7K-1

テストケース抽出の一方式
「原因流れ図」

松尾谷　徹　（日本電気　システム技術本部）

1．はじめに

ソフトウェア開発工程におけるテストは、信頼性を決定する重要な工程である。かつテスト工程に費やされる期間、費用共大きく、全体の生産性を支配する要因に成っている。

テストの生産性について観測をすると、テストケースの数が摘出エラー数、テストの工数を支配している。1),2)

テストケースを抽出する技法としては、ブラックボックステストにおける「原因－結果グラフ技法」、「実験計画法を用いたテストケース抽出技法」などが実用化されている。3)
テストケース抽出技法は「ＳＶＡ」等のツールにより自動化され、テストの生産性向上がはかられている。4)

本稿では実際のテストにおけるテストケースと発見エラーについて観測を行い、テストケースの冗長性について考察を加え、テストケースを合理的に減少させテストの生産性を向上させる改善案として「原因流れ図によるテストケース抽出技法」を提案する。

提案する技法は機能仕様から分析を行い、テストケースの抽出と共に設計工程においてプログラムの制御構造を規定し、テスト容易なプログラムを作ることにより大巾なテストケースの削減、ひいては信頼性と生産性の向上を目指している。

2．テストケースと発見エラーの関係

ソフトウェアのテストを探索理論から考えると、平行探索とランダム探索の問題がある。「原因－結果グラフ技法」のような網羅性を重視する技法は平行探索を導入しテスト効率を改善するアプローチととらえられる。5)

しかし原因－結果グラフ技法を導入したテストにおいても、実施テストケース数と摘出エラー数との関係は図１に示す通り直線ではなく飽和する傾向を示している。

そこで摘出されたエラーとテストケースの因果関係について詳細な調査を行った。

対象は７９５の原因－結果グラフによる３４１８件のテストケースで２３２件のエラーを発見した事例である。

表１は発見されたエラーとテストケースの対応を示す。表の対応していなとは、テストケースがテストの目的とした機能以外の部分、たとえばテストの準備や前処理においてエラーを発見することである。

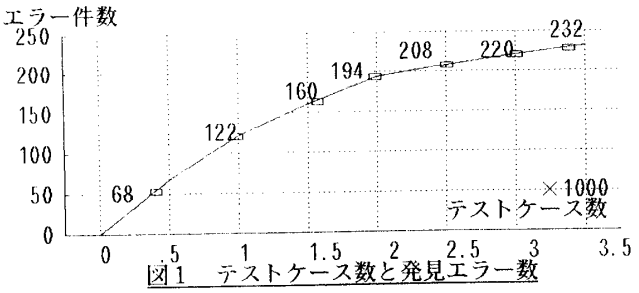

図１　テストケース数と発見エラー数

A Derivation Method for Test Case

Toru MATSUODANI NEC,Corporation

表１　テストケースとエラーの対応関係

エラーとテストケースとの対応	件数	率%
一つのテストケースが対応	１０８	４６
複数のテストケースが対応	４１	１８
直接テストケースが対応しない	８３	３６
合計	２３２	１００

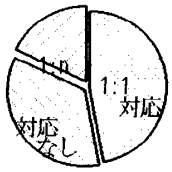

図2エラーとの対応

3．テストケースの冗長性

観測された現象がら次の推測を行った。

①目的テストケース以外のエラー検出
テストケースの準備段階でエラーをみつけている場合であるが、これは原因－結果グラフを作る時、機能分割を行い小さな機能範囲でテストケースを抽出する為、その機能を実行するために必要な別の機能がテストデータ作成時に入り込む。

図３目的外のテストケース混入例

図３にコンパイラの構成を例に示す、たとえばオブジェクト生成に対するテストケースは、必ず構文解析と意味解析の正常系のテストケースを暗黙の内に含むことになる。

②エラーの重複検出
同じエラーが複数のテストケースで検出されるのは次のような原因－結果グラフにおけるテストケースに多く見られる。（図４）

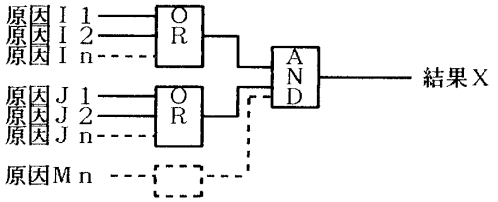

図４　Ｍ個のＯＲと１個のＡＮＤによるグラフ

この場合のテストケース数は原因－結果グラフ技法でｎのＭ乗となる、また実験計画法の直交配列でも指数的に増加する。（実際には制約条件を導入している）3)6)

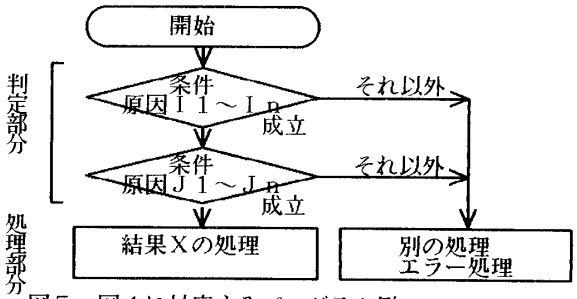

図５　図４に対応するプログラム例

図４のケースに対応するプログラム例を図５に示す。条件や構文を判定する部分と対応する処理（結果Ｘ）を分離したプログラム構造が考えられる。
この例の場合、原因どうしを積の組み合わせとしてテストケースとするのは冗長であり重複が発生する。

