Vol. 27 No. 1 情報処理学会論文誌 Jan. 1986

エラー修正率を考慮したソフトウェア信頼度成長モデル†

山田 茂†† 山根 勉††† 尾崎俊治†††

ソフトウェアの開発において，ソフトウェア内のエラー数の推移を把握することは，ソフトウェアの信頼性を評価する上で重要である．近年，この目的のために多くの信頼性モデルが開発され，実用化されている．本論文では，ソフトウェアの試験段階で発見されたエラーは，そのときの修正で必ずしも完全に修正されないことに注目し，エラー修正率を考慮した信頼性モデルについて議論する．このとき，エラー発見期間の単位として，カレンダー時間やＣＰＵ時間を用いる場合と，テストラン試行回数を用いる場合の，それぞれについてモデルを提案する．モデルの基本となる確率過程は，非同次ポアソン過程である．各モデルに対し，信頼性評価に有用な定量的尺度として，ソフトウェア内の期待残存エラー数や，ソフトウェア信頼度などを導出する．モデルの未知パラメータは，最尤法により推定される．さらに，実際のソフトウェアエラーデータへの適用例を示し，エラー修正率がエラー発見過程に与える影響について考察する．

1. まえがき

今日，コンピュータシステムの開発において，信頼性およびコストの両面でソフトウェアの占める割合が増大している．そのなかでも，ソフトウェアの試験に費やされる時間が非常に増加している．したがって，ソフトウェアの試験段階において，ソフトウェアの信頼性を効率よく評価する方法を確立することは急務となっている．

試験段階では，コンパイルの完了したソフトウェアに対し，エラー*の発見と修正のための試験が行われる．そして，その結果により，ソフトウェアの信頼性が評価される．このとき，発見されたエラーの修正により，ソフトウェア内に新しいエラーは作り込まれないものと仮定すると，試験時間の経過とともにソフトウェア内の残存エラー数は減少し，その結果，ソフトウェアの信頼度は向上する．このようなエラー発見過程を記述するモデルは，ソフトウェア信頼度成長モデル(software reliability growth model)と呼ばれる[1]．試験時間の単位，すなわち，エラー発見期間の単位としては，カレンダー時間やＣＰＵ時間のような実行時間，あるいはテストラン試行回数が用いられる．これまでに，この分野に属する多くのモデルが提案されてきた[2)~7)]．これらのモデルでは，発見されたエラーは，そのときの修正により完全に修正されるものと仮定している．しかし，一般に，発見されたエラーは，そのときの修正で必ずしも完全に修正されるとは限らない．そこで，不完全な修正を考慮してモデルを作成することは興味ある問題である．

本論文では，ソフトウェア信頼度成長モデルのエラー発見期間の単位として，試験時間を用いるモデル（以下，連続時間モデルと呼ぶ）と，テストラン試行回数を用いるモデル（以下，離散時間モデルと呼ぶ）を，それぞれ提案する．各モデルの基本となる確率過程は，非同次ポアソン過程(nonhomogeneous Poisson process, 以下NHPPと略す(Ascher and Feingold[8], Ross[9]参照))である．各モデルのもとで，ソフトウェアの信頼性を評価するために有用な定量的尺度を導出し，モデルの未知パラメータの最尤法 (method of maximum likelihood) による推定方法について述べる．さらに，実際のソフトウェアエラーデータへ適用し，モデルのデータへの適合性，およびエラー修正率がエラー発見過程に与える影響について考察する．

2. NHPP

モデル化のための確率過程として，以下の NHPP を仮定する．

2.1 連続時間モデル

$\{N_c(t), t\geq 0\}$ を，時刻 t までに発見された累積ソフトウェアエラー数を表す計数過程とする．この計数過程に対し，平均値関数 $M(t)$ をもち次の条件を満足する NHPP を仮定する．

ⅰ) $N_c(0)=0$. (1)

ⅱ) $\{N_c(t), t\geq 0\}$ は，独立増分をもつ．

ⅲ) 任意の時刻 t_i, t_j $(0<t_i<t_j)$ に対し，

† Software Reliability Growth Models with Error Debugging Rate by SHIGERU YAMADA (Okayama University of Science), TSUTOMU YAMANE and SHUNJI OSAKI (Department of Industrial and Systems Engineering, Faculty of Engineering, Hiroshima University).

