離散型ソフトウェア信頼度成長モデルと最適リリース問題への応用†

山田 茂††

ソフトウェアの信頼性や品質を定量的に評価し，その結果をソフトウェアの開発管理に反映することは重要な問題である．このために本論文では，ソフトウェア開発の最終段階であるテスト工程におけるソフトウェアエラー発見事象を，ソフトウェア信頼度成長モデルにより，発見された総ソフトウェアエラー数と実行された総テストラン回数との関係において取り扱う．まず，実際のテスト工程でよく経験するような，テストランに伴うエラー発見率がテストの進行とともに減少していく現象を，非同次ポアソン過程に基づいて一般的にモデル化できることを示す．特別な場合として，エラー発見率が幾何級数的に減少していくとする幾何減少型エラー発見率モデルを議論することができる．このモデルを用いて，ソフトウェアの開発管理上の問題の1つである，テストランを打ち切って運用段階へ移行するのに必要な総テストラン回数を決定する方法，すなわちソフトウェアの最適リリース政策について考察する．ここで，総期待ソフトウェアコストおよびソフトウェア信頼度という2つの評価基準が導入される．最後に，テストデータの解析例および最適リリース政策の数値例を与える．

1. まえがき

ソフトウェア開発のテスト工程は，作り込まれたソフトウェアエラー（以下エラーと略す）の除去作業により，開発結果としての品質・信頼性を最終的に確認する重要な工程となっている．この工程では詳細な計画の下に，テストケースの作成，テストラン，エラーの発見と修正，テスト結果の解析が実施される．ここで，1組のテストケースを実行することを1回のテストランと定義する．従来から，テスト結果をもとにソフトウェアの信頼性評価を行う信頼性モデルとして，ソフトウェア信頼度成長モデル (software reliability growth model)[1] が提案されてきた．これはテストに費やされたカレンダー時間や CPU 時間などの連続的時間と，発見された総エラー数との関係によりエラー発見過程を記述し，それをソフトウェアの信頼度成長過程として把握するものである．したがって，テストにおけるエラー発見期間の単位は連続的である．

一方，ソフトウェア信頼度成長モデルに基づいてテストデータを解析した後の興味ある問題の1つに，ソフトウェアの最適リリース (release) 問題がある．これは，所定の評価基準により，テスト工程を終了して実際の運用段階へ移行するのに最適な時期を決定するものである．

本論文では，エラー発見期間の単位をテストラン回数などの離散的時間とする離散型ソフトウェア信頼度成長モデル[2] について議論する．まず，非同次ポアソン過程 (nonhomogeneous Poisson process, 以下 NHPP と略す)[3] に基づいてモデルを一般的に記述し，ソフトウェアの信頼性評価尺度およびモデル・パラメータの推定法を与える．つぎに特別な場合として，実際のテスト工程においてよく経験されるエラー発見事象を表すために，信頼度成長過程を特徴づけるテストラン1回当りのエラー発見率が幾何級数的に変化していくものと仮定する，幾何減少型エラー発見率モデル (geometrically-decreasing error detection rate model)[4] を考える．さらに，このモデルの実際問題への応用として，コスト評価基準によりソフトウェアの最適リリース問題を考察する．これは，総期待ソフトウェアコストを最小にするような，ソフトウェアのリリースに必要な総テストラン回数を求める問題として定式化され，その最適政策を与える．また，ソフトウェアの最適リリース問題を考えるにあたっては，コスト要因だけでなく開発されたソフトウェアの信頼性の達成度合をも考慮するべきである．そこで，信頼性評価尺度の1つであるソフトウェア信頼度もその評価基準として，コストおよび信頼度の両評価基準を同時に考慮した最適リリース政策を議論する．最後に，実際のテスト工程において観測されたソフトウェアエラーデータを使って，データ解析例および最適リリース政策の数値例も与える．

† Discrete Software Reliability Growth Model and Its Application to Optimal Release Problems by SHIGERU YAMADA (Department of Electronic Engineering, Faculty of Engineering, Okayama University of Science).

