潜入フォールトを考慮した不完全デバッグモデルと適合性評価

山田 茂† 三木貴史†

従来より，ソフトウェアの動的環境におけるソフトウェア故障発生現象あるいはフォールト発見事象を記述するために，多くのソフトウェア信頼度成長モデルが提案されてきた．しかし，これらの多くは，発生したソフトウェア故障の発生原因となったフォールトの修正作業は完全に行われるという完全デバッグの仮定を前提としていた．しかし，現実には，いつも確実に修正作業が行われるとは限らない．よって，本論文では，フォールトの修正時において新規フォールトが潜入するという不完全デバッグ環境を考慮したソフトウェア信頼度成長モデルを取り上げ，モデルの適合性比較と妥当性を考察する．特に，潜入フォールトによるソフトウェア故障の発生を，陽に記述するモデルと暗黙のうちに仮定するモデルとの両者について議論する．モデルの推定法としては，最尤法を取り上げる．

Imperfect Debugging Models with Introduced Software Faults and their Comparisons

SHIGERU YAMADA† and TAKASHI MIKI†

Assuming that the detected faults are perfectly removed by debugging without introducing new faults is considered to be unrealistic in software reliability modeling. In this paper, applying a nonhomogeneous Poisson process, we discuss several software reliability growth models considering the newly introduced faults to overcome this unrealistic assumption. These models explicitly or implicitly assume that new faults are randomly introduced when the detected faults are corrected and removed by imperfect debugging. In the models which explicitly recognize this assumption it is assumed that there are two types of the detected faults: the inherent faults originally latent in a software system before testing or operation and the faults introduced by imperfect debugging. The models discussed above are compared by some evaluation criteria in terms of goodness-of-fit to several cited data sets of actual fault detection during testing.

1. まえがき

ソフトウェアの開発者は，開発プロセスの全工程に対して確実な管理活動を実施していく必要がある．特に，ソフトウェアの品質管理において，出荷品質はソフトウェア開発プロセスの下流工程であるテスト工程あるいは実際の運用段階での，ソフトウェア内の潜在フォールト数やソフトウェア故障発生時間により評価できることから，その評価技術が重要となる．ここで，ソフトウェア故障（software failure）とは，ソフトウェアが期待どおりに動作しないことであり，ソフトウェア内に潜在する欠陥や誤りなどのソフトウェアフォールト（software fault）により引き起こされる．

そこで，ソフトウェア開発のテスト工程や実際の運用段階における，フォールト発見事象やソフトウェア故障発生現象を数学的にモデル化できれば，定量的品質/信頼性評価の際に便利である．このようにして，ソフトウェアの信頼性評価のために開発された数理モデルはソフトウェア信頼性モデル（software reliability model）[1)~5)]と呼ばれている．特に，ソフトウェア開発のテスト工程では，大量のテスト資源を投入してフォールトの発見と修正を行っているので，潜在しているフォールト数は時間が経過すれば減少していく．すなわち，テスト時間の経過とともにソフトウェア故障の発生する確率は減少し，その結果ソフトウェア故障の発生しない確率，言い換えればソフトウェア信頼度が増加していくことになる．そこで，このような現象や事象を記述するソフトウェア信頼性モデルは，ソフトウェア信頼度成長モデル（software reliability growth model）と呼ばれている．これまでに構築されてきたソフトウェア信頼度成長モデルの多くは，発見されたフォールトはすべて修正・除去され，修正時に新規フォールトは作り込まれないと仮定している．

† 鳥取大学工学部社会開発システム工学科
Department of Social Systems Engineering, Faculty of Engineering, Tottori University

この仮定は，非現実的なものと考えられ，実際にはいつも確実に修正作業が行われるとは限らず，新たなフォールトを作り込む場合が多い．これは，テスト工程あるいは実際の運用・保守段階は，不完全デバッグ（imperfect debugging）環境にあるということを意味している[6),7)]．

そこで，本論文ではフォールトの修正時において新規フォールトが潜入するという不完全デバッグ環境を考慮したソフトウェア信頼度成長モデルについて議論する．特に，潜入フォールトによるソフトウェア故障の発生を，陽に記述するモデルと暗黙のうちに仮定するモデルとの両者について議論する．前者のモデルにおいては，ソフトウェアの動的環境において発生するソフトウェア故障には，稼動前に潜在する固有フォールトに起因するものと，稼動中にランダムに潜入するフォールトに起因するものとの2種類があると仮定する．このフォールト発見事象に対して，非同次ポアソン過程（nonhomogeneous Poisson process，以下NHPPと略す）[8),9)]を導入してソフトウェア信頼度成長モデルを構築し，実測データを用いてモデルパラメータを最尤法により推定する．さらに，ソフトウェアの信頼性評価に有用な定量的評価尺度を導出し，実測データを用いて得られたモデルパラメータの推定結果より適用例を示すとともに，従来モデルとの適合性比較を行う．

2. 不完全デバッグモデルの記述

本論文では，ソフトウェアの動的実行環境，すなわちソフトウェア開発プロセスのテスト工程または実際の運用段階で発生するソフトウェア故障には，次の2種類があるものと仮定する[10)]．

