ソフトウェア信頼性評価における
動的メトリクス加速寿命試験モデル

林　坂　弘 一 郎[*1]
ウォン　エリック[*2]

概要

本論文ではハードウェアの加速寿命試験の考え方に基づいたソフトウェア信頼性評価モデルを定式化する．またテスト期間中に観測され，テスト環境を特徴づけるテスト作業者数，テスト実行時間，投入テストケース数などの動的メトリクスデータを有機的に利用した動的メトリクス加速寿命試験 (DMALT) モデルを提案する．DMALT モデルではフォールト検出個数データと動的メトリクスデータから NHPP モデルのパラメータとテスト環境に関する係数パラメータを同時に最尤法によって推定できるようになる．数値例では実際のフォールトデータおよび動的メトリクスデータを利用した適合性評価によって DMALT モデルが高い適合性を有していることを示す．

キーワード：ソフトウェア信頼性モデル，DMALT モデル，加速寿命試験，メトリクスデータ，環境係数，チェンジポイント，最尤推定

1.　まえがき

今日の社会基盤の多くがソフトウェアによって支えられていることは周知のとおりであり，高品質なソフトウェアを開発することは極めて重要である．ソフトウェア開発の主体が人間であるため，人間のミスによってフォールトが埋め込まれることは避けられない．このためソフトウェア開発の最終段階であるソフトウェアテスト工程では可能な限りフォールトを検出し，それが修正される．さらにソフトウェアの品質を確保するためには，テスト工程で得られた

[*1] 本学経営学部准教授
[*2] テキサス大学ダラス校コンピュータサイエンス学科教授

フォールト検出に関するデータから当該ソフトウェアの信頼性を評価し，リリース可否の判断などの意思決定も必要となる．

ソフトウェアの信頼性を評価するための代表的なモデルが非同次ポアソン過程 (Non-Homogeneous Poisson Process; NHPP) モデルである．NHPP モデルは Goel and Okumoto [1] によって提案された指数形ソフトウェア信頼性モデルに始まり，遅延 S 字型ソフトウェア信頼性モデル [2] や一般化 NHPP ソフトウェア信頼性モデル [3] など，これまでに数多くのモデルが提案されている [4–9]．これらの多くのモデルはフォールト検出率を表す強度関数に連続的な関数を仮定している．しかしながらテスト期間中のテスト環境は必ずしも一定であるとは限らない．Yamada ら [10–13] はテスト環境を記述するテスト労力関数に基づいた NHPP モデルであるテスト労力依存型モデルを提案した．これはテスト工程において投入されるテスト労力をワイブル関数によって表現することで，テスト環境の変化を NHPP モデルに取り入れている．しかし，テスト労力依存型モデルでもテスト労力関数，強度関数はどちらも連続関数となる．

テスト期間中にはテスト作業者の配置転換など管理者の何らかの意思決定によってしばしばテスト環境が非連続的に変化する．このようなテスト環境の変化はチェンジポイントと呼ばれ，チェンジポイントを考慮したソフトウェア信頼性モデルもこれまでに提案されている [14–28]．多くのモデル [14–18] はチェンジポイント前後で障害発生率に関するパラメータが変化する NHPP モデルである．また Kapur ら [19] や Inoue and Yamada [20, 21] では複数のチェンジポイントが考慮されている．また Kapur ら [22]，Lin and Huang [23], Gupta ら [24], Huang [25] はテスト労力関数がチェンジポイント前後で変化するモデルを提案している．

一方で運用段階でのソフトウェア信頼性を評価するためのモデルも提案されている [26, 28–36]．岡村ら [36] はハードウェアに対する加速寿命試験 [37, 38] を NHPP ソフトウェア信頼性モデルに適用することで，ソフトウェアリリース後の運用段階における信頼性評価モデルを提案した．

また Inoue and Yamada [20, 21] のチェンジポイントモデルは岡村ら [36] の加速寿命試験モデルにおけるリリース時点をソフトウェアテスト中のチェンジポイントとみなして応用したモデルであると考えられる．このモデルでは，ソフトウェアフォールト検出時刻データのみからチェンジポイント前後の環境係数を推定しており，テスト期間中に観測されるメトリクスデータを利用できない．一方でテスト労力関数を用いるチェンジポイントモデル [22–25] ではメトリクスデータを利用できるものの，メトリクスデータからテスト労力関数のパラメータを最小二乗法によって推定した後，フォールトデータから NHPP モデルのパラメータを最尤法によって推定するという 2 段階の推定手法が採用されているため，推定値の統計的な意味合いが薄れ

