解　説

超幾何分布にもとづくソフトウェア残存フォールト数推定モデル†

当　麻　喜　弘††

1. はしがき

周知のようにプログラムの開発は，通常，(1)仕様の決定，(2)設計，(3)コーディング，(4)テストとデバッグ，という過程を経て行われる．以下のモデル化の話は，最後の(4)に関するものである．

コーディングされたプログラムはもちろんプログラマによってテストされ，その間，フォールト☆が見つかれば除去される．しかし，プログラマとは別のテスト作業者によって行われるテストで，さらにフォールトが発見されるのが普通である．このテストを継続すると，発見／除去されるフォールトの累積数は増加していく．この累積数の増加の過程をモデル化するのである．

このテスト／デバッグ過程のモデル化は次のような意味をもつ．まず第1に，テスト／デバッグを始めた時点のフォールト数（以後，これを初期フォールト数という）はだれにも分かっていないので，デバッグを継続しても，いつの時点でプログラム中のフォールトを全て取り除いたかを決定的に断定することができない．それゆえ，いつプログラムをサービスにリリースしたらよいかという判断に苦労する．モデル化を通してプログラムの初期フォールト数を推定できれば，発見／除去されたフォールト数から残存フォールト数を推定できるから，（たとえば，以後のテスト／デバッグの手数とサービス時におけるクレームとの兼ね合いから）リリース時期を定める手がかりが得られる．

第2に残存フォールト数が推定されるから，プログラムの信頼性が（厳密に数量的にではないが）ある程度推察される．

これまで，傾向が似ている数式で発見／除去されるフォールトの累積数の増加を単純になぞるもの，フォールトの発見／除去過程を非斉次ポアソン過程とみなし，ハザード率を適当に定めるもの，といったモデルが提案され，また一部に実用されてきた．われわれはこれらとは異なり，超幾何分布の観点から新しいモデルを提案している．

2. 基本的考え方

われわれのモデルの基本的考え方は，フォールトがテストに誤りとして反応することと，テストが加えられたときにフォールトとして新たに発見されることを区別する点にある．さらに，発見／除去されるフォールト数の増加を表すときの横軸（テストの経過を意味する）に，時間をとる代わりに1回目，2回目，…というように加えたテストインスタンスの回数を用いる．

テストの経過を時間で表すと，本当にテストを実行している時間か，単なる経過時間（アイドル時間も含まれる）なのかはっきりしないことが多い．

われわれは，まず，テストを個々のテストインスタンスの集合と考える．すなわち，テスト T は

$$T=\{t_i\} \qquad (1)$$

t_i は第 i 回目に加えられるテストインスタンスを表す．テストインスタンスはテスト作業のまとまった単位である．したがって，テストインスタンスとして1日に行うテスト作業をとることもあるし，1週間のテスト作業をとることもある．必ずしも，細かなテスト入力の実行のみをテストインスタンスとするのではない点に注意を要する．

初期フォールト $f_1, f_2, \cdots, f_m$ の集合を F で表す．すなわち，

$$F=\{f_i \mid i=1, 2, \cdots, m\} \qquad (2)$$

したがって，初期フォールトの総数は m である．これはもちろん不明で，以下に述べる方法で推定する．

テストインスタンス t_i に対して，フォールトに

† Hyper-Geometric Distribution Model for Estimating the Number of Residual Software Faults by Yoshihiro TOHMA (Department of Computer Science, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

☆ バグとも言う．

よっては誤りを生じたり，生じなかったりする．フォールト f_j が t_i に対して誤りとして反応する場合を

$$r(i,j)=1 \tag{3}$$

で表す．誤りとして反応しない場合は

$$r(i,j)=0 \tag{4}$$

である．これより t_i に対する反応係数 (sensitivity factor) $w(i)$ を次のように定義する．

$$w(i)=|\{f_j\in F|r(i,j)=1\}| \tag{5}$$

すなわち，$w(i)$ は t_i に誤りとして反応する初期フォールトの数を表す．

この $w(i)$ は，必ずしも，i 番目に加えられた t_i によって新たに発見／除去されるフォールトの数ではない点に注意しよう．第1回目のテストインスタンス t_2 を加えたときに新たに発見／除去されるフォールトの数はもちろん $w(1)$ である．しかし，第2回目の t_1 によって新たに発見／除去されるフォールト数は，必ずしも，$w(2)$ ではない．$w(2)$ のあるものは t_1 によってすでに発見／除去されているかも知れないからである．

