組込みソフトウェア開発課題への挑戦　～網羅度～

渡辺　政彦†

組込みシステムに占めるソフトウェアの割合が高くなるにつれて，モデルやコードを網羅的に検査，テスト，検証することが重要になっている．しかしながら全てを網羅することは，巨大化・複雑化する組込みソフトウェアでは容易なことではない．そこで，ある部分に限定して 100%網羅する技法やより最適解の近似値を求める技法が用いられるようになってきた．そこで本稿では最近よく見かける 3 つの技法，「直交表」，「MC/DC」，そして「GA」をついての入門的な解説をする．

Challenge to embedded software development problem –coverage-

MASAHIKO WATANABE†

In an embedded system, due to the increase of software involved in it nowadays, it has become important to inspect, test and verify its model and code exhaustively. It is not easy, however, to cover all the test cases in the model or code in a large-scale complicated embedded software. Therefore recently those methods that target exhausting test cases of a restricted coverage or finding a better approximation of the optimal solution are commonly well used. In this paper, we explain three such methods, called orthogonal tables, MC/DC and GA.

1. はじめに

ソフトウェアの特徴として，同じものを作ることは極めてたやすく，また，ソフトウェアそのものに物理的な経年変化は起こらないことはよく知られている．つまり，ソフトウェアの問題は製造上の問題，物理的現象上の問題ではなく，設計上の問題かつ論理的現象上の問題である．設計かつ論理上の問題であれば，「正しく」設計し，「正しく」テストすれば良い．しかしながら，ソフトウェアにおける問題は，「正しく」設計し，「正しく」テストすることが非常に困難なことである．

Myers は『“完全な”テストは不可能であるから、いかなるプログラムのテストも不完全にならざるをえない』と述べている[1]．そこで本稿では，「正しい」もしくは「完全な」テストに近づける方法を議論する．

さらに Davis は『網羅的でとりこぼしのないテストは不可能である』と述べている[2]．特に，機器に組込まれるソフトウェア（組込みソフトウェア）では，データの値の他に，制御信号の値，またその値を更新する時間を網羅的にテストすることが要求されるため，網羅すべき範囲が大きくなる．そこで本稿では，網羅すべき範囲をどのように絞り込むかについて議論する．

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

2. 技法紹介

本稿では，網羅度に関して以下の 3 つの技法を紹介する．

1. 直交表
2. MC/DC (Modified Condition / Decision Coverage)
3. GA (Genetic Algorithm)

直交表は「実験計画法」の1つであり，1920 年代から農業実験で用いられてきた．農業実験では，例えば収穫を最大限にするために，品種，肥料，温度，雨量，そして土壌など数多くの因子を組み合わせて，最適な配合や条件を見つける実験を行う．しかしながら，現実に全ての実験を行うことは時間的，費用的に困難であるため，実験の組合せを削減するために直交表を用いる．3 章では，農業におけるこの実験項目削減に倣い，組込みシステムにおけるテスト項目を削減することについて議論する．

MC/DC カバレッジは，航空無線技術委員会(RTCA)によって作られたソフトウェアの開発用ガイドライン DO-178B において採用され注目を集めている．全ての条件の組合せを確認するためのテストケースは前述のように困難であるから，他の条件を固定化することでテストケースを削減する MC/DC について 4 章で議論する．

GA とは，「発見的手法」の1つである．発見的手法とは，「完全」であるとは言い切れないが，現実的な時間内で，それに近い解を求める手法である．遺伝子が優秀な子孫を残すことを模倣し，遺伝子を解の候補とすることで，最適に近い解を求める GA について 5 章で議論する．

3. 直交表

3.1. 考え方（交互作用）

直交表は実験計画法の1つである．実験計画法は実験を効率よく行うための方法で，1920 年代に農業での実験に使われていた．

例えば，収穫を最大限にするために品種，温度，水をどのように組み合わせればよいかを実験する．品種，温度，水の 3 つの因子にそれぞれに 2 つの水準があるとする．品種(東・西),温度(高・低),水(硬・軟)である．全ての因子の全ての水準の組合せ実験をしようとすると 2*2*2=8 通りの実験が必要となる(表 1).