てしまうという問題がある．

本論文では岡村ら [36] の運用段階でのソフトウェア信頼性モデルや Inoue and Yamada [20,21] のチェンジポイントモデルと同様の観点から，加速寿命試験に基づいたソフトウェア信頼性評価モデルを定式化する．さらにテスト期間中のテスト環境を特徴づける動的メトリクスデータを有機的に利用した動的メトリクス加速寿命試験 (Dynamic Metrics-based Accelerated Life Testing; DMALT) モデルを提案する．DMALT モデルではフォールト検出個数データと動的メトリクスデータから，NHPP モデルのパラメータとテスト環境に関する係数パラメータを同時に最尤法によって推定できるようになる．これによって AIC や BIC などの情報量基準によってモデルの適合性を評価できる．

本論文の構成は次のとおりである．**2.**では既存の NHPP ソフトウェア信頼性モデルと岡村ら [36] の加速寿命試験による運用段階信頼性モデルを概観する．**3.**ではテスト期間中の各期終了時点をチェンジポイントとして捉えることが可能な加速寿命試験型ソフトウェア信頼性モデルを定式化し，動的メトリクス加速寿命試験 (DMALT) モデルを提案するとともに，そのモデルパラメータ推定手法について議論する．**4.**では実際のソフトウェアフォールトおよびメトリクスデータを用いた数値例によって DMALT モデルの適合性を評価する．

2.　NHPP モデルと運用段階信頼性モデル

2.1　NHPP ソフトウェア信頼性モデル

ここでは NHPP ソフトウェア信頼性モデルについて概観する．ソフトウェアテストにおける時刻 t までに検出されたソフトウェアフォールト数を表す計数過程を $\{N(t);\ t \geq 0\}$ とし，$\lim_{t\to\infty} N(t) = N\ (> 0)$ をテスト開始前にソフトウェアに潜在する初期フォールト数とする．各ソフトウェアフォールトの検出時刻は互いに独立な同一分布に従い，分布関数を $F(t)$, 密度関数を $f(t) = dF(t)/dt$ とする．このとき，時刻 t までに検出されるフォールト数の確率関数は次の二項分布によって与えられる．

$$\Pr\{N(t) = m|N\} = \left(\begin{array}{c} N \\ m \end{array} \right) F(t)^m \overline{F}(t)^{N-m}, \quad m = 0, 1, 2, \cdots \tag{1}$$

ここで，$\overline{F}(\cdot) = 1 - F(\cdot)$ である．

初期フォールト数 N がパラメータ $\omega\ (> 0)$ のポアソン分布に従うとすると，時刻 t までのフォールト検出個数は

$$\Pr\{N(t) = m\} = \sum_{x=0}^{\infty} \Pr\{N(t) = m|x\} \frac{\omega^x}{x!} e^{-\omega}$$

$$= \frac{[\omega F(t)]^m}{m!} e^{-\omega F(t)}, \quad m = 0, 1, 2, \cdots \tag{2}$$

となる．式 (2) より，ソフトウェアフォールト検出個数の平均値関数は

$$H(t) = \mathrm{E}[N(t)] = \omega F(t) \tag{3}$$

となり，強度関数は

$$\lambda(t) = dH(t)/dt = \omega f(t) \tag{4}$$

となる．

上記の定式化は NHPP ソフトウェア信頼性モデルの一般的な表現となっており，フォールト検出時間分布 $F(t)$ に様々な分布を仮定することで既存の多くの NHPP ソフトウェア信頼性モデルを表現できる．例えば $F(t)$ に指数分布 $F(t) = 1 - \exp(-bt)$ $(b > 0)$ を仮定すれば Goel and Okumoto [1] の指数形ソフトウェア信頼性モデルに，次数 2 のアーラン分布 $F(t) = 1 - (1 + bt)\exp(-bt)$ $(b > 0)$ を仮定すれば Yamada ら [2] の遅延 S 字型ソフトウェア信頼性モデルに帰着される．さらに Goel [3] は $F(t)$ に形状パラメータ c (> 0) のワイブル分布 $F(t) = 1 - \exp(-bt^c)$ $(b > 0)$ を仮定した一般化 NHPP ソフトウェア信頼性モデルを提案した．なお，NHPP の独立増分の性質から明らかに

$$\Pr\{N(t + u) - N(t) = m\} = \frac{[H(t + u) - H(t)]^m}{m!} e^{-[H(t+u)-H(t)]}, \quad m = 0, 1, 2, \cdots \tag{5}$$

なる関係が成立する [39].