†† 岡山理科大学

††† 広島大学工学部第二類（電気系）

* ソフトウェアエラーとは，ソフトウェアが実行中に期待どおりの動作をしないというソフトウェア故障を引き起こす，プログラミング論理の誤りまたは欠陥と定義する．

$$P_r\{N_c(t_j)-N_c(t_i)=x\}$$
$$=\frac{\{M(t_j)-M(t_i)\}^x}{x!}\exp[-\{M(t_j)-M(t_i)\}],$$
$$(x=0,1,2,\cdots). \quad (2)$$

ここで，平均値関数 $M(t)$ は，時刻 t までに発見される累積期待ソフトウェアエラー数を表す．

2.2 離散時間モデル

$\{N_d(n), n=0,1,2,\cdots\}$ を，n 回のテストランにより発見された累積ソフトウェアエラー数を表す離散型計数過程とする．この計数過程に対し，平均値関数 $D(n)$ をもち，次の条件を満足する NHPP を仮定する．

i) $N_d(0)=0.$ (3)

ii) $\{N_d(n), n=0,1,2,\cdots\}$ は，独立増分をもつ．

iii) 任意のテストラン試行回数 n_i, n_j $(0<n_i<n_j)$ に対し，

$$P_r\{N_d(n_j)-N_d(n_i)=x\}$$
$$=\frac{\{D(n_j)-D(n_i)\}^x}{x!}\exp[-\{D(n_j)-D(n_i)\}],$$
$$(x=0,1,2,\cdots). \quad (4)$$

ここで，平均値関数 $D(n)$ は，n 回のテストランにより発見される累積期待ソフトウェアエラー数を表す．

3. 連続時間モデル

2.1節で定義した計数過程 $\{N_c(t), t\geq 0\}$ に基づく連続時間モデルについて議論する．

3.1 モデルの記述

時刻 t において発見される累積期待ソフトウェアエラー数は，そのときのソフトウェア内の期待残存エラー数に比例するものと仮定する．すなわち，

$$\frac{dM(t)}{dt}=b(a-PM(t)),\ 0<b<1,\ 0<P<1, \quad (5)$$

とする．ここで，パラメータ a は，試験前のソフトウェア内に潜在する総期待ソフトウェアエラー数を表し，b は時刻 t におけるエラー1個当りの発見率を表す．Pは，発見されたソフトウェアエラーのうち，そのときの修正により完全に修正される割合，すなわち完全修正率を表す．ここで，$M(0)=0$ とする．式(5)より，平均値関数 $M(t)$ は，

$$M(t)=\frac{a}{P}(1-e^{-Pbt}),\ 0<b<1,\ 0<P<1, \quad (6)$$

となる．したがって，

$$\lim_{t\to\infty} M(t)=\frac{a}{P}, \quad (7)$$

を得る．これは，最終的に発見される累積期待ソフトウェアエラー数を表す．

3.2 信頼性評価尺度

式(6)の平均値関数 $M(t)$ をもつ連続時間モデルに対し，ソフトウェアの信頼性を評価するための定量的尺度を導出する．

3.2.1 残存ソフトウェアエラー数

時刻 t におけるソフトウェア内の残存エラー数を，

$$\bar{N}_c(t)=N_c(\infty)-N_c(t), \quad (8)$$

と定義する．このとき，$\bar{N}_c(t)$ の期待値および分散は，

$$r(t)\equiv E[\bar{N}_c(t)]=\mathrm{Var}[\bar{N}_c(t)]=\frac{a}{P}e^{-Pbt}, \quad (9)$$

となる．また，時刻 t までに発見された累積ソフトウェアエラー数が n_c 個であるとき，$\bar{N}_c(t)$ の条件付き確率は，

$$P_r\{\bar{N}_c(t)=x\,|\,N_c(t)=n_c\}=\frac{\{r(t)\}^x}{x!}\exp[-r(t)],$$
$$(x=0,1,2,\cdots), \quad (10)$$

となり，n_c に無関係な関数となる．

3.2.2 ソフトウェア信頼度

$(k-1)$ 番目と k 番目のソフトウェア故障発生時間間隔を表す確率変数を $\{X_k, k=1,2,\cdots\}$ とする．このとき，$S_k\equiv\sum_{i=1}^{k} X_i$ $(k=1,2,\cdots)$ は k 番目のソフトウェア故障発生時刻を表す．したがって，$S_{k-1}=s$ が与えられたときの X_k の条件付き信頼度関数は，

$$R_{X_k|S_{k-1}}(x\,|\,s)\equiv R(x\,|\,s)=\exp[-M(x)e^{-Pbs}], \quad (11)$$

となる．これは，時刻 s でソフトウェア故障が発生したときに，時間間隔 $(s, s+x]$ でソフトウェア故障が発生しない確率を表す．

3.3 パラメータの推定

モデルの未知パラメータ a および b の最尤法による推定方法について述べる．

k 番目のソフトウェア故障発生時刻 $S_k=s_k$ $(k=1,2,\cdots,n: 0<s_1<s_2<\cdots<s_n)$ が観測されたものとする．このとき，尤度関数は，$\{S_1, S_2, \cdots, S_n\}$ の同時確率密度関数により与えられ，

$$f_{S_1,S_2,\cdots,S_n}(s_1,s_2,\cdots,s_n)$$
$$=\exp[-M(s_n)]\prod_{k=1}^{n} abe^{-Pbs_k}, \quad (12)$$

