ソフトウェア工学 31－2
（１９８３．８．３１）

バグ摘出難易度に関する尺度の適用性について

伊土　誠一　　馬場　正和　　林　孝樹
（日本電信電話公社　横須賀電気通信研究所）

１．はじめに

ソフトウェア品質を推定するいくつかの定量的な手法が提案されている。１つの方法は、テストが開始されて以来のバグ摘出履歴を観察することによってバグ数を予想する方法であり、代表例は、ソフトウェア信頼性の成長曲線を用いる方法である［ＧＯＥＬ７９，ＧＯＥＬ８０］。本方法では、テストの内容を成長曲線に反映することがむずかしいため予測精度が悪いという問題がある。もう１つの方法は、テスト工程期間の一部で、何らかの手法を用いてプログラムの品質を推定する方法であり、Ｃａｐｕｔｕｒｅ－Ｒｅｃａｐｔｕｒｅ法（以降、Ｃ＆Ｒ法と略す）がその代表例である［ＳＣＨＩ７８］。Ｃ＆Ｒ法は、統計上の理論にもとづいていており、次の点を前提としている。

（１）　ある推定精度を保証するため、統計的に必要な数以上の埋込みバグが、プログラム中に挿入されていること。

（２）　埋込みバグと本来のバグとの間で摘出難易度に大きな差がないこと。

（１）は純粋な統計上の問題であり、埋込みバグ数により予測値の信頼度がきまる。（２）は重要な問題であるが、埋込みバグの選定に対するよい目安がいまだないのが現状である。

本報告の目的の１つは、Ｃ＆Ｒ法の推定精度をあげるために、埋込みバグの選定に対する尺度を提案することである。

また、ソウトウェア品質を早期に向上させるためには、各テスト工程（単体、結合、システムテスト）において発見すべきバグを次の工程に持ち越さないで確実に見つけることが重要である。しかし、摘出されたバグが発見された工程以前に摘出されるべきであったか否かを判定し、テストの着実性を評価する尺度はまだ提案されていない。

本報告のもう１つのねらいは、そのような尺度を提案することである。

以上に述べた、２つの目的を達成するために、バグ発見過程を解析し、バグ摘出難易度に関する尺度を導出した。

２．　バグ発見過程

ソフトウェア中に潜在するバグをマシンにより摘出する場合は、何らかの手段（ドライバ、スタブを用いることもある）によりバグの潜んでいるエッジを通過させる必要がある。しかし、バグにはバグの潜在する部分を単に通過するだけで発生する単純なバグから、いくつかの条件が組合わさって発生する複雑なバグがある。

バグ発見過程の議論に入る前に、“バグの発生”と“バグの顕在化”が違うことを明確にしておく。バグの摘出にとって、バグ発生は必要条件ではあるが、それだけでは十分な条件ではない。バグを摘出するためには、さらにテスト者にバグが顕在化する必要がある。バグの発生時点と顕在化時点とは空間的、時間的にズレることがある。一般にこのズレが大きいとバグは発見しにくい。

以上述べたことにより、バグ摘出難易度を考察する場合、次の３つの要因を考慮する必要がある。

（１）　バグ潜在点を通過させる困難さ（到達難易度）

（２）　バグ発生条件の複雑さ（発生条件複雑度）

（３）　バグ発見のしにくさ（顕在化難易度）

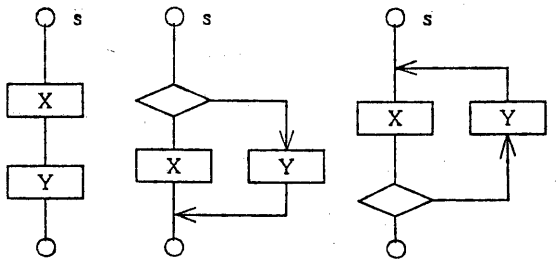
３．　摘出難易度

３．１　到達難易度

到達難易度とは、プログラム中のバグ潜在点を通過させることの難しさを表わす。プログラムは一般に図１に示すように３つの基本制御構造（連続、判断分岐、繰返し）の組合せからなっている。ここでは、それぞれの構造を到達難易度の観点から考察する。