(F1) 稼動開始以前にソフトウェア内に潜在するフォールト（固有フォールトと呼ぶ）により引き起こされるソフトウェア故障．

(F2) ソフトウェアの稼動中に不完全デバッグ等により潜入したフォールトにより引き起こされるソフトウェア故障．

また，1つのソフトウェア故障は1個のフォールトにより引き起こされるものとし，発生したソフトウェア故障の原因となるフォールトは，F1とF2のいずれであるか区別はできないものとする．F1のソフトウェア故障発生率は，稼動時間 t の経過とともに検出される固有フォールトの2種類の減少傾向を考えるため $a_i(t)\,(i=1,2)$ とする．また，F2のソフトウェア故障発生率は，その発生は稼動時間に関してランダムであり，一定の $\lambda(\lambda>0)$ により表す．したがって，F1およびF2のソフトウェア故障の発生現象を同時に考えるとき，稼動時間 t におけるソフトウェア故障発生率は

$$h_i(t)=\lambda+a_i(t)\quad(i=1,2) \tag{1}$$

により与えられる．式(1)より，時間区間 $(0,t]$ において発生する総期待ソフトウェア故障数（または発見される総期待フォールト数）は

$$\left.\begin{aligned} H_i(t)&=\lambda t+A_i(t)\\ A_i(t)&=\int_0^t a_i(x)dx\quad(i=1,2)\end{aligned}\right\} \tag{2}$$

となる．

そこで本論文では，式(1)の $h_i(t)$ を強度関数，式(2)の $H_i(t)$ を平均値関数とするNHPP

$$\Pr\{N(t)=n\}=\frac{\{H_i(t)\}^n}{n!}\exp[-H_i(t)]\quad(n=0,1,2,\cdots), \tag{3}$$

$$H_i(t)=\int_0^t h_i(x)dx \tag{4}$$

に基づくソフトウェア信頼度成長モデルを議論する $(i=1,2)$．ここで，$N(t)$ は時間区間 $(0,t]$ において発生する総ソフトウェア故障数を表す確率変数であり，$\Pr\{\cdot\}$ は確率を表す．式(3)および式(4)で与えられるNHPPモデルの強度関数 $h_i(t)\,(i=1,2)$ は，ランダムに起こる現象を記述するのに有用な同次ポアソン過程（homogeneous Poisson process）[8),9)]の強度関数 λ（一定）を $a_i(t)\,(i=1,2)$ に合成したものとなっている．

2.1 指数形ソフトウェア信頼度成長モデルに基づく不完全デバッグモデル

NHPPに基づく指数形ソフトウェア信頼度成長モデルは，単位時間あたりに発見されるフォールト数が，任意のテスト時刻においてソフトウェア内に残存するフォールト数に比例するものと仮定して平均値関数を記述するものである．NHPPの平均値関数および強度関数は，それぞれ次式で与えられる[11)]．

$$\left.\begin{aligned} m(t)&=a\left(1-e^{-bt}\right)\quad(a>0,b>0)\\ h_m(t)&\equiv\frac{dm(t)}{dt}=abe^{-bt}\end{aligned}\right\}. \tag{5}$$

ここで，a は初期内蔵固有フォールト数の期待値，b は固有フォールトに起因する1個あたりのソフトウェア故障率を表す．式(5)の $m(t)$ および $h_m(t)$ を式(2)に適用すると，不完全デバッグ環境におけるNHPPの平均値関数は

$$H_1(t)=\lambda t+a\left(1-e^{-bt}\right)\quad(\lambda>0,a>0,b>0) \tag{6}$$

となる．

2.2 遅延S字形ソフトウェア信頼度成長モデルに基づく不完全デバッグモデル

NHPPに基づく遅延S字形ソフトウェア信頼度成長モデルは，テスト工程において，ソフトウェア故障の現象を観測した後に，その原因解析を十分に行わなければフォールトの発見には至らないと仮定する．そこで，フォールト発見事象を，ソフトウェア故障の発生現象を観測するソフトウェア故障発見過程と，そのソフトウェア故障の原因解析を行ってフォールトの発見がなされるフォールト認知過程という2つの過程により記述する．ここで，時刻 t までに発見される総期待フォールト数を $M(t)$ とすると，NHPPの平均値関数および強度関数は，それぞれ次式で与えられる[12]．

$$\left.\begin{aligned} M(t) &= a\left[1-(1+bt)e^{-bt}\right] \\ h_M(t) &\equiv \frac{dM(t)}{dt} = ab^2te^{-bt} \\ &(a>0, b>0) \end{aligned}\right\}. \tag{7}$$

ここで，a は初期内蔵固有フォールト数の期待値，b は固有フォールトに起因する1個あたりのソフトウェア故障率を表す．式(7)の $M(t)$ および $h_M(t)$ を式(2)に適用すると，不完全デバッグ環境におけるNHPPの平均値関数は

$$H_2(t) = \lambda t + a\left[1-(1+bt)e^{-bt}\right] \quad (\lambda>0, a>0, b>0) \tag{8}$$