直交表では，因子間で「交互作用」がないことを前提として，組合せ数を削減することができる．「交互作用」について図 1 に示す．図 1 には因子（品種，温度）の収穫高への影響を実験した結果が示されている．

図 1-(a)は，品種，温度は収穫効果なし（「主効果なし」と言う）の場合を示している．

図 1-(b)は，品種が西より東の場合に収穫効果あり，温度は高いほど収穫効果あり（「主効果あり」と言う）の場合を示している．

図 1-(c)は，品種は西が良いとも，東が良いとも言えず，温度は高ければ良いとも，低ければ良いとも言えない．しかし，西の品種は温度が高いほど収穫効果があり，東の品種は温度が低いほど収穫効果がある．

図 1-(c)では，どの品種で収穫が高いのかそれ単独ではわからず，また温度においても低い方がよいのか，高い方がよいのかが分からないので，両因子に主効果なしとなる．西の品種は温度が高くなれば収穫が良くなり，東の品種は温度が低くなれば収穫が良くなる．この依存関係を交互作用と呼ぶ．

図 1-(a)では，品種と温度に交互作用はない．図 1-(b)では，どの品種にも温度が高ければ同じ比率で収穫が良くなるため，品種と温度に交互作用はない．あくまで温度という主効果があるということになる．品種の線が平行にならなければ，交互作用があるとなる．

表 1　農業実験

	品	温	水
列 / No	1	2	3
1	東	高	硬
2	東	高	軟
3	東	低	硬
4	東	低	軟
5	西	高	硬
6	西	高	軟
7	西	低	硬
8	西	低	軟

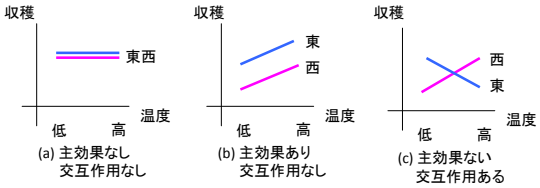

図 1　交互作用

交互作用がなければ，因子間全ての組合せをとらずに，2つの因子間で全ての水準の組合せをすれば良いという考え方が直交表である．

『網羅的でとりこぼしのないテストは不可能である』が，全因子のうち直交する因子成分に関しては，網羅的でとりこぼしのないテストができる可能性はある．

3.2. 組合せ削減方法(L4)

組み合わせる因子の中で最大水準数を持つ 2 つの因子の各水準を 2 のべき乗に切り上げてそれらの積をとり，それを最少の直交表サイズとする．因子数は直交表のサイズより少ない．

表 1 を例にすると最も大きい水準数は 2 なので 2*2=4 となり，直交表 L4 を適用すれば良いことが分かる（表 2）．L は Latin square の頭文字を示し，続く数字は実験数（行数）を示す．

L4 は 2 因子間で 2 水準の全ての組合せを示す．表 2 の例では，(11),(12),(21),(22)の組合せは，どの 2 列でもともに1回ずつあり，直交である．「AとB」，「AとC」，そして「B と C」で全ての水準が組合せられている．水

準値が 1 か 2 の 2 種類しかもたないことを 2 水準系と呼ぶ.

表 2　L4

L4	A	B	C
列 / No	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

表 3 に, 表 2 の全ての組合せから L4 が適用された組合せと削減された組合せを示す. No2,3,5,8 が削減された組合せである.

表 3　全組合せ

	A	B	C	
列 / No	1	2	3	
1	1	1	1	
2	1	1	2	これらの組合せを削減する.
3	1	2	1	
4	1	2	2	
5	2	1	1	
6	2	1	2	
7	2	2	1	
8	2	2	2	

全組合せ
2*2*2=8通り

繰り返しになるが, 表 4 に, 表 1 へ L4 を適用した場合の組合せを示す.

表 4　L4 適用

L4	品	温	水
列 / No	1	2	3
1	東	高	硬
2	東	低	軟
3	西	高	軟
4	西	低	硬

3.3. テストへの応用(L16)

直交表をテストに適用する. 組込み機器には, 複写機, 自動車, そして家電製品などユーザインタフェースを持つものは多い. ユーザインタフェースを持つということは, ユーザにどのような操作をされても問題ないことをテストしなければならない. しかしながら, 全てのユーザインタフェースの組合せをテストすることは難しい.

