解　説

非定常ポアソン過程モデル†

尾　崎　俊　治††

1. はじめに

情報処理産業は現在最も成長性の高い産業の一つであり，今後も高い成長性が期待できる．Boehm[1]の1976年の論文で予測したように，コンピュータシステムのコスト比でみるかぎり，ソフトウェアの比重は年年増加している．コンピュータシステムの役割はますます重大となり，その障害は経済的損失のみでなく社会的信用の失墜や社会的混乱をも引き起こす．文献2)の最近のアンケート調査によれば，平成元年調査の重大障害の要因分類の第1位はソフトウェアバグで27%，システム設計不良（主としてソフト）の7%を加えると，システムの重大障害の約1/3のソフトウェアに起因していると言ってよい．

今日のコンピュータシステムはますます大規模化し，ネットワーク化も進んでいる．それにともなってソフトウェアも大規模化している．大規模なシステムを動作させるときソフトウェア故障をできるだけ少なくするため，ソフトウェアに関するいろいろなフォールトトレランス技術[3]が採用されるようになった．しかし，フォールトトレランスによって故障を覆い隠すことはかぎりがあるので，ソフトウェアの内部に立ち入って信頼性を高めることが基本的であり，ソフトウェア信頼性評価技術の一つであるソフトウェア信頼性モデルを研究することが最も急務な課題である．本稿においては，**ソフトウェア信頼性モデル** (Software Reliability Model) の中でも代表的なモデルである**ソフトウェア信頼度成長モデル** (Software Reliability Growth Model) に限定して議論を進める．

一般に，ソフトウェアにおいては，ソフトウェア故障（現象）はソフトウェアエラー（原因）により引き起こされるとする．これらの用語はフォールトトレラントシステムの用語と一致していない（たとえば，向殿[4]参照）．本稿においては，エラーが原因であり，故障はその結果であるとする．

ソフトウェア信頼性を主題，あるいはそれを含めたソフトウェア工学の本は内外で数多く出版されている．邦書としては，三浦[5]，菅野[6),7)]，大場[10]，山田[8),9)]などがある．洋書としては，Shooman[11]，Musa et al.[12]などがある．解説あるいは2次資料としては，Ramamoorthy and Bastani[13]，Dhillon[14]，Shanthikumar[15]，Mellor[16]，尾崎[17]，当麻[18),19)]，三道[20]などがある．ソフトウェア信頼性モデルについては上記の文献を参照されたい．

2. 非定常ポアソン過程

2.1 ソフトウェア信頼度成長モデル

ソフトウェア開発のテスト段階において，ソフトウェアに内在するエラーはソフトウェア故障として発見され，エラーは分離・修正される．テスト時間の経過とともに，エラーは減少し，ソフトウェアの信頼度は成長する．このような過程を数式で表したものをソフトウェア信頼度成長モデルとよぶ．

ソフトウェア信頼度成長モデルに共通する基本的な仮定を列挙する．これらの仮定は多くのモデルに共通するものであるが，一部のモデルにおいてはこれらの仮定を緩めることもできる．

（i） 各ソフトウェア故障は一つのエラーにより引き起こされる．

（ii） 発見された各ソフトウェアエラーはすべて分離・修正され，修正時に新しいエラーは作り込まれない．

（iii） 各ソフトウェアエラーの修正時間は無視する．

（iv） 各ソフトウェア故障は互いに独立である（すなわち，ソフトウェア故障は他のソフトウェアエラーに影響を及ぼさない）．

（v） エラー発見のためのソフトウェアへの入力

† Nonhomogeneous Poisson Process Models by Shunji OSAKI (Department of Industrial and Systems Engineering, Hiroshima University).

†† 広島大学工学部第二類（電気系）

はランダムであり，ソフトウェア故障は一つずつ起こる．

このほかに厳密にはモデルに応じていろいろな仮定が必要である．これらの仮定の下で，いろいろなソフトウェア信頼度成長モデルが提案されている．テスト段階で，$(i-1)$ 番目と i 番目の間のソフトウェア故障発生間隔を X_i $(i=1,2,\cdots,n)$ とする．X_i の分布を

$$F(x_i)=P\{X_i\leq x_i\} \quad (x_i\geq 0) \qquad (1)$$

としよう．さらに，密度 $f(x_i)=dF(x_i)/dx_i$ は存在するとすれば，故障率すなわち瞬間エラー発見率は

$$z(x_i)=\frac{f(x_i)}{1-F(x_i)} \quad (x_i\geq 0) \qquad (2)$$

で与えられる．分布 $F(x_i)$ あるいは故障率 $z(x_i)$ から，時間間隔 $(0, x_i]$ でソフトウェア故障の発生しない確率すなわちソフトウェア信頼度 $R(x_i)$ は

$$R(x_i)=\exp\left[-\int_0^{x_i} z(x)dx\right] \quad (x_i\geq 0) \qquad (3)$$

および i 番目のソフトウェアに対する平均故障時間間隔 $(MTBF)$ は

$$MTBF=\int_0^{\infty}[1-F(x_i)]dx_i \qquad (4)$$

によって計算できる．このような基本的な信頼性理論に基づいてさまざまなモデルが提案されている．

Jelinski and Moranda[21] は一定の故障率

$$z(x_i)=\phi[N-(i-1)] \quad (N>0, \phi>0) \qquad (5)$$

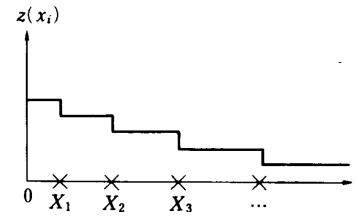
を仮定したモデルを提案した．Littlewood and Verrall[22] および Littlewood[23] はベイズ理論 (Bayesian Theory) に基づくベイズモデルを提案した．一方，Moranda[24] は故障率は一定で，幾何級数的に減少すると仮定して