となる．

2.3 ワイブル過程モデル

このモデルは，累積ソフトウェア故障率（cumulative failure rate，総ソフトウェア故障数/総テスト時間）と総テスト時間との間には対数線形性が成り立つものと仮定し，任意のテスト時刻までに発生する総ソフトウェア故障数をNHPPにより記述している．総テスト時間 t に対して，NHPPに基づく計数過程 $\{N(t), t\geq 0\}$ を用いた累積故障率 $N(t)/t$ の平均値が，実証分析の結果，総テスト時間 t との間に対数線形性の関係を有することから，NHPPの平均値関数および強度関数は，それぞれ次式で与えられる[13]．

$$\left.\begin{aligned} \gamma(t) &= \alpha t^{\beta} \quad (\alpha>0, 0<\beta<1) \\ h_\gamma(t) &= \alpha\beta t^{\beta-1} \end{aligned}\right\}. \tag{9}$$

ここで，α および β は定数パラメータである．式(9)において，

$$\lim_{t\to\infty}\gamma(t) = \infty \tag{10}$$

となることから，最終的に発見されるフォールト数は無限大となり，固有フォールト以外に不完全デバッグ等による潜入フォールトの存在を暗黙のうちに認めていることになる．

2.4 対数型ポアソン実行時間モデル

このモデルもワイブル過程モデルと同様に，稼動中に潜入するフォールトに起因するソフトウェア故障発生現象を暗黙のうちに考慮している．実行時間で計測された時間区間 $(0,t]$ において発生した総期待ソフトウェア故障数を $\mu(t)$ とし，単位実行時間あたりに発生するソフトウェア故障数を表す強度関数 $h_\mu(t)$ が，$\mu(t)$ に関して指数関数的に減少するものと仮定する．このとき，NHPPの平均値関数および強度関数は，それぞれ次式で与えられる[14]．

$$\left.\begin{aligned} \mu(t) &= \frac{1}{\theta}\ln(\lambda_0\theta t+1) \\ h_\mu(t) &= \lambda_0/(\lambda_0\theta t+1) \\ &(\lambda_0>0, \theta>0) \end{aligned}\right\}. \tag{11}$$

ここで，パラメータ λ_0 は初期故障強度，θ はソフトウェア故障1個あたりの故障強度の減少率を表す．式(11)において，

$$\lim_{t\to\infty}\mu(t) = \infty \tag{12}$$

であり，ワイブル過程モデルと同様に潜入フォールトの存在を暗黙のうちに認めている．

3. 信頼性評価尺度とパラメータの推定

2章で議論したNHPPに基づくソフトウェア信頼度成長モデルから，以下のような定量的な信頼性評価尺度を導出することができる．ここで，平均値関数を $J(t)$，強度関数を $j(t)$ とする．

総稼動時間が $t\,(t\geq 0)$ のとき，時間区間 $(t, t+x]$ $(x\geq 0)$ においてソフトウェア故障の発生しない条件付き確率は，

$$\begin{aligned} R(x|t) &\equiv \Pr\{N(t+x)-N(t)=0|N(t)=n\} \\ &= \exp[J(t)-J(t+x)] \\ &\quad (t\geq 0, x\geq 0) \end{aligned} \tag{13}$$

により与えられ，ソフトウェア信頼度（software reliability）と呼ばれる．ここで，$\Pr\{A|B\}$ は，事象Bが生起したという条件のもとで，事象Aが生起する条件付き確率を表す．

ソフトウェア故障の発生率や発生頻度を表すハザードレート（hazard rate）は，

$$z(x|t) \equiv \frac{-\frac{d}{dx}R(x|t)}{R(x|t)} = j(t+x) \tag{14}$$

により与えられ，ソフトウェア故障の発生パターンを知ることができる．

式 (13) を用いて，総稼動時間が t のときの平均ソフトウェア故障発生時間間隔（mean time between software failures，以下 MTBF と略す）を導出することができ，

$$\mathrm{E}[X|t] = \int_0^\infty R(x|t)dx, \tag{15}$$

により与えられる．これは，条件付き MTBF となっている．

また，NHPP の強度関数の逆数

$$\mathrm{MTBF}_I(t) = \frac{1}{j(t)} \tag{16}$$

を瞬間 MTBF，さらに総テスト時間 t を平均値関数 $J(t)$ で割った

$$\mathrm{MTBF}_C(t) = \frac{t}{J(t)} \tag{17}$$

を累積 MTBF として求めることができる．

4. モデルパラメータの推定方法

2 章で議論したソフトウェア信頼度成長モデルのパラメータの推定方法について議論する．このとき，時間区間 $(0, t_k]$ において発生した総ソフトウェア故障数 y_k に関する n 組の実測データ $(t_k, y_k)\,(k = 1, 2, \cdots, n;\ 0 < t_1 < t_2 < \cdots < t_n)$ が観測されたものとする．

最尤法では，平均値関数 $J(t)$ を持つ NHPP モデルの尤度関数は，NHPP の性質より

$$L = \prod_{k=1}^{n} \frac{\{J(t_k) - J(t_{k-1})\}^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!} \cdot \exp[-\{J(t_k) - J(t_{k-1})\}] \tag{18}$$