2.2　加速寿命試験による運用段階信頼性モデル

次に岡村ら [36] によって提案された加速寿命試験に基づいた運用段階の信頼性評価モデルについて概観する．

テスト段階での i $(= 1, 2, \cdots)$ 番目のフォールト検出時刻を X_i とし，その時間間隔 $S_i = X_i - X_{i-1}$ に対する時間分布を $\Pr\{S_i \leq t\} = K_i(t)$ とする．また，仮にテストを行うことなくリリースしたソフトウェアが存在し，その i 番目のフォールトが検出される時刻を Y_i，その時間間隔 $T_i = Y_i - Y_{i-1}$ に対する確率分布を $L_i(t)$ とする．ここで次の仮定を設定する．

$$Y_i = \alpha(X_i) \tag{6}$$

$$T_i = \alpha(S_i) \tag{7}$$

$$L_i(t) = K_i(\alpha(t)) \tag{8}$$

確率分布の逆関数 K_i^{-1} が存在すると仮定すれば，

$$\alpha(t) = K^{-1}L(t) \tag{9}$$

と記述することが可能であり，$\alpha(t)$ は加速関数と呼ばれる．岡村ら [36] は Barlow and Scheuer [37] と同様に $\alpha(t) = \alpha t$ $(\alpha > 0)$ を仮定し，α を環境係数と呼んだ．

ソフトウェアのテスト期間 $[0, \tau)$ におけるフォールト検出数の平均値関数を $H(t)$ とすると，リリース後の運用段階における検出フォールト数の平均値関数は次のとおりである．

$$H_O(t) = H\left(\tau + \frac{t-\tau}{\alpha}\right) - H(\tau) \tag{10}$$

ここで，τ はソフトウェアのリリース時刻であり，$\alpha > 1$ (< 1) はテスト環境がリリース後の運用環境よりも厳しい（緩い）ことを意味する．なお，岡村ら [36] では過去の同種のソフトウェア開発経験から環境係数を推定する方法が提案されている．

一方で Inoue and Yamada [21] は岡村ら [36] の加速寿命試験モデルを応用することでテスト期間中のチェンジポイント (CP) モデルを提案した．CP モデルは岡村ら [36] のモデルと本質的には類似したモデルであるが，フォールトターゲットの変更，テスト計画の変更など，テスト期間中におけるテスト管理者の何らかの行動をチェンジポイントとして捉え，チェンジポイント前後のテスト環境を環境係数としてモデル化している．また，テスト期間中に複数のチェンジポイントを設定でき，ソフトウェアのフォールト検出データからソフトウェア信頼性モデルのパラメータと環境係数を同時に推定している．しかしながら，CP モデルではソフトウェアフォールトデータのみからパラメータを推定するため，ソフトウェアテストにおいて観測可能なメトリクスデータを有効利用できないという問題点がある．

3.　動的メトリクス加速寿命試験 (DMALT) モデル

ここではテスト期間中の各期（1 日，1 週など）終了時点をチェンジポイントとして捉えることもできる加速寿命試験型ソフトウェア信頼性モデルを定式化する．さらにフォールト検出データだけでなくテスト期間中に観測可能な動的メトリクスを有機的に利用した動的メトリクス加速寿命試験 (DMALT) モデルを展開する．

テストの初期や運用段階など何らかの標準的なテスト（または運用）環境を想定し，その環境下における時刻 t での累積フォールト検出個数が平均値関数 $H(t) = \omega F(t)$ の NHPP に従うものと仮定する．また，ソフトウェアのテスト期間中の時間区間 $(t_{i-1},\ t_i]$ $(i = 1, 2, \cdots, n)$ に投入されたテスト労力から得られる環境係数を α_i (> 0), そのベクトルを $\boldsymbol{\alpha}^{\mathrm{T}} = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ とする．ただし，$t_0 = 0$ とする．

はじめに，テスト時刻 t_1 までに検出されるフォールト数の確率関数は，式 (2) および時間区間 $(t_0\ t_1]$ の環境係数 α_1 を用いて次のようなる．

$$\Pr\{N(t_1)=m\}=\frac{\left[H\left(t_1/\alpha_1;\boldsymbol{\theta}\right)\right]^m}{m!}\exp\left[-H\left(\frac{t_1}{\alpha_1};\boldsymbol{\theta}\right)\right],\quad m=0,1,2,\cdots \tag{11}$$