と表すことができる．したがって，対数尤度関数は，

$$L=n\ln a+n\ln b-Pb\sum_{k=1}^{n} s_k-\frac{a}{P}(1-e^{-Pbs_n}), \quad (13)$$

となる．未知パラメータ a および b の最尤推定値 $\hat{a}$ および $\hat{b}$ は，式(13)において，$\partial L/\partial a=\partial L/\partial b=0$ として得られる尤度方程式

$$\frac{n}{a}=\frac{1}{P}(1-e^{-Pbs_n}), \tag{14}$$

$$\frac{n}{b}=as_ne^{-Pbs_n}+P\sum_{k=1}^{n}s_k, \tag{15}$$

を数値的に解くことにより得られる.

4. 離散時間モデル

2.2 節で定義した離散型計数過程 $\{N_d(n), n=0, 1, 2, \cdots\}$ に基づく離散時間モデルについて議論する.

4.1 モデルの記述

1回のテストランにおいて発見される累積期待ソフトウェアエラー数は, そのときのソフトウェア内の期待残存エラー数に比例するものと仮定する. すなわち,

$$D(n+1)-D(n)=b(a-PD(n)),$$
$$0<b<1,\ 0<P<1, \tag{16}$$

とする. 式(16)より, 平均値関数 $D(n)$ は,

$$D(n)=\frac{a}{P}(1-(1-Pb)^n),\ 0<b<1,\ 0<P<1, \tag{17}$$

となる. ここで, 1回のテストランによるエラー1個当りの発見率を表す b 以外のパラメータ a および P は, 連続時間モデルの場合と同じ意味をもち,

$$\lim_{n\to\infty}D(n)=\frac{a}{P}, \tag{18}$$

となり, 連続時間モデルの結果に一致する.

4.2 信頼性評価尺度

式(17)の平均値関数 $D(n)$ をもつ離散時間モデルに対し, ソフトウェアの信頼性を評価するための定量的尺度を導出する.

4.2.1 残存ソフトウェアエラー数

n 回目のテストラン終了後のソフトウェア内の残存エラー数を,

$$\overline{N}_d(n)=N_d(\infty)-N_d(n), \tag{19}$$

と定義する. このとき, $\overline{N}_d(n)$ の期待値および分散は,

$$r(n)\equiv E[\overline{N}_d(n)]=\mathrm{Var}[\overline{N}_d(n)]=\frac{a}{P}(1-Pb)^n, \tag{20}$$

となる. また, n 回のテストランにより発見された累積ソフトウェアエラー数が n_d 個であるとき, $\overline{N}_d(n)$ の条件付き確率は,

$$P_r\{\overline{N}_d(n)=x\mid N_d(n)=n_d\}=\frac{\{r(n)\}^x}{x!}\exp[-r(n)],$$
$$(x=0, 1, 2, \cdots), \tag{21}$$

となり, n_d に無関係な関数となる.

4.2.2 ソフトウェア信頼度

n 回のテストランにより発見された累積ソフトウェアエラー数が n_0 個であるとき, n 回目と $(n+h)$ 回目のテストラン試行間隔において, ソフトウェアエラーが発見されない確率は,

$$R(n, h)=\exp[-D(h)(1-Pb)^n], \tag{22}$$

となり, n_0 に無関係な関数となる.