$$z(x_i)=Dk^{i-1} \quad (D>0, 0<k<1) \qquad (6)$$

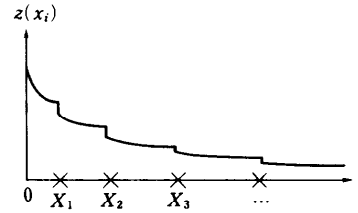
となる Geometric Purification Process モデルを提案した．**図-1** はこれらの三つのモデルの故障率 $z(x_i)$ の標本関数を示したものである．これらの図から分かるように故障率 $z(x_i)$ は不連続点があるので，不連続点のない滑らかな曲線で故障率曲線を仮定してモデル化したものが，**非定常ポアソン過程** (Nonhomogeneous Poisson Process) **モデル**である．**図-2** は典型的な非定常ポアソン過程モデルの故障率（強度関数）曲線とそれに対する平均値関数を示す．

2.2 非定常ポアソン過程

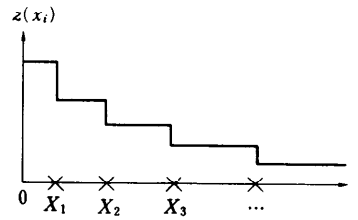
発見されたソフトウェアエラーを一つ二つと数えてゆく確率過程を計数過程 $\{N(t), t\geq 0\}$ という．ここで，$N(t)=k$ は時刻 t で k 個のソフトウェア故障が生起していることを示す．



(a) De-eutrophication process model[21]



(b) Bayesian model[22),23)]



(c) Geometric purification process model[24]

図-1 時間計測モデルの故障率 $z(x_i)$

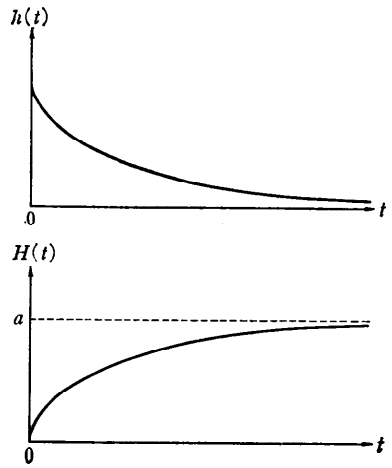


図-2 典型的な非定常ポアソン過程モデルの故障率 $h(t)$ と平均値関数 $H(t)$

計数過程 $\{N(t), t\geq 0\}$ は

(i) $N(0)=0$,

(ii) 過程は独立増分をもつ,

(iii) $P\{N(t+\Delta t)-N(t)=1\}=h(t)\Delta t+o(\Delta t)$,

(iv) $P\{N(t+\Delta t)-N(t)\geq 2\}=o(\Delta t)$,

を満たすならば，**強度関数** (intensity function) $h(t)$ をもつ非定常ポアソン過程とよばれる．ただし，(i) は $t=0$ でソフトウェア故障はない．(ii)は相異なる時間区間（増分）で発生するソフトウェア故障は統計的に独立である．(iii)微少時間区間 $(t, t+\Delta t]$ で一つの故障の発生する確率は $h(t)$ に比例する．(iv)微少時間区間で二つ以上の故障の発生する確率は無視できることを示している．ここで，非定常ポアソン過程は定常ポアソン過程 (Homogeneous Poisson Process) から定常増分の仮定を取り除いたものであり，非同次あるいは非斉次ポアソン過程 (Nonhomogeneous Poisson Process あるいは Nonstationary Poisson Process, 以下 NHPP と略す) ともよばれる．定常増分の仮定を取り除くことによって，ソフトウェア故障の発生を記述する強度関数 $h(t)$ は時刻 t の関数として一般化できる．強度関数 $h(t)$ はソフトウェア信頼度成長モデルにおいては単位時間当たりのエラー発見率を表す．

$N(t)$ の期待値 $H(t)=E[N(t)]$ は**平均値関数** (Mean Value Function) とよばれる．なぜなら，$N(t)=n$ となる確率は

$$P\{N(t)=n\}=\frac{[H(t)]^n}{n!}e^{-H(t)} \quad (n=0, 1, 2, \cdots) \tag{7}$$

となり，パラメータ $H(t)$ のポアソン分布に従う．ここで，

$$H(t)=\int_0^t h(x)dx \tag{8}$$

は式(7)のポアソン分布の平均になるからである（たとえば，Osaki[25] 参照）．

これまでに，$H(t)$ が実際のエラー発見過程によりよく適合するように工夫をこらし，さまざまな NHPP モデルが提案されている．これらのモデルについて，**2.5～2.9** に紹介する．テスト開始前に潜在する総期待エラー数は当然 $H(\infty)=a$ となり，そのとき $H(\infty)$ はパラメータ a の定常ポアソン分布に従う．時刻 t における期待残存エラー数は

$$n_r(t)=E[N(\infty)-N(t)]=a-H(t) \tag{9}$$

となる．時刻 t でソフトウェア故障が起こったという条件の下で，時間区間 $(t, t+x]$ においてソフトウェア故障の起こらない確率は

$$R(x|t)=\frac{P\{\text{時間区間 } (t, t+x] \text{ でソフトウェア故障なし}\}}{P\{\text{時刻 } t \text{ でソフトウェア故障生起}\}}$$
$$=\exp\{-[H(t+x)-H(t)]\} \quad (t\geq 0, x\geq 0) \tag{10}$$