図 2 にテスト対象のカーオーディオ機器を示す. 操作ボタン 3 種類(A, B, C)ある. A ボタンには 2 種類の操作, B ボタンには 4 種類の操作, そして C ボタンには 4 種類の操作がある. ボタンの種類が因子, ボタン操作種類が水準として直交表を適用する. 全てのボタン操作の組合せをテストするには 2*4*4=32 通りのテストケースが必要となる.

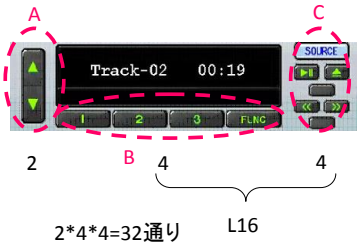

図 2　カーオーディオ機器

図 2 において最も大きい水準数は 4 なので 4*4=16 となり, 直交表 L16 を適用すれば良い. 直交表から因子と水準を選択する場合, 不用意に割り当てすると直交性が実現しないので注意が必要である.

表 5 の L16 の列 1-3 にボタン B を割り当て, 列 4-6 にボタン C を割り当てる. ボタン A は残りの列 7-15 のどこか 1 列を割り当てれば良い. ボタン B に割り当てた列 1-3 では, 4 水準が実現できている. しかしながら, 列 4-6 に割り当てられたボタン C では, 4 水準がボタン B に直交していない.

表 5　L16 割り当て失敗例

L16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
5	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
7	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
9	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
10	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1
	B	C	A												

割り当てを正しく行うには点線図を用いる. 表 6 では点線図から組み合わせて使用できない列(5,10,15) (6,11,13) (7,9,14)を除外して, 列 4,8,12 にボタン C を割り当てることで, 2 つの因子間で全ての水準の組合せのテストケースを導出することができる. 直交表を用い

ることで，全ての組合せでは 32 のテストケースを半分の 16 のテストケースにできる．

表 6　L16 割り当て成功例

L16

No \ 列	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
5	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1
7	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2
9	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
10	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1

B C A　点線図 (5,10,15) (6,11,13) (7,9,14)

3.4. 適用事例

富士ゼロックスにおける適用結果では，組合せ網羅率が従来(経験と勘)と比較して 2.7 倍，組合せ市場不具合ほぼ 0 件との報告が公表されている[3,4].

3.5. 詳細

点線図を含めさらに詳細な直交表については，別稿で述べる．→リンク先：川崎

4. MC/DC (Modified Condition / Decision Coverage)

4.1. 考え方(条件と判定)

MC/DC カバレッジは，航空無線技術委員会(RTCA)によって作られたソフトウェアの開発用ガイドライン DO-178B で規定される．

MC/DC は Modified Condition / Decision Coverage の略語である．判定(Decision)は条件判定文の判定であり，条件(Condition)は条件判定文を構成する条件である(図 3)．

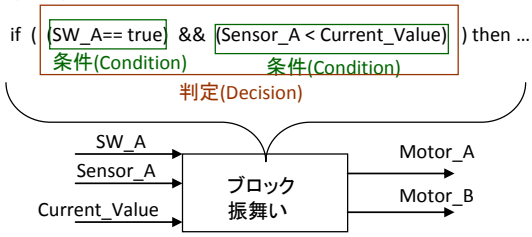

図 3　条件と判定

MC/DC は DC(Decision Coverage)や CC(Condition Coverage) よりも網羅度が高く，MCC(Multiple Condition Coverage)よりは網羅度が低い．

『網羅的でとりこぼしのないテストは不可能である』が，MC/DC に関して網羅的でとりこぼしのないテストができる可能性はある．

4.2. DC

DC は，全ての判定に関して true/false の全ての組合せを網羅する．

表 7　DC

テストケース			条件		判定
SW_A	Sensor_A	Current_Value	SW_A == true	Sensor_A < Current_Value	
T	50	80	T	T	T
F	50	80	F	T	F