†† 岡山理科大学工学部電子工学科

2. モデルの一般的記述

ソフトウェアのテストは，あらかじめ用意されたテスト用入力データ（テストケースと呼ばれる）を実行して，ソフトウェア内に潜在するエラーを発見・修正することを目的としている．このとき，テスト工程における総テストラン回数の増加に伴う発見された総エラー数の傾向・変化により，信頼度成長の様相を把握するために次の仮定をおく．

（1） 1回のテストランにより発見されるエラー数は，その時点でのソフトウェア内の残存エラー数に比例する．

（2） テストラン1回当りのエラー発見率は，総テストラン回数に関して非増加である．

（3） 発見されたエラーの修正時には，新しいエラーは作り込まれない．

（4） 任意のテストラン区間内で発見されるエラーは互いに独立である．

以上の仮定の下で，n 回のテストランにより発見された総期待エラー数を $M(n)$ とする $(n=0,1,2,\cdots)$．仮定(1)および(2)から，

$$M(n+1)-M(n)=b_{n+1}[a-M(n)] \quad (n=0,1,2,\cdots), \tag{1}$$

$$a>0, 1\geq b_1\geq b_2\geq\cdots\geq b_n\geq b_{n+1}\geq\cdots>0, \tag{2}$$

を得る．ここで，

a＝テスト開始前にソフトウェア内に潜在する総期待エラー数，

b_n＝n 回目のテストランにおける（エラー1個当りの）エラー発見率，

とする．式(1)の差分方程式を，初期条件 $M(0)=0$ の下で $M(n)$ について解くと

$$M(n)=a[1-\prod_{i=1}^{n}(1-b_i)], \tag{3}$$

を得る．テスト工程におけるエラー発見事象を確率モデルとして取り扱うために，n 回のテストランにより発見された総エラー数を表す離散型計数過程 $\{N(n), n\geq 0\}$ $(n=0,1,2,\cdots)$ を導入する（Yamada and Osaki[2] 参照）．そこで，式(3)の $M(n)$ を平均値関数として NHPP により，このエラー発見事象をモデル化すると

$$Pr\{N(n)=x\}=\frac{\{M(n)\}^x}{x!}\exp[-M(n)] \quad (n,x=0,1,2,\cdots), \tag{4}$$

となる．ここで，$Pr\{\cdot\}$ は確率を表す．

式(3)および式(4)で定義された NHPP に基づく離散型ソフトウェア信頼度成長モデルから，ソフトウェアの信頼性評価に有用な定量的尺度が導出できる（Goel and Okumoto[5] および Yamada and Osaki[6] 参照）．n 回目のテストランが終了した後にソフトウェア内に潜在する総期待エラー数は

$$r(n)\equiv a-M(n)=a\prod_{i=1}^{n}(1-b_i), \tag{5}$$

により与えられる．また，$N(n)=x_0$ すなわち n 回のテストランにより x_0 個のエラーが発見されたという条件の下で，n 回目と $(n+h)$ 回目のテストラン間隔においてエラーが発見されない確率を考える．この条件付き確率は

$$R(n,h)\equiv Pr\{N(n+h)-N(n)=0\mid N(n)=x_0\}=\exp[-\{M(n+h)-M(n)\}] \quad (n,h=0,1,2,\cdots), \tag{6}$$

により与えられ，離散型ソフトウェア信頼度成長モデルに対するソフトウェア信頼度 (software reliability) と呼ばれる[2]．ここで，ソフトウェア信頼度 $R(n,h)$ は x_0 に無関係であることに注意する．

さらに，式(3)および式(4)で定義された離散型ソフトウェア信頼度成長モデルに含まれるモデル・パラメータ a および b_i $(i=1,2,\cdots)$ の推定法について議論する（Yamada and Osaki[6] 参照）．テスト工程において，n_j 回のテストランにより発見された累積エラー数が x_j 個であるというテスト・データ (n_j,x_j) が k 組観測されたものとする $(j=1,2,\cdots,k;\ 0<n_1<n_2<\cdots<n_k)$．このとき，$\{N(n_1)=x_1, N(n_2)=x_2,\cdots, N(n_k)=x_k\}$ の同時確率関数，すなわち尤度関数は NHPP の性質から

$$L\equiv Pr\{N(n_1)=x_1, N(n_2)=x_2,\cdots, N(n_k)=x_k\}=\prod_{j=1}^{k}\frac{\{M(n_j)-M(n_{j-1})\}^{(x_j-x_{j-1})}}{(x_j-x_{j-1})!}\cdot\exp[-\{M(n_j)-M(n_{j-1})\}], \tag{7}$$