となり，ソフトウェア信頼度とよばれる．NHPP モデルにおいては平均値関数 $H(t)$ (あるいは強度関数 $h(t)$) を決めれば，その確率的挙動は決まる．

2.3 指数形モデル

NHPP モデルの最も基本的なモデルは Goel and Okumoto[26] Goel[27] により提案された．その基本的な考え方は単位時間当たりに発見されるエラー数は，その時刻の残存エラー数に比例するとして，微分方程式

$$h(t)=\frac{dH(t)}{dt}=b[a-H(t)] \tag{11}$$

を導入した．ここで，a はテスト開始前に潜在する総期待エラー数であり，b はエラー発見率を表す比例定数である．$H(0)=0$ として，式(11)の微分方程式を解けば，

$$H(t)=a(1-e^{-bt}) \quad (a>0, b>0) \tag{12}$$
$$h(t)=abe^{-bt} \tag{13}$$

となる．

式(2)の故障率の考え方を一般化して，

$$d(t)=\frac{h(t)}{a-H(t)} \tag{14}$$

という指標[28]を導入する．$d(t)$ は時刻 t における残存エラー1個当たりの**エラー発見率**を表す．式(12)および(13)から，$d(t)=b$ となり，指数形モデルにおいては，任意時刻においてエラー1個当たりのエラー発見率は一定であることを示している．図-2 に $h(t)$ と $H(t)$ のグラフを示す．指数形モデルは物理的意味も簡単であり，後に述べるパラメータ推定も容易であり，アメリカの AT & T ベル研を中心に広く用いられている．われわれの経験では，大規模ソフトウェアによく適合する NHPP モデルである．

2.4 修正指数形モデル

前節で議論した指数形モデルでは，式(14)の $d(t)=b$ で一定であった．そこで，Yamada and Osaki[29], Yamada et al.[30] はテストにより発見容易なエラー（タイプ1エラー）と発見困難なエラー（タイプ2エラー）の2種類のエラーがあると仮定して，修正指数形 NHPP モデルを提案した．すなわち，

$$H(t)=a\sum_{i=1}^{2}p_i(1-e^{-b_it})$$
$$(a>0,\ 0<b_2<b_1<1,\ p_1+p_2=1,\ 0<p_i<1) \quad (15)$$

という平均値関数を導入した．ここで，b_i および p_i は各タイプ i のエラー発見率およびソフトウェア内のエラー含有率を表す $(i=1,2)$．時刻 t におけるエラー発見率および式(14)の1個当たりの発見率は

$$h(t)=a\sum_{i=1}^{2}p_ib_ie^{-b_it} \quad (16)$$

$$d(t)=\sum_{i=1}^{2}p_ib_ie^{-b_it}\Big/\sum_{i=1}^{2}p_ie^{-b_it} \quad (17)$$

となる．

図-3 は修正指数形モデルの $h(t)$, $H(t)$ および $d(t)$ のグラフを図示したものである．前小節の $d(t)$ と比較すると，$d(t)$ が単調に減少している様子がよく分かる．事実，

$$d(0)=p_1b_1+p_2b_2>d(\infty)=b_2 \quad (18)$$

であり，テスト時間の経過とともに，エラー発見が困難になってゆく様子を表している．

Ohba[31], Matsumoto et al.[32] は修正指数形モデルを一般化して，n 個のタイプ（あるいはクラスタ）のエラーの存在を仮定したモデルを提案した．現実には，後で述べるようにパラメータの数が多くなるとパラメータ推定が困難となる．その意味で，タイプの種類は少なくすることが現実のモデルを解析する場合には重要である．

2.5 遅延S字形モデル

2.3 で示した指数形モデルはその平均値関数曲線は指数形になっている（図-2 参照）．それに対して，**図-4** に示すようなS字形を示す平均値関数

$$H(t)=a[1-(1+bt)e^{-bt}] \quad (a>0,b>0) \quad (19)$$

をもつ NHPP モデルを遅延S字形モデルとよぶ．このモデルは山田・尾崎[44]，Yamada et al.[39]により提案された日本独自のモデルである．経験的に中小規模のソフトウェアの信頼性モデルによく適合すると言われている．

式(19)の $H(t)$ の満たす微分方程式

$$\frac{dH(t)}{dt}=b[a(1-e^{-bt})-H(t)] \quad (a>0,b>0) \quad (20)$$

の初期条件 $H(0)=0$ の解となる．式(20)はテストの実行過程がソフトウェア故障の現象を観測する過程（故障過程）とその原因解析を行ってエラーの発見に至る過程（エラー認知）という二つの過程からなると考えればよい．すなわち，$dH(t)/dt$ は故障として発見されていながらエラーとして認知されていないものに比例している．あるいは制御理論における2次遅れと考えることもできる．さらに，心理学における学習曲線（S字形）と考えてもよい．

