ソフトウェア工学 113－8
（1997. 3. 7）

限界値に着目したテスト網羅度の提案と評価

中村 寿彦　臼井 和敏　中本 幸一　門田 浩

NEC マイコンソフト開発環境研究所

E-mail:{tosihiko, usui, nakamoto, h-monden}@ccs.mt.nec.co.jp

ソフトウェアの品質向上のために、テストが重要となってきている。このテスト工程の終了判定ののために、ソースコードをもとにした指標であるテスト網羅度が使われている。これまで、命令網羅度、分岐網羅度などが実際のソフトウェア開発で使われてきた。これらの測定基準では、未実行部分が実行されるだけで網羅度は向上する。しかし、境界条件の誤りなど、実行したかどうかだけではなく、その実行時にどのようなテストデータを使ったかが重要となる場合がある。このような誤りは、境界近くを通過するテストデータで実行しないと発見が困難であるが、これまでこのようなテスト実行を調べ、指標化する手段は無かった。そこで、このようなテスト実行を行なったかどうかを数値化した限界値網羅度を考えた。評価の結果、制御不良に関する誤りに対して特に効果があることが確かめられた。

Boundary Value Test Coverage, Proposal and Evaluation

Toshihiko NAKAMURA, Kazutoshi USUI
Yukikazu NAKAMOTO, Hiroshi MONDEN

Microcomputer Software Engineering Laboratories, NEC Corporation

E-mail:{tosihiko, usui, nakamoto, h-monden}@ccs.mt.nec.co.jp

To improve the quality of software, software testing become very important. To decide when software testing is complete, test coverage based on the source code is used. Statement coverage and branch coverage are two popular criteria that are widely used in the actual development of software. Unfortunately, they are not reliable when there are faults that can be found only by executing the program in particular conditions because they are insensible to the execution condition. Until now, there are no criteria that is sensible to the execution condition. Then we propose a new criterion called boundary value coverage. As the result of the evaluation, we have verified that this criterion is effective in the faults which concerned with control structures.

1 はじめに

ソフトウェアの品質向上への取り組みの一つとして、テスト工程が重要になってきている。テストの進捗管理のためには、ソースコードをもとにしてプログラムのどの部分をテストしたかを割合で表すテスト網羅度 (以下、単に網羅度と記す) が使われている。この網羅度は、プログラムのどの部分に着目して測定するかという違いでさまざまな測定基準がある。なかでも、プログラムの命令に着目した命令網羅度、分岐に着目した分岐網羅度などが実際のソフトウェア開発で広く使用されている。また、データフロー網羅度なども有効性が認められつつある [1, 2]。

これらの測定基準では、未実行部分が実行されるだけで網羅度は向上する。一方では、境界条件の誤りを発見したい場合など、実行したかどうかだけではなく、その実行時にどのようなテストデータが使われたかが重要となる場合がある。このような検出できない誤りは、分岐の境界値付近を通過するようなテストデータで実行すると検出できる場合がある。しかし、これまでこのようなテスト実行を行ったかどうかを調べ、指標化する手段が無かった。そこで我々は、境界条件の誤りを検出できるテストデータで実行したかどうかを網羅度に反映させることができるような測定基準を考えた。この網羅度が限界値網羅度である。

以下では、まず限界値網羅度の諸定義を示し、この網羅度を測定するためのコードを組み込み、実際に網羅度を測定するツール群の実装について述べる。そして、これらのツールを使った限界値網羅度の評価について説明する。最後に、これらの実装、評価から得られた結論を述べる。

2 限界値網羅度

ソフトウェアの誤りに関する多くの調査で、制御に関連した誤りが特に高い割合を示している [3, 4]。これらの誤りは、偶然正しいパスを通ると正しい結果が得られてしまうので、分岐を単にテスト実行しただけでは、誤りを検出できないことがある。このような検出できない誤りは、分岐の境界値付近を通過するようなテストデータで実行すると検出できる場合がある。しかし、これまでこのようなテスト実行を行ったかどうかを調べ、指標化する手段が無かった。そこで、分岐の境界値付近を通過するテスト実行を行なったかどうかを網羅度という形で数値化することを考えた。この網羅度を限界値網羅度と呼ぶ。

