ソフトウェア工学　46－12
（１９８６．２．７）

残存誤り数の一推定法

若杉忠男
三菱電機株式会社，コンピュータシステム製作所

１　概要

ここで紹介するプログラムの残存誤り個数の推定法は，次のようなものである．プログラムに一連のテストデータを入力し，重複を含む誤り発見回数と重複を除いた発見個数との比率から残存誤りを推定する．これは誤り累積曲線のあてはめによる推定法と違い，発見誤りの記録を継続的にとる煩わしさがない．ここで，その理論的根拠，予測に使う数表と使い方，とシミュレーションの結果を述べる。

この方法の考え方を簡単に述べると次の様なものである．

テスト中のプログラムに何ケースかのテストデータを入力する．その間プログラムの修正や変更をしないものとすれば，同じ誤りを別のテストケースで重複して発見することがある．誤りが残り少なくなれば，発見される新規の誤りは少なくなり，同じ誤りを重複して発見する割合は増加するであろう．

ここに，

N1 ：重複を含めた誤り発見回数．

N2 ：重複を除いた誤りの発見個数．

と定義すれば，その比N1 ／N2 はテストが進捗するにつれ増大するであろう．従ってこれによりテストの進捗度が掴め，これとN2 から，残存誤り数が推定できる．更に，それまでに入力したテストケース数から，あと何ケースのテストデータが必要かの見積りも得られる．

この方法はその特長から考えて，テスト途中で行うプログラムの品質評価，いわゆる探針（probe)に有効と思われる．またテストケースを査読者におきかえて考えれば，ドキュメントの 品質の評価にも使える．

この方法のメリットは，ロジスティック曲線やゴンペルツ曲線などによる曲線あてはめによる評価方法と違い，長期間にわたる連続したテスト記録をとらなくても残存誤り個数を推定できる点にある．

２　理論

２．１　前提条件と定義

テストすべきプログラムと一連のテストケースとに対し，次の仮定を置く．

（１）各テストケースは互いに独立で，同じ誤り発見能力（以下Ｐと記す）を持つものとする．

ここで誤り発見能力とは一ケースで発見できる誤りの,残存誤り数に対する割合である．そしてこの仮定ではそれが残存誤り数によらず一定とする．いいかえれば，このＰでテストデータ作成者の技術力の質とプログラムのテスト難易度が表されると考えられる．

（２）誤りは互いに確率的に独立とする．

たとえばある誤りをとらなければ，他の誤りがとれないというようなことはないし，一つの誤りがとれれば，他の誤りもとれるというようなことはない．

（3）この一連のテスト作業中，これらの確率的性質は変わらないものとする．
すなわち，テスト中プログラムの状態は変えないものとする．
これらの仮定のもとに，次のように定義する．

K　：テストデータのケース数
Xi　：i番目のテストケースで見つけた誤りの個数．このケースで見つけたものは他のテストケースで見つけたものと重複しても含める．
N1　：重複を含めた誤り発見回数．

$$N1 = X1 + X2 + \cdots + Xk \qquad (1)$$

たとえば1つの誤りを3つのテストケースで見つけた場合，3と数える．ただし，1つの誤りを同じテストケースで2度以上見つけても，1と数える。
N2　：重複を除いた誤りの発見個数．すなわち1つの誤りを2つ以上のテストケースで見つけても，1と数える．
N2 ≦N1　である．
W　：テスト指数．W＝N1／N2　と定義する．　（2）
これが大きなほどテストが捗っていることが分かる．
Wi　：Wのサンプル値．W＝E（Wi），Eは平均を表す．
N　：誤り総数．残存誤り個数＋今考察しているテストで発見した誤り個数．過去に見つけた誤りは含めない．
U　：テストケース1ケース当たりの誤り発見個数の平均．

