ソフトウェア工学 46−13
（1986. 2. 7）

離散型ソフトウェア信頼度成長モデルの解析とソフトウェア信頼性評価ツールの作成

北岡 武司
（広島大学）

山田 茂
（岡山理科大学）

尾崎 俊治
（広島大学）

1．はじめに

高信頼性コンピュータソフトウェアを作ることはソフトウェア工学の分野で最も重要で急務のテーマの1つである．要求仕様定義－設計－コーディング－試験という一連のソフトウェア開発過程の各段階で様々な技術が開発され適用されている．特に，試験段階では開発されたソフトウェアが運用段階で期待通りに動作することを保証するために繰り返しテストされる．

このとき，ソフトウェアの信頼性を評価する立場から，ソフトウェア故障発生（ソフトウェアエラー発見）現象を解析・モデル化することは興味深い問題である．ソフトウェア故障とはソフトウェア内に潜在するエラーによってソフトウェアが期待通りに動作しなくなることと定義する．既に，開発されたソフトウェアの信頼性・処理性を予測・評価するソフトウェア信頼性モデルがいくつか提案されてきた．ソフトウェア信頼度成長モデルもその1つで，ソフトウェア内に残存するエラー数やソフトウェア故障発生時間間隔を推定するモデルである（例えば，Goel and Okumoto[1]，Littlewood[2]，Musa[3]，Yamada et al.[4] 等）．この分野の主な目的は，ソフトウェア内に残存するエラーが発見・修正されていく過程をモデル化することである．このとき，カレンダータイムやCPU時間がソフトウェアエラー発見の単位としてよく用いられてきた．しかしながら，テストラン試行回数（実行テストケース数）を用いた方が良い場合がある（Ramamoorthy and Bastani[5] 参照）．そのような場合のソフトウェア信頼度成長モデルは離散型ソフトウェア信頼度成長モデルと呼ばれる．

本研究では，3タイプの離散型ソフトウェア信頼度成長モデルについて考察する（Yamada and Osaki[6]，Kitaoka et al.[7] 参照）．モデルは，確率変数がn回のテストランによって発見されるエラー数として表される非定常ポアソン過程（Non-Homogeneous Poisson Process，以下NHPPと略す）を用いてエラー発見過程を表わしている．このモデルの下で，定量的信頼性評価尺度を導出し，実際のエラーデータをこれらのモデルに適用した結果を示す．最後に，これらのモデルを用いて作成したソフトウェア信頼性評価ツールの概要を示す．

2．離散型NHPPモデルの一般的記述

試験段階ではソフトウェア内に潜在するエラーを発見・修正するために繰り返しテストケースが実行される．このとき，エラーの修正によって新たにエラーを作り込まないものと仮定し，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わす離散型計数過程（discrete counting process）を$\{N(n),\ n\geq 0\}$ $(n=0,1,2,\ldots)$ と定義する．ソフトウェアエラー発見現象を記述するために，以下に示すような平均値関数 $D(n)$ をもつNHPPに基づいて，離散型ソフトウェア信頼度成長モデルを提案する：

(i) $N(0)=0$.

(ii) $\{N(n),\ n\geq 0\}$ は独立増分をもつ．

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

$$\Pr\{N(n_j)-N(n_i)=x\} = \frac{\{D(n_j)-D(n_i)\}^x}{x!}\cdot\exp[-\{D(n_j)-D(n_i)\}] \quad (x=0,1,2,\ldots).$$

平均値関数 $D(n)$ はn回のテストランにより発見される期待累積エラー数を表わし，上に有界な非減少関数である．ここで，$D(0)=0$ である．このとき，$D(n)$ をもつ離散型NHPPモデルは

$$Pr\{N(n)=x\}=\frac{\{D(n)\}^x}{x!}\cdot\exp[-D(n)]$$

$$(x,n=0,1,2,\ldots),\quad(1)$$

として定式化される．また，n回目のテストラン終了後のエラー発見率は

$$q(n)=[D(n+1)-D(n)]/[A-D(n)],\quad(2)$$

で与えられる．ここで，Aは初期に潜在している期待エラー数である．エラー発見率 $q(n)$ が既知のとき，(2)式の差分方程式を解くと，モデルの平均値関数 $D(n)$ は