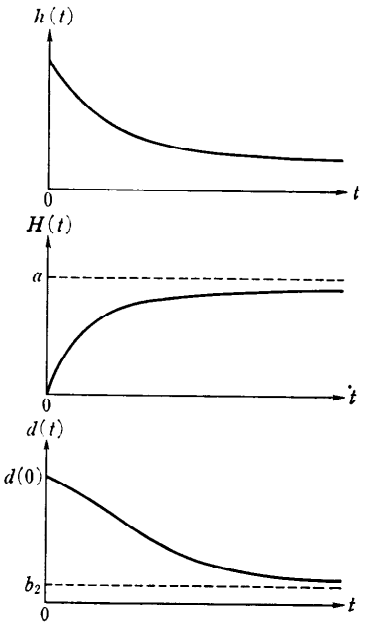


図-3 修正指数形モデルの $h(t)$, $H(t)$ および $d(t)$

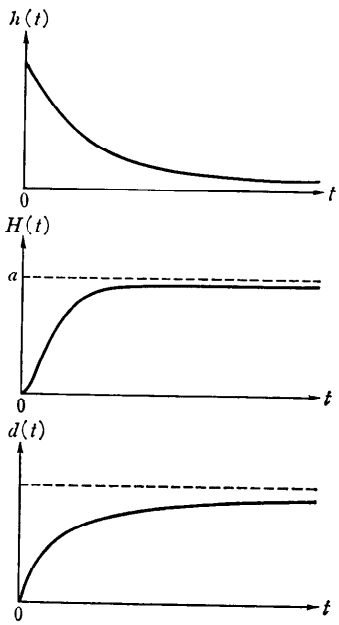


図-4 遅延S字形モデルの $h(t)$, $H(t)$ および $d(t)$

時刻 t におけるエラー発見率 $h(t)$ および1個当たりのエラー発見率 $d(t)$ は,

$$h(t)=ab^2te^{-bt} \tag{21}$$

$$d(t)=\frac{b^2t}{1+bt} \tag{22}$$

となる. **図-4** から分かるように, $d(t)$ のグラフは0から時間とともに増加して $d(\infty)=b$ に近づく. すなわち, 1個当たりのエラー発見率は時間とともに増加して, 最終的には b に近づく. 学習効果によって1個当たりのエラー発見率が徐々によくなることを示している.

式(9)および(10)の $n_r(t)$ および $R(x|t)$ はそれぞれ

$$n_r(t)=a(1+bt)e^{-bt} \tag{23}$$

$$R(x|t)=\exp\left[-a\{(1+bt)e^{-bt}-(1+b(t+x))e^{-b(t+x)}\}\right] \tag{24}$$

となる.

2.6 習熟S字形モデル

Ohba[33], 山田・大場[34]はテストチームのソフトウェアに対する習熟度を考慮した習熟S字形モデルを提案した. すなわち, 総期待ソフトウェア故障数 $H(t)$ は

$$\frac{dH(t)}{dt}=b\left\{r+(1-r)\frac{H(t)}{a}\right\}[a-H(t)] \qquad (a>0, b>0, 0<r\leq 1) \tag{25}$$

を満たす. ここで, 式(25)の右辺の $b\{r+(1-r)H(t)/a\}$ は $t=0$ で br, $t=\infty$ で b となる線形関数で, 習熟によりエラー発見率の向上を考慮している. r はエラー発見能力に関する習熟係数 $(0<r\leq 1)$ である. 式(25)の微分方程式を初期条件 $H(0)=0$ の下で解けば,

$$H(t)=\frac{a(1-e^{-bt})}{1+ce^{-bt}} \qquad (a>0, b>0, c>0) \tag{26}$$

となる. ただし, $c=(1-r)/r$ である. $c=0(r=1)$ ならば, 指数形モデルとなることは明らかである.

時刻 t におけるエラー発見率 $h(t)$ および1個当たりのエラー発見率 $d(t)$ は

$$h(t)=\frac{ab(1+c)e^{-bt}}{(1+ce^{-bt})^2} \tag{27}$$

$$d(t)=\frac{b}{1+ce^{-bt}} \tag{28}$$

となる. **図-5** は習熟S字形モデルの $h(t)$, $H(t)$ および $d(t)$ を示したものである. $d(t)$ は $d(0)=b/(1+c)$, $d(\infty)=b$ で時間とともにS字形の単調増加関数となっている. すなわち, 1個当たりのエラー発見率が増加して, b に近づいてゆくことを示している.

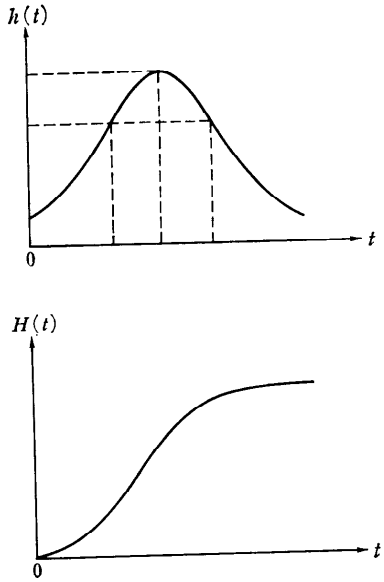


図-5 習熟S字形モデルの $h(t)$ と $H(t)$

2.7 テスト労力依存型モデル

ソフトウェア開発において, テスト労力の果たす役割は大きい. そこで, テスト労力の時間的な変動をワイブル曲線で記述しよう. ワイブル曲線はその特別な場合として指数曲線およびレイリー曲線を含み, 利用範囲の広い曲線である. 時刻 t におけるテスト労力の投入量を $w(t)$ として, テスト労力の投入量当たりのエラー発見率はその時刻における残存期待エラー数に比例すると仮定して, 微分方程式

$$\frac{\dfrac{dH(t)}{dt}}{w(t)}=r[a-H(t)] \qquad (a>0, r>0) \tag{29}$$

を得る. Yamada et al.[35], 山田[36]は式(29)の微分方程式を導入して, 初期条件 $H(0)=0$ の下で解いて

$$H(t)=a\{1-\exp[-r\cdot W(t)]\} \qquad (a>0, 0<r<1) \tag{30}$$

となる平均値関数をもつ NHPP モデルをテスト労力依存型モデルを提案した. ここで,

$$W(t)=\int_0^t w(x)dx \tag{31}$$

である. $W(t)$ はテスト時間区間 $(0, t]$ で投入された総テスト労力量を表す.