$$E(Xi) = U \qquad (3)$$

P　：テストケースの平均誤り発見能力．テストケース1ケースで残存誤り数の何割を見つけられるか，その割合．

$$P = U / N \qquad (4)$$

Ui　：Uのサンプル値

$$Ui = N1 / K = (X1 + X2 + \cdots + Xk) / K$$

Pi　：Uiから算出したPの推定値

前提条件で述べたように，U，Pは当該プログラムテスト中，変わらないものと仮定する．

2．2　式の誘導

2．1で述べた条件から，次のような定理がなりたつ．

定理1

先に述べた条件のもとに次の式が成り立つ．

$$W = K \times P / \{1 - (1 - P)^k\} \qquad (5)$$

定理2

Pは　$P_0 = W / K$　を初期値として，（5）式を変形した下の式で反復代入して求めることができる．

$$p_i = \{1 - (1 - p_{i-1})^k\} \times W / K \qquad (6)$$

図２．２－１はこうして得たW，KとPの関係をグラフで示したものである．

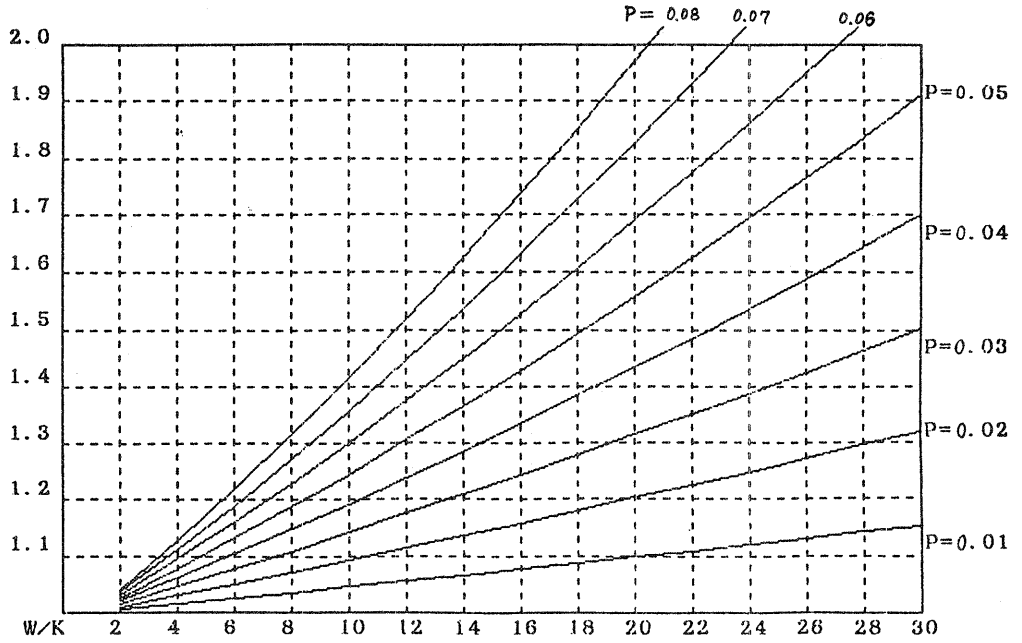


図２．２－１　W，K，Pの関係

定理３

定理２によって求めたPを使って，誤りの総数Nは次の式で求められる．

$$N = E(N1) / (P \times K)$$
$$= E(N2) \times W / (P \times K) \qquad (7)$$

表２．２－１は上記の定理によって求めた数表である．この表には縦方向にW，横方向にKをとり，それに対応する　S＝W／（P×K）　がのせてある．したがって，誤り総数Nは，

（誤り発見個数N2）×（表から読んだS）　で求められる．

右端には次に述べるGoel，Okumotoの曲線から得られた数値をのせてある．

ここで、T＝P×Kをあらたに，テスト進捗度と定義すると，

テスト進捗度＝平均誤り発見能力×テストケース数　である．

これを使って（５）式を書き直すと，Kが大なるとき次の近似式が成り立つ．

$$E(N2) = N \times (1 - e^{-T}) \qquad (8)$$

これはGoel，Okumotoの曲線であり，発見した誤りの累積を表している．これを図２．２－２に示す．このグラフによって，テストの進捗度Tが増加するにつれて，重複して発見される誤りの割合Wが増加することが分かる．