考案にあたっては以下の点を考慮した。

- 境界の通過が実行時に容易に判別可能であること
- 網羅度 100% にするためのテストデータの量が、これまでの網羅基準を 100% にするためのテストデータの量から大幅に増えないこと

2.1 定義

まず、用語を定義する。次節で、これらの定義を例を使って説明する。

ソースコードにおいて、実行パスが同じになる入力値の集合をドメインと呼ぶ。ドメインはいくつかの条件式の論理演算 (条件式群) で表現される。但し、ここで条件式とは、真偽

値を得られる最小単位の式のこととする。例えば、プログラムの条件判定文中の不等式が条件式である。

また、個々の条件式で規定される面で、別のドメインと接する部分をドメインの境界と呼ぶ。但し、else など条件式が明示されていない場合にもそのドメインの境界は存在する。

ある境界がドメインに含まれる場合は、その境界に関してドメインが閉じていると表現し、ある境界がドメインに含まれない場合は、その境界に関してドメインが開いていると表現する。

これらの定義を使って、以下では、限界値、限界値網羅度を定義する。

限界値とは: ある境界に関して、ドメインが閉じている場合は、その境界上の値が限界値である。また、開いている場合は、その境界からドメイン側にある微小量 ϵ の距離以内の値が限界値である。

ϵ は、ドメインの境界を定めている条件式で使用されている変数のとりうる最小の増分値であることが理想である。例えば、変数が整数型であれば、ϵ を 1 と扱えばよい。しかし、実数型の場合は、このように単純に扱うことはできない。そこで、変数の型などにより、システム、又はユーザが分岐の条件式ごとに決定する。

限界値網羅度とは: 限界値網羅度は、以下の式で定義できる。

$$\text{限界値網羅度}(\%) = \frac{F}{W} \times 100$$

W はテスト対象の総数で、ドメインの 1 つの境界に対して、テスト対象を 1 つとして算出する。一方、F はテスト対象のテスト済み総数である。テスト実行時に、このテスト対象の境界を実行し、かつ、その時の値が限界値であれば、そのテスト対象をテスト済みとする。

2.2 テスト対象の例

```
1: a = getchar(); b = getchar();
2: x = func1(a); y = func2(b);
3: if(y > 0 && x + y < 0) ...
4: else ...
```

図 1: プログラム例

図 1のプログラムにおける 3 行目の命令文では、y > 0 and x + y < 0 が条件式群となる。ここで、限界値は x,y という変数を用いて判定される。この if 文が成立するパスのドメインを表したものが、図 2で塗りつぶされた部分である。条件式が 2 つあるので、このドメインには、テストの対象となる境界が 2 つ存在する。テスト対象の一つが、(a) の領域の 1 点であり、もう一つが (b) の領域の 1 点である。また、4 行目の else に対しては、太線で示された 2 つの境界上で、それぞれ 1 点ずつがテスト対象となる。

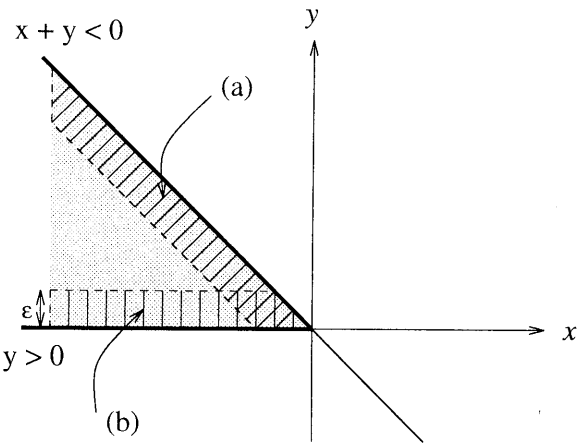

図 2: 限界値網羅度のテスト対象

2.3 網羅基準の強弱比較

網羅度に使われる基準 (網羅基準) を比較する時に、強弱を使うことがある [5]。網羅基準 A が、網羅基準 B よりも強いとは、網羅基準 A で 100% 網羅した時、網羅基準 B でも必ず 100% 網羅することをいう。