$$D(n)=A\sum_{k=1}^{n}\frac{f(n)}{f(k)}\cdot q(n-1),\quad(3)$$

ただし，

$$f(k)=\prod_{i=1}^{k}[1-q(i-1)],\quad(4)$$

と表わされる．

次に，(3)式の平均値関数 $D(n)$ をもつ離散型ＮＨＰＰモデルに基づいて，いくつかの定量的信頼性評価尺度を導出する．$\overline{N}(n)$ をn回目のテストラン終了後，システム内に残存するエラー数とする．すなわち，

$$\overline{N}(n)=N(\infty)-N(n).\quad(5)$$

このとき，$\overline{N}(n)$ の期待値および分散は

$$r(n)=E[\overline{N}(n)]=Var[\overline{N}(n)]=A-D(n),\quad(6)$$

となる．また，n回までのテストランでm個のエラーが発見されたという条件の下でそれからh回のテストランによりエラーが発見されない確率は

$$R(n,h)=Pr\{N(n+h)-N(n)=0\mid N(n)=m\}=\exp[-\{D(n+h)-D(n)\}]$$

$$(n,h=0,1,2,\ldots),\quad(7)$$

となり，mに無関係である．この $R(n,h)$ はソフトウェア信頼度と呼ばれる信頼性評価関数である．

３．モデル

各モデルは表１に示した仮定に基づいている．モデルは１回のテストランにより発見されるエラー数が幾何級数的に減少するモデル（幾何型モデル，　モデル１，２），１回のテストランにより発見されるエラー数がその時点でソフトウェア内に潜在するエラー数に比例するモデル（エラー内包量比例発見率モデル，　モデル３，４）および前者２つのタイプを統合したモデル（幾何減少型発見率モデル，　モデル５）の３タイプである．さらに前者２タイプのモデルについては，発見難易度が一様なもの（基本モデル）と発見難易度の異なる２種類のエラー群をもつもの（拡張モデル）とのそれぞれ２モデルずつとした．仮定は表１の２段目に示される差分方程式として表わされる．これらの差分方程式を解くことにより各モデルの平均値関数が得られ，さらに(2)式に代入することによってエラー発見率が得られる．これらをそれぞれ表１の３および４段目に示した．各モデルは(1)式の $D(n)$ のかわりにそれぞれの平均値関数を代入することにより定式化される．

次に，それぞれの平均値関数により定式化された各モデルについて，定量的信頼性評価尺度を導出する．１つは残存エラー数の期待値および分散で，もう１つはソフトウェア信頼度である．それぞれ(6)式および(7)式に各モデルの平均値関数を代入することにより得られ，表１の５および６段目に示した．

４．実際データへの適用

各モデルに実際の試験段階で観測されたエラーデータを適用する．　データはＰＬ／Ｉおよびアセンブラで書かれた約50,000 LOC (lines of code) から成るあるアプリケーション・プログラムの試験段階で実際に観測されたものである．データの組数は１８組で，エラー発見の単位は実行されたテストケース数である．各モデルのパラメータは最尤法により推定し，その結果を表２に示した．また，テストラン試行回数と累積発見エラー数との関係をモデル５を例にとり実際のエラーデータと共に図１に示した．

次に，各モデルのデータへの適合度を統計的に検定するために χ^2 検定(Yamada[8])