表２．２－１　ＷとＫからＳを求める表

W	K=5	10	15	20	25	30	35	40	45	G-O
1.05	8.60	9.62	9.96	10.12	10.21	10.27	10.31	10.34	10.35	10.65
1.10	4.61	5.14	5.32	5.41	5.46	5.49	5.52	5.53	5.55	5.67
1.15	3.29	3.65	3.77	3.83	3.87	3.89	3.91	3.92	3.93	4.02
1.20	2.63	2.91	3.00	3.05	3.08	3.09	3.11	3.12	3.12	3.19
1.25	2.24	2.46	2.54	2.58	2.60	2.62	2.63	2.63	2.64	2.69
1.30	1.98	2.17	2.23	2.27	2.29	2.30	2.31	2.32	2.32	2.36
1.35	1.79	1.96	2.02	2.05	2.06	2.07	2.08	2.09	2.09	2.13
1.40	1.66	1.81	1.86	1.88	1.90	1.91	1.91	1.92	1.92	1.96
1.45	1.55	1.69	1.73	1.75	1.77	1.78	1.78	1.79	1.79	1.82
1.50	1.47	1.59	1.63	1.65	1.67	1.67	1.68	1.68	1.69	1.72
1.55	1.40	1.51	1.55	1.57	1.58	1.59	1.60	1.60	1.60	1.63
1.60	1.35	1.45	1.49	1.50	1.51	1.52	1.53	1.53	1.53	1.56
1.65	1.30	1.40	1.43	1.45	1.46	1.46	1.47	1.47	1.48	1.50
1.70	1.26	1.35	1.38	1.40	1.41	1.42	1.42	1.42	1.43	1.45
1.75	1.23	1.32	1.34	1.36	1.37	1.37	1.38	1.38	1.38	1.40
1.80	1.20	1.28	1.31	1.32	1.33	1.34	1.34	1.34	1.35	1.37
1.85	1.18	1.25	1.28	1.29	1.30	1.31	1.31	1.31	1.31	1.33
1.90	1.16	1.23	1.25	1.27	1.27	1.28	1.28	1.28	1.29	1.30
1.95	1.14	1.21	1.23	1.24	1.25	1.25	1.26	1.26	1.26	1.28
2.00	1.13	1.19	1.21	1.22	1.23	1.23	1.24	1.24	1.24	1.26

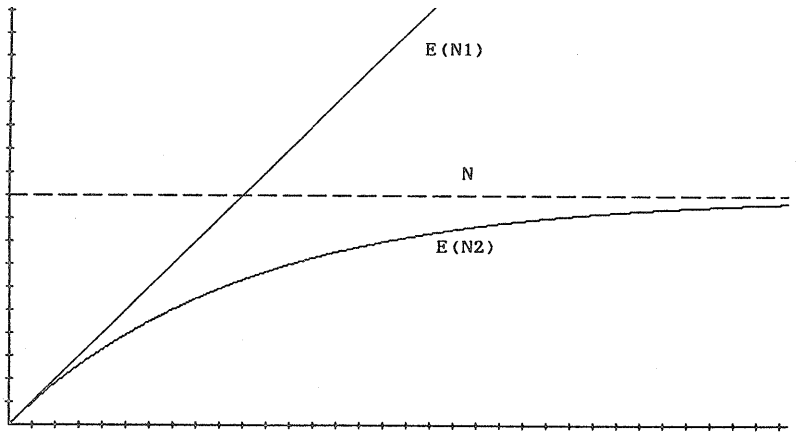


図２．２－２　Ｅ（Ｎ1）とＥ（Ｎ2）の関係

３　適用

３．１　適用の仕方

今ここにテスト中のプログラムがあるとする．すでに自明な誤りは取りつくされ，また一部分にかたまっているような誤りもなくなり，テスト作業が軌道にのったとする．このような状況で，テストデータを２０件ほど偏らないように選んで入力し，その間プログラムを変更しないとすれば，２．１の前提が成り立つと思われる．　これらのテストケースによりテスト指数Ｗiを求め，これとＫから表２．２－１によりＮの推定値を求めることができる．