により与えられる．ここで，$x_0=0$ および $n_0=0$ とする．式(7)の自然対数をとり対数尤度関数 $\ln L$ を求めれば，最尤法によりモデル・パラメータの最尤推定値 $\hat{a}$ および $\hat{b}_i$ $(i=1,2,\cdots)$ は

$$\partial\ln L/\partial a=\partial\ln L/\partial b_i=0 \quad (i=1,2,\cdots), \tag{8}$$

と置いて，これらの同時尤度方程式を数値的に解けば得られる．

3. 幾何減少型エラー発見率モデル[4]

一般にソフトウェアのテスト工程では，テスト初期に発見されるエラー数に比較して，後に発見されるエラー数はテストの進行とともに小さくなっていくものと考えられる．このような現象を離散型ソフトウェア信頼度成長モデルに反映させるために，テストラン1回当りのエラー発見率はソフトウェア内に残存するエラー数に比例して幾何級数的に減少すると仮定する（Moranda[7] および Musa and Okumoto[8] 参照）．したがって，エラー発見率 b_n は

$$b_n \equiv d(n) = D \cdot r^{n-1} \quad (n=1,2,\cdots), \tag{9}$$

$$0<D<1,\ 0<r<1, \tag{10}$$

と具体的に書ける．ここで，

D＝テストラン1回当りの初期エラー発見率，

r＝テストラン1回当りのエラー発見率減少係数，

である．このとき，式(3)の平均値関数 $M(n)$ は

$$M(n) \equiv G(n) = a\left[1-\prod_{i=1}^{n}(1-Dr^{i-1})\right], \tag{11}$$

と書くことができる．同様にして，式(5)の期待残存エラー数 $r(n)$ および式(6)のソフトウェア信頼度 $R(n,h)$ は，それぞれ

$$r(n) \equiv r_G(n) = a\prod_{i=1}^{n}(1-Dr^{i-1}), \tag{12}$$

$$R(n,h) \equiv R_G(n,h) = \exp\left[-r(n)\left\{1-\prod_{i=n+1}^{n+h}(1-Dr^{i-1})\right\}\right], \tag{13}$$

となる．さらに，式(11)の平均値関数 $G(n)$ をもつ NHPP に基づく幾何減少型エラー発見率モデルのモデル・パラメータ a, D，および r の最尤推定値 $\hat{a}$，$\hat{D}$，および $\hat{r}$ は，式(7)において $M(n) \equiv G(n)$ と置くことにより，

$$\partial \ln L/\partial a = \partial \ln L/\partial D = \partial \ln L/\partial r = 0, \tag{14}$$

から導出される同時尤度方程式を数値的に解けば求められる．

4. ソフトウェアの最適リリース政策（I）

ソフトウェアの運用段階における品質や信頼性は，実施されるテスト方法およびテストツールなどとともに実行される総テストラン回数に依存していると考えられる．すなわち，総テストラン回数を増加させればさせるほどエラーは数多く発見され，運用段階における信頼性は高くなる．また，総テストラン回数を際限なく増加させればソフトウェアの運用が遅れ，テストに要するコストは余分にかかる．一方テスト期間を短くし，したがって総テストラン回数を減少させればさせるほど，テスト工程において未発見のエラーが増加し，運用段階で発見されるエラーに対する保守コストがかさむことになる．一般に，テスト工程で発見されるエラー1個当りの修正に要するコストは，運用段階で発見し修正するよりも安い．上述のことは，適当な評価基準を設定して，ソフトウェアをテスト工程から運用段階へ移行させるのに最適な総テストラン回数を求める問題，すなわち最適リリース問題（Koch and Kubat[9]，Okumoto and Goel[10]，および山田[11]参照）が存在していることを意味する．

まず，式(11)の平均値関数 $G(n)$ をもつ幾何減少型エラー発見率モデルに対して，コスト評価基準によりソフトウェアの最適リリース問題を議論する．そこで，次のソフトウェアコストを導入する．

c_1＝テストラン1回当りのコスト，

c_2＝テスト工程において未発見となったエラー1個当りの保守コスト．

ソフトウェアをリリースするために必要な総テストラン回数を N とする．このとき，テスト工程で必要な期待コストは c_1N であり，運用段階で必要な期待コストは $c_2 r_G(N)$ である．ここで $r_G(N)$ は，式(12)で表される N 回のテストランが終了したときの期待残存エラー数である．したがって，総期待ソフトウェアコストは

$$C(N) \equiv c_1 N + c_2 r_G(N) = c_1 N + c_2 a\prod_{i=1}^{N}(1-Dr^{i-1}), \tag{15}$$