表1．　モ　デ　ル

モデル	幾何型モデル		エラー内包量比例発見率モデル		幾何減少型発見率モデル
	Model 1	Model 2	Model 3	Model 4	Model 5
仮定	1.任意のテストランにおける発見数とその1つ前のテストランにおける発見数の比は一定で1未満である． 2.テストラン間隔は統計的に独立である． 3.エラー1個当り（テストラン1回当り）の発見率は一定である．	3.発見難易度の異なる2種類のエラー群が存在する． 4.各タイプのエラー1個当り（テストラン1回当り）の発見率は一定である．	1.1回のテストランにより発見される期待エラー数は，その時点でソフトウェア内に潜在するエラー数に比例する． 2.テストラン間隔は統計的に独立である． 3.エラー1個当り（テストラン1回当り）の発見率は一定である．	3.発見難易度の異なる2種類のエラー群が存在する． 4.各タイプのエラー1個当り（テストラン1回当り）の発見率は一定である．	1.1回のテストランにより発見される期待エラー数はその時点でソフトウェア内に潜在しているエラー数に比例して幾何級数的に減少する． 2.テストラン間隔は統計的に独立である．
差分方程式	$G(n+1)-G(n)=Dr^n$ $(n=0,1,2,\ldots)$, $D>0$, $0<r<1$ D=初期発見エラー数の期待値 r=期待発見エラー数の減少率	$Gp_i(n+1)-Gp_i(n)=p_iDr_i^n$ $(n=0,1,2,\ldots)$, $D>0$, $0<r_i<1$, $p_1+p_2=1$, $0<p_i<1$ $(i=1,2)$ D=初期発見エラー数の期待値 r_i=各エラータイプの期待発見エラー数の減少率 p_i=タイプiエラーの含有率	$M(n+1)-M(n)=b(a-M(n))$ $(n=0,1,2,\ldots)$, $a>0$, $0<b<1$ a=初期潜在エラー数の期待値 b=エラー1個当りの発見率	$Mp_i(n+1)-Mp_i(n)=b_i(p_ia-Mp_i(n))$ $(n=0,1,2,\ldots)$, $a>0$, $0<b_i<1$, $p_1+p_2=1$, $0<p_i<1$ $(i=1,2)$ a=初期潜在エラー数の期待値 b_i=各エラータイプのエラー1個当りの発見率 p_i=タイプiエラーの含有率	$C(n+1)-C(n)=Dr^n(a-C(n))$ $(n=0,1,2,\ldots)$, $a>0$, $0<D<1$, $0<r<1$ a=初期潜在エラー数の期待値 D=エラー1個当りの初期発見率 r=エラー発見率の減少率
平均値関数	$G(n)=D(1-r^n)/(1-r)$	$Gp(n)=Gp_1(n)+Gp_2(n)$ $Gp_i(n)=p_iD(1-r_i^n)/(1-r_i)$ $(i=1,2)$	$M(n)=a(1-(1-b)^n)$	$Mp(n)=Mp_1(n)+Mp_2(n)$ $Mp_i(n)=p_ia(1-(1-b_i)^n)$ $(i=1,2)$	$C(n)=Da[r^{n-1}+\sum_{i=1}^{n-1}\{r^{i-1}\cdot\prod_{j=i}^{n-1}(1-Dr^j)\}]$
エラー発見率	$q(n)=1-r$ (Constant)	$q(n)=(p_1r_1^n+p_2r_2^n)/w(n)$ $w(n)=p_1r_1^n/(1-r_1)+p_2r_2^n/(1-r_2)$ (Decreasing)	$q(n)=b$ (Constant)	$q(n)=(b_1z_1(n)+b_2z_2(n))/(z_1(n)+z_2(n))$ $z_i(n)=p_i(1-b_i)^n$ $(i=1,2)$ (Decreasing)	$q(n)=Dr^n$ (Decreasing)
残存エラー数	$r(n)=Dr^n/(1-r)$	$r(n)=Dw(n)$	$r(n)=a(1-b)^n$	$r(n)=a(z_1(n)+z_2(n))$	$r(n)=a-C(n)$
ソフトウェア信頼度	$R(n,h)=\exp[-G(h)r^n]$	$R(n,h)=\exp[-Gp_1(h)r_1^n-Gp_2(h)r_2^n]$	$R(n,h)=\exp[-M(h)(1-b)^n]$	$R(n,h)=\exp[-Mp_1(h)(1-b_1)^n-Mp_2(h)(1-b_2)^n]$	$R(n,h)=\exp[-(C(n+h)-C(n))]$

参照）を実施し，各モデルとも有意水準５％で適合性が確認された（表２）．さらに，偏差二乗和と最終発見エラー数の期待値を比較基準として各モデルの比較をした．偏差二乗和の比較においてはモデル５が最も良い結果を与え，最終発見エラー数については８５～９５個であることが確認されており，モデル２，４および５が良い結果を与えている（表２）．

各モデルの特徴を表わす１つの指標としてエラー発見率がある．表１の４段目に示されたエラー発見率の関数に推定により得られた各パラメータの値（表２参照）を代入し，これらを図２に示した．モデル１および３は発見率が一定なモデルであり，モデル２，４および５は発見率が試験と共に減少するモデルである．