ここで次を仮定する．

【仮定】

1. t_i によって発見されたフォールトは次のテストインスタンスが加えられる前に完全に除去される．
2. 発見されたフォールトの除去に際して新たにフォールトが挿入されることはない．
3. t_i に誤りとして反応する $w(i)$ 個のフォールトは m 個の初期フォールトからランダムに選ばれたものとする．

本来，t_i とそれに誤りとして反応する $w(i)$ 側のフォールトとの関係は明確に定まっているものであるが，この関係を考慮に入れるのは困難であること，また，どのテストインスタンスをいつの時点で加えるかは一定していず，むしろランダムに選ばれることが多いと思われるので，上の 3. を仮定したのである．

第 $i-1$ 回目までのテストインスタンスを加えた時点までに発見／除去されたフォールトの累積数を $C(i-1)$ とする．すると 第 i 回目の t_i によって新たに発見/除去されるフォールトの数 $N(i)$ が x となる確率 Prob $\{x|m, C(i-1), w(i)\}$ は，上の仮定から

$$\text{Prob}\{x|m,C(i-1),w(i)\}=\frac{\binom{m-C(i-1)}{x}\binom{C(i-1)}{w(i)-x}}{\binom{m}{w(i)}} \tag{6}$$

となる[1]．ここで x は

$$\left.\begin{aligned}&0\le x\le U_x\\ &\text{ただし，}U_x=\min\{w(i),\ m-C(i-1)\}\end{aligned}\right\} \tag{7}$$

の範囲にある．式(6)が表す確率分布を超幾何分布 (Hyper-Geometric Distribution) という．

3. 発見/除去されたフォールトの累積数推定

式(6)から $N(i)$ の期待値 $\overline{N(i)}$ が次のように求まる．

$$\overline{N(i)}=\{m-C(i-1)\}\frac{w(i)}{m} \tag{8}$$

$C(i)$ は，明らかに

$$C(i)=\sum_{k=0}^{i}N(k) \tag{9}$$

であるが，$N(k)$ の代わりに $\overline{N(k)}$ を用いた次式を $C(i)$ の推定値 $E[C(i)]$ とする．

$$E[C(i)]=\sum_{k=0}^{i}\overline{N(k)} \tag{10}$$

また，式(8)においても $C(i-1)$ の代わりに $E[C(i-1)]$ を用いる．すると次の漸化式が得られる．

$$E[C(i)]=E[C(i-1)]\left\{1-\frac{w(i)}{m}\right\}+w(i) \tag{11}$$

この解は[2]

$$E[C(i)]=m\left\{1-\prod_{j=1}^{i}\left(1-\frac{w(j)}{m}\right)\right\} \tag{12}$$

近似式として[3]

$$E[C(i)]=m\{1-e^{-\frac{1}{m}\int_0^i w(j)dj}\} \tag{13}$$

ハードウェアの信頼度を評価する場合の古典的信頼性理論においてハザードを一般の $\lambda(t)$ とすると不信頼度（障害発生の時間に関する分布関数）は $F(t)=1-e^{-\int_0^t\lambda(\tau)d\tau}$ となるが，式(13)がこれと酷似した形式をしているのは非常に興味深い．

$E[C(i)]$ を実際に数値的に求めるためには，m および $w(j)$, $j=1,2,\cdots,i$ 中のパラメータの値を定めなければならない．この点については後述する．

4. 反 応 係 数

実際のテストデータとして $w(i)$ が陽に与えられることはまずない．そこで与えられたテストデータの内容に合わせて $w(i)$ を定式化する．

1回のテストインスタンスに誤りとして反応するフォールトの数は，そのテストインスタンスにおけるテスト項目数，実際の処理時間，従事したテスト作業者の数などによって変わるであろう．また，テストの