表 7 に図 3 のブロック図から DC を用いたテストケースを示す．判定が true/false の全ての組合せを網羅するようにテストケースを作成する．

4.3. CC

CC は，全ての条件に関して true/false の全ての組合せを網羅する．

表 8　CC

テストケース			条件		判定
SW_A	Sensor_A	Current_Value	SW_A == true	Sensor_A < Current_Value	
T	90	80	T	F	F
F	50	80	F	T	F

表 8 に図 3 のブロック図から CC を用いたテストケースを示す．条件が true/false の全ての組合せを網羅するようにテストケースを作成する．判定は考慮しない．表 8 では判定は FF となっている．

4.4. MCC

MCC は全ての条件に関して true と false の全ての組合せと判定の true/false の全ての組合せを網羅する．

表 9　MCC

No	テストケース			条件		判定
	SW_A	Sensor_A	Current_Value	SW_A == true	Sensor_A < Current_Value	
①	T	50	80	T	T	T
②	T	90	80	T	F	F
③	F	50	80	F	T	F
④	F	90	80	F	F	F

表 9 に図 3 のブロック図から MCC を用いたテストケースを示す．条件が true と false の全ての組合せと，判定の true/false の全ての組合せを網羅するようにテストケースを作成する．

4.5. MC/DC

MCC では全ての条件の組合せを確認するため，テストケースが増大する．そのため，MC/DC では，他の条件の組合せの交互作用を無視することで，組合せ数の削減を行う．他条件の組合せの交互作用の無視とは，他条件を固定化し，自条件の全ての組合せをとることである．これを全ての条件で行い，最後に判定の全ての組合せを求める．

表 10　MC/DC

No	テストケース			条件		判定
	SW_A	Sensor_A	Current_Value	SW_A == true	Sensor_A < Current_Value	
①	T	50	80	T	T	T
②	T	90	80	T	F	F
③	F	50	80	F	T	F

表 10 に図 3 のブロック図から MC/DC を用いたテストケースを示す．処理判定ロジックのコンパイラ生成実行順序のため，条件(condition)の組合せは必要である．つまり DC ではなく，CC が必要になる．また，論理積（＆＆）と論理和（｜｜）の結果(decision)の違いをテストする必要もあり，DC が必要である．両方を満足する MCC では組合せ数が増大するので，他条件を固定化，つまり，他条件の組合せが自条件の組合せには影響しないとして，組合せ数を削減するのが MC/DC である．

表 10 がどのように求められるかの手順を以下に示す．条件「SW_A == true」を A とし，条件「Sensor_A < Current_Value」を B とする．

(1) A に関する確認（ここでは、B = true に固定して確認）

A が true となるテストケース：①

A が false となるテストケース：③

(2) B に関する確認（ここでは、A = true に固定して確認）

B が true となるテストケース：①

B が false となるテストケース：②

(3) 判定が true となるテストケース:①

判定が false となるテストケース:②, ③

(1)から(3)の操作からテストケース①, ②, ③が導出され，表 9-④を削除することができる．

仮に A と B を false に固定すると，

(1)’ A に関する確認（ここでは、B = false に固定して確認）

A が true となるテストケース：②

A が false となるテストケース：④

(2)’ B に関する確認（ここでは、A = false に固定して確認）

B が true となるテストケース：③

B が false となるテストケース：④

(3)’ 判定が true となるテストケース：①

判定が false となるテストケース：②, ③, ④

この場合，MCC と同じ数のテストケースとなり，削除には失敗となる．

4.6. MC/DC 練習問題

MC/DC の理解を深めるため，練習問題を解く[5]．下記 If 文の MC/DC を求めよ．

If (A && (B || C)) then…

表 11 に解答を示す．

表 11　MC/DC 表

No	A	B	C	判定	A	B	C
1	T	T	T	T	5		
2	T	T	F	T		4	
3	T	F	T	T			4
4	T	F	F	F		2	3
5	F	T	T	F	1		
6	F	T	F	F			
7	F	F	T	F			
8	F	F	F	F			

(1) A に関して No.1 で TTT, No.5 で FTT

(2) B に関して No.2 で TTF, No.4 で TFF

(3) C に関して No.3 で TFT, No.4 で TFF

(4) 判定は No.1,2,3 で T, No.4,5 で F

これから(1,2,3,4,5)が MC/DC テストケースとなる．

A に関して No.2 の TTF と No.6 の FTF を選択すれば，さらにテストケースを最小化した(2,3,4,6)となる．

4.7. 適用事例

英国のソフトウェア信頼センターでは，MC/DC テストとランダム入力テストと既存のシステムテストにおける不具合検出数の比較を行った[6]．その結果が表 12 である．