4.3 パラメータの推定

モデルの未知パラメータ a および b の最尤法による推定方法について述べる.

n_i 回のテストランにより発見された累積ソフトウェアエラー数が y_i 個であるようなデータ (n_i, y_i) $(i=1, 2, \cdots, k)$ が観測されたものとする. このとき, 尤度関数は, $\{N_d(n_1)=y_1, N_d(n_2)=y_2, \cdots, N_d(n_k)=y_k\}$ の同時確率関数により与えられ,

$$P_r\{N_d(n_1)=y_1, N_d(n_2)=y_2, \cdots, N_d(n_k)=y_k\}$$
$$=\exp[-D(n_k)]\prod_{i=1}^{k}\frac{\{D(n_i)-D(n_{i-1})\}^{y_i-y_{i-1}}}{(y_i-y_{i-1})!}, \tag{23}$$

と表すことができる. ただし, $n_0=0$ および $y_0=0$ とする. したがって, 対数尤度関数は,

$$L=y_k\ln a+\sum_{i=1}^{k}(y_i-y_{i-1})\cdot\ln[(1-Pb)^{n_{i-1}}-(1-Pb)^{n_i}]-y_k\ln P-\frac{a}{P}(1-(1-Pb)^{n_k})-\sum_{i=1}^{k}\ln[(y_i-y_{i-1})!], \tag{24}$$

となる. 未知パラメータ a および b の最尤推定値 $\hat{a}$ および $\hat{b}$ は, 式(24)において, $\partial L/\partial a=\partial L/\partial b=0$ として得られる尤度方程式

$$\frac{y_k}{a}=\frac{1}{P}(1-(1-Pb)^{n_k}), \tag{25}$$

$$an_k(1-Pb)^{n_k-1}=P\sum_{i=1}^{k}\frac{(y_i-y_{i-1})\{n_i(1-Pb)^{n_i-1}-n_{i-1}(1-Pb)^{n_{i-1}-1}\}}{[(1-Pb)^{n_{i-1}}-(1-Pb)^{n_i}]}, \tag{26}$$

を数値的に解くことにより得られる.

5. 実際データへの適用

実際に観測されたデータに, 連続時間モデルおよび離散時間モデルを適用し, データ解析を行う.

5.1 連続時間モデル

Goel and Okumoto[2]の引用した NTDS (Naval

表 1 NTDS のソフトウェアエラーデータ
(Goel and Okumoto[21]より引用)
Table 1 Software error data from NTDS
(From Goel and Okumoto[21]).

Error No. k	Time between errors x_k(days)	Cumulative time s_k(days)
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250

Tactical Data System) トラブル・レポート・データにモデルを適用しデータ解析を行う．このデータは，リアルタイム・マルチコンピュータ・システムのためのソフトウェア開発において観測されたものである．**表 1** に示す 26 個のエラーデータ $s_k'(k=1,2,\cdots,26)$ が観測されている．

表 2 連続時間モデルのモデルパラメータの最尤推定値
Table 2 Maximum likelihood estimates of model parameters for the continuous time model.

	Estimates of model parameters	
P	$\hat{a}$	$\hat{b}$
0.10	3.3993510	0.0579016
0.20	6.7987020	0.0289508
0.30	10.1980600	0.0193005
0.40	13.5974000	0.0144754
0.50	16.9967900	0.0115803
0.60	20.3961200	0.0096503
0.70	23.7954600	0.0082717
0.80	27.1948200	0.0072377
0.90	30.5941700	0.0064335
1.00	33.9935100	0.0057902

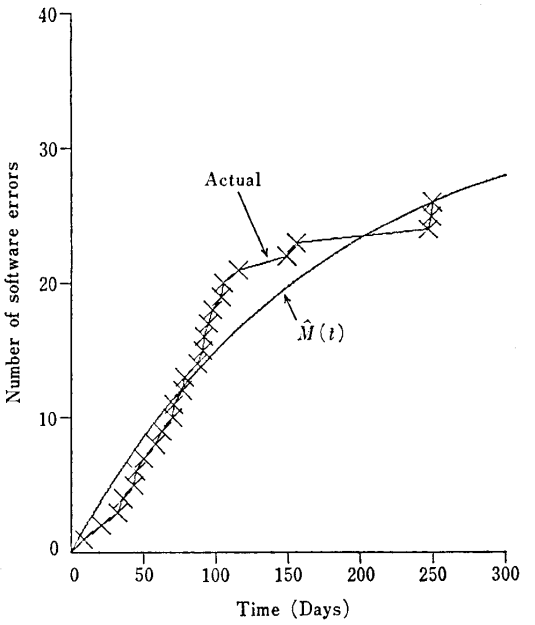

図 1 NTDS データに対する推定された平均値関数 $\hat{M}(t)$
Fig. 1 Estimated mean value function $\hat{M}(t)$ for the NTDS data.