となる．ここで，$t_0 = 0$，$y_0 = 0$ とする．式 (18) の両辺の自然対数をとると，対数尤度関数として

$$\ln L = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[J(t_k) - J(t_{k-1})] - J(t_n) - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!] \tag{19}$$

を得る．たとえば，式 (6) を式 (19) に代入すると

$$\ln L = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[\lambda(t_k - t_{k-1}) + a(e^{-bt_{k-1}} - e^{-bt_k})] - \lambda t_n - a(1 - e^{-bt_n}) - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!] \tag{20}$$

となる．したがって，このモデルの未知パラメータ λ，a，および b の最尤推定値 $\hat{\lambda}$，$\hat{a}$，および $\hat{b}$ は，

$$\partial \ln L/\partial a = \partial \ln L/\partial b = \partial \ln L/\partial \lambda = 0 \tag{21}$$

とおくことにより得られる同時尤度方程式を数値的に解くことにより与えられる．

5. 適合性評価基準 [5),15)]

今まで議論してきた 4 つの不完全デバッグモデルと，従来の NHPP に基づく式 (5) および式 (7) の不完全デバッグ環境を考慮しない指数形および遅延 S 字形ソフトウェア信頼度成長モデルの合わせて 6 つのモデルの実測データに対する適合性比較を行い，各モデルの実測データに対する適合性と特徴を考察する．適合性評価基準としては，以下のものを用いる．ここで，一定のテスト時刻 t_k までに発生した総ソフトウェア故障数 y_k に関する n 組のソフトウェア故障数データ $(t_k, y_k)\,(k = 1, 2, \cdots, n)$ が観測されているものとする．

- 平均偏差二乗和：平均偏差二乗和は，実測値と推定値との二乗誤差の総和をデータ数で平均化したものであり，観測データ数を n とすると

$$\mathrm{MSE} = \frac{1}{n} \sum_{k=1}^{n} (y_k - \hat{y}_k)^2 \tag{22}$$

により計算される．式 (22) の $\hat{y}_k$ は，テスト時刻が $t_k\,(k = 1, 2, \cdots, n)$ のときの各モデルの平均値関数の推定値である．平均偏差二乗和は，その値が小さいほど実測データによく適合していることを意味する．

- 対数尤度関数値：対数尤度関数値は，式 (18) で与えられるような実測データに対する同時確率密度関数（すなわち尤度関数）の自然対数をとったものであり，その値が大きいほど実測データによく適合している．
- AIC：AIC は，最尤法によってパラメータ推定を行ういくつかのモデルがあるときに，最適なモデルを選ぶ 1 つの規準となり，次式により与えられる．

$$\mathrm{AIC} = -2 \times (\text{モデルの最大対数尤度}) + 2 \times (\text{モデルの自由パラメータ数}).$$

AIC は，その値自体の大小よりも，AIC の値の差の大小に意味がある．各モデルの AIC の値を求めて，その差が 1〜2 程度以上あるなら AIC の値の差は有意と考えられ，値の小さい方のモデルの適合性が良いといえる．しかし，AIC の値の差が 1 よりも小さい場合は，それらのモデルの優劣の判断はできず，どちらも同程度であることを意味する．

- 予測相対誤差：予測相対誤差は，テスト終了時刻における累積フォールト数の予測値と実測値の相対誤差のことであり，次式で示される．

$$R_e = \frac{\hat{A}(t_q) - q}{q}. \tag{23}$$

ここで，t_q は総テスト時間，q は時間区間 $(0, t_q]$ において発見されたフォールト数の実測値，$\hat{A}(t_q)$ は任意のテスト時刻 t_e $(0 \leq t_e \leq t_q)$ までの実測データを使って求めたテスト終了時刻 t_q における平均値関数 $A(t)$ の推定値である．