分岐網羅度では、複数の分岐を通過するテストデータで実行すると、そのすべての分岐をテスト済みとすることができる。一方、限界値網羅度では複数の分岐を通過しても、その分岐を限界値で通過しなければ、テスト済みとすることができない。そのため、複数の分岐を通過するパスに対して、それぞれの分岐ごとに、限界値を通過するデータを用意しなければならない可能性がある。一方で、すべての分岐は、限界値での通過という形でテストされる。従って、限界値網羅度は分岐網羅度より強い基準である。

また、一般的にパス網羅度が最も強い網羅基準とされている。このパス網羅度に比べて、テストデータの数では、少なくなると考えられるが、パス網羅度で 100% 網羅するテストデータ集合が、限界値網羅度で常に 100% 網羅するテストデータ集合になるとは限らない。従って、限界値網羅度は、パス網羅度に対して比較不可能である。

2.4 実装

限界値網羅度評価のために限界値通過を判定するコードを自動で組み込むツールを実装した。コード組み込みの対象とするプログラム言語は、C 言語とした。実装に用いた言語は、C 言語と lex、yacc を使用した。

今回のコード組み込みツールの実装に際して、ϵ は以下のように仕様を決めた。

- ϵ の決定は、変数の型などを考慮せず、ファイル全体で同じ値として扱う。[1]

[1]テスト対象のプログラムは浮動小数点型を使用していないものを選んだ。浮動小数点型への対処は今後検

- ϵはコンパイル時に決定する。

ϵをファイル全体で同じ値として扱うのは、非常に単純で、理解しやすいことから決定した。また、ϵの決定に関しては、プログラムへの影響を最小限にすることからコンパイル時とした。

例として、2.2節で取り上げた図1のプログラムをコード組み込みツールに通した結果を図3に示す。

```
1: a = getchar(); b = getchar();
2: x = func1(a); y = func2(b);
3: if(y > 0 && x + y < 0){
     if(y <= 0 + epsilon) bvc_judge(1, 1);
     if(x + y >= 0 - epsilon) bvc_judge(1, 2);
     ...
   }
4: else{
     if(y = 0 && !(x + y >= 0)) bvc_judge(-1, 1);
     if(!(y <= 0) && x + y = 0) bvc_judge(-1, 2);
     ...
   }
```

図 3: コード組み込み後のプログラム例

3 評価

3.1 評価方法

網羅度は、テストの進捗を表す指標として使われることがある。このためには、網羅度と発見できた誤りの数との関連性が重要である。今回は、主に分岐条件網羅度との比較という点を中心にして、発見できた誤りの数との関連で限界値網羅度の評価を行なった。

今回テスト対象としたプログラムは、以下の2つである。

- クイックソートプログラム (109 line)
- 勤務時間算出プログラム (429 line)

評価の方法として、ある一定条件下で誤りを挿入したプログラムを用意した。挿入する誤りに関しては、参考文献 [4] の付録を参照した。但し、プログラマによるユニットテスト工程を想定し、この工程で起こり得ない誤りは除外した。結果として、機能不良、構造不良、データ不良だけが対象となった。この3つのグループ中の、サブグループに関しては、一定

討する予定である。

の割合以上の誤りを取り上げた。各誤りの数の比率は、[4] に記された割合からから大きく異ならないようにした。この割合は、表1に示した。

プログラムに挿入する誤りの数は、その誤りの影響が判別出来るようにという理由から、各テスト実行で1つづつ挿入するようにした。また、以下の3つの場合をテストの結果で誤りが判別されたと判断した。