26 個のエラーデータにより，式(14)および式(15)の尤度方程式を完全修正率 $P=0.1(0.1)0.9$ のそれぞれについて数値的に解いて得られた未知パラメータ a および b の最尤推定値 $\hat{a}$ および $\hat{b}$ を**表 2** に示す．この最尤推定値を用いると，推定された平均値関数は，

$$\hat{M}(t)=33.99(1-e^{-0.005790t}), \tag{27}$$

となり，観測データとともに**図 1** に示す．$\hat{M}(t)$ の観測データへの適合性を判定するために，コルモゴロフ・スミルノフ適合度検定 (Kolmogorov-Smirnov goodness-of-fit test (Yamada and Osaki[10]参照)) を行うと，5％有意水準で十分な適合性のあることを確認できた．

各最尤推定値 $\hat{a}$ および $\hat{b}$ に対し，修正が完全に行われていたものと仮定すると，そのときの推定された平均値関数は，Goel and Okumoto[2]より

$$\hat{M}_p(t)=\hat{a}(1-e^{-\hat{b}t}), \tag{28}$$

となる．試験段階において，発見されるエラーには，初めて発見されたエラーと，不完全な修正により再度発見されたエラーがある．後者の不完全修正エラー数は，$\hat{M}(t)$ と $\hat{M}_p(t)$ との差，すなわち，

$$\Delta\hat{M}(t)=\hat{M}(t)-\hat{M}_p(t), \tag{29}$$

により表すことができる．$\Delta\hat{M}(t)$ は，時刻 t までに発見されたソフトウェアエラーのうち，不完全な修正により再度発見された累積不完全修正エラー数を表して

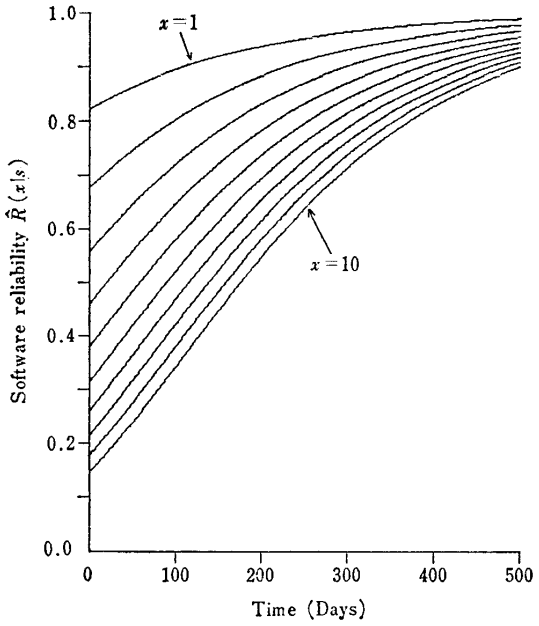

図 2 NTDS データに対する推定されたソフトウェア信頼度 $\hat{R}(x|s)(x=1(1)10)$

Fig. 2 Estimated software reliability $\hat{R}(x|s)$ for the NTDS data.

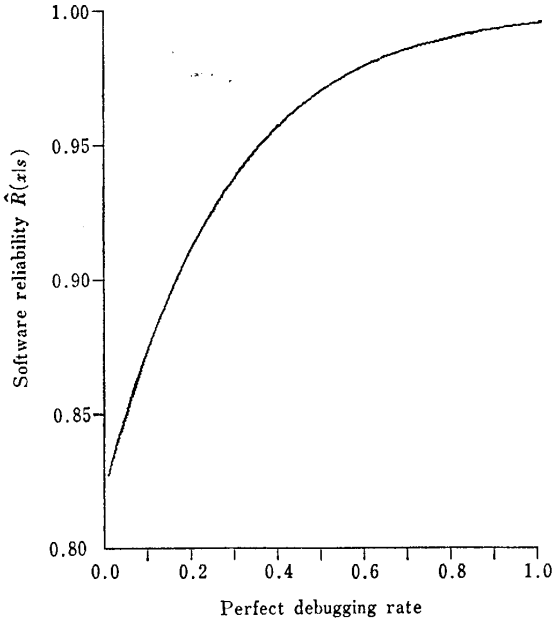

図 3 完全修正率Pに対する推定されたソフトウェア信頼度 $\hat{R}(1|450)$

Fig. 3 Dependence of perfect debugging rate P in the estimated software reliability $\hat{R}(1|450)$.