経過に応じたテスト難易度の変化（たとえば，テストの後半ではテスト作業者の熟練度が上がることが考えられるし，逆に，発見困難なフォールトのみが残るということも起こりえよう）も考慮に入れたほうがよいであろう．そこで

$$w(i)=u(i)\cdot v(i) \tag{14}$$

とし，実際のテストデータに，各 t_i におけるテスト作業者の数 $tester(i)$，テスト項目数 $I(i)$，処理時間 $h(i)$ などが与えられていれば，それらを $u(i)$ に代入する．また，$v(i)$ は難易度を表す項で，われわれは簡単のために，次のような直線的に変化するものと飽和特性をもつものを試みた[3]．

$$v(i)=a_1 i+b_1 \tag{15}$$

$$v(i)=a_2 i^2+b_2 i+c_2 \tag{16}$$

しかし，両者の間にはほとんど差異が認められなかったので，現在は式(15)を用いればよいと考えている．

このように，いろいろな形式のテストデータへの適用可能性がわれわれのモデルの一つの特色である．実際のテストデータへの適用例は 8. に示す．

5. パラメータの推定

さて，式(15)を用いると $E[C(i)]$ を計算するために得なければならないパラメータは $\{m, a_1, b_1\}$ の三つである．現在は，これらの値を3次元的にしらみつぶしに探索する方法を用いている．その際，暫定的な値を用いて計算した $E[C(i)]$，$i=1,2,\cdots$ を実際に観測される $C(i)$ と比較して，次の式で評価する☆．

$$EF1=\frac{1}{n}\sum_{i=1}^{n}|C(i)-E[C(i)]| \tag{17}$$

この $EF1$ を最小にするように m，a_1，b_1 の値を選び，それぞれの推定値 $E[m]$，$E[a_1]$，$E[b_1]$ として用いる．この段階で，本来，目的としていた初期フォールトの数 m の推定値が得られる．

われわれのモデルに関して，このパラメータの決定法が難しいという批判がある．3次元探索は計算機が行い，内容的にはきわめて単純な作業である．計算機の労力は大変であるが，困難であるというのは当たらないとわれわれは考えている．

しかしながら，パラメータ推定をもう少し解析的に行い，推定値を高速に得る試みを行っている．若干の結果を得ているが，これらについては改めて報告したい．

☆ 誤差自乗和法なども試みたが大差なかった．

6. 構造的取り扱い

テストインスタンスによってはテストの性格が他と異なることがある．たとえば，テストの初期ではモジュール機能テストを行い，後半でそれらを組み合わせた総合テストを行うということもあろう．性格の異なるテスト結果を他と同一に扱うのは適当ではないと考えられる．

そこでテストインスタンスをテストの性格／内容によって類別し，F を部分集合に分割することが考えられる．この部分集合ごとに上に述べた超幾何分布モデルを適用するのである．われわれはこの考え方に立って，まず，単純にテスト期間をいくつかの部分区間に分け（これを segmentation と呼んでいる[1]），部分区間の境界における $E[C(i)]$ の値の受渡しに留意しつつ☆，部分期間ごとにそれぞれのパラメータを決定する．この方法によって，性格を異にするテストインスタンスを含むテストの場合，$C(i)$ と $E[C(i)]$ の適合状況（fitness）は著しく向上する．

さらに一歩踏み込めば，テストインスタンスによってはプログラム全体をテストしているのではなく，構成要素である一部のモジュール，あるいはモジュールのいくつかをテストしているに過ぎないこともあろう．したがってテストインスタンスに関してその領域（domain）を定め，領域ごとにテストインスタンスを類別し，それらについて超幾何分布を適用することが考えられる．この場合，注意しなければならないのはこれらの領域が互いに disjoint であるとは限らないことである．モジュールによっては複数のテストインスタンスによって重複してテストされることもあり得るからである．したがって，各領域 $D(k)$，$k=1,2,\cdots$ ごとに初期フォールト数を推定しても，それらの和をもってプログラム全体の初期フォールト数とするわけにはいかない．領域の重なり部分を重複して数えている数を推定し，重複分を差し引かなければならない（われわれはこれを composite estimation と呼んでいる[1]）．