- 相関係数：実測データと平均値関数による推定値との相関係数のことであり，その値が大きいほど適合性が良いことを表している．

6. 数値例および考察

本論文では，実測データとして，文献 16)~19) から引用した 12 セットの実際のテスト工程で観測されたフォールト発見数データを不完全デバッグ環境下にあったものとして，これらに各モデルを適用し，上記の適合性評価基準を用いて適合性比較を行う．なお，4 章で議論した各モデルパラメータの最尤推定値を求めるにあたり，数式処理システムの Mathematica[20] を用いた．まず，DS1 と呼ばれる 25 組の実測データ (t_k, y_k) $(k = 1, 2, \cdots, 25;\ t_{25} = 25$ (日), $y_{25} = 136)$ に，各 NHPP モデルを適用した場合について考える．なお，平均値関数 $H_1(t)$ を持つ NHPP モデルを，モデル H_1 と定義し，平均値関数 $\gamma(t)$ および $\mu(t)$ を持つ NHPP モデルを，それぞれモデル γ およびモデル μ と定義する．**表 1** には，各モデルのモデルパラメータの推定結果を示した．また，**図 1** には，DS1 に対する既存の指数形ソフトウェア信頼度成長モデル（式 (5) の $m(t)$ を平均値関数とする NHPP モデルのことで，モデル m と定義することにする）の推定結果と，$H_1(t)$ の推定値 $\hat{H}_1(t)$ を示した．$\hat{\lambda}$=2.7074 より，テスト工程で潜入したフォールトは $\hat{\lambda} \times 25 \simeq 68$ であると推定され，テスト工程で発見された固有フォールト数は $136 - 68 = 68$ と推定される．なお，図 1 から分かるように，実測データの成長曲線は指数関数形を示すため，平均値関数 $M(t)$ および $H_2(t)$ を持つ NHPP モデルを適用するのは妥当ではなく，実際にもモデルパラメータの値は求まらなかった．また，**図 2** には DS1 に対する $\gamma(t)$ の推定値 $\hat{\gamma}(t)$ と，$\mu(t)$ の推定値 $\hat{\mu}(t)$ を示した．図 1 および図 2 より，不完全デバッグ環境を考慮していない既存のモデルにおいては，発生するソフトウェア故障数（あるいは発見されるフォールト数）が収束傾向にあるのに対して，不完全デバッグ環境を考慮したモデルにおいては，発散傾向にあるということが分かる．また，**表 5** には DS1 に対する各モデルの適合性比較の結果を示した．表 5 よりすべての適合性評価基準に対して，モデル μ の適合性が良いことが分かる．さらに，**図 3** にはモデルの実測データに対する適合度を見るための予測相対誤差を示した．この図より，テスト進捗率 40% 以降，各モデルの推定値が安定していることが分かる．特にモデル μ においては，どの段階においても推定値が安定していることが分かる．また，DS5 と呼ばれる 15 組の実測データ (t_k, y_k) $(k = 1, 2, \cdots, 15;\ t_{15} = 15$ (日), $y_{15} = 1138)$ に対する同様の結果を**表 2**，**表 6**，**図 4** に示した．このデータについては，モデル H_1 の適合性が最も良いことが分かる．これらの適合性比較結果から考えると，

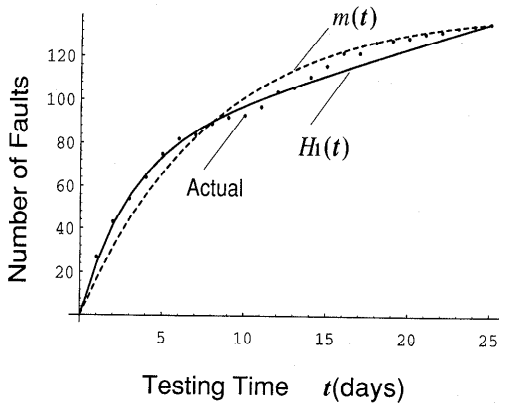


図 1 DS1 に対するモデル m およびモデル H_1 の推定された平均値関数

Fig. 1 The estimated mean value functions $\hat{m}(t)$ and $\hat{H}_1(t)$ for DS1.

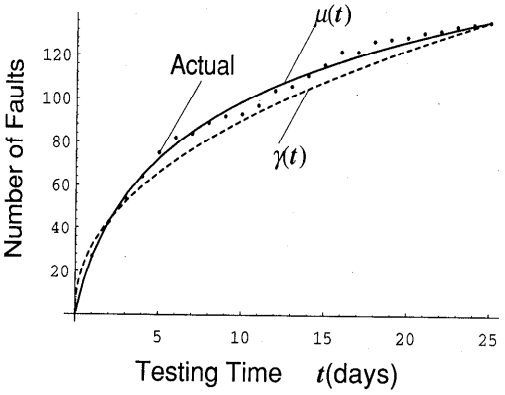


図 2 DS1 に対するモデル γ およびモデル μ の推定された平均値関数

Fig. 2 The estimated mean value functions $\hat{\gamma}(t)$ and $\hat{\mu}(t)$ for DS1.

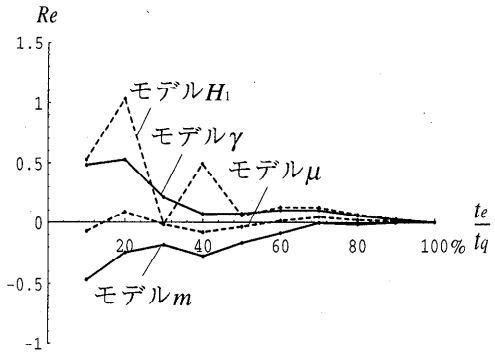


図3 DS1に対する予測相対誤差
Fig. 3 Predictive validity for DS1.

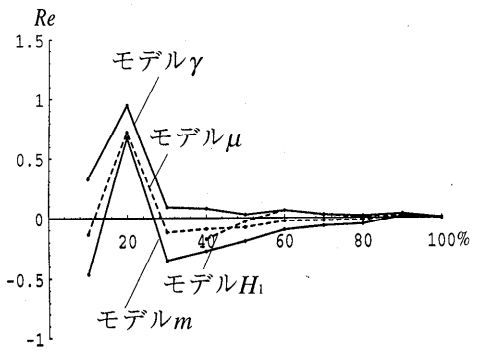


図4 DS5に対する予測相対誤差
Fig. 4 Predictive validity for DS5.