テスト労力関数 $w(t)$ および総テスト労力量をワイブル曲線と仮定すれば,

$$w(t)=\alpha\beta m t^{m-1}\exp(-\beta t^m) \quad (\alpha>0, \beta>0, m>0) \tag{32}$$

$$W(t)=\alpha[1-\exp(-\beta t^m)] \tag{33}$$

となる．ここで，αはテストに必要な総テスト労力量であり，βは尺度パラメータ，mは形状パラメータである．式(30)を用いて，最終的にテストにより発見される総期待エラー数は

$$H(\infty)=a(1-e^{-r\alpha})<a \tag{34}$$

となり，最終的に発見されないエラーが存在することを示している．

2.8 対数型実行時間モデル

Musa and Okumoto[37),38)] はエラー発見率は指数的に減少すると仮定して，微分方程式

$$\frac{dH(t)}{dt}=\lambda_0 e^{-\theta H(t)} \quad (\lambda_0>0, \theta>0) \tag{35}$$

を初期条件 $H(0)=0$ の下で解いて，

$$H(t)=\frac{1}{\theta}\ln(\lambda_0\theta t+1) \tag{36}$$

となるモデルを提案した．ここで，λ_0 は初期故障強度，θ は故障1個当たりの故障強度関数 $h(t)$ の減少率を表す[12)]．

故障強度関数 $h(t)$ は

$$h(t)=\lambda_0/(\lambda_0\theta t+1) \tag{37}$$

となる．**図-6** は対数形実行時間モデルの $h(t)$ および $H(t)$ を示す．式(36)から明らかなように，$H(\infty)=\infty$ となるから，このモデルは潜在エラー数が無限大となるモデルである．

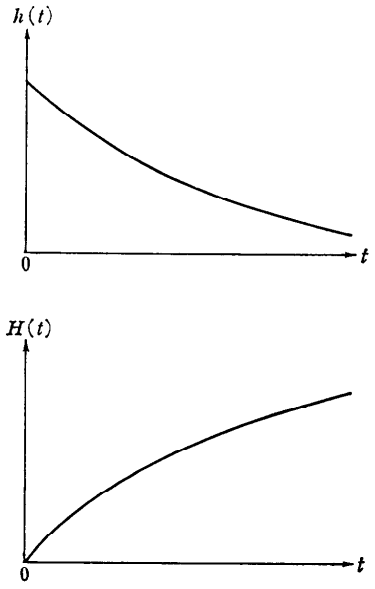


図-6 対数形実行時間モデルの $h(t)$ と $H(t)$

2.9 その他のモデル

前節までに6つの代表的な NHPP モデルを述べたが，このほかに多くの NHPP モデルが提案されている．しかし，それらのモデルの有用性の検証はこれからの課題である．特に，パラメータが多くなると後に述べるように，パラメータ推定が困難となる．さらに，モデルの直観的意味（本稿では主に微分方程式の物理的意味）が明白なモデルが望ましい．それらの点を考慮して指数形モデルあるいは遅延S字形モデルがよく用いられる．

以上述べた NHPP モデルはすべて連続時間 NHPP モデルであった．Yamada and Osaki[39)], Kitaoka et al.[40)] は離散時間 NHPP モデルについても議論した．ソフトウェアの発見過程をテストランによって数える離散時間 NHPP モデルも実用の観点から興味あるモデルである．

3. パラメータ推定

3.1 故障データと最尤推定法

前章で述べたいろいろな NHPP モデルについて，そのパラメータ推定について述べる．一般に，平均値関数 $H(t)$ はパラメータ a とその他の N 個のパラメータ w_i $(i=1, 2, \cdots, N)$ があるとする．ここで，$W=(w_1, \cdots, w_N)$ としよう．

故障データは n 個のソフトウェア故障生起時刻 s_k $(k=1, 2, \cdots, n)$ が与えられたとする．ここで，$\boldsymbol{s}=(s_1, s_2, \cdots, s_n)$ としよう．$\boldsymbol{s}$ が与えられたときの NHPP モデルの尤度関数（$\boldsymbol{s}$ の同時密度関数）は

$$L(a, \boldsymbol{w}|\boldsymbol{s})=\exp[-H(s_n)]\prod_{k=1}^{n}h(s_k) \tag{38}$$

となるから，この確率を最大にするようにパラメータ a と $\boldsymbol{w}$ を決める方法が最尤法である．式(38)の対数をとれば，

$$l(a, \boldsymbol{w}|\boldsymbol{s})=\sum_{k=1}^{n}\ln h(s_k)-H(s_n) \tag{39}$$

となる．$(n+1)$ 個のパラメータ a, $\boldsymbol{w}$ について微分すれば，

$$\frac{\partial l(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})}{\partial a}=\frac{\partial l(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})}{\partial w_i}=0 \quad (i=1, 2, \cdots, N) \tag{40}$$