最も理論的に厳格な立場は segmentation と composite estimation を組み合わせることであろう．このためには，テストデータの記録の形式をあらかじめ注意深く整えておく必要がある．しかし，上の両者を組み合わせた推定作業はかなり面倒なものになると思われ，そこまですることが実用的であるか疑念が残るところである．

☆ j 番目の部分区間の最後の $E[C(i)]$ を，次の $j+1$ 番目の部分区間の $[EC(0)]$ とする．

7. 他モデルとの関係

式(12)もしくは式(13)を用いて，われわれのモデルと他のモデルとの関係を明らかにすることができる．ここでは後者を用いることとし，前者を用いたより厳格な比較は文献 2) を参照されたい．

1. $w(i)$ を定数 w とおけば，式(13)は

$$E[C(i)]=m(1-e^{-\frac{w}{m}i}) \tag{18}$$

これは，Goel-Okumoto モデルの平均値関数(ハザード関数) の a, b をそれぞれ m, m/w とおいたものに等しい．したがって，われわれのモデルは，テストインスタンスごとに b の変化を考慮して Goel-Okumoto モデルを拡張したものとみなせる．

2. $w(i)$ を

$$w(i)=w(ai+b) \tag{19}$$

とおくと

$$E[C(i)/m]=1-e^{-\{x+(kx)^2\}} \tag{20}$$

ただし，

$$x=\frac{w}{m}bi,\quad k=\frac{1}{b}\sqrt{\frac{ma}{2w}} \tag{21}$$

x に対して $E[C(i)/m]$ の値は図-1 のように変化する．これより，われわれのモデルも，大場[5]，山田[6]らのモデルと同様に，累積数のS字型の変化に対応できることが分かる．