となる．式(15)を最小にする最適解 $N=N^*$ が最適総テストラン回数である．式(15)から，

$$C(N+1)-C(N) = c_2\{c_1/c_2 - W(N)\}, \tag{16}$$

を得る．ここで，

$$W(N) \equiv d(N+1)\cdot r_G(N) = Dr^N \cdot a\prod_{i=1}^{N}(1-Dr^{i-1}), \tag{17}$$

とする．式(17)の $W(N)$ は，テストラン1回当りに発見される総期待エラー数を意味する．また，$N>0$ なる整数値 N について

$$W(N+1)<W(N),\ W(0)=Dr(0),\ W(\infty)=0, \tag{18}$$

となり，$W(N)$ は N に関する単調減少関数である．以上のことから，最適総テストラン回数 N^* に関する次の定理を得る．

[定理1] $c_1>0, c_2>0$, および $r_G(0)=a$ とする.

(i) もし $W(0)\leq c_1/c_2$ ならば, $N^*=0$ である.

(ii) もし $W(0)>c_1/c_2$ ならば,

$$W(N-1)>c_1/c_2,\quad W(N)\leq c_1/c_2, \tag{19}$$

を満たす有限かつ唯一の解 N_0 が存在し, この解 N_0 は $C(N)$ の最適解 N^* となる.

(証明) もし $W(0)\leq c_1/c_2$ ならば, $W(N)$ は単調減少関数であるので式(16)から, $N>0$ なる N に対して常に $C(N+1)>C(N)$ となり, $N^*=0$ である. もし $W(0)>c_1/c_2$ ならば, $W(N)$ は単調減少関数であることから式(19)を満たす有限かつ唯一の N_0 が存在する. このとき,

$$\left.\begin{array}{l} C(N+1)<C(N)\ (0<N<N_0) \\ C(N+1)>C(N)\ (N>N_0), \end{array}\right\} \tag{20}$$

となる. したがって, N_0 は $C(N)$ を最小にする最適解 N^* である. Q. E. D.

5. ソフトウェアの最適リリース政策 (II)

前章では, 式(15)の総期待ソフトウェアコスト $C(N)$ を評価基準として, 幾何減少型エラー発見率モデルに対する最適リリース政策を議論した. 実際問題としては, ソフトウェアの最適リリース問題をコスト要因のみを考慮して論じるのでなく, テストにより達成された信頼度をも同時に考慮することが重要である. そのために, 信頼度評価基準として式(13)のソフトウェア信頼度を考え, コストおよび信頼度の両者を評価基準として (山田[11], および Yamada and Osaki[12] 参照), 幾何減少型エラー発見率モデルに対する最適リリース政策を議論する. このとき, 2つの評価基準を同時に考慮したリリースに必要とされる最適総テストラン回数 $N=N^*$ は, テストにより向上していくソフトウェア信頼度 $R_G(N,h)$ が目標信頼度 R_0 を満足しながら, 総期待ソフトウェアコスト $C(N)$ を最小にするように決定されるものとする. したがって, このような最適リリース政策は, 指定された追加テストラン回数 h ($h\geq 0$) に対して

$$\left.\begin{array}{l} \text{minimize } C(N) \\ \text{subject to } R_G(N,h)\geq R_0,\ N\geq 0, \end{array}\right\} \tag{21}$$

を満足する総テストラン回数 $N=N^*$ を求める問題として定式化される.

式(13)において $n=N$ と置き換えたソフトウェア信頼度 $R_G(N,h)$ は, $N>0$ なる整数値 N について

$$R_G(N+1,h)>R_G(N,h),\ R_G(0,h)=\exp[-G(h)],\quad R_G(\infty,h)=1, \tag{22}$$

となり, $R_G(N,h)$ は N に関する単調増加関数である. したがって, もし $R_G(0,h)<R_0$ ならば

$$R_G(N-1,h)<R_0,\ R_G(N,h)\geq R_0, \tag{23}$$

を満たす有限かつ唯一の解 N_1 が存在する. もし $R_G(0,h)\geq R_0$ ならば, $N>0$ なる N について $R_G(N,h)\geq R_0$ となる.