３．２　シミュレーションの結果

この方法の有効性を見るために，ＢＡＳＩＣ言語のプログラムにより，乱数を使ってシミュレーションした結果を示す．

プログラムの手順は次の通りである．

（１）　１００個のデータエリアを用意し，その内の５０個に１を埋め他は０とする．この５０が誤りの総数Ｎにあたる．

（２）　１から１００までの一様乱数を一組につきＬ個ずつ発生させ，その数にあたるエリアを見る．この一組がテストケースの一件にあたる．

（３）　そのエリアに０が入っていたら，誤りが見つからなかったと見なす．

そこに１が入っていたら誤りを見つけたものとみなし，その１を－１に置き替える．

もしエリアに－１が入っていたら、それは誤りを重複して見つけたことになる。

今，乱数をＬ個ずつ発生させたとすると，誤り発見能力Ｐ＝Ｌ／１００となる．このＰの値は初めにうめこんだＮの値には依存しない。なんとなれば、誤りを見つける確率はＬ×Ｎ／１００であり，ＰはＮの中の何％を１ケースのデータで見つけることができるかを表す数値であるから，Ｌ×Ｎ／１００をＮで割って
Ｌ／１００となる．

ここでは次の条件で計算した．

（１）　誤りの総数　Ｎ＝５０

（２）　乱数発生個数／一組　Ｌ＝２，４　の２種．
すなわちＰ＝０．０２，０．０４の二種

（３）　テストケース数　Ｋ＝２０，３０　の２種について

（４）　１００回繰り返してシミュレーションする

（５）　１００回のシミュレーションの中から，Ｎ1＝Ｎ2となるケースは除いて集計した．

こうして得た誤り総数Ｎのヒストグラムと平均と標準偏差を次に示す．

これらの結果から，次のようなことが分かった．

（１）　テストケース数Ｋと誤り発見能力Ｐが大なほどＮの推定値の精度はよくなる．これらの積　Ｔ＝Ｐ×Ｋ　をテスト進捗度と定義した理由はここにある．

（２）　Ｋ＝３０，Ｐ＝０．０４　くらいの場合では，Ｎの真の値が５０であるのに対し，平均が５０．６，標準偏差が７．６となり相当の精度で見積れる．（表３．３－１のナンバー４参照）．

（３）　Ｋ＝２０，Ｐ＝０．０２の場合，平均６２．６，標準偏差３５．５となり，ケースによっては数倍の見積り誤差を生じることがある．（表３．３－１のナンバー１）．

その状況を調べてみると，Ｎ1－Ｎ2＝１の場合，すなわち重複して見つけた誤りが１個しかない場合には，誤差が大きくなることが分かった．

（４）　上記の例について　Ｎ1－Ｎ2＞１　という条件を満たした場合のみのデータをとってグラフ化したところ，Ｎについては平均５３．５，標準偏差２０．２と大幅な改善がみられたが，Ｐの推定値は逆に０．０２１から０．０２４と悪い方に変化した．（表３．３－１のナンバー５）．

表3．3－1　シミュレーション結果

ナンバー	ケース数K	誤り数N	発見能力P	Nの平均，標準偏差		Pの平均，標準偏差	
1	20	50	0.02	62.6	35.5	.021	.010
2	30	50	0.02	59.2	31.7	.020	.007
3	20	50	0.04	52.5	13.0	.041	.011
4	30	50	0.04	50.6	7.6	.041	.007
5	20	50	0.02	53.5	20.2	.023	.009