となる $(N+1)$ 個の非線形連立方程式すなわち尤度方程式を得る．

次に，時間区間 $(0, t_k]$ $(k=1, 2, \cdots, n;\ 0<t_1<t_2<\cdots$

$<t_n$) で y_k 個のソフトウェア故障が観測されたとしよう．ここで，$\boldsymbol{t}=(t_1, t_2, \cdots, t_n)$ および $\boldsymbol{y}=(y_1, y_2, \cdots, y_n)$ とする．そのとき，$(\boldsymbol{t}, \boldsymbol{y})$ が与えられたときの NHPP モデルの尤度関数は

$$L(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})=\prod_{k=1}^{n}\frac{\{H(t_k)-H(t_{k-1})\}^{y_k-y_{k-1}}}{(y_k-y_{k-1})!}\cdot\exp\left[-\{H(t_n)-H(t_{k-1})\}\right] \tag{41}$$

となる．ここで，$t_0=0$, $y_0=0$ とする．式(41)の対数をとれば，

$$l(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})=\sum_{k=1}^{n}(y_k-y_{k-1})\ln\left[H(t_k)-H(t_{k-1})\right]-H(t_n)-\sum_{k=1}^{n}\ln\left[(y_k-y_{k-1})!\right] \tag{42}$$

となる．$(n+1)$ 個のパラメータ a, $\boldsymbol{w}$ について微分すれば，

$$\frac{\partial l(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})}{\partial a}=\frac{\partial l(a, \boldsymbol{w}|\boldsymbol{t}, \boldsymbol{y})}{\partial w_i}=0 \quad (i=1, 2, \cdots, N) \tag{43}$$

となる $(N+1)$ 個の非線形連立方程式すなわち尤度方程式を得る．

式(38)あるいは(41)の尤度関数が最尤になるようにパラメータを推定する方法が最尤法である．したがって，故障データ $\boldsymbol{s}$ が与えられた場合は式(40)の尤度方程式を解けばよい．同様に，故障データ $(\boldsymbol{t}, \boldsymbol{y})$ が与えられた場合には式(43)の尤度方程式を解けばよい．厳密には式(40)あるいは(43)の尤度方程式の解は最大となるための必要条件であるから，十分条件（全域で最大）になっていることも証明しなければならない．

前章の **2.3** から **2.7** で述べた5つの NHPP モデルに対する尤度方程式（連立非線形方程式）は表示できるが，ここでは省略する（山田[8]，表 5.3 および表 5.4 参照）．

3.2 数値計算法と実例

前節で示したように，式(40)あるいは(43)の尤度方程式を解けば，$(N+1)$ 個の最尤推定値 $\hat{a}$, $\hat{\boldsymbol{w}}$ を求めることができる．一般にパラメータは少ないほうが数値計算的にも楽である．たとえば，指数形モデルあるいは遅延S字形モデルはパラメータは a と b の二つであるので，数値計算は容易であり，その精度も高い．しかし，パラメータが三つ以上になると数値計算も決して容易ではない．

尤度方程式を解くための代表的な数値計算法としては，

- **Newton 法**（あるいは Newton-Raphson 法）
- **割線法**（あるいは二分法，はさみうち法）

がある．Newton 法は（連立）非線形方程式としてよく知られている．割線法は簡単なアルゴリズムによって計算できる．われわれの経験によれば，Newton 法よりも割線法のほうが使いやすく，収束も速い．いずれの方法も初期値を設定して数値計算しなければならない．初期値の設定がよければ，当然収束も速くなる．よりよい初期値を設定するためには，各モデルごとのパラメータのおおよその値についてあらかじめ分かっていれば，それを使えばよい．そのために，今までの経験を有効に活用すべきである．

次に尤度方程式の十分条件について考えてみよう．解析的に十分条件を証明するのは不可能であるので，数値的に証明することになる．その場合，極大値であることは証明できても十分でなく，全域で最大値であることを証明しなければならない．さらに，尤度方程式の解が多根である場合もあるので，一般に十分条件を証明するのは決して容易ではない．モデルが簡単でパラメータが少ないときは尤度方程式の解は十分条件になっている場合が多い．この点でも簡単なモデルは有効である．

故障データを用いた実例をあげる[41),42)]．10 組の実測データ (t_k, y_k) $(k=1, 2, \cdots, 10)$ が観測された．ここでは，遅延S字形モデルをこの故障データ $(\boldsymbol{t}, \boldsymbol{y})$ を用いてパラメータを推定すれば，$\hat{a}=37.9$, $\hat{b}=0.3118$

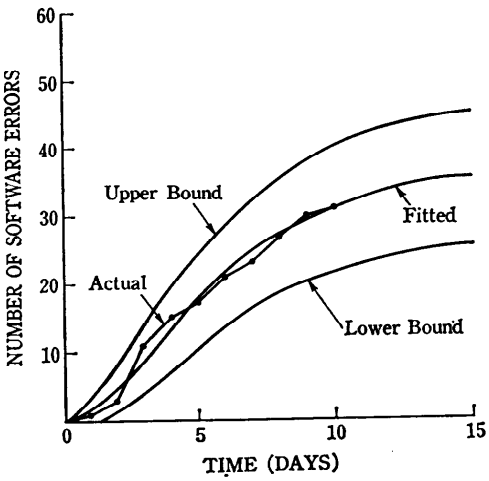


図-7 故障データ，平均値関数 $\hat{H}(t)$ および $\hat{H}(t)$ の 90% 信頼限界の実例（山田[8]，147 頁，図 5.4 より引用）