定理1と合わせて以上のことから, 最適総テストラン回数 N^* に関する次の定理を容易に得る.

[定理2] $c_1>0, c_2>0$, および $r_G(0)=a$ とする. また, $0<R_0<1$ および $h\geq 0$ とする.

(i) もし $W(0)>c_1/c_2$ かつ $R_G(0,h)<R_0$ ならば, $N^*=\max\{N_0, N_1\}$ である.

(ii) もし $W(0)>c_1/c_2$ かつ $R_G(0,h)\geq R_0$ ならば, $N^*=N_0$ である.

(iii) もし $W(0)\leq c_1/c_2$ かつ $R_G(0,h)<R_0$ ならば, $N^*=N_1$ である.

(iv) もし $W(0)\leq c_1/c_2$ かつ $R_G(0,h)\geq R_0$ ならば, $N^*=0$ である.

(証明) もし $W(0)>c_1/c_2$ かつ $R_G(0,h)<R_0$ ならば, $N_1>N_0$ のとき $R_G(N_0,h)<R_0$ かつ $R_G(N,h)>R_0$ $(N>N_1)$ であるから $N^*=N_1$ となり, $N_1<N_0$ のとき $R_G(N,h)>R_0$ $(N>N_1)$ であるから $N^*=N_0$ となる. これらをまとめると, $W(0)>c_1/c_2$ かつ $R_G(0,h)<R_0$ ならば $N^*=\max\{N_0, N_1\}$ となる. つぎに, $R_G(0,h)\geq R_0$ ならば式(23)を満たす有限な解は存在せず, $R_G(N,h)>R_0$ $(N>0)$ である. したがって, もし $W(0)>c_1/c_2$ かつ $R_G(0,h)\geq R_0$ ならば, コスト評価基準のみで決まり $N^*=N_0$ となる.

一方, $W(0)\leq c_1/c_2$ かつ $R_G(0,h)<R_0$ ならば, $R_G(N,h)>R_0$ $(N>N_1)$ であるから信頼度評価基準のみで決まり $N^*=N_1$ となる. また, $W(0)\leq c_1/c_2$ かつ $R_G(0,h)\geq R_0$ ならば, $R_G(N,h)>R_0$ $(N>0)$ であるから $N^*=0$ となる. Q. E. D.

6. 適 用 例

NHPP により記述された幾何減少型エラー発見率モデルに基づき, 実際のテストデータを使ったソフトウェアの信頼性解析および最適リリース問題への適用例を示す. 適用されるテストデータは, Yamada and Osaki[2] により引用されたもので, 図1 に示すような18組のデータ (n_i, x_i) $(i=1,2,\cdots,18)$ から成っている. 本データは, PL/I およびアセンブラ言語で書かれた約 50,000 コード行 (lines of code) から成るアプリケーション・プログラムのテスト段階で実際に観

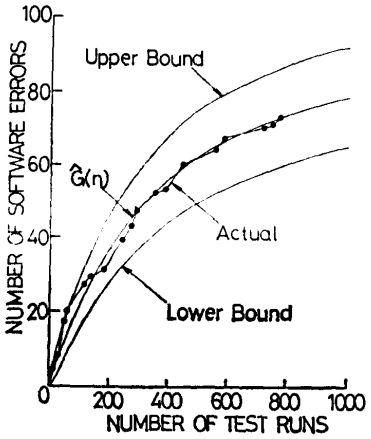

図 1 実際データに対して推定された平均値関数 $\hat{G}(n)$ とその 90% 信頼限界

Fig. 1 The estimated mean value function $\hat{G}(n)$ and the 90 percent confidence bounds for actual data.

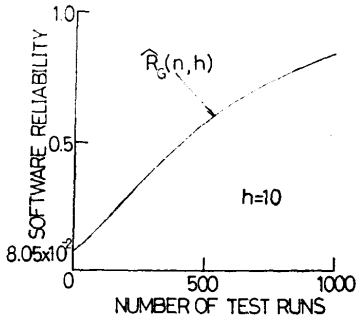

図 2 $h=10$ に対する推定されたソフトウェア信頼度 $\hat{R}_G(n, h)$

Fig. 2 The estimated software reliability $\hat{R}_G(n, h)$ for $h=10$.

測されたものである.