＜連続構造＞

連続構造は連続なステートメントの集合であり、同一エッジである。よってバグ潜在点を通過する確率は、バグが図１（ａ）に示すＸあるいはＹ点のどちらに潜んで



(a) 連続構造　(b) 判断分岐構造　(c) 繰返し構造

図１　基本的制御構造

いようとも、エッジの開始地点Sを通過する確率に等しい。即ちP（α）＝P（S）となり連続構造の場合X、Yとも到達難易度は同じである。ここでP（α）は、バグ潜在点αを通過する確率、P（S）は、エッジの開始地点を通過する確率である。

＜判断分岐構造＞

図1の（b）の場合、XあるいはYを通過する確率はXとYの各々のエッジの機能分担に依存する。つまり、各々のエッジがメインルートかそうでないかによって通過確率は異なってくる。また、この確率はテスト内容によっても変化する。即ち、単体テストの段階では、テストの主要な目的の一つは、全パスあるいは全エッジを実行させることである。従って、この様な場合XあるいはYの通過確率が各々のエッジの機能とはあまり関係しないことになる。

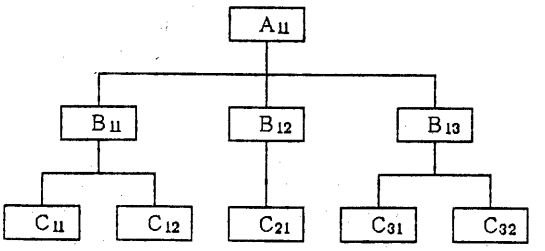


図2　プログラムモジュール例

＜繰返し構造＞

図1（c）のケースにおいて、プログラムを実行することにより、エッジの開始点Sを通過するならば、明らかにXの通過確率は1であり、Yの通過確率は分岐条件に依存する。XとYとの関連が強ければ、Yの通過確率は1に近く、弱ければ0に近くなる。

この概念は、図2に示す構造化プログラムを用いることによって説明できる。すなわち、モジュールA_{11}の中のα点を通過する確率P_{a11}（α）は以下のように表現できる。

$$P_{a11}(\alpha) = P_{a11}(S) * P_{a11}(A_\alpha)$$

$P_{a11}(S)$　：　モジュールA_{11}の開始地点の通過確率

$P_{a11}(A_\alpha)$　：　α点の通過確率（前提：α点の属しているモジュールA_{11}の開始地点を通過する確率が1）

モジュールB_{11}の開始地点の通過確率P_{b11}（S）は、モジュールA_{11}がモジュールB_{11}をコールする地点の通過確率に依存する。

$$P_{b11}(S) = P_{a11}(S) * P_{a11}(L_{b11})$$

L_{b11}　：　モジュールA_{11}がモジュールB_{11}をコールする地点

$P_{a11}(L_{b11})$　：　モジュールA_{11}がモジュールB_{11}をコールする地点の通過確率

従ってモジュールB_{11}中のα点を通過する確率P_{b11}（α）は次のようになる。

$$P_{b11}(\alpha) = P_{b11}(S) * P_{b11}(B_\alpha)$$
$$= P_{a11}(S) * P_{a11}(L_{b11}) * P_{b11}(B_\alpha)$$

同様な方法でモジュールC_{11}中のα点を通過する確率P_{c11}（α）は以下のように表現できる。

$$P_{c11}(\alpha) = P_{c11}(S) * P_{c11}(C_\alpha)$$
$$= P_{b11}(S) * P_{b11}(L_{c11}) * P_{c11}(C_\alpha)$$
$$= P_{a11}(S) * P_{a11}(L_{b11}) * P_{b11}(L_{c11}) * P_{c11}(C_\alpha)$$

３．２　発生条件複雑度

大規模プログラムのテストは、一般に単体、結合、システムテストの順におこなわれる。そのため、発見されるバグもテスト工程が進むに従って単にバグの存在するエッジを通過しただけでバグが発生する簡単なレベルのものから、色々な条件が組合わされないと発生しない複雑なレベルのものへと変化していく。

この節ではバグの発生条件について述べる。テスト工程の各段階の目的を考慮することによって、表１に示すようにバグの発生条件複雑度をレベル１からレベル４まで定義することができる。