となるから，NHPP の平均値関数 $H(t)$ は

$$\hat{H}(t)=37.9[1-(1+0.3118t)e^{-0.3118t}] \quad (44)$$

となる．**図-7** は故障データ（折線），平均値関数 $\hat{H}(t)$ および $H(t)$ の 90% 信頼度限界

$$\hat{H}(t)\pm\kappa\sqrt{\hat{H}(t)} \quad (45)$$

を示したものである．ただし，$\kappa=1.64$ である．

3.3 モデルの適合度検定

NHPP モデルが故障データに適合しているかどうかを統計的に調べるために，適合度検定を行う．最もよく知られている適合度検定法は χ^2（カイ二乗）検定法である．χ^2 検定法は故障データが多ければよいが，故障データが少ないと精度が落ちる．データが少ない場合にはノンパラメトリックな検定法の一つであるコルモゴロフ・スミルノフ（Kolmogorov-Smirnov）適合度検定法を用いる（山田[8]，第5章参照）．

故障データに対する χ^2 統計量を求めるため，観測したテスト時間 T を m 個の区間に分割して

$$0=T_0<T_1<T_2<\cdots<T_{m-1}<T_m=T$$

とする．各時間区間 $(T_{i-1}, T_i]$ $(i=1,2,\cdots,m)$ でのソフトウェア故障の観測数 O_i と NHPP モデルより求めた期待度数（理論値）E_i を用いて，χ^2 統計量

$$\chi^2=\sum_{i=1}^{m}\frac{(O_i-E_i)^2}{E_i} \quad (46)$$

となる．自由度 $(m-k-1)$ で有意水準 α の χ^2 分布の棄却限界 $\chi_\alpha^2(m-k-1)$ の値を用いて

$$\chi^2<\chi_\alpha^2(m-k-1) \quad (47)$$

ならば，有意水準 α で故障データに対してモデルは適合している．ここで，k はパラメータの数を表し，χ^2 分布表は統計の数値表[43]に与えられる．

次にコルモゴロフ・スミルノフ検定量について述べよう．故障データ s に対して

$$D_i=\max\left\{\left|\frac{H(s_i)}{H(s_n)}-\frac{i}{n-1}\right|,\left|\frac{H(s_i)}{H(s_n)}-\frac{i-1}{n-1}\right|\right\} \quad (i=1,2,\cdots,n-1) \quad (48)$$

$$D=\max_{1\le i\le n-1}\{D_i\} \quad (49)$$

となる．同様に，故障データ $(\boldsymbol{t},\boldsymbol{y})$ に対して

$$D_i=\max\left\{\left|\frac{H(t_i)}{H(t_n)}-\frac{y_i}{y_n}\right|,\left|\frac{H(t_i)}{H(t_n)}-\frac{y_{i-1}}{y_n}\right|\right\} \quad (i=1,2,\cdots,n-1) \quad (50)$$

$$D=\max_{1\le i\le n-1}\{D_i\} \quad (51)$$

となる．有意水準 α で自由度 n および $(n-1)$ の棄却限界 $D_{n;\alpha}$ および $D_{n-1;\alpha}$ を用いて

$$D<D_{n;\alpha} \quad \text{あるいは} \quad D<D_{n-1;\alpha} \quad (52)$$

ならば，有意水準 α で故障データに対してモデルは適合している．ここで，$D_{n;\alpha}$ および $D_{n-1;\alpha}$ はコルモゴロフ・スミルノフ検定表は統計の数値書[43]に与えられる．

4. おわりに

NHPP を用いたソフトウェア信頼度成長モデルについて述べた．NHPP モデルの特長は平均値関数 $H(t)$ を定めることによっていろいろなモデルが提案できる．そのため，非常に数多くのモデルが提案されている．本稿では5つのモデルについて述べたが，そのほかにも有効なモデルもあるであろう．Musa et al.[12] はよいソフトウェア信頼度成長モデルは次の特長

1. 将来の故障挙動のよい予測を与える．
2. 有用な尺度が計算できる．
3. 簡単である．
4. 広く適用できる．
5. 適切な仮定に基づいている．

を備えていると言っている．これらの特長を備えた NHPP モデルは限られてくるであろう．

さらに，NHPP モデルを用いたソフトウェアの最適リリース問題も興味ある問題であるが，本稿では議論しない（山田[8]，第8章参照）．

最後に，本文について有益なご助言をいただいた本誌編集委員会各位に深甚の謝意を表する．

参考文献

1) Boehm, B. W.: *Software Engineering*, IEEE Trans. Computer, Vol. C-25, No. 12, pp. 1226-1241 (1976).
2) 平成元年度システムの高信頼化技術に関する調査研究（電子応用システム）成果報告書，日本規格協会 (1990).
3) 尾崎・山田：ソフトウェアの信頼性とフォールトトレランス，フォールト・トレラント・コンピューティング，当麻喜弘監修（向殿政男編），第7章，丸善，東京 (1989).
4) 向殿政男編：コンピュータシステムの高信頼化技術入門，日本規格協会 (1988).
5) 三浦 武：ソフトウェアの品質評価法，日科技連出版社，東京 (1981).
6) 菅野文友：ソフトウェア・エンジニアリング，日科技連出版社，東京 (1979).
7) 菅野文友（編集）：ソフトウェアの品質管理，日科技連出版社，東京 (1986).
8) 山田 茂：ソフトウェア信頼性評価技術，HBJ出版局，東京 (1989).