いる．また，最終的な累積不完全修正エラー数の最尤推定値は，式(29)より，

$$\lim_{t\to\infty} \Delta\hat{M}(t)=\frac{\hat{a}}{P}(1-P)=33.99q, \tag{30}$$

となる．ここで，$q=1-P$とする．したがって，最終的に発見される総期待ソフトウェアエラー数の$(100q)$%が不完全な修正によるものであることがわかる．例えば，$P=0.8$のとき，$\Delta\hat{M}(\infty)=6.8$となり，最終的に発見される34個の総期待ソフトウェアエラーのうち，7個が不完全な修正によるものである．

式(11)から，推定されたソフトウェア信頼度は，

$$\hat{R}(x|s)=\exp[-33.99(1-e^{-0.005790x})e^{-0.005790s}], \tag{31}$$

となり，$x=1(1)10$に対する推定値$\hat{R}(x|s)$を**図2**に示す．最終試験時刻sを固定して，運用時間xを増加させると，推定値$\hat{R}(x|s)$は減少することがわかる．

ソフトウェア試験担当者にとって，目標信頼度を達成するために要する時間を推定することは興味ある問題である．目標信頼度Rを達成するために要する時間は，式(11)から，

$$s=\frac{1}{Pb}\left[\ln M(x)-\ln\left(\ln\frac{1}{R}\right)\right], \tag{32}$$

となる．例えば，このデータにおいて，目標信頼度$R=0.98$，および運用時間$x=1$とすると，

$$\hat{s}=\frac{1}{0.005790}\left[\ln\{33.99(1-e^{-0.005790(1)})\}-\ln\left(\ln\frac{1}{0.98}\right)\right]$$
$$=393, \tag{33}$$

となり，目標信頼度0.98を達成するために393日を要す．現在250日間の試験が行われているので，目標信頼度を達成するためには，さらに393−250=143日間の試験が必要である．

最終試験時刻$s=450$および運用時間$x=1$としたときの，完全修正率Pに対するソフトウェア信頼度を**図3**に示す．このとき，$a=23.80$および$b=0.008272$とする．図3より，完全修正率が低いとき，完全修正率の改善がソフトウェア信頼度に与える効果はより大きいことがわかる．

5.2 離散時間モデル

ここで取り扱うソフトウェアエラーデータは，約50,000 LOC (lines of code) から成る PL/1 およびアセンブリ言語で書かれたアプリケーションプログラムの試験段階において観測されたものである．n_i回のテストランにより発見された累積ソフトウェアエラー数 y_i $(i=1,2,\cdots,18)$ を示す18組のエラーデータが観測されている．

18組のエラーデータにより，式(25)および式(26)の尤度方程式を $P=0.1(0.1)0.9$ のそれぞれについて数

表 3 離散時間モデルのモデルパラメータの最尤推定値
Table 3 Maximum likelihood estimates of model parameters for the discrete time model.

	Estimates of model parameters	
P	$\hat{a}$	$\hat{b}$
0.10	8.1950380	0.0286054
0.20	16.3901700	0.0143028
0.30	24.5852600	0.0095352
0.40	32.7803400	0.0071515
0.50	40.9754200	0.0057212
0.60	49.1705100	0.0047675
0.70	57.3656000	0.0040865
0.80	65.5606800	0.0035757
0.90	73.7557700	0.0031783
1.00	81.9508500	0.0028606

値的に解いて得られた未知パラメータ a および b の最尤推定値 $\hat{a}$ および $\hat{b}$ を**表 3** に示す．この最尤推定値を用いると，推定された平均値関数は，

$$\hat{D}(n)=81.95(1-(1-0.002861)^n), \tag{34}$$

となり，観測データとともに**図 4** に示す．$D(n)$ の観測データへの適合性を判定するために，カイ二乗適合度検定 (chi-square goodness-of-fit test, (Yamada and Osaki[10]参照)) を行うと，5％有意水準で十分な適合性があることを確認できた．

各最尤推定値 $\hat{a}$ および $\hat{b}$ に対し，修正が完全に行われていたものと仮定すると，そのときの推定された平均値関数は，Yamada et al.[5]より，

$$\hat{D}_p(n)=\hat{a}(1-(1-\hat{b})^n), \tag{35}$$

となる．n 回のテストランにより発見されたソフトウェアエラーのうち，不完全な修正により再度発見された累積不完全修正エラー数は，式(34)および式(35)より，

$$\Delta\hat{D}(n)=\hat{D}(n)-\hat{D}_p(n), \tag{36}$$

と表すことができる．したがって，最終的な累積不完全修正エラー数は，

$$\lim_{n\to\infty}\Delta\hat{D}(n)=\frac{\hat{a}}{P}(1-P)=81.95q, \tag{37}$$

となる．ここで，$q=1-P$ とする．式(37)から，最終的に発見される総期待ソフトウェアエラー数の $(100q)$％が不完全な修正によるものであることがわかる．例えば，$P=0.8$ のとき $\Delta\hat{D}(\infty)=16.4$ となり，最終的に発見される 82 個の総期待ソフトウェアエラーのうち，17 個が不完全な修正によるものである．