- 誤り挿入前の結果と異なる結果が得られた時
- プログラムが終了しなかった時
- プログラムが異常終了した時

評価は、2通りの方法で行った。一つは、限界値網羅度、分岐条件網羅度と誤り発見数との関連性の評価であり、もう一つは、限界値網羅度と分岐条件網羅度との違いを評価するものである。以下では、それぞれの評価ごとに、用意したテストデータと、得られた結果を示す。

[評価 1]

最初の評価は、限界値網羅度、分岐条件網羅度と誤り発見数との関連性を調べるものである。テストデータはランダムに作成し、そのテストデータの分岐条件網羅度、限界値網羅度を測定した。そして、誤りを含んだプログラムでテストをし、その誤りが発見できたかどうかを調べ、発見できた誤りの数を記録した。この結果をそれぞれの網羅度ごとにグラフ化したものが図4である。

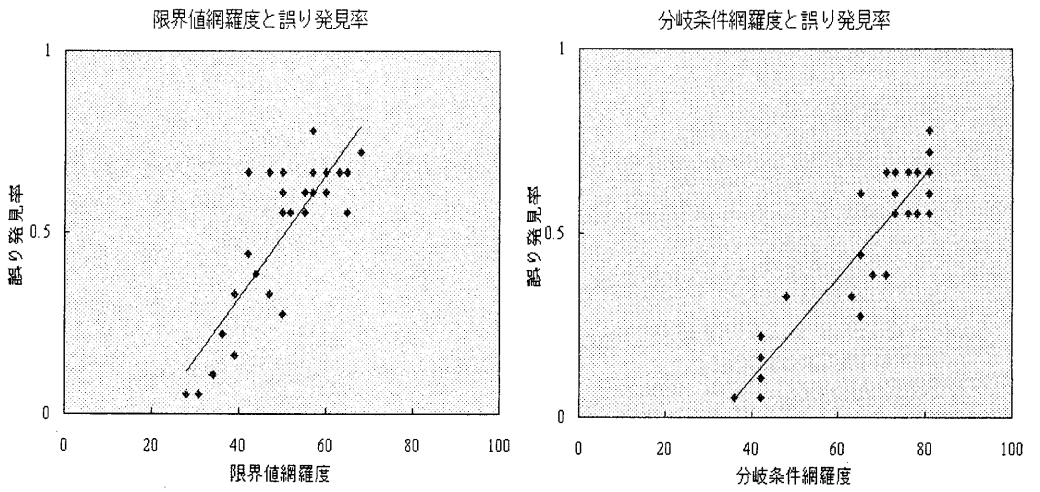

図 4: 誤り発見数と網羅度のグラフ

[評価 2]

評価 2 は、限界値網羅度と分岐条件網羅度との違いを明らかにするものである。そのために、分岐条件網羅度が高い割合を示し、限界値網羅度が低い割合を示すテストデータと、そのデータから分岐条件網羅度を一定にしたまま、限界値網羅度だけを高い割合にあげたテストデータを用意した。これらのテストデータで実行した結果から、見つかった誤りの数を記録した。これをまとめたものが、表 1である。テストデータは全部で 4 セット用意した。その各網羅度は表 1の 2 行目、3 行目に示している。

表 1: 限界値網羅度と誤り発見数

誤りの分類	割合 %	全回数	Set 1	Set 2	Set 3	Set 4
限界値網羅度 (%)			40	50	60	70
分岐条件網羅度 (%)			80	80	80	80
開発された機能不良	16.19					
概要機能の完全性	(1.43)	1	1	1	1	1
場合分けの完全性	(1.19)	2	2	2	2	2
領域不良	(4.80)	4	1	2	3	3
ユーザ／診断メッセージ	(5.29)	2	2	2	2	2
構造不良	25.18					
構造不良とシーケンス不良	(12.82)	6	2	3	4	5
処理	(12.36)	5	4	4	4	4
データ	22.44					
データ定義、構造、宣言	(11.14)	4	1	1	1	1
データのアクセスと取り扱い	(11.30)	4	3	3	3	3