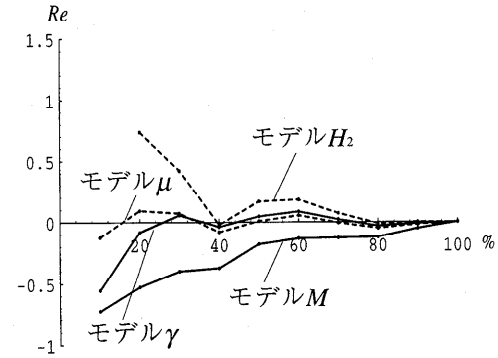


図5 DS10に対する予測相対誤差
Fig. 5 Predictive validity for DS10.

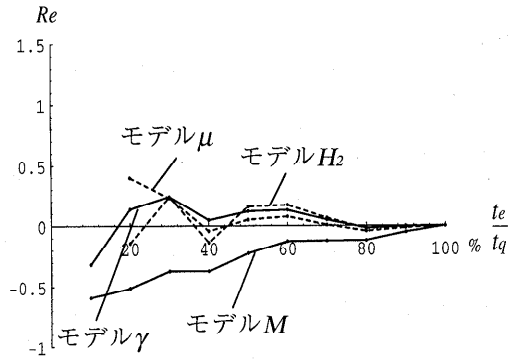


図6 DS11に対する予測相対誤差
Fig. 6 Predictive validity for DS11.

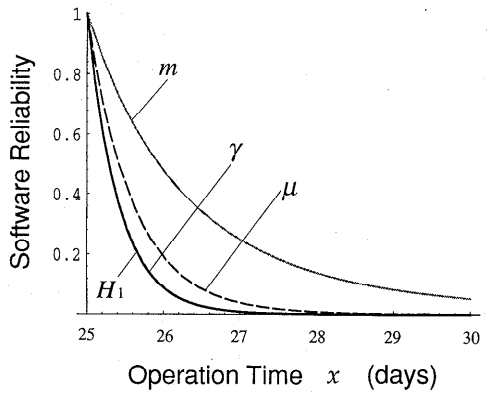


図7 DS1に対するソフトウェア信頼度
Fig. 7 The estimated software reliability functions for DS1.

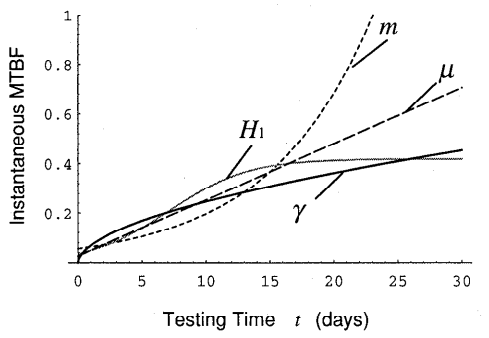


図8 DS1に対する瞬間MTBF
Fig. 8 The estimated instantaneous MTBF's for DS1.

従来のモデル，すなわちモデル m に比べて，不完全デバッグ環境を考慮したモデル，すなわちモデル H_1 および μ の実測データに対する適合性がかなり良いことが分かった．

次に，3章で議論した信頼性評価尺度について，実測データDS1に対する推定例を示す．テスト終了時点 t=25（日）以降の，各モデルのソフトウェア信頼度を**図7**に示す．図7より，運用を開始して1日後のソフトウェア信頼度を見ても分かるように，テスト工程での潜入フォールトを考慮すると，考慮しない場合に比べて，リリース後のソフトウェア信頼度は低下することが分かる．さらに，**図8**には瞬間MTBF，すな

表1 DS1 $(t_k, y_k)(k = 1, 2, \cdots, 25; t_{25} = 25, y_{25} = 136)$ に対するモデルパラメータの推定結果
Table 1 The estimated model parameters for DS1 (t_k, y_k) $(k = 1, 2, \cdots, 25; t_{25} = 25, y_{25} = 136)$.

	$\hat{a}$	$\hat{b}$	$\hat{\lambda}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\lambda_0}$	$\hat{\theta}$
m	142.315	0.1246					
H_1	76.1759	0.3302	2.3938				
γ				31.987	0.4496		
μ						36.661	0.0226

表2 DS5 $(t_k, y_k)(k = 1, 2, \cdots, 15; t_{15} = 15, y_{15} = 1138)$ に対するモデルパラメータの推定結果
Table 2 The estimated model parameters for DS5 (t_k, y_k) $(k = 1, 2, \cdots, 15; t_{15} = 15, y_{15} = 1138)$.

	$\hat{a}$	$\hat{b}$	$\hat{\lambda}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\lambda_0}$	$\hat{\theta}$
m	1347.84	0.124					
H_1	471.167	0.425	44.509				
γ				231.73	0.5877		
μ						237.93	0.0017

表3 DS10 $(t_k, y_k)(k = 1, 2, \cdots, 20; t_{20} = 100, y_{20} = 369)$ に対するモデルパラメータの推定結果
Table 3 The estimated model parameters for DS10 (t_k, y_k) $(k = 1, 2, \cdots, 20; t_{20} = 100, y_{20} = 369)$.