この 18 組のテストデータを用いて式(14)から導出される同時尤度方程式を数値的に解くと，モデル・パラメータの最尤推定値として

$$\hat{a}=112.9,\ \hat{D}=2.27\times10^{-3},\ \hat{p}=9.985\times10^{-1}, \tag{24}$$

を得る．式(24)から，平均値関数 $G(n)$ の最尤推定値を求めると

$$\hat{G}(n)=112.9\left[1-\prod_{i=1}^{n}\{1-(2.27\times10^{-3})\cdot(9.985\times10^{-1})^{i-1}\}\right], \tag{25}$$

となり，これを 90% 信頼限界とともに図 1 に示した．この推定結果は観測データに良く適合しており，χ^2(カイ 2 乗) 検定法[11]により統計的に確認した．このテスト工程では，総テストラン回数 773 により 73 個のエラーが発見されている．すなわち，$n_{18}=773$ および $x_{18}=73$ である．したがって，$\hat{a}-x_{18}\fallingdotseq40$ 個のエラーがさらにソフトウェア内に潜在していることが推定され，このことは当該テスト終了後のエラー追跡調査により妥当な結果であることがわかった．また，ソフトウェアの信頼性評価尺度の 1 つであるソフトウェア信頼度について，$h=10$ として最尤推定値 $\hat{R}_G(n, 10)$ を図 2 に示した．図 2 から，テスト終了時点におけるソフトウェア信頼度は $\hat{R}_G(773, 10)=7.550\times10^{-1}$ であることがわかる．

上述の幾何減少型エラー発見率モデルに基づくテストデータ解析結果を使って，ソフトウェアの最適リリース問題に関する数値例を示す．まず，第 4 章のコスト評価基準による最適リリース問題を考える．ソフトウェアコストの相対的な大きさにより議論するために，テストラン 1 回当りのコスト c_1 を標準値として $(c_1=1)$，定理 1 の最適総テストラン回数 N^* を求めると表 1 の結果を得る．表 1 から，テスト工程において未発見となったエラー 1 個当りの保守コスト c_2 が大きくなればなるほど，N^* も大きくなりテストを十分に行う必要があることがわかる．具体的に数値例を示すと，$c_1\fallingdotseq1$ および $c_2=2$ のとき $W(0)<c_1/c_2$ であるから，定理 1 (i) より $N^*=0$ である (図 3 参照)．また，$c_1=1$ および $c_2=25$ のとき $W(0)>c_1/c_2$ であるから，定理 1 (ii) より $W(623)>c_1/c_2$ かつ $W(624)<c_1/c_2$ となり $N^*=N_0=624$ である (図 4 参照)．

表 1 コスト評価基準による最適リリース政策の数値例

Table 1 Numerical examples of optimal software release policies based on cost criteria.

c_1	c_2	N^*	$W(N^*)$	c_1/c_2
1	1	0	2.563×10^{-1}	1.0
1	2	0	2.563×10^{-1}	5.000×10^{-1}
1	3	0	2.563×10^{-1}	3.333×10^{-1}
1	5	68	1.998×10^{-1}	2.000×10^{-1}
1	10	281	9.980×10^{-2}	1.000×10^{-1}
1	15	423	6.664×10^{-2}	6.667×10^{-2}
1	20	534	4.988×10^{-2}	5.000×10^{-2}
1	25	624	3.999×10^{-2}	4.000×10^{-2}

つぎに，コスト評価基準だけでなく信頼度評価基準をも考慮した第 5 章の最適リリース問題を考える．ここで，コスト評価基準による最適リリース政策は，上述の $c_1=1$ および $c_2=25$ の場合であるものと仮定する．第 1 例として，$h=10$ および $R_0=0.8$ のとき，最適リリース問題は

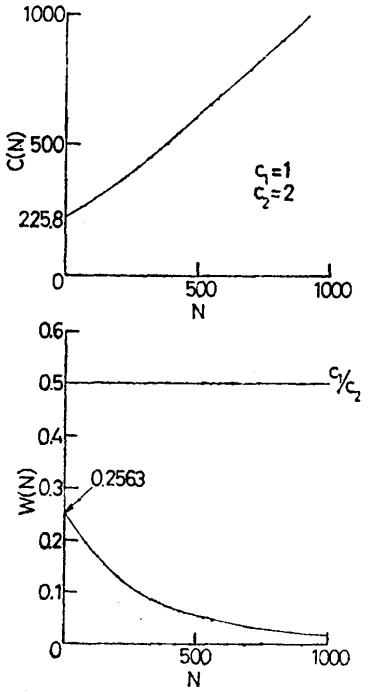

図 3 $c_1=1$ および $c_2=2$ のときの最適総テストラン回数

Fig. 3 The optimum total number of test runs for $c_1=1$ and $c_2=2$.