レベル１は、バグの潜在点を通過するだけでバグが発生するケースである。問題は、バグ潜在点を通過させる困難さであるがそれに関しては前節で述べている。レベル１のバグは単体テストで完全に発見できる。

レベル２は、バグの潜在点を通過するだけではバグが発生しないケースである。いくつかの特殊な条件、例えば特殊な変数値、あるいは通過するエッジの特定の組合わさり方等によって発生するケースである。

レベル２は、レベル３、４と違って発生条件が１モジュールに閉じていてかつ時間要因とも独立であるケースである。レベル２のバグを発見するためにテスト者はパス、変数とそれらの領域について考慮してテストケースを抽出する必要がある。したがって、全てのテストケースを網羅することは困難である。この種のバグの大部分は単体テストで発見できる。

レベル３は、複数のモジュールに渡ったエッジ、あるいは変数の組合わさり方によってバグが発生するケースである。本レベルとレベル４との違いは、発生条件が時間要因と独立であることである。レベル３のバグは主として結合試験で発見される。

レベル４は、レベル３の発生条件に時間要因が加わったものであり、ガーベジバグがこれに属する。この種のバグは適切な発生条件を作成することが非常に困難であり、バグの発見が遅れることが多い。場合によっては、保守工程まで残ることがある。

表１　　発生条件複雑度のレベル

	バグ発生条件	考慮要因
レベル１	バグの潜在点を通過するだけで発生	E
レベル２	バグの潜在点を通過し、かつ１モジュールに閉じていくつかの特殊な条件が重なって発生	＋Pi，Vi，Di
レベル３	バグの潜在点を通過し、かつ複数モジュールに渡っていくつかの特殊な条件が重なって発生	＋Pj，Vj，Dj
レベル４	レベル３の発生条件に時間要因が加わって発生	＋T

E　：エッジ数　　Pj ：プログラム中のパス数
Pi ：モジュール内のパス数　Vj ：プログラム中の変数
Vi ：モジュール内の変数　　Dj ：Vj の変域
Di ：Vi の変域　　T　：時間

３．３　顕在化難易度

プログラムが走行し、バグが発生してもその時点で必ずしもテスト者にバグとして顕在化せず、バグが発生した所と別の部分が走行してはじめて顕在化するケースがある。バグが発生してから顕在化する困難さを顕在化難易度と定義する。顕在化難易度は、バグ発生点と顕在化点の空間的、時間的差に着目する必要がある。

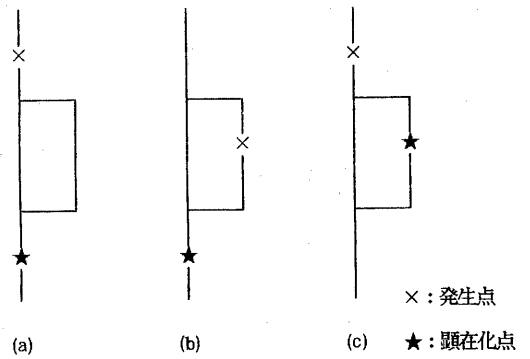


図３　顕在化確率の違い

＜空間的観点＞

バグが発生してから顕在化する所を通過する確率を顕在化確率と呼ぶこととする。顕在化難易度は、この顕在化確率に反比例する。これを図３を用いて説明する。（ａ）と（ｂ）の顕在化確率は１であるが（ｃ）の確率は１より小さい。よって、発生点と顕在化点との間の分岐数が多いほど顕在化確率は小さくなる。すなわち、バグの顕在化確率 $P_{ac}(\delta)$ は以下のように表わせる。

$$P_{ac}(\delta) = P(b_1) * P(b_2) * \cdots\cdots * P(b_n)$$

$P(b_i)$ ： 分岐ｉにおけるバグδの潜在しているエッジ側への分岐確率
ｎ ： 発生点と顕在化点との間にある（但し、発生点と顕在化点との間に対応する結合点のある分岐は除く）分岐点の数