式(22)より，推定されたソフトウェア信頼度は，

$$\hat{R}(n,h)=\exp[-81.95(1-(1-0.002861)^h)\times(1-0.002861)^n], \tag{38}$$

となる．**図 5** に，$h=1(1)10$ に対する $\hat{R}(n,h)$ を示す．最終テストラン試行回数 n を固定して，試行回数 h を増加させると，$\hat{R}(n,h)$ は減少することがわかる．

目標信頼度 R を達成するために要するテストラン試行回数は，式(22)から，

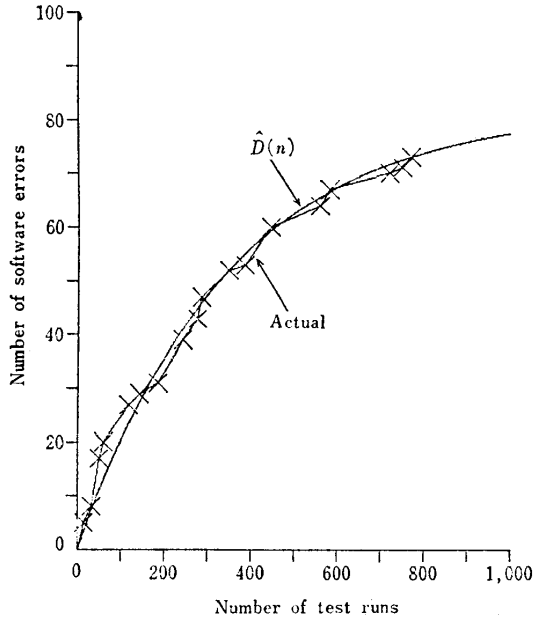

図 4 観測データに対する推定された平均値関数 $\hat{D}(n)$
Fig. 4 Estimated mean value function $\hat{D}(n)$ for the observed data.

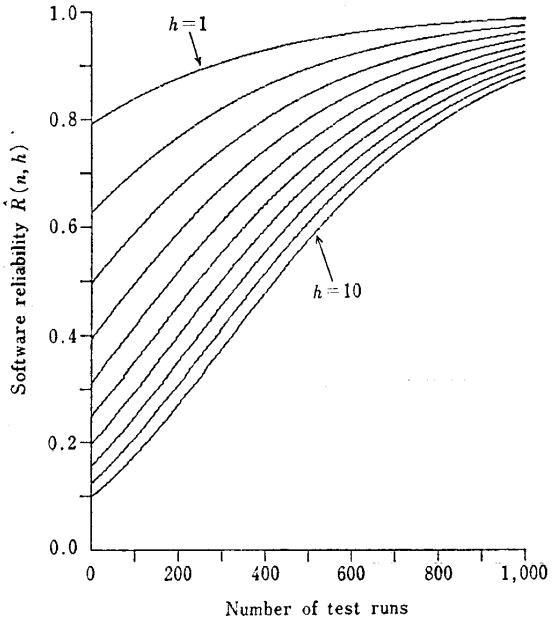

図 5 観測データに対する推定されたソフトウェア信頼度 $\hat{R}(n,h)(h=1(1)10)$
Fig. 5 Estimated software reliability $\hat{R}(n,h)$ for the observed data $(h=1(1)10)$.

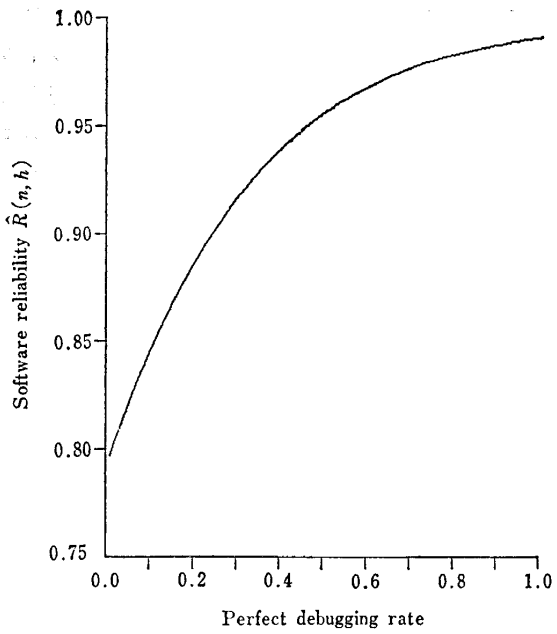

図 6 完全修正率Pに対する推定されたソフトウェア信頼度 $\hat{R}(800,1)$

Fig. 6 Dependence of perfect debugging rate P in the estimated software reliability $\hat{R}(800,1)$.