表 12　MC/DC テスト評価

テスト数	不具合検出数		
	MC/DCテスト	ランダム入力	既存システムテスト
10	4	0	?
100	5	0	?
486	6	0	4
1,000	6	0	
3,000	6	1	
9,514	6	4	
10,000	6	4	
100,000	6	6	
1,000,000	6	6	

PLC(Programmable Logic Controller)に MC/DC を適用し，全不具合数 6 件を他のテスト方法に比べて早期に発見し，その有効性を示した．

4.8. 詳細

MC/DC テストケースを最小化する方法，MC/DC ランダムテストに確率値誤差逆伝播法（バックプロパゲーション）を用いる方法については，別稿で述べる．→リンク先：松本

5. GA (Genetic Algorithm)

5.1. 考え方（遺伝子）

GA(Genetic Algorithm)とは，発見的手法分野におけるアルゴリズムである．発見的手法とは，「完全」であるとは言い切れないが，現実的な時間内で，それに近い解を求める手法である．

GA は解の候補を遺伝子として，選択（結婚）・交叉（生殖）し，優秀な子孫が生き残り，まれに突然変異を発生することで，最適解を発見する．遺伝子である解の候補を何にするかが，GA をどのように適用するかの鍵となる．

『“完全な”テストは不可能であるから、いかなるプログラムのテストも不完全にならざるをえない』が，完全に近い解を生命の営みのメカニズムから得ることができる．

5.2. GA 適用(WCET)

組込みソフトウェアがリアルタイムシステムである場合，最悪実行時間(WCET: Worst-Case Execution Time)を保証することが求められる．RM(Rate Monotonic)などのリアルタイムスケジューリング理論で計算できるシステムであれば良い．しかしながら，ネットワークやバスにより，複数のユニットが並列に動作し，かつユニット間の通信に遅延時間が発生し，かつ，駆動するメカの状態によってユニットの反応が異なるような複雑なシステムにおいて，一定時間内の反応を保証することは難しい．そこで，最適解を求める GA の遺伝子である解の候補を，最悪実行時間と考えた例を図 4 に示す．

従来の最悪実行時間を検証するには，ランダムにテストケースを作成し，実行後の反応時間を測定し，最悪実行時間値を守れているかを検証していた．ランダムにテストケースを作成するのではなく，実行したテストケースで時間がかかったテストケースを掛け合わせて，より優秀な，つまり時間のかかるテストケースを導出し効果的なテストケース生成を GA で実現する．

GA では突然変異を除き，概して次の 6 つの操作を行う．

①初期集団
②結果
③適合度計算
④選択
⑤交叉
⑥次世代

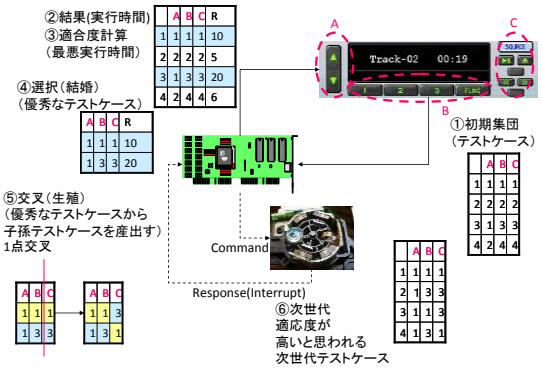

図 4　最悪実行時間検証

①初期集団では，最初の親（テストケース）を決める．親の優性遺伝（ここではより実行時間のかかるテストケース）が，子に引き継がれる．このためできるだけ優秀な親を最初に作ることが，GA の効率に大きく影響を与える．解をヒトとした場合，最初の親に魚を与えるのとサルを与えるのでは，求めるまでの時間に大きな差が出るのと同じである．初期集団はテスト実行者が考えて，GA に与えるものである．

②結果では，テストケースを実行した結果を得る．図 4 では，テストケース 1 が 10ms，テストケース 2 が 5ms，テストケース 3 が 20ms，テストケース 4 が 6ms，実行時間がかかったとしている．

③適合度計算では，結果から優秀（最悪実行時間

に近い)な遺伝子(テストケース)がどれかを計算する.