但し、該当する分岐点のない場合の$P_{ac}(\delta)$は１である。

実際には顕在化確率の計算は面倒であるので、表２に示す様なテスト工程を考慮した簡単な尺度を用いるのが現実的である。

表　２　　顕在化難易度のレベル

レベル１	発生点と顕在化点は同一エッジか同一モジュールにある。 但し、$P_{ac}=1$	容易
レベル２	発生点と顕在化点は同一モジュールにある。 但し、$0<P_{ac}<1$：単体テスト $P_{ac}=0$：その他のテスト	↑
レベル３	発生点と顕在化点は別モジュールにある。 但し、$0<P_{ac}<1$	↓
レベル４	発生点と顕在化点は時間要因に依存する。	難易

＜時間的観点＞

バグの発生点から顕在化点までの経過時間が長いほどバグ発見が困難である。特に、一回のテスト時間で発見できないようなバグを発見することは非常に困難である。ガーベジバグがこの範疇に属する。この種のバグの発見は、顕在化難易度が時間に相関しないバグの発見よりも困難であり、テスト工程の早期にこの種類のバグを発見するためにはメモリ、ファイルの内容をダンプし、ガーベジの存在をチェックしなければならない。

４．　実験例と考察

本実験対象プログラムは、ＯＳの一部であり５章で述べるＰＵ（Ｐｒｏｄｕｃｔ　Ｕｎｉｔ）の１つにあたる。このＰＵは、１７モジュール、１０００　ｌｉｎｅからなり、モジュールのＣｙｃｌｏｍａｔｉｃ　Ｎｕｍ－ｂｅｒ［ＭｃＣＡ７６］の平均値は６．９である。テストは、コーディング後の机上デバグ、単体、結合、システムテストの順に進めた。その結果、机上デバグで１０個のバグを、マシンによるテストで１５個のバグを発見した。実験結果を図４に示す。

到達難易度を求める際、バグ潜在点の通過確率を算定することが困難なため、分岐後のエッジの通過確率は分岐前の半分になる、と仮定した。

（１）　到達難易度

マシンテストによって発見されるバグの到達難易度A_{mh}は、テスト工程間で差がないことが図４（ａ）よりわかる。到達難易度のスケールを１、１／２、１／４、１／８、１／１６から、１、２、３、４、５に変換するとA_{mh}はほぼ変換後の到達難易度A_{pr}の平均値を中心とした正規分布となる。
プログラムの平均到達難易度A_{pr}の平均値は次のように表わされる。

$$A_{pr} = \frac{\sum_{i=1}^{m} A_i S_i}{\sum_{i=1}^{m} S_i}$$

A_i ： エッジｉの到達難易度
S_i ： エッジｉのステップ数
ｍ ： プログラム中のエッジ数

また、机上デバグで発見したバグの平均到達難易度A_{dk}も、A_{pr}、A_{mh}と大きな差のないことがわかる。

（２）　発生条件複雑度

図４（ｂ）より明らかなように、発生条件複雑度は、テスト工程の各段階のそれぞれの特徴を的確にあらわしうる。つまり、単体テストと結合テストの前半の段階では、簡単なバグが発見され、結合テストの後半とシステムテストの段階で複雑なバグが発見される。

（３）　顕在化難易度

図４（ｃ）で示すように、テスト工程の早期は右上りの勾配になり、中途で右下がりの勾配（顕在化が容易）になる。この現象は一見奇異に見えるが発生条件複雑度と顕在化難易度の関係を見ると理解できる。すなわち試験当初の単体テスト中は、モジュール内に閉じるバグの摘出により顕在化難易度は徐々に高くなるが、その後の結合テストの初期には単純なモジュール間インタフェースのミスによる顕在化しやすいバグが、摘出されるためである。