	$\hat{a}$	$\hat{b}$	$\hat{\lambda}$	$\hat{\alpha}$	β	$\hat{\lambda_0}$	$\hat{\theta}$
M	383.191	0.0511					
H_2	128.901	0.156	2.401				
γ				31.159	0.5367		
μ						10.824	0.0051

表4 DS11 $(t_k, y_k)(k = 1, 2, \cdots, 20; t_{20} = 100, y_{20} = 369)$ に対するモデルパラメータの推定結果
Table 4 The estimated model parameters for DS11 (t_k, y_k) $(k = 1, 2, \cdots, 20; t_{20} = 100, y_{20} = 369)$.

	$\hat{a}$	$\hat{b}$	$\hat{\lambda}$	$\hat{\alpha}$	β	λ_0	$\hat{\theta}$
M	384.414	0.0501					
H_2	137.788	0.131	2.3122				
γ				23.828	0.595		
μ						9.7452	0.0046

わち任意のテスト時刻における平均ソフトウェア故障発生時間間隔の推定値を示した．図8において，既存の指数形ソフトウェア信頼度成長モデルではテスト時間の経過とともに発散していくが，不完全デバッグモデルの場合は瞬間MTBFの成長が遅いことが分かる．

次に，実測データの成長曲線がS字形を示す場合について考える．ここでは，DS10と呼ばれる20組の実測データ (t_k, y_k) $(k = 1, 2, \cdots, 20;\ t_{20} = 100(\%),\ y_{20} = 369)$ に，各NHPPモデルを適用した場合について考える．なお，平均値関数 $M(t)$ を持つNHPPモデルをモデル M と定義し，平均値関数 $H_2(t)$ を持つNHPPモデルをモデル H_2 と定義する．**表3**には，各モデルのモデルパラメータの推定結果を示した．また，**表7**にはDS10に対する各モデルの適合性比較の推定結果を示した．表7より，すべての適合性評価基準に対して，モデル μ の適合性が良いことが分かる．さらに，**図5**には予測相対誤差を示した．図5より，テスト進捗率40%以降，各モデルの推定値が安定していることが分かる．特にモデル μ においては，どの段階においても推定値が安定していることが分かる．また，DS11と呼ばれる20組の実測データ (t_k, y_k) $(k = 1, 2, \cdots, 20;\ t_{20} = 100(\%),\ y_{20} = 369)$ に対する同様の結果を**表4**，**表8**，**図6**に示した．このデータについては，モデル H_2 の適合性がかなり良

表5 DS1に対する適合性比較結果
Table 5 Comparision among estimated models m, H_1, γ, and μ for DS1.

	モデルm	モデルH_1	モデルγ	モデルμ
平均偏差二乗和	35.5183	13.8601	39.1075	6.8729
対数尤度値	-57.219	-53.766	-55.358	-52.024
AIC	118.438	113.532	114.715	108.047
相関係数	0.9983	0.9995	0.9994	0.9997

表6 DS5に対する適合性比較結果
Table 6 Comparision among estimated models m, H_1, γ, and μ for DS5.

	モデルm	モデルH_1	モデルγ	モデルμ
平均偏差二乗和	1682.27	467.643	1044.35	741.865
対数尤度値	-138.87	-119.62	-130.97	-121.53
AIC	281.746	245.232	265.934	247.062
相関係数	0.9986	0.9996	0.9995	0.9994

表7 DS10に対する適合性比較結果
Table 7 Comparision among estimated models m, H_2, γ, and μ for DS10.

	モデルm	モデルH_1	モデルγ	モデルμ
平均偏差二乗和	829.496	97.6068	127.929	63.8506
対数尤度値	-148.6	-84.201	-90.455	-82.099
AIC	301.196	174.403	184.911	168.197
相関係数	0.9942	0.9993	0.9989	0.9995

表8 DS11に対する適合性比較結果
Table 8 Comparision among estimated models m, H_2, γ, and μ for DS11.

	モデルm	モデルH_1	モデルγ	モデルμ
平均偏差二乗和	688.315	74.7169	184.978	72.747
対数尤度値	-127.55	-77.791	-92.811	-80.72
AIC	259.09	161.583	191.622	165.411
相関係数	0.9951	0.9994	0.9986	0.9994

好であることが分かる．これらの適合性比較結果から考えると，従来のモデル，すなわちモデル M に比べて，不完全デバッグ環境を考慮したモデル，すなわちモデル H_2，および μ の実測データに対する適合性がかなり良いことが分かった．

以上のことより，ソフトウェア故障発生現象やフォールト発見事象を記述する際に，不完全デバッグ環境を前提とすれば，本論文で議論した不完全デバッグモデルを用いることは，十分有効であると考えられる．すなわち，固有フォールト（F1）と潜入フォールト（F2）を区別したいときには，モデル H_1 および H_2 により信頼性評価を行い，モデルの簡便性を重視しF1とF2の区別を必要としないときは，モデル γ および μ により信頼性評価を行えばよい．このとき，モデル γ については，適合性比較結果が良好でなかったことより，稼動中に潜入するフォールトに起因するソフトウェア故障発生現象を暗黙のうちに仮定しているモデルとしては，モデル μ を用いる方が妥当であるといえる．