ナンバー5は，ナンバー1と同じ条件のもとで重複発見誤り数が一個以下のケース16件を除いて集計したもの．

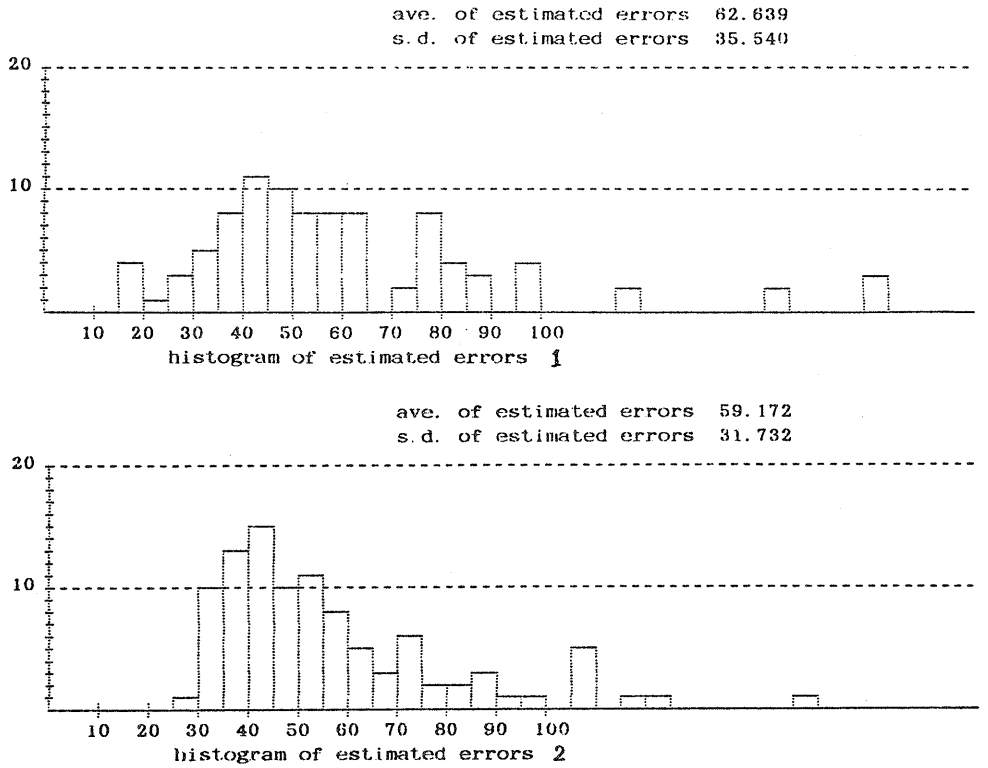
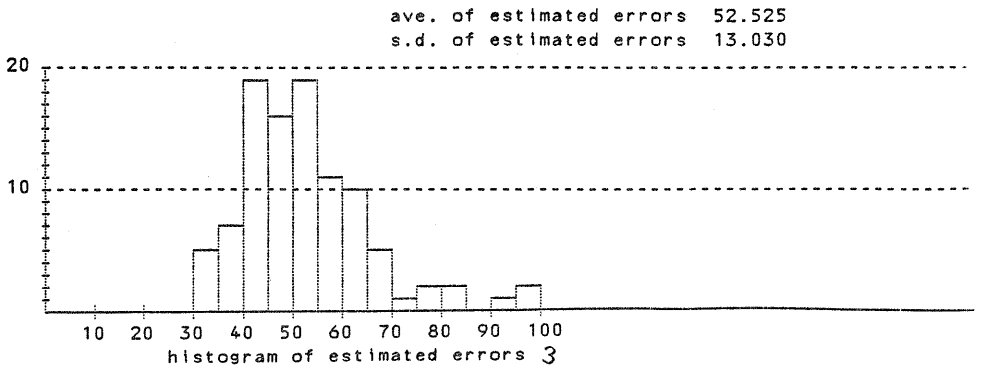
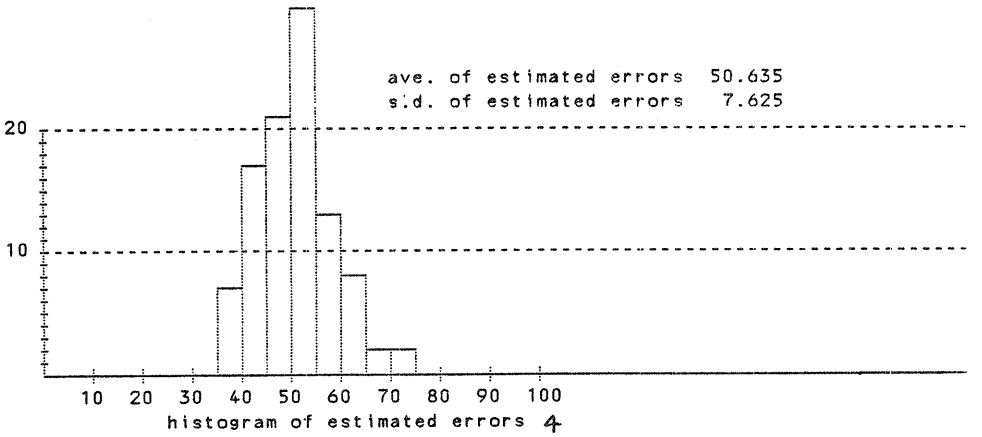