※ 割合の列の括弧内の数値は括弧無しの上段の数値の内数である。

3.2 考察

図 4と、表 1の結果より以下のことが考察される。

- 分岐条件網羅度を十分高い割合にしただけでは見つからなかった誤りが、限界値網羅度を十分高い割合にすることで見つけられる。特に、多くの割合を占めている「制御不良とシーケンス不良」に関する誤りをより高い精度で発見できる。
- 限界値網羅度は、バグの発見数との関連性では、十分とはいえない。
- 限界値網羅度を使用した場合、正常なプログラムであっても網羅度の上限が 100% よりかなり低い値で留まる場合がある。

4 結論と課題

限界値網羅度は、バグの発見数との関連性では、十分とはいえず、現状では、既存の網羅度と併用し、補助的な使い方が中心となるかもしれない。しかし、小規模なプログラムが中心とはいえ、分岐条件網羅度で高い割合になるテストデータでテストしても見落とされる多くの誤りが、限界値網羅度の割合を高くすることで、見つけられることは確かめられた。特に、統計的に誤りが多いとされている制御不良でその効果を確かめることができた。

また、網羅度がかなり低い値で上限となる場合があるのは、等式(==)、不等式(!=)の扱いが主な原因になっている。例えば、フラグとして使われる変数など、構文上その変数が取りうる変数の値がかなり限定されている場合に、!= の条件式の両端をテスト対象と考えることが問題となる。対策として、不等式(!=)の扱いを変えるか、限界値網羅度の対象外とするなど、限界値網羅度測定の変数を選別する必要がある。

今後は、これらの改善の他に、ϵの決定も実際の適用にあたって重要な問題であるので、ϵをどのように決定するか、いつ決定するかなどを検討していきたい。テスト実行時、網羅度計測時に決定するような方法や、変数がどういう値をとりうるかというような情報をプログラムから得て、システム側がϵを自動で設定する方法などを考えていきたい。

また、今回はかなり小規模なプログラムを使用して、アルゴリズムの正当性や、分岐条件網羅度との比較を中心に評価したが、今後、実際のプロジェクトなどで開発される大規模なプログラムに適用してみて、この限界値網羅度の有効性を検証したいと考えている。さらに、この網羅度をC++や、JAVAへ適用するに際しての問題点なども検討したいと考えている。

参考文献

[1] Weyuker, E. J.: The Cost of Data Flow Testing: An Empirical Study, *IEEE Trans. Softw. Eng.*, Vol. 16, No. 2, pp. 121–128 (1990).

[2] Frankl, P. G. and Weiss, S. N.: An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing, *IEEE Trans. Softw. Eng.*, Vol. 19, No. 8, pp. 774–787 (1993).

[3] 下村隆夫：CASEツールの開発におけるソフトウェアバグの分析, 情報処理学会論文誌, Vol. 35, No. 7, pp. 1380–1389 (1994).

[4] Beizer, B., (小野間彰, 山浦 恒央訳)：ソフトウェアテスト技法, 日経BP出版センター (1994).

[5] Frankl, P. G. and Weyuker, E. J.: An Applicable Family of Data Flow Testing Criteria, *IEEE Trans. Softw. Eng.*, pp. 1483–1498 (1988).

[6] White, L. J. and Cohen, E. I.: A Domain Strategy for Computer Program Testing, *IEEE Trans. Softw. Eng.*, Vol. SE-6, No. 3, pp. 247–257 (1980).

[7] 中村寿彦, 臼井和敏, 中本幸一, 門田浩：限界値に着目したテスト網羅度の提案, 情報処理学会第53回全国大会講演論文集1, pp. 205–206 (1996).