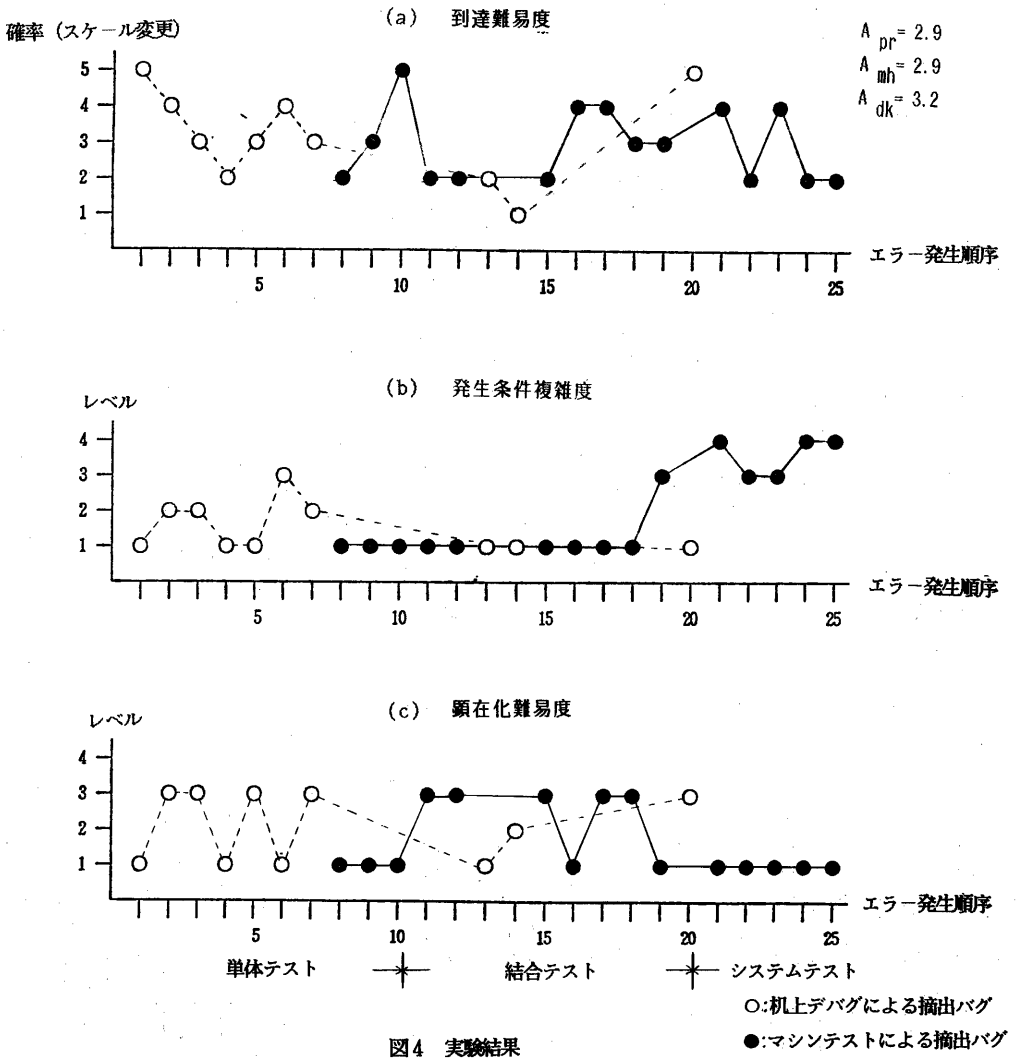


図４　実験結果

（４）　発生条件複雑度と顕在化難易度の関係

発生条件複雑度と顕在化難易度の関係を図５に示す。マシン摘出バグは、テスト工程が進むにつれてＡ、Ｂ、Ｃ、Ｄの順に遷移していく。テストが順調に行われるならば、発見バグは決してＡ、Ｂ、Ｄとは遷移せず、ＢとＤの間には必ずＣのステータスがあることに注目すべきである。また、机上デバグによって発見したバグは、マシンによるテストによって発見したバグのような特性は示さないこともわかる。

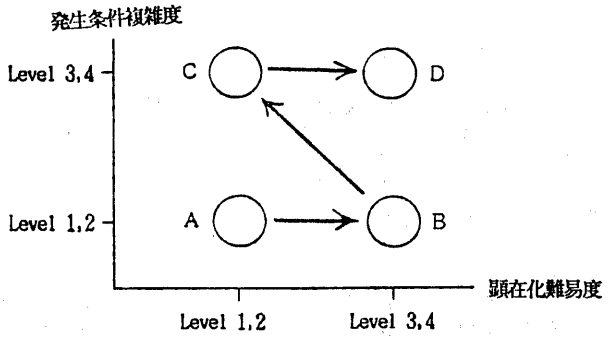


図5　発生条件複雑度と顕在化難易度の相関

5．　適用性

本章では、実験結果をもとにＣ＆Ｒ法における埋込みバグの選定に対する尺度、およびテスト工程の妥当性に関する尺度について考察する。

＜Ｃ＆Ｒ法における埋込みバグの選定尺度＞

Ｍｉｌｌｓが１９７２年にＣ＆Ｒ法のソフトウェア品質管理への適用を提案［ＭＩＬＬ７２］して以来、この方法は統計面［ＴＡＵＳ７７］、信頼度面［ＤＵＲＡ８１］、予測数の補正［ＯＨＢＡ８２］の観点等から研究されてきた。しかし、本格的にソフトウェアの品質管理に適用するには、まだ多くの解決すべき事項が残っている。