3. $w(i)$ を

$$w(i)=m\gamma\alpha\beta bi^{b-1}e^{-\beta i^b} \tag{22}$$

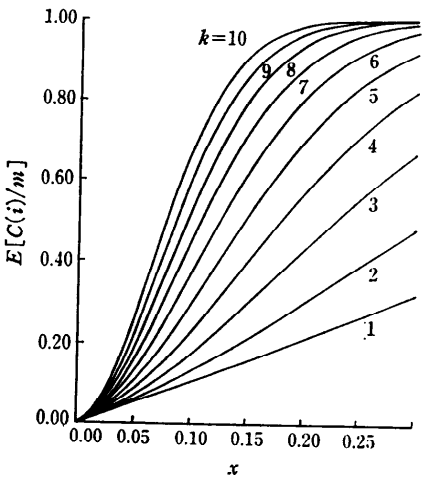

図-1 $E[C(i)/m]$ の変化

表-1 テストデータ 1 (* 印は内挿したもの)．

Days	Faults	Cumulated Faults $C(i)$	Number of Test Workers *tester*(i)	Days	Faults	Cumulated Faults $C(i)$	Number of Test Workers *tester*(i)
1	5*	5*	4*	57	2	448	4*
2	5*	10*	4*	58	3	451	4
3	5*	15*	4*	59	2	453	4
4	5*	20*	4*	60	7	460	4
5	6*	26*	4*	61	3	463	4
6	8	34	5	62	0	463	4*
7	2	36	5	63	1	464	4*
8	7	43	5	64	0	464	4*
9	4	47	5	65	1	465	4*
10	2	49	5	66	0	465	3*
11	31	80	5	67	0	465	3*
12	4	84	5	68	1	466	3*
13	24	108	5	69	1	467	3
14	49	157	5	70	0	467	3*
15	14	171	5	71	0	467	3*
16	12	183	5	72	1	468	3*
17	8	191	5	73	1	469	4
18	9	200	5	74	0	469	4*
19	4	204	5	75	0	469	4*
20	7	211	5	76	0	469	4*
21	6	217	5	77	1	470	4*
22	9	226	5	78	2	472	2
23	4	230	5	79	0	472	2*
24	4	234	5	80	1	473	2*
25	2	236	5	81	0	473	2*
26	4	240	5	82	0	473	2*
27	3	243	5	83	0	473	2*
28	9	252	6	84	0	473	2*
29	2	254	6	85	0	473	2*
30	5	259	6	86	0	473	2*
31	4	263	6	87	2	475	2*
32	1	264	6	88	0	475	2*
33	4	268	6	89	0	475	2*
34	3	271	6	90	0	475	2*
35	6	277	6	91	0	475	2*
36	13	290	6	92	0	475	2*
37	19	309	8	93	0	475	2*
38	15	324	8	94	0	475	2*
39	7	331	8	95	0	475	2*
40	15	346	8	96	1	476	2*
41	21	367	8	97	0	476	2*
42	8	375	8	98	0	476	2*
43	6	381	8	99	0	476	2*
44	20	401	8	100	1	477	2*
45	10	411	8	101	0	477	1*
46	3	414	8	102	0	477	1*
47	3	417	8	103	1	478	1*
48	8	425	4	104	0	478	1*
49	5	430	4	105	0	478	1*
50	1	431	4	106	1	479	1*
51	2	433	4	107	0	479	1*
52	2	435	4	108	0	479	1*
53	2	437	4	109	1	480	1*
54	7	444	4	110	0	480	1*
55	2	446	4	111	1	481	1*
56	0	446	4*				

とおくと，式(13)は山田のモデル[7]の平均値関数と同じになる．上式はbの値によって異なった変化をするので，iによって直線的に変化するとした式(19)よりさらに柔軟にいろいろな場合に対応できるように拡張されたものと考えることができる．

4. $w(i)$を

$$w(i)=\frac{mb}{\eta^b}i^{b-1} \tag{23}$$

とおくと，式(13)は，福島らによって提案されたモデル[8]の検出フォールトの累積数の成長を表す式と同じものになる．

8. 適用例

表-1は，リアルタイム制御および監視システム用に開発されたプログラムのテストデータである．プログラムは高級言語で書かれ，全体で約200キロ行である．テストデータは1日単位でまとめられているので，この場合，1日ごとのテスト作業をテストインスタンスとする．また，テスト作業者数$tester(i)$が記録されているので，式(14)を

$$w(i)=tester(i)\ (a_1 i+b_1) \tag{24}$$

とする．

まず，全テストデータを一括して扱って$E[m]$，$E[a_1]$，$E[b_1]$を求め，これらを式(12)に代入して$E[C(i)]$を計算し，これを実際に観測された$C(i)$と比較したものが図-2である．太線が$C(i)$を，細線が$E[C(i)]$を示す．

さらに，テストの性格が

$t(1)$から$t(10)$までが組み合わせテスト1．

$t(11)$から$t(27)$までが組み合わせテスト2．

$t(28)$から$t(47)$までが全体テスト

$t(48)$から$t(55)$までが装置組み込みテスト

$t(56)$から$t(111)$までが総合テスト

であるので，それぞれのテスト区間を部分区間とし，6.で述べたsegmentationの考え方に従って推定を行った結果が図-3である．

その他，テスト項目数，処理時間が与えられている場合のテストデータへの適用例などについては，文献3)を，またcomposite estimationの適用例については，文献1)を参照されたい．