9) 山田・大寺：ソフトウェアの信頼性：理論とその実践的応用，ソフト・リサーチ・センター，東京 (1990).
10) 大場 充：ソフトウェアの開発技術，オーム社，東京 (1988).
11) Shooman, M. L.: *Software Engineering: Design, Reliability and Management*, McGraw-Hill, New York (1983).
12) Musa, J. D., Iannino, A. and Okumoto, K.: *Software Reliability-Measurement, Prediction, Application*, McGraw-Hill, New York (1987).
13) 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).
14) Dhillon, B. S.: *Microelectron. Reliab.*, Vol. 22, No. 3, pp. 625-640 (1982).
15) Shanthikumar, J. G.: Software Reliability Model: A Review, *Microelectron. Reliab.*, Vol. 23, No. 5, pp. 903-943 (1983).
16) Mellor, P.: Software Reliability Modeling: The State of the Art, *Information and Software Technology*, Vol. 29, No. 2, pp. 81-98 (1987).
17) 尾崎俊治：ソフトウェア信頼性の現状と動向，情報処理学会ソフトウェア工学研究会資料，62-2 (1988).
18) 当麻喜弘：ソフトウェアの信頼性，情報処理，Vol. 31, No. 4, pp. 508-517 (1990).
19) 当麻喜弘：ソフトウェア信頼性モデルと評価，電子情報通信学会誌，Vol. 73, No. 5, pp. 461-466 (1990).
20) 三道弘明：ソフトウェアの定量的信頼性評価法，システム/制御/情報，Vol. 34, No. 6, pp. 316-323 (1990).
21) Jelinski, Z. and Moranda, P. B.: Software Reliability Research, in *Statistical Computer Performance Evaluation*, Freiberger, W. (ed.), pp. 465-484, Academic Press, New York (1972).
22) Littlewood, B. and Verrall, J. L.: A Bayesian Reliability Growth Model for Computer Software, *Applied Statistics*, Vol. 22, No. 3, pp. 332-346 (1973).
23) 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).
24) Moranda, P. B.: Prediction of Software Reliability During Debugging, *Proc. 1975 Annual Reliability and Maintainability Symp.*, pp. 327-332 (1975).
25) Osaki, S.: *Stochastic System Reliability Modeling*, World Scientific, Singapore (1985).
26) 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).
27) Goel, A. L.: Software Error Detection Model with Application, *J. Systems and Software*, Vol. 1, pp. 243-249 (1980).
28) Yamada, S. and Osaki, S.: An Error Detection Rate Theory for Software Reliability Growth Models, *Trans. IECE Japan*, Vol. E 68, No. 5, pp. 292-296 (1985).
29) Yamada, S. and Osaki, S.: Nonhomogeneous Error Detection Rate Models for Software Reliability Growth, in *Stochastic Models in Reliability Theory*, Osaki, S. and Hatoyama, Y. (eds.), pp. 120-143, Springer-Verlag, Berlin (1984).
30) Yamada, S., Osaki, S. and Narihisa, H.: A Software Reliability Growth Model with Two Types of Errors, *R. A. I. R. O.-Operations Research*, Vol. 19, No. 1, pp. 87-104 (1985).
31) Ohba, M.: Software Reliability Analysis Models, *IBM J. R & D*, Vol. 28, No. 4, pp. 428-443 (1984).
32) Matsumoto, K., Inoue, K., Kikuno, T. and Torii, K.: Experimental Evaluation of Software Reliability Growth Models, *Proc. of 18th FTCS*, pp. 148-153 (1988).
33) Ohba, M.: Inflection S-Shaped Software Reliability Growth Model, in *Stochastic Models in Reliability Theory*, Osaki, S. and Hatayama, Y. (eds.), pp. 144-162, Springer-Verlag, Berlin (1984).
34) 山田・大場：エラー発見率に基づくS字形ソフトウェア信頼度成長モデルの考察，情報処理学会論文誌，Vol. 27, No. 8, pp. 821-828 (1986).
35) Yamada, S., Ohtera, H. and Narihisa, H.: Software Reliability Growth Models with Testing-Effort, *IEEE Trans. Reliab.*, Vol. R-35, No. 1, pp. 19-23 (1986).
36) 山田 茂：テスト労力の投入量を考慮したソフトウェア信頼度成長モデルと適合性比較，電子情報通信学会論文誌，Vol. J 70-D, No. 12, pp. 2438-2445 (1987).
37) Musa, J. D. and Okumoto, K.: A Logarithmic Poisson Execution Time Model for Software Reliability Measurement, *Proc. 7th Int. Conf. Software Eng.*, pp. 230-238 (1984).
38) Musa, J. D. and Okumoto, K.: Application of Basic and Logarithmic Poisson Execution Time Models in Software Reliability Measurement, in *Software System Design Methods*, Skwirzynski, J. K. (ed.) (NATO ASI Series, Vol. F 22), pp. 275-298, Springer-Verlag, Berlin (1986).

39) Yamada, S. and Osaki, S.: Discrete Software Reliability Growth Models, *Applied Stochostic Models and Data Analysis*, Vol. 1, No. 1, pp. 65-77 (1985).
40) Kitaoka, T., Yamada, S. and Osaki, S.: A Discrete Non-Homogeneous Error Detection Rate Model for Software Reliability, *Trans. IECE Japan*, Vol. E 69, No. 8, pp. 859-865 (1985).
41) 大場 充：プログラム・テストの妥当性評価に関する実験報告，情報処理学会ソフトウェア工学研究会資料，21-4 (1981年11月).
42) Ohba, M.: Software Quality=Test Coverage ×Test Accuracy, Proc. 6th Int. Conf. Software Engineering, pp. 287-293 (1982).
43) 山内二郎他編：簡約統計数値表，日本規格協会 (1972).
44) 山田・尾崎：ソフトウェアの信頼度成長モデルとその比較，電子通信学会論文誌，Vol. J 65 D, No. 7, pp. 906-912 (1982).

(平成2年7月6日受付)