図3．3－1　シミュレーション結果





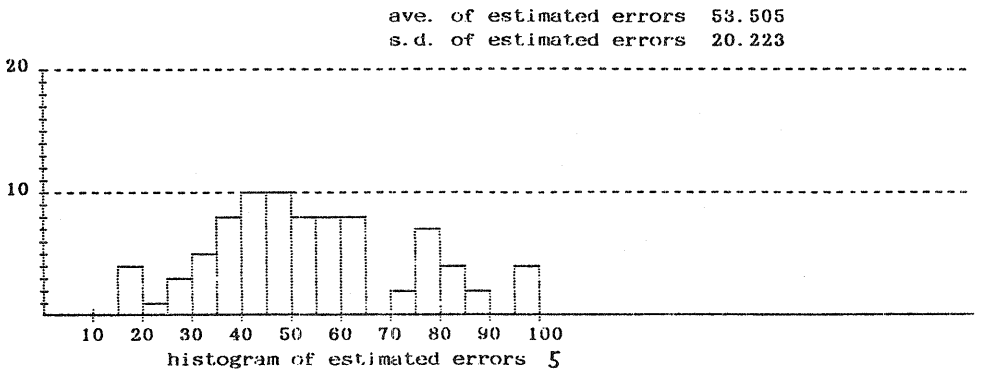


図3．3－1　シミュレーション結果（続）

3．3　考察

（1）　一般にテストを行う場合には，作業効率をあげるためテスト項目が重複しないようにテスト計画をたてる．したがって誤りを重複して見つけないように意図的な操作がなされるので，ここで述べた方法を良く考えずに使うのは効果的ではないであろう．

しかし，テスト作業の進捗とプログラムの品質把握のために開発者以外の第三者が行う探針作業とか，システムの出荷の可否を決定するシステム検査などには，この方法は向くと思われる．それ以外にも，たとえば単体テストの最終仕上げとか，サブシステムのテストの最終段階などに有効に用いることができよう．

そのためにはテストのやりかたから考える必要がある．

（2）　この方法は，2．1で述べた条件のもとで使えるわけであるが，この条件はテストが安定した状態に入っていることを示す．したがって，この方法はテスト作業が軌道に乗った状態，すなわちテストの後半の段階に向くであろう．そういう点から考えても，この方法は，テストの進捗状態の確認，すなわち探針や第三者検査に向くと考えられる．

（3）　この方法はドキュメントのチェックにも使うことができる．

今までKをテストケース数と考えてきたが，これをレビューする人の数，Pをその人達の平均の誤り発見能力と見なせば，Nはドキュメントチェックによる誤り総数の推定になる．すなわち，K人の人が手分けしてドキュメントの査読を行い，それから得られた誤りの発見データから，誤りがどのくらいあるかを推定できる．

この場合，Kは5，6人で，Pは0．3から0．6といった数値になるであろう．

（4）　シミュレーションの結果などから，次のことがいえる．

a　重複して見つかった誤りの個数が1個の場合には，重複個数が2個以上になるまでテストケース数を追加してからこの方法を使うと良い．

b　Pが小さいときにはケース数を大きくしなければならない．
$P \times K \geqq 1$　くらいになるようにKを選ぶと良さそうである．

c　Kが大な場合は，誤り総数をGoel，Okumotoの式により近似的に，
$N = E(N2) / (1 - e^{-PI})$　で見積ってさしつかえない．

4　参考文献

（1）　近代確率論　国沢清典　1958年
（2）　近代数理統計学通論　宮沢光一　1958年
（3）　ソフトウェアの品質評価法　三はし武　1981年
（4）　BASICによる統計　石原辰雄　1984年

以上