対象となるソフトウェアの信頼性を定量的に評価する尺度として，テストラン試行回数と残存エラー数の期待値との関係およびテストラン試行回数とソフトウェア信頼度との関係をモデル５を例にとってそれぞ

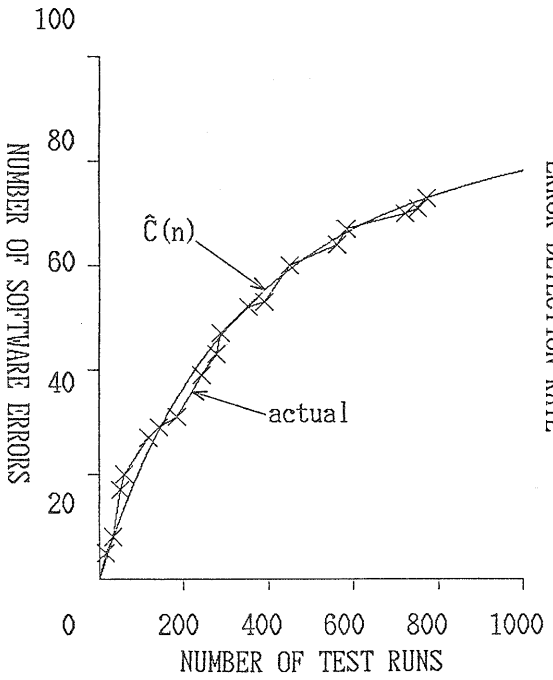

図１．　平均値関数(モデル５)とエラーデータ

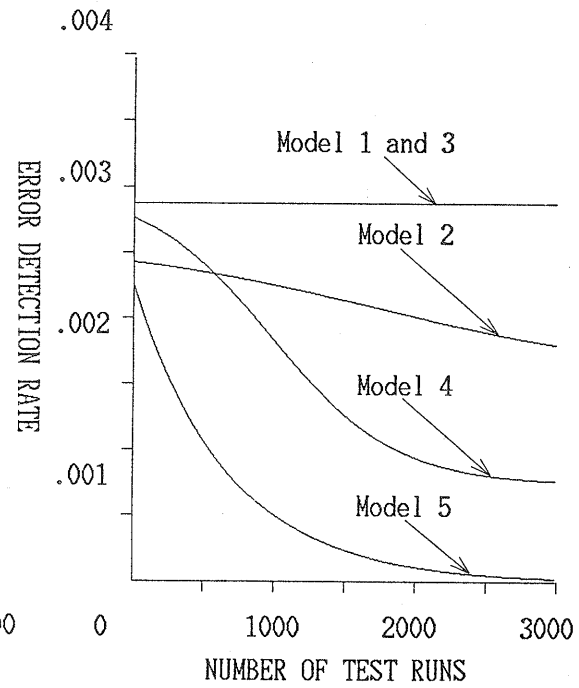

図２．　各モデルのエラー発見率

表２．　各モデルのパラメータの最尤推定値と比較基準

	Model 1	Model 2 (p_1=.9,p_2=.1)	Model 3	Model 4 (p_1=.9,p_2=.1)	Model 5
パラメータの最尤推定値	D=0.2344 r=0.9971	D =0.2121 r_1=0.9974 r_2=0.9984	a=81.95 b=0.0029	a= 85.22 b_1=0.00301 b_2=0.00075	a=112.87 D=0.00227 r=0.99850
χ^2検定量	6.35	5.89	6.35	6.39	6.59
偏差二乗和	133.63	192.82	133.55	130.55	113.93
最終発見エラー数の期待値	81.95	86.66	81.95	85.22	88.09

れ図３および図４に示した．図３では，試験の進行と共にソフトウェア内に残存するエラー数が減少していく過程がわかる．また，図４では，テストラン試行回数ｎを固定したときそれ以後の実行回数ｈが増加するにつれてソフトウェア信頼度が低下し，逆に目標とするソフトウェア信頼度を固定したときソフトウェア実行回数ｈに従って必要なテストラン試行回数を読み取ることができる．