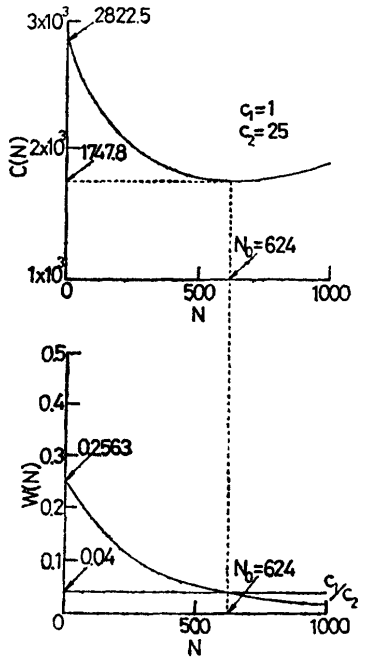

図 4 $c_1=1$ および $c_2=25$ のときの最適総テストラン回数

Fig. 4 The optimum total number of test runs for $c_1=1$ and $c_2=25$.

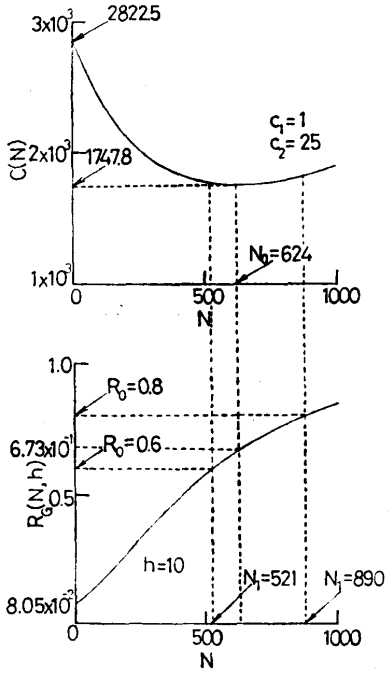

図 5 コストおよび信頼度評価基準を同時に考慮した最適リリース政策

Fig. 5 The optimal software release policies based on both cost and reliability criteria.

$$\left.\begin{array}{l}\text{minimize } C(N)\\ \text{subject to } R_G(N,10)\geq 0.8,\ N\geq 0\end{array}\right\}, \tag{26}$$

となる．このとき，$R_G(889,10)<R_0$ かつ $R_G(890,10)>R_0$ を満たすことから $N_1=890$ となる．したがって，$W(0)>c_1/c_2$ かつ $R_G(0,10)=8.05\times10^{-2}<R_0$ であるから，定理2(i)より $N^*=\max\{N_0,N_1\}=\max\{624,890\}=890$ となる（図5参照）．もしコスト評価基準のみによる最適総テストラン回数 $N_0=624$ を採用するならば，このときソフトウェア信頼度は $R_G(624,10)=6.73\times10^{-1}$ であり目標信頼度 $R_0=0.8$ を達成することはできない．ゆえに，テストはさらに継続されることになる．このような場合，コストおよび信頼度評価基準の両者を同時に考慮することが重要であることがわかる．同様に第2例として，$h=10$ および $R_0=0.6$ のときは，$N_1=521$ であるから定理2(i)より $N^*=\max\{N_0,N_1\}=\max\{624,521\}=624$ となる（図5参照）．この場合は，コスト評価基準のみに基づく最適総テストラン回数 $N_0=624$ を採用したときには目標信頼度 $R_0=0.6$ を達成しており，テストを終了してソフトウェアをリリースすることができる．

7. むすび

ソフトウェア開発の最終段階であるテスト工程において観測されたソフトウェアエラーデータを使って，ソフトウェアの信頼性モデルにより開発されたソフトウェアの信頼性評価を実施するときには，データの種類に応じたデータ解析を行うことが重要である．本論文では，エラー発見期間の単位をテストラン試行回数などの離散的時間とするようなテストデータを解析するために，離散型ソフトウェア信頼度成長モデルについて考察した．エラー発見事象のモデル化にあたり重要な役割を果たす平均値関数，すなわち任意の総テストラン回数により発見される総期待エラー数を導出した上で，NHPP に基づく確率モデルとして一般的に記述した．さらに，ソフトウェアの信頼性評価尺度およびモデル・パラメータの推定法を与えた．このようなモデルの一般的記述に対する特別な場合として，エラー発見率がテストの進行とともに減少していくという実際にもよく経験される現象を表すために，幾何減少型エラー発見率モデル[4]について総括した．