$$n=\frac{1}{1-Pb}\left[\ln\left(\ln\frac{1}{R}\right)-\ln D(h)\right], \tag{39}$$

となる．例えば，このエラーデータにおいて目標信頼度 $R=0.98$ および試行回数 $h=1$ とすると，

$$\hat{n}=\frac{1}{1-0.002861}\left[\ln\left(\ln\frac{1}{0.98}\right)-\ln\{81.95(1-(1-0.002861)^{h})\}\right]$$
$$=856, \tag{40}$$

となり，目標信頼度0.98を達成するために856回のテストランを要す．現在773回のテストラン試行が行われているので，目標信頼度を達成するためには，さらに856−773=83回のテストラン試行が必要である．

最終テストラン試行回数 $n=800$ および運用回数 $h=1$ としたときの，完全修正率Pに対するソフトウェア信頼度を**図6**に示す．このとき，$a=57.37$ および $b=0.004087$ とする．図6より，完全修正率が低いとき完全修正率の改善がソフトウェア信頼度に与える効果は，より大きいことがわかる．

6. むすび

本論文では，試験において発見されたソフトウェアエラーに対し，その修正率を考慮したソフトウェア信頼度成長モデルを提案した．このモデルのもとで，ソフトウェアの信頼性評価に有用な定量的尺度を導出できた．エラー修正率の導入により，試験前のソフトウェアに潜在する総期待エラー数の推定ができた．この結果，試験の進行に伴う累積不完全修正エラー数の推定が可能となった．また，ソフトウェア信頼度により，目標信頼度を達成するために必要な試験時間あるいはテストラン試行回数を推定することができた．

本論文では，エラー修正率はソフトウェア試験担当者の経験，およびソフトウェア規模などから推定できる既知の値として取り扱っている．実際には，このエラー修正率の推定は重要であるが，困難な問題であるかもしれない．したがって，この推定方法を確立することが今後の課題である．さらに，提案したモデルの妥当性を検証するために，より多くのソフトウェアエラーデータに適用し，データ解析を行う必要がある．

謝辞 貴重なソフトウェアエラーデータを提供していただいた日本IBM(株)サイエンス・インスティチュートのソフトウェア・エンジニアリング担当 大場充氏に深く感謝の意を表します．

参考文献

1) Ramamoorthy, C. V. and Bastani, F. B.: Software Reliability—Status and Perspectives, *IEEE Trans. Softw. Eng.*, Vol. SE-8, No. 4, pp. 354-371 (1982).
2) Goel, A. L. and Okumoto, K.: Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, *IEEE Trans. Reliab.*, Vol. R-28, No. 3, pp. 206-211 (1979).
3) Musa, J. D.: A Theory of Software Reliability and Its Application, *IEEE Trans. Softw. Eng.*, Vol. SE-1, No. 3, pp. 312-327 (1975).
4) Littlewood, B.: Theories of Software Reliability: How Good Are They and How Can They Be Improved?, *IEEE Trans. Softw. Eng.*, Vol. SE-6, No. 5, pp. 489-500 (1980).
5) Yamada, S., Osaki, S. and Narihisa, H.: Software Reliability Growth Modeling with Number of Test Runs, *Trans. IECE Japan*, Vol. E-67, No. 2, pp. 79-83 (1984).
6) 山田 茂，尾崎俊治：ソフトウェアエラー発見過程に関する信頼性モデル，情報処理学会論文誌，Vol. 24, No. 3, pp. 376-378 (1983).
7) Yamada, S., Ohba, M. and Osaki, S.: S-Shaped Reliability Growth Modeling for Software Error Detection, *IEEE Trans. Reliab.*, Vol. R-32, No. 5, pp. 475-478, 484 (1983).
8) Ascher, H. and Feingold, H.: *Repairable Systems Reliability: Modeling, Inference, Misconceptions and Their Causes*, Marcel

Dekker, Inc., New York (1984).
9) Ross, S. M. : *Stochastic Processes*, John Wiley & Sons, Inc., New York (1983).
10) Yamada, S. and Osaki, S. : Reliability Growth Models for Hardware and Software Systems Based on Nonhomogeneous Poisson Processes : A Survey, *Microelectron. Reliab.*, Vol. 23, No. 1, pp. 91-112 (1983).

(昭和 60 年 4 月 4 日受付)
(昭和 60 年 6 月 20 日採録)