なお，$\boldsymbol{\theta}$ は NHPP モデル平均値関数 $H(t)$ のパラメータである．式 (11) より，時間区間 $(t_0,t_1]$ での平均検出フォールト数は $M_1=H(t_1/\alpha_1;\boldsymbol{\theta})$ である．

次に，時間区間 $(t_1,t_2]$ に検出されるフォールト数の確率関数は，式 (5) および環境係数を用いて

$$\begin{aligned}&\Pr\{N(t_2)-N(t_1)=m\}\\&=\frac{\left[H(t_1/\alpha_1+(t_2-t_1)/\alpha_2;\boldsymbol{\theta})-H(t_1/\alpha_1;\boldsymbol{\theta})\right]^m}{m!}\\&\quad\times\exp\left\{-\left[H\left(\frac{t_1}{\alpha_1}+\frac{t_2-t_1}{\alpha_2};\boldsymbol{\theta}\right)-H\left(\frac{t_1}{\alpha_1};\boldsymbol{\theta}\right)\right]\right\},\quad m=0,1,2,\cdots\end{aligned} \tag{12}$$

となる．すなわち，同時間区間の平均検出フォールト数は $M_2=H(t_1/\alpha_1+(t_2-t_1)/\alpha_2;\boldsymbol{\theta})-H(t_1/\alpha_1;\boldsymbol{\theta})$ によって与えられる．

一般に，時間区間 $(t_{i-1},t_i]$ に検出されるフォールト数の確率関数は，$i=1,2,\cdots,n$ に対して

$$\begin{aligned}&\Pr\{N(t_i)-N(t_{i-1})=m\}\\&=\frac{\left[H(A_{i,\boldsymbol{\alpha}};\boldsymbol{\theta})-H(A_{i-1,\boldsymbol{\alpha}};\boldsymbol{\theta})\right]^m}{m!}\\&\quad\times\exp\left\{-\left[H(A_{i,\boldsymbol{\alpha}};\boldsymbol{\theta})-H(A_{i-1,\boldsymbol{\alpha}};\boldsymbol{\theta})\right]\right\},\quad m=0,1,2,\cdots\end{aligned} \tag{13}$$

となる．ここで

$$A_{i,\boldsymbol{\alpha}}=\sum_{k=1}^{i}\frac{t_k-t_{k-1}}{\alpha_k},\quad i=0,1,2,\cdots,n \tag{14}$$

である．すなわち，時間区間 $(t_{i-1},t_i]$ に検出される平均フォールト数は

$$M_i=H(A_{i,\boldsymbol{\alpha}};\boldsymbol{\theta})-H(A_{i-1,\boldsymbol{\alpha}};\boldsymbol{\theta}) \tag{15}$$

である．したがって，ソフトウェアテストの開始時刻 $t_0=0$ から時刻 t_i までに検出される平均総フォールト数は

$$\begin{aligned}\Lambda(t_i;\boldsymbol{\theta},\boldsymbol{\alpha})&=\sum_{k=1}^{i}M_k\\&=\sum_{k=1}^{i}\left[H(A_{k,\boldsymbol{\alpha}};\boldsymbol{\theta})-H(A_{k-1,\boldsymbol{\alpha}};\boldsymbol{\theta})\right]\end{aligned}$$

$$= H(A_{i,\boldsymbol{\alpha}};\boldsymbol{\theta}), \quad i = 0, 1, 2, \cdots, n \tag{16}$$

の NHPP となる．

これまでに議論した環境係数は n 期のテスト期間に対して n 個存在することになり，すべてのパラメータを独立に推定することも考えられる．まずは，$s = 1$ や $s = n$ など基準となるテスト期間の環境係数を $\alpha_s = 1$ に固定して，その他のパラメータを推定すればよい．しかしながら，フォールト個数データではデータ数に対して推定パラメータ数が大きくなり，推定は不可能である．一方で，フォールト時間データでフォールト検出数がテスト期間 n よりも十分に大きければ推定は可能になる．しかし，その場合においても，自由度が大きくなりモデルの障害データへの適合度は高くなる反面，予測能力が低下することが予測される．したがって，環境係数に何らかの制約を置くことが考えられる．これには 2 通りの方法が考えられる．

ひとつはテスト期間中のテスト作業者数の増減などの事象が発生した時点をテスト環境のチェンジポイントとして捉え，環境係数に制約を追加する方法である．例えば，チェンジポイント発生時刻を t_c $(c = 1, 2, \cdots, n-1)$ として

$$\begin{aligned}
&\alpha_1 = \alpha_2 = \cdots = \alpha_c, \\
&\alpha_{c+1} = \alpha_{c+2} = \cdots = \alpha_n
\end{aligned}$$