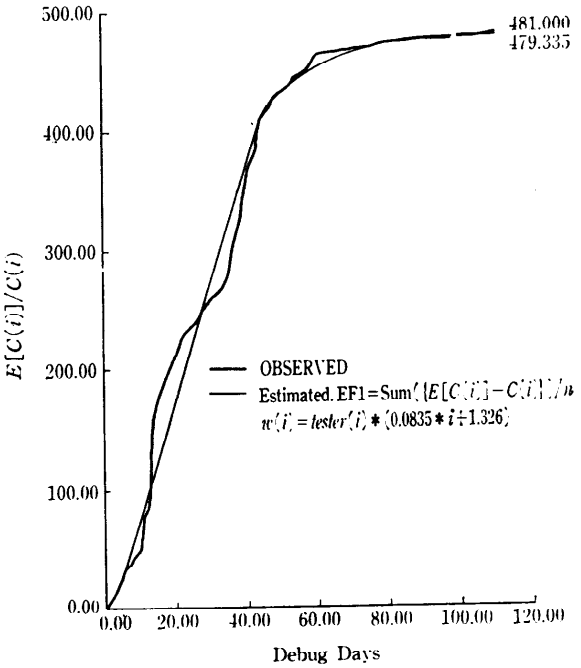

図-2 $E[C(i)]$(細線)/$C(i)$(実線)—その1

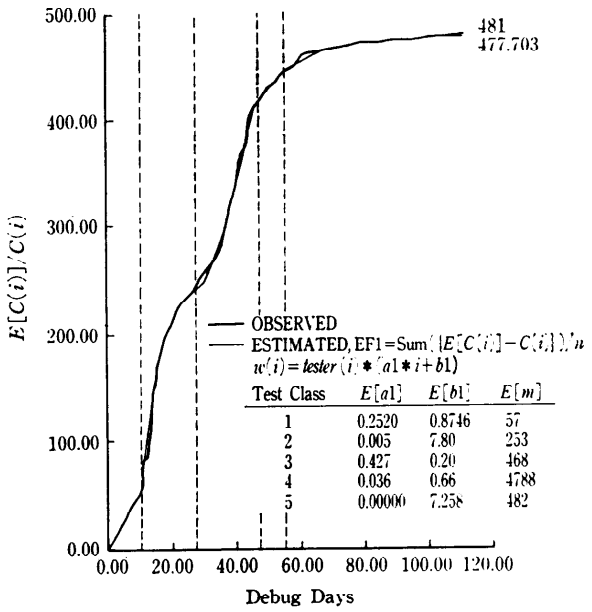

図-3 $E[C(i)]$(細線)/$C(i)$(実線)—その2

9. 今後の課題

われわれのモデルに限らず，一般に次のような問題提起がある．

1. 発見されたフォールトの除去に際して，不注意による新たなフォールトが挿入されるケースがままある．したがって，【仮定】の2．は十分に現実的とはいえないのではないか．

2. 初期フォールト数を推定するのにどの程度のテストインスタンスで十分なのであろうか．

3. テスト／デバッグ過程のみならず，フィールドサービス時の誤り生起の時間間隔を推定できないか．

上記の諸点に関連し，今後，われわれのモデルをさらに改良したいと思っている．

参 考 文 献

1) Tohma, Y., Tokunaga, K., Nagase, S. and Murata, Y.: Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution, IEEE Trans. Software Engg. Vol. 15, No. 3, pp. 345-355 (Mar. 1989).

2) Jacoby, R. and Tohma, Y.: The Hyper Geometric Distribution Software Reliability Growth Model (HGDM) : Precise Formulation and Applicability, To appear, COMPSAC 90, (Sep. 1990).

3) Tohma, Y., Jacoby, R., Murata, Y. and Yamamoto, M.: Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults, Proc. COMPSAC 89, Orlando, pp. 610-617 (Sep. 1989).

4) Goel, A. L. and Okumoto, K.: Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, IEEE Trans. Rel. Vol. R-28, No. 3, pp. 206-211 (Aug. 1979).

5) Ohba, M., Yamada, S., Takeda, K. and Osaki, S.: S-shaped Software Growth Curve: How good is it?, Proc. COMPSAC 1982, Chicago, pp. 38-44 (1982).

6) Yamada, S., Ohba, M. and Osaki, S.: s-Shaped Software Reliability Growth Models and Their Applications, IEEE Trans. Rel. Vol. R-33, No. 4, pp. 289-292 (Oct. 1984).

7) 山田 茂，大寺浩司：ソフトウェアの信頼性～理論と実践的応用～（第8章），ソフト・リサーチ・センター（1990年）．

8) 福島勝弘，岸田陽二：OA 機器ソフトウェアの実験的回帰分析によるバグ推定，シャープ技報，第37巻，pp. 71-75 (1987).

（平成2年7月12日受付）