５．ソフトウェア信頼性評価ツール

前述してきた５つのモデルについて信頼性評価ツールを作成した．その概要は図５に示す通りである．　ツールの対応機種はIBM5550 シリーズであり，使用言語は MS-DOS BASIC（コンパイラ用）である．モデルパラメータの推定は最尤法によるもので，計算手法としては Newton-Raphson 法を採用した．各モデルの計算時間は，イテレーション回数を１０回として前述のデータを使って推定した場合，モデル１～４では約１５秒，モデル５では約2,500秒であった．

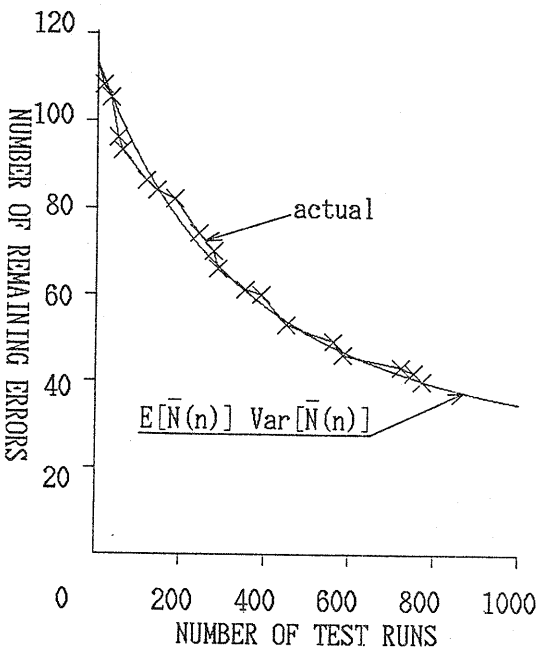

図３．　残存エラー数の期待値および分散

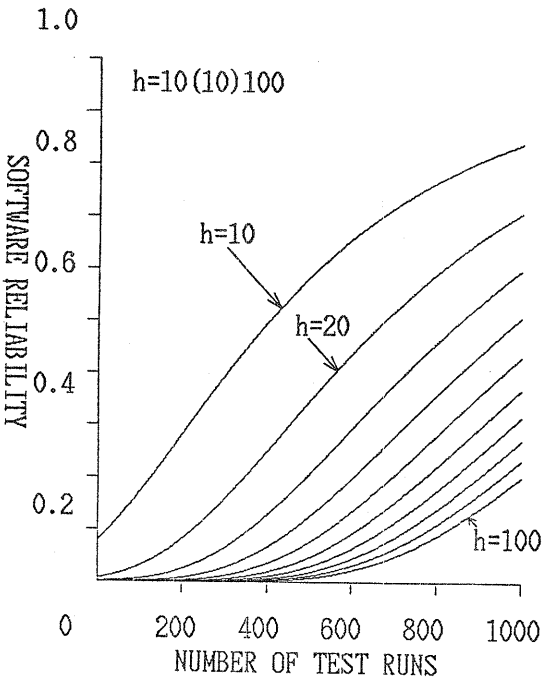

図４．　ソフトウェア信頼度(h=10(10)100)

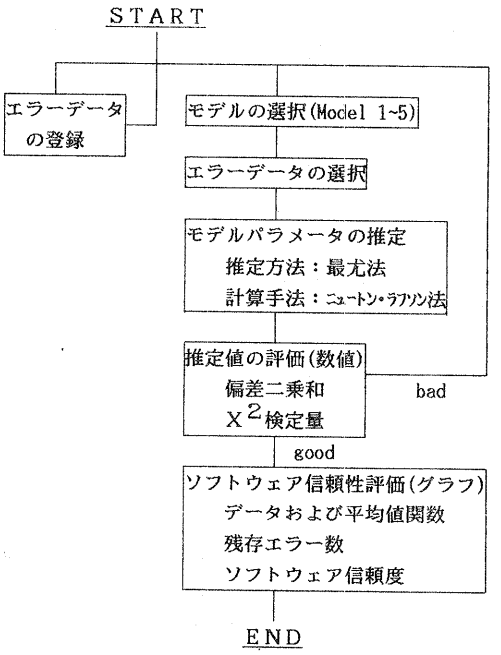

図５．　ソフトウェア信頼性評価ツールの概要

また，ツールは推定により得られたモデルパラメータの値を用いて，ソフトウェア信頼性を評価した結果をグラフおよび数値で示すものである．図１，３および４はＣＲＴに描かれた結果を画面コピーしたものであり，この他にエラーデータとモデルとの適合度を確認するために偏差二乗和およびχ^2検定量が数値として得られる．