7. むすび

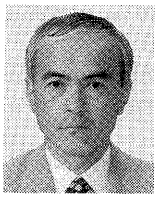
本論文では，フォールトの修正時に新規フォールトが潜入するという不完全デバッグ環境を考慮したソフトウェア信頼度成長モデルを取り上げ，特に潜入フォールトによるソフトウェア故障の発生を，陽に記述したモデルと暗黙のうちに仮定しているモデルの両者について議論した．また，これらのモデルと従来のモデルとの適合性比較を行い，不完全デバッグモデルの有効性も確かめた．

今後は，本モデルを用いて，より多くの適用例を示し，有効性を確認していく必要がある．

参考文献

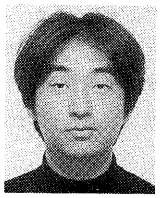
1) Malaiya, Y.K. and Srimani, P.K. (Eds.): *Software Reliability Models: Theoretical Developments, Evaluation and Applications*, IEEE Computer Society, Los Alamitos, CA (1991).
2) Musa, J.D., Iannino, A. and Okumoto, K.: *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York (1987).
3) Lyu, M.R. (Ed.): *Handbook of Software Reliability Engineering*, McGraw-Hill, New York (1995).
4) Xie, M.: *Software Reliability Modeling*, World Scientific, Singapore (1991).
5) 山田 茂：ソフトウェア信頼性モデル—基礎と応用，日科技連出版社 (1994).
6) Goel, A.L. and Okumoto, K.: A Markovian model for reliability and other performance measures of software systems, *Proc. National Computer Conf.*, pp.769–774 (1979).
7) Shooman, M.L.: *Software Engineering: Design, Reliability, and Management*, McGraw-Hill, New York (1983).
8) Ross, S.M.: *Introduction to Probability Models* (5th ed.), Academic Press, San Diego (1993).
9) 山田 茂：ソフトウェア信頼性評価技術—ソフトウェア信頼度成長モデル入門，HBJ出版局 (1989).
10) 山田 茂：潜入フォールトによる不完全デバッグを考慮したソフトウェア信頼度成長モデル，電子情報通信学会論文誌，Vol.J80-A, No.2, pp.363–370 (1997).
11) Goel, A.L. and Okumoto, K.: Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliability*, Vol.R-28, No.3, pp.206–211 (1979).
12) Yamada, S., Ohba, M. and Osaki, S.: S-shaped reliability growth modeling for software error detection, *IEEE Trans. Reliability*, Vol.R-32, No.5, pp.475–478, 484 (1983).
13) Crow, L.H.: On tracking reliability growth, *Proc. 1975 Annu. Reliability and Maintainability Symp.*, pp.438–443 (1975).
14) Musa, J.D. and Okumoto, K.: A logarithmic Poisson execution time model for software reliability measurement, *Proc. 7th Int. Conf. Software Engineering*, pp.230–238 (1984).
15) Iannino, A., Musa, J.D., Okumoto, K. and Littlewood, B.: Criteria for software reliability model comparisons, *IEEE Trans. Softw. Eng.*, Vol.SE-10, No.6, pp.687–691 (1984).
16) 山田 茂，大寺浩志：ソフトウェアの信頼性—理論と実践的応用，ソフト・リサーチ・センター (1990).
17) Misra, P.N.: Software reliability analysis, *IBM System Journal*, Vol.22, No.3, pp.262–270 (1983).
18) Brooks, W.D. and Motley, R.W.: Analysis of Discrete Software Reliability Models, Final Technical Report, RADC-TR-80–84, IBM Corporation (1980).
19) 上村松男：ソフトウェアの信頼度評価手法，インターフェース，Vol.14, No.2, pp.238–254 (1988).
20) Wolfram, S.: *Mathematica: A System for Doing Mathematics by Computer* (2nd ed.), Wolfram Research, Champaign, IL (1991).

(平成9年 6 月19日受付)
(平成9年11月 5 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．昭和58～63年岡山理科大学勤務．昭和63～平成5年広島大学工学部第二類（電気系）助教授．平成5年鳥取大学工学部社会開発システム工学科教授，現在に至る．工学博士．ソフトウェア信頼性工学，ソフトウェアマネジメントモデル，品質管理などの研究に従事．平成3年度情報処理学会Best Author賞，第8回電気通信普及財団賞（テレコムシステム技術賞）受賞．著書「ソフトウェア信頼性評価技術」（HBJ出版局），「ソフトウェアの信頼性～理論と実践的応用～」（ソフト・リサーチ・センター），「ソフトウェアマネジメントモデル入門」（共立出版），「ソフトウェア信頼性モデル—基礎と応用」（日科技連出版社）など．電子情報通信学会，日本OR学会，日本経営工学会，日本信頼性学会，日本応用数理学会，日本品質管理学会，IEEE各会員．



三木　貴史

昭和50年生．平成9年鳥取大学工学部社会開発システム工学科卒業．同年，同大学大学院工学研究科社会開発システム工学専攻入学，現在に至る．ソフトウェアの信頼性の研究に従事．電子情報通信学会会員．