストすることを求めている.

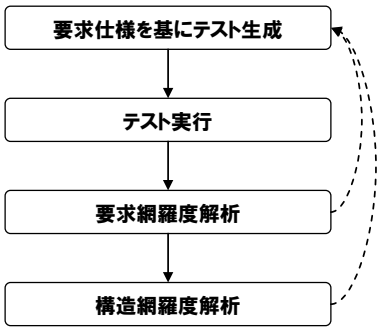

図 1 DO-178B のテストプロセス

DO-178B のテストプロセスは図 1になる. まず, 要求仕様を基にテストを生成する. 次に, 生成したテストを用いてテスト実行する. その後, 要求網羅度解析で, 要求仕様の何%がテストされたのかを解析する. もし, 100%でなかった場合には, テストされていなかった要求仕様に対しテストを生成し, 再度テスト実行する. 要求網羅度解析で, 要求仕様の 100%がテストされたことを確認できたら, 構造網羅度解析で, テスト網羅度が何%かを解析する. もし, 100%でなかった場合には, テスト網羅度を100%にするために必要なテストを生成し, 再度テスト実行する. 構造網羅度解析で, テスト網羅度が 100%になったことを確認できたら, テストプロセスは終了である.

要求網羅度(要求仕様の何%がテストされたのかの指標)だけでは, 以下の理由でテストとして不十分である.

1. 要求仕様では, プログラムの完全で正確な仕様を書き尽くしていない
2. 関数の振舞いを保証するために必要な粒度で要求仕様が記述されていない
3. 要求仕様だけでは, 意図しない機能が含まれていないことを確認できない

このため, DO-178B では, テスト網羅度 100%テストでのテストを求めている.

DO-178B における, MC/DC の位置付けの, 更なる詳細に関しては[1]を参照のこと.

付録 2. MC/DC と類似する概念

MC/DC の適用可能性を理解していただくため, 付録 2 では, MC/DC と類似する概念について説明する.

MC/DC はソフトウェアテストにおけるテスト網羅度であるが, PLC ロジックネットワークのテストの分野に, MC/DC と類似する入出力ペア網羅度(input-output pair coverage)[2]というテスト網羅度がある.

PLC ロジックネットワークは, AND ゲート, OR ゲート, NOT ゲートから構成されるハードウェアのことである.

与えられたテストが, 入出力ペア網羅度(input-output pair coverage)100%である, の定義は, 以下を満たすことである.

① PLC ロジックネットワークの全入力に対し, 他の入力値は固定したまま, 入力値を変更(true／false)し, 出力値を変更させる(true／false または false／true)テストをすること.

この入出力ペア網羅度(input-output pair coverage)の①は, MC/DC の④に対応している.

この入出力ペア網羅度(input-output pair coverage)を 100%に出来るだけ近いパーセンテージのテストを生成する方法として確率値誤差逆伝播法(バックプロパゲーション)がある.

確率値誤差逆伝播法(バックプロパゲーション)では, 次の①～③の方法で, テストを作成する. 5節で説明した MCDC100%テストとは異なり, 確率に基づいてテストを作成する. また, 作成されたテストのことを MC/DC ランダムテストと呼ぶ.

① 出力が true になる確率を 0.5 とする.
② 各ゲートの入力が true になる確率(Pin)を, 次のようにして, 出力が true になる確率(Pout)から求める(ゲートの入力数を n 本とする).
   (1) AND ゲートの場合
   Pin = Pout^(1/n)　(x^n は, x の n 乗のこと)
   (2) OR ゲートの場合
   Pin = 1 – (1 – Pout)^(1/n)
   (3) NOT ゲートの場合
   Pin = (1 – Pout)
③ ②を繰り返すことで最終的に得られる PLC ロジックネットワークの入力確率に基づきテストケースを生成する.

確率値誤差逆伝播法（バックプロパゲーション）の①, ②について, 例 13 を用いて説明する.

例 13）PLC ロジックネットワークを図 2とする.

① 全体の出力が true になる確率を図 2のように, 0.5 にする.

② （1）まず, AND ゲートに着目する. AND ゲートの出力が true になる確率は, ①より 0.5 であり, この AND ゲートの入力は 2 本なので, AND ゲートの入力が true になる確率は, 0.5^(1/2)=0.71 である.
（2）次に, 2つの OR ゲートに着目する. OR ゲートの出力が true になる確率は共に, ②（1）より 0.71 であり, OR ゲートの入力は共に 2 本なので, OR ゲートの入力が true になる確率は, 共に 1－（1－0.71）^(1/2)=0.46 である.

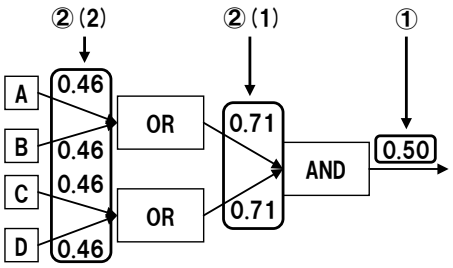

図 2 例 13 の PLC ロジックネットワーク

入出力ペア網羅度（input-output pair coverage）の更なる詳細に関しては[2]を参照のこと.