④選択では, 適合度計算から優秀なテストケースを選択する. 図 4 では, 実行結果が 10ms かかったテストケース 1 と 20ms かかったテストケース 3 を選択(結婚)する. 選択にはルーレット選択, ランキング選択, トーナメント選択があるが, ここでは単純に上位から選択するランキング選択としている.

⑤交叉では, テストケース 1 とテストケース 3 を「AB」と「C」の1点の箇所で切って, 交叉させる. 交叉にはここで用いた 1 点交叉以外にも, 2 点交叉, 多点交叉, 一様交叉がある.

⑥次世代では, 親のテストケース 1 とテストケース 2 と 1 点交叉によって生まれたテストケース 3 とテストケース 4 を実行させ, ②の結果へと繰り返す.

この繰り返しテストで最悪実行時間以内に処理が完了することを確認する.

5.3. 適用事例

あるプログラムに対して, ランダムに作成されたテストケースと GA を用いて作成されたテストケースで, MC/DC の網羅率を比較した事例を表 13 に示す[7].

表 13　MC/DC 網羅率比較

プログラム	ランダム	GA
バイナリサーチ Binary search	53.3	66.7
バブルソート1 Bubble sort 1	100	100
バブルソート2 Bubble sort 2	44.4	44.4
日にち計算 Number of days between two dates	35.3	39.2
ユークリッド最大公約数 Euclidean GCD	100	100
挿入ソート Insertion sort	100	100
中央値計算 Computing the median	100	100
解の公式 Quadratic formula	75	75
ワーシャル-フロイド法 Warshall's algorithm	91.7	100
三角形分割 Triangle classification	48.6	84.3

5.4. 詳細

本稿の GA では非常に単純なランキング選択および 1 点交叉を用いたが, これら以外にも選択にはルーレット選択やトーナメント選択があり, 交叉には2点交叉, 多点交叉, 一様交叉がある. また突然変異についても別稿で述べる. →リンク先:目時

6. むすび

組込みソフトウェア開発における課題の1つである「網羅度」に対して解を与えるであろう 3 つの技法について紹介した. 最初に「直交表」, 次に「MC/DC」, 最後に「GA」を解説した. 網羅すべき範囲をどのように絞り込むかについて, 「直交表」と「MC/DC」の技法と適用事例を紹介した. 「正しい」もしくは「完全な」テストに近づける方法について「GA」の技法と適用事例を紹介した.

無作為に時間と費用をかけて「網羅度」を上げるより, 工学的技法を用いて効率的に「網羅度」を上げる取り組みがこれからますます重要になるであろう.

参考文献

[1] Glenford J. Myers, 長尾真監訳, 「ソフトウェアテストの技法」, 近代科学社, 1980.

[2] Alan M. Davis, "201 Principles of Software Development," McGraw-Hill, 1995.

[3] 秋山浩一,「直交表を活用したソフトウェアテストの効率化 ー HAYST 法の活用 ー」, http://www.swtest.jp/jasst05w/S4-1.pdf

[4] 小川正樹,「技術者に必要な統計知識」, http://www.fiberbit.net/user/masa-2ogawa/crmin020.html

[5] Christopher Ackermann,"MC/DC in a nutshell," http://www.cs.umd.edu/~atif/Teaching/Fall2006/StudentSlides/MCDC_Coverage_02.ppt

[6] 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.

[7] Christoph C. Michael, Gary E. McGraw, Michael A. Schatz, and Curtis C. Walton, "Genetic Algorithms for Dynamic Test Data Generation," Proceedings of the 12th International Conference on Automated Software Engineering (ASE'97), pp.307, 1997.

渡辺政彦
2009年九州大学大学院システム情報科学府博士後期課程終了.
博士(工学). 情報処理学会会員. 電子情報通信学会会員.
現在, キャッツ株式会社取締役副社長 兼 最高技術責任者 兼 CAL 所長,
九州工業大学大学院情報工学研究院客員教授.