６．まとめ

本研究では，３タイプ５通りの離散型ソフトウェア信頼度成長モデルを定式化し，これらのモデルに基づいて定量的信頼性評価尺度を導出した．また，実際のエラーデータを適用した結果を示し，それに使用したソフトウェア信頼性評価ツールの概要を述べた．

各モデルはエラー発見の単位としてテストラン試行回数（実行テストケース数）を用いた．カレンダータイムをエラー発見の単位として用いる場合，試験努力にムラがあるときあるいはエラー発見・修正過程に学習要素が含まれると考えられるときには人間的要因による影響が強く，発見・修正過程をモデル化することが困難なことが起こりうる．また，ＣＰＵ時間をエラー発見の単位として用いる場合，エラー修正の際のソフトウェア実行時間を含むことが考えられ，修正に手間取るあるいは修正が不完全で同じエラーを２度カウントする等人間的要素の影響によりエラーデータに偏りを生じる可能性がある．一方，テストラン試行回数（実行テストケース数）をエラー発見の単位として用いる場合，１つのテストケースにより発見されたエラーは同じテストケースでエラーが発見されなくなるまで修正された後次のテストケースが実行されるので，エラーデータには時間的要素が含まれず，修正において人間的要素の影響を受けにくいという利点をもっている．しかしながら，テストケースが互いに似通っているものばかりのときエラーデータに偏りを生じるあるいはテストケースの実行順によってエラー発見過程が変化するなどの問題点があるため，各テストケースの独立性および実行順のランダム化に配慮する必要がある．

考察した５つのモデルは発見率の性質によって発見率の一定なモデル（モデル１，３）と発見率の減少するモデル（モデル２，４，５）の２つに大別されるが，試験と共に試験効率が低下すると思われるので，発見率が試験と共に減少すると考える方がより実際的である．また，モデル１～４は初期潜在エラー数と最終発見エラー数が一致しており，ソフトウェア内に潜在するエラーはすべて発見されることを表わしている．一方，モデル５は最終発見エラー数が初期潜在エラー数を下回っており，どのように試験しても発見できないエラーが存在することを表わしている．これらより，モデル５が大規模で複雑なソフトウェアに対しては最も実際的であると考えられる．適用したデータに関しても，モデル５は総合的に良い結果を与えていると評価できる．しかし，モデル５は他のモデルに比べ平均値関数が複雑で，モデルパラメータの推定に時間がかかるため，エラーデータのテストラン回数が大きいときは実用的でない．モデル２，４も総合的に良い結果を与えていると評価できるが，これらのモデルは発見の容易・困難なエラーの割合を経験的に決定してやる必要がある．これらの割合を決定することが困難なことがしばしばあり，このような問題を解決していかなければならない．

今後，数多くのエラーデータをこれらのモデルに適用しモデルの評価・検討を行うことが課題である．さらに，エラーデータ採取時刻の決定およびモデルの評価を精度良く与えるにはどの状態までエラーデータを採取すればよいかという問題についても考察していきたい．

参考文献

(1) 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).

(2) Littlewood, B.:Theories of Software Reliability: How Good Are They and How Can They Be Improved?, IEEE Trans. Software Eng., Vol. SE-6, No.5, pp.489-500 (1980).

(3) Musa, J.D.:The Measurement and Management of Software Reliability, IEEE Proc., Vol. 68, No. 9, pp. 1131-1143 (1980).

(4) 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).

(5) Ramamoorthy, C.V. and Bastani,F.B.: Software Reliability-Status and Perspectives, IEEE Trans. Software Eng., Vol. SE-8, No. 4, pp. 354-371 (1982).

(6) 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).

(7) 北岡武司・山田茂・尾崎俊治：２種類のエラーを考慮した離散的ソフトウェア信頼度成長モデル，信学論(D)，Vol. J68-D, No. 6, pp. 1242-1247 (1985).

(8) 山田茂：ソフトウェア信頼性評価，ソフトリサーチセンター，東京 (1985).