なる制約を追加すればよい．また，l (> 1) 回のチェンジポイントが存在する場合も，チェンジポイントを $t_{c_1} < t_{c_2} < \cdots < t_{c_l}$ として

$$\begin{aligned}
&\alpha_1 = \alpha_2 = \cdots = \alpha_{c_1}, \\
&\alpha_{c_1+1} = \alpha_{c_1+2} = \cdots = \alpha_{c_2}, \\
&\vdots \\
&\alpha_{c_l+1} = \alpha_{c_l+2} = \cdots = \alpha_{c_n}
\end{aligned}$$

なる制約を追加すればよい．なお，l 個のチェンジポイントを仮定し $\alpha_1 = \cdots = \alpha_{c_1} = 1$ とすれば，Inoue and Yamada [21] らの CP モデルに帰着され，さらに，チェンジポイントの発生回数を 1 回に限定し，チェンジポイント以前をテスト期間として，チェンジポイント以降を運用期間として捉えれば，岡村ら [36] の運用段階信頼性モデルに帰着される．しかしながら，CP モデルではチェンジポイント数の増加に伴い推定パラメータ数も多くなるため，最尤推定においてパラメータの非負制約違反が発生しやすくなる．よって最尤推定での初期値の最適な選択が肝要になるとともに，推定にメトリクスデータを利用しないため，テスト環境の変化が直接的にはモデルに反映されることはない．

環境係数に制約を置くもうひとつの方法は環境係数がソフトウェアテスト工程のテスト環境を特徴づける動的メトリクスデータ (テスト時間, テスト作業者数, 投入テストケース数など) に依存 (比例) するとみなした定式化を行う方法である．いま, 時間区間 $(t_{i-1}, t_i]$ $(i = 1, 2, \cdots, n)$ において投入された d $(= 1, 2, \cdots)$ 種類のメトリクスデータ $\boldsymbol{z}_i = (z_{i1}, \cdots, z_{id})^{\mathrm{T}}$ が観測可能であるとする．

本論文では動的メトリクスデータを考慮した環境係数を

$$\frac{1}{\alpha_i} = g(\boldsymbol{z}_i; \boldsymbol{\beta}), \quad i = 1, 2, \cdots, n \tag{17}$$

と定義する．ここで $g(\boldsymbol{z}_i; \boldsymbol{\beta})$ は共変量関数，$\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d)^{\mathrm{T}}$ は係数ベクトルである．以下では比例ハザードモデル [38, 40] や比例強度モデル [41, 42] と同様に

$$g(\boldsymbol{z}_i; \boldsymbol{\beta}) = \exp(\boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{\beta}), \quad i = 1, 2, \cdots, n \tag{18}$$

を仮定する．このとき，DMALT モデルの平均値関数 $\Lambda_m(t_i; \boldsymbol{\theta}, \boldsymbol{\beta})$ は

$$\Lambda_m(t_i; \boldsymbol{\theta}, \boldsymbol{\beta}) = H(B_{i,\boldsymbol{\beta}}; \boldsymbol{\theta}), \quad i = 0, 1, 2, \cdots \tag{19}$$

と書くことができる．ここで

$$B_{i,\boldsymbol{\beta}} = \sum_{k=1}^{i} \exp(\boldsymbol{z}_i^{\mathrm{T}} \boldsymbol{\beta}) \, (t_k - t_{k-1}) \tag{20}$$

である．

式 (19) の DMALT モデルでは時間区間 $(t_{i-1}, t_i]$ の環境係数はその時間区間に投入されたメトリクス $\boldsymbol{z}_i$ に比例する．また，すべての j $(= 1, 2, \cdots, d)$ に対して $\beta_j = 0$ のとき，式 (19) のモデルは式 (3) の NHPP モデルに帰着することに注意を要する．さらに，式 (20) より，メトリクスデータに変化がない場合は同じテスト環境であることになり，メトリクスデータに変化があった時点がチェンジポイントになる．もちろんすべての期間でメトリクスデータに変化があっても推定パラメータ数が増加することはない．