本方法における、最重要問題は埋込みバグの選定法の確立である。そのためには、埋込みバグの分布に

① 図６（ａ）に示すような機能的な偏り、

② 図６（ｂ）に示すようなバグ摘出難易度の観点からの偏り、

をなくし、潜在バグ数の予測精度を上げるような埋込みバグ選定のための指標を確立しなければならない。

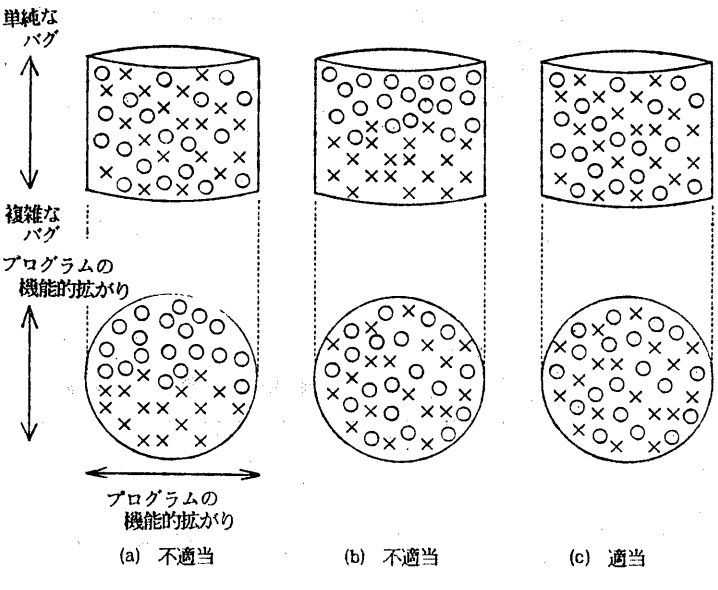


図6　埋込みバグ選択の基準

①に関しては、プログラムを同様な属性（作成言語・機能・開発期間等）をもつグループに分割するＰＵという概念（図７を参照）を提案し、良好な実験結果も得ている。［ＩＤＯ８１］

②に関しては、実バグと埋込みバグとの間で発生条件複雑度と顕在化難易度に関して有意差がないように、埋込みバグを選定することである。

そのためには次の点に注意を払う必要がある。

Ⅰ）　Ｃ＆Ｒ法の適用時期

Ⅱ）　Ｃ＆Ｒ法を適用するプログラムの特徴

Ⅲ）　Ｃ＆Ｒ法適用までに実施したテストの着実度

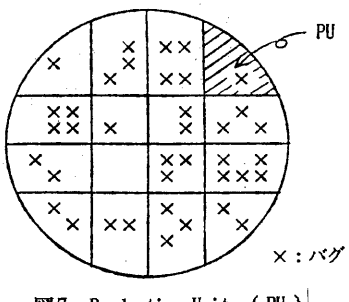


図7　Production Unit （ＰＵ）

＜テスト実施の着実度評価＞

ソフトウェアの品質をできるだけ早く向上させるためには、バグを発見すべきテスト工程で確実に見つけて行くことが重要である。ここではその評価尺度を提案する。この尺度はＤＤＴＤ（ｅｒｒｏｒ Ｄｅｔｅｃｔｉｏｎ Ｄｉｆｆｉｃｕｌｔｙ Ｔｒａｎｓｉｔｉｏｎ Ｄｉｓｔａｎｃｅ）と呼び、以下のように定義する。