ソフトウェアの信頼性モデルに基づくテストデータの解析結果を，開発管理に反映することは興味ある問題である．本論文では，上述の幾何減少型エラー発見率モデルを用いて，テスト工程におけるテストランを打ち切って運用段階へ移行するのに最適な総テストラン回数を求める最適リリース問題も議論した．その最適リリース政策を考えるにあたって必要な評価基準として，総期待ソフトウェアコストおよびソフトウェア信頼度を導入した．まず，前者のコスト評価基準のみを採用する場合，つぎに前者だけでなく後者の信頼度評価基準をも採用する場合について，これらの最適リリース政策を定理としてまとめた．その結果，ソフトウェアの最適リリース問題を考える際には，コスト評価基準および信頼度評価基準を同時に考慮することの重要性がわかった．

今後は，いくつかのソフトウェア・プロジェクトに本論文で議論したソフトウェア信頼度成長モデルを適用し，信頼性データ解析法とその応用である最適リリース政策の実際問題に対する妥当性を検証していかなければならない．

謝辞 日頃，暖かいご指導を頂く広島大学工学部の尾崎俊治教授に深謝致します．また，有益なご助言を頂いた名城大学理工学部の中川覃夫助教授に感謝の意を表します．なお，本研究の一部は文部省科学研究費（奨励研究A）の補助を受けたことを付記する．

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) Yamada, S. and Osaki, S.: Discrete Software Reliability Growth Models, *J. Applied Stochastic Models and Data Analysis*, Vol. 1, No. 1, pp. 65-77 (1985).
3) Ascher, H. and Feingold, H.: *Repairable Systems Reliability: Modeling, Inference, Misconceptions and Their Causes*, Marcel Dekker, Inc., New York (1984).
4) Kitaoka, T., Yamada, S. and Osaki, S.: A Discrete Non-Homogeneous Error Detection Rate Model for Software Reliability, *Trans. IECE Japan (Section E)*, Vol. E 69, No. 8, pp. 859-865 (1986).
5) 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).
6) Yamada, S. and Osaki, S.: Software Reliability Growth Modeling: Models and Applications, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 12, pp. 1431-1437 (1985).
7) Moranda, P. B.: Predictions of Software Reliability during Debugging, *Proc. Ann. Reliability and Maintainability Symp.*, pp. 327-332 (1975).
8) Musa, J. D. and Okumoto, K.: A Logarithmic Poisson Execution Time Model for Software Reliability Measurement, *Proc. Int. Conf. Software Engineering*, pp. 230-238 (1984).
9) Koch, H. S. and Kubat, P.: Optimal Release Time of Computer Software, *IEEE Trans. Softw. Eng.*, Vol. SE-9, No. 3, pp. 323-327 (1983).
10) Okumoto, K. and Goel, A. L.: Optimum Release Time for Software Systems Based on Reliability and Cost Criteria, *J. Syst. Softw.*, Vol. 1, pp. 315-318 (1980).
11) 山田 茂：ソフトウェアの信頼性評価法，ソフト・リサーチ・センター，東京 (1985).
12) Yamada, S. and Osaki, S.: Cost-Reliability Optimal Release Policies for Software Systems, *IEEE Trans. Reliab.*, Vol. R-34, No. 5, pp. 422-424 (1985).

(昭和61年11月10日受付)
(昭和62年3月25日採録)

山田　茂（正会員）

昭和27年生．昭和50年広島大学工学部経営工学科卒業．昭和52年同大学院修士課程修了．昭和52～55年，日本電装(株)品質保証部勤務．昭和58年広島大学大学院博士課程修了．同年より岡山理科大学勤務，現在工学部電子工学科専任講師．工学博士．信頼性工学，システム工学，ソフトウェアの信頼性などの研究に従事．著書「ソフトウェアの信頼性評価法」など．電子情報通信学会，日本ＯＲ学会，日本経営工学会各会員．