以上テストケースの冗長性について考察を行った、この問題を解決するため従来とは異なる次の考え方によりテストケースの数を合理的に減少させることが可能である。

ブラックボックス・テストは対象となるプログラムが仕様に対していかなる制御構造で実現したとしてもテスト可能なテストケースを抽出しょうとしている。

一方グラスボックス・テストは作られた制御構造に従ってテストケースを抽出する。

そこでテストケースが少なくてテスト可能な制御構造を考え、それにしたがってプログラムの制御構造とテストケースを決める方法が考えられる。

4．原因流れ図による表現

テストケースの分析結果によりプログラムの制御構造に制約条件を与えようとしても、原因－結果グラフ技法では処理の順序について定義出来ない。

そこでプログラム言語の構文則を定義する構文流れ図を基にしてプログラムの外部仕様（原因とその組み合わせ）を記述する方法を考える。　原因の非終端記号（取り得る状態の集合）を用いて原因の順序と組み合わせを現す図を定義し、これを「原因流れ図」と呼ぶことにする。

原因流れ図により文献３）で原因－結果グラフ技法の例題となっているＬコマンドを表記したものを図６に示す。

構文流れ図と原因流れ図の違いは、非終端記号にある状態の取り得る集合を対応させること以外にも幾つかの違いがある、一部を以下に示す。

①補集合の表現
エラーケースの記述を簡略化するため非終端記号に補集合を表す出力端子を定義する。　これにより条件式も非終端記号で表現する。

②結果の表現
一つの流れ図の中に複数の結果を記述するため結果を表す記号を定義する。

③ループの明示
有向線およびその交わりを簡略化し線とする、流れは左上から右下としループの場合にはループを有向線で表す。

原因－結果グラフ技法と比較すると、原因間の関係が流れ図の中に流れとして表現されている。

5．流れ図からテストケース抽出と制約条件

テストケースのためのデシジョンテーブル作成は、原因－結果グラフ技法に近い、まず結果側から流れ図を見ると木構造になっており、開始に至る原因のパスとしてテストケースを抽出する。

さらにテストケースの選択基準として「関係する木構造上の終端記号を最低一回は通る」および「エラーケースの場合、正常系で選択したテストケースを重複選択しない」などを定義する。

図６をもとにデシジョンテーブルを求めた例を表２に示す。

この例題に対する原因－結果グラフ技法、実験計画法、原因流れ図それぞれのテストケース数を表３に比較する。

次にテストケース選択基準の妥当性を保証するためプログラムの制御構造に制約条件を付ける。つまり示された原因流れ図を基にプログラムの詳細設計を行う。

原因流れ図は、外部変数の取り得る値の集合とその組み合わせについて定義している。これに処理のアルゴリズムを組み込むが、外部変数の取り扱いについて原因流れ図を守る必要がある。

表２　デシジョンテーブル

項番	原因と結果＼テストケース	1	2	3	4	5	6	7	8	9	10	11	12	
1.	コマンドの後のブランク指定	1	1	1	1	1	1	1	1	0	1	1	1	
2.1	第一パラメータ　省略	1			1			1				1		
.2	＊		1			1			1				1	
.3	ファイル名			1			1			1				
.4	異常値										1			
3.1	第二パラメータ　省略	1					1							
.2	行1		1					1						
.3	行1－行2			1					1					
.4	－行2				1					1				
.5	行1－					1					1			
.6	異常値											1		
4.	区切り記号　,		1	1	1	1							1	
5.	行番号異常でない行1＜行2			1				1	0					
6.	ファイルの存在	1	1	1	1	1	0	1						
7.	行番号の存在		1	1	1	1		0	1					
91	指定された動作を実行	1	1	1	1	1								
92	エラー　FILE NOT FOUND						1							
93	エラー　LINE NO. NOT FOUND							1						
94	エラー　LINE NO. ERR								1					
95	エラー　INVALID SYBTAX									1	1	1	1	

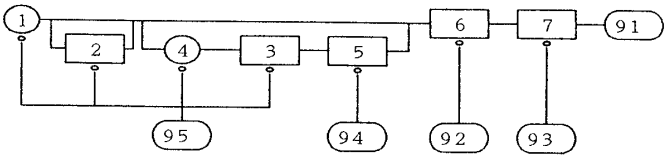

図６　Ｌコマンドの原因流れ図による表現

表３　各技法によるテストケース数の比較

	原因結果技法	実験計画法	原因流れ図
正常系	１５(100%)	１５(100%)	５(33%)
異常系	１０(100%)	７(70%)	７(70%)
合計	２５(100%)	２２(88%)	１２(48%)

6．まとめ

テストケース抽出方法として外部状態（原因）の流れ図を作成し、その原因流れ図から従来の方法より少ないテストケースでテスト可能なプログラム制御構造を規定する方法論を提案した。本稿では主にテストケース抽出について紹介を行った。

参考文献

［１］松尾谷、真野、塩川：ソフトウェア検査モデル　その１－３、情報処理学会第２９回全国大会.
［２］松尾谷、真野，他：ソフトウェア検査の外部発注における管理技術、ソフトウェアシンポジュム８４.
［３］石井　康雄編：ソフトウェアの検査と品質保証、日科技連ソフトウェア品質管理シリーズ第４巻.
［４］織田他：原因－結果グラフ技法を用いたレビュー支援ツールＳＶＡ、情報処理学会第３７回全国大会.
［５］松尾谷：予防保守のモデル化、情報処理学会第３６回全国大会.
［６］佐藤、下川：実験計画法を用いたソフトウェアのテスト項目設定法、ソフトウェアシンポジウム８５.
［７］平井、真野：実験計画法を用いたテスト項目設計技法－問題点とその改善への取組み－、ｿﾌﾄｳｪｱｼﾝﾎﾟｼﾞｳﾑ88.