“ＤＤＴＤは隣接した期間の２つの発見バグの間の遷移距離の平均”であり

$$DDTD = \frac{\sum_{i=1}^{m-1} D_{i,i+1}}{m-1}$$

ここで $D_{i,i+1}$ ： ｉ番目に発見されたバグとｉ＋１番目に発見されたバグの間の遷移距離
（遷移距離は図８により計算できその遷移距離マトリックスは表３の様に表わせる。図８は、図５のＡ、Ｂ、Ｃ、Ｄ間の遷移距離のダイヤグラムである。）

ｉ ： ｉ番目の発見バグ

ｍ ： 評価期間中に発見された全バグ数

と表わせる。

ＤＤＴＤの値が小さいほど、テストは抜けなく着実に実施されていると判断できる。逆にＤＤＴＤの値が大きいほど各工程でのテスト漏れの恐れがあり、バグが摘出しきれていない可能性が大きい。

４章に示した実験では、マシンテストによって発見したバグのＤＤＴＤは０．３５であり、机上デバグによって発見されるバグのＤＤＴＤは１．００であった。我々の他の実験結果も考え合せると、着実なマシンデバグにおけるＤＤＴＤは０．５以下であると言える。

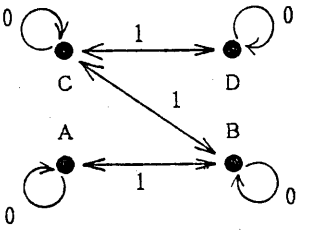


図８　遷移距離ダイヤグラム

表３　遷移距離マトリックス

Ｉ番目のバグの属すグループ ＼ Ｉ＋１番目のバグの属すグループ	Ａ	Ｂ	Ｃ	Ｄ
Ａ	0	1	2	3
Ｂ	1	0	1	2
Ｃ	2	1	0	1
Ｄ	3	2	1	0

６．　結論

本報告で提案した２つの尺度を、いくつかのプログラムに適用した結果以下のことがいえる。

すなわち、バグ摘出難易度尺度はＣ＆Ｒ法の埋込みバグの選定に有効である。

バグ摘出難易遷移距離ＤＤＴＤは各工程でのテストの着実性を評価するのに役だつ。

これらの尺度の問題点は計算がむずかしいことであるが、以下の手順に従うことによって容易に導出できる。

（１）　分岐毎に分岐確率を計算するのは困難であるので、分岐確率を５０％と仮定する。そうすれば、到達難易度はコンパイルリストやフローチャトを用いて容易に求められる。

（２）　発生条件複雑度と顕在化難易度も３章（表１、表２）に示す様な尺度を設ければ発見バグの分類が容易にできる。

（３）　ＤＤＴＤも、これらの発生条件複雑度や顕在化難易度の尺度を用いることによって容易に計算できる。

[文献]

[DURA81] J.W.DURAN,:'Caputure-Recapture Sample for Estimating Software Error Content.' IEEE Trans.SE,Vol.SE-7,NO.1,Jan. 1981,pp 147-148

[GOEL79] A.L.GOEL and K.OKUMOTO,:'Time-dependent Error-Detection Rate Model for Software Reliability and Other Performance Measure.', Vol.R-28,1979,pp 206-211

[GOEL80] A.L.GOEL,:'Software Error Detection Model with Applications.' J.Systems & Software, Vol.1,1980,pp 243-249

[IDO81] S.IDO and N.MURATA, :'Estimating Hidden Bugs using the Capture & Recaputure Method and its Application.'(in Japanese),IPS-J proc.WGSC Meeting, Vol.19,1981,pp 1-10

[MILL72] H.D.MILLS,: 'On the Statistical Validation of Computer Programs' IBM FDS Rep.,1972

[McCA76] T.J.McCABE,:'A Complexity Measure.' IEEE Trans.SE,Vol.SE-2,No.4 ,Dec.1976

[OHBA82] M.OHBA,:'Software Quality=Test Accuracy ×Test Coverage.'6th ICSE,Sep.1982

[SCHI78] G.J.SCHICK and R.W.WOLVERTON,:'An Analysis of Computing Software Reliability Models. IEEE Trans,SE,Vol.SE-4,No.2,MAR.1978,pp 104-120

[TAUS77] R.C.TAUSWORTHE,:'Standardized Development of Computer Software.'Prentice-Hall, Englewood Cliffs,N.J.,1977