DMALT モデルにおけるパラメータ $\boldsymbol{\theta}$ および $\boldsymbol{\beta}$ の推定には最尤法が利用できる．総テスト時間を t_n，テスト期間中の各時刻 t_i における累積検出フォールト数を y_i $(i = 1, 2, \cdots, n)$ とする．式 (19) の平均値関数に対する尤度関数 $L(\boldsymbol{\theta}, \boldsymbol{\beta})$ は

$$\begin{aligned} L(\boldsymbol{\theta}, \boldsymbol{\beta}) &= \Pr\{N(t_1) = y_1, N(t_2) = y_2, \cdots, N(t_n) = y_n\} \\ &= \exp\left[-\Lambda_m(t_n; \boldsymbol{\theta}, \boldsymbol{\beta})\right] \prod_{k=1}^{n} \frac{\left[\Lambda_m(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) - \Lambda_m(t_{k-1}; \boldsymbol{\theta}, \boldsymbol{\beta})\right]^{y_k - y_{k-1}}}{(y_k - y_{k-1})!} \end{aligned} \tag{21}$$

である．ここで $(t_0, y_0) = (0, 0)$，$z_{0j} = 0$ $(j = 1, 2, \cdots, d)$ である．式 (21) の両辺について対数をとれば次式の対数尤度関数が得られる．

$$\mathrm{LLF}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln \left[\Lambda_m(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) - \Lambda_m(t_{k-1}; \boldsymbol{\theta}, \boldsymbol{\beta})\right] - \Lambda_m(t_n; \boldsymbol{\theta}, \boldsymbol{\beta}) - \sum_{k=1}^{n} \ln \left[(y_k - y_{k-1})!\right] \tag{22}$$

したがって，式 (22) を最大にする最尤推定値 $\hat{\boldsymbol{\theta}}$，$\hat{\boldsymbol{\beta}}$ を準ニュートン法 [43] やネルダー・ミード法 [44] などの最適化アルゴリズムを利用して求めればよい．

4.　数値例

ここでは実際のソフトウェア開発プロジェクトのテスト工程において観測されたソフトウェアフォールトおよびメトリクスデータを用いて本論文で提案した DMALT モデルの適合性について評価する．評価にあたっては式 (22) の対数尤度 (LLF) のほかに，平均二乗誤差 (Mean Square Error; MSE)，赤池情報量基準 (Akaike's Information Criterion; AIC)，ベイズ情報量基準 (Bayesian Information Criterion; BIC) を利用する．ここで，

$$\mathrm{MSE} = \frac{1}{n} \sum_{k=1}^{n} \left[y_k - \Lambda_m(t_k; \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})\right]^2, \tag{23}$$

$$\mathrm{AIC} = -2\mathrm{LLF}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}) + 2\pi, \tag{24}$$

$$\mathrm{BIC} = -2\mathrm{LLF}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}) + \pi \ln \psi \tag{25}$$

であり，π はパラメータ数，ψ はデータ数である．LLF は大きいほど，MSE, AIC, BIC は小さいほど適合度が高いといえる．また以降では式 (3) で与えられる平均値関数の $F(t)$ に指数分布，2 次アーラン分布，およびワイブル分布の形状パラメータを $c = 2$ としたレイリー分布をそれぞれ適用することで得られる指数形ソフトウェア信頼性モデル [1], 遅延 S 字型ソフトウェア信頼性モデル [2]，レイリーソフトウェア信頼性モデル [3] を考える．

表 1にはフォールトおよびメトリクスデータの概要を示す．DS-T1 は Tohma ら [45] によって報告されたリアルタイム制御および監視システム用に開発されたソフトウェアのテストデータであり，フォールトデータとともにテスト作業者数のメトリクスデータが記録されている．DS-SS は Misra [46] によって報告されたスペースシャトルのソフトウェア開発プロエジェクトにおけるフォールト数およびテスト時間データである．DS-R1〜4 は Tandem Computer 社で

表 1　フォールト／メトリクスデータの概要

データセット	テスト期間	フォールト数	メトリクスデータ
DS-T1 [45]	111 days	481	Number of Test Workers
DS-SS [46]	38 weeks	231	Test Hours
DS-R1 [47]	20 weeks	100	Execution Hours
DS-R2 [47]	19 weeks	120	Execution Hours
DS-R3 [47]	12 weeks	61	Execution Hours & Test Cases
DS-R4 [47]	19 weeks	42	Execution Hours

表 2　適合性評価結果 (DS-T1)

DMALT	LLF	MSE	AIC	BIC
Exponential	**-298.6**	**213.8**	**603.1**	**613.3**
S-shaped	-307.9	347.3	621.7	631.9
Rayleigh	-347.1	853.3	700.2	710.4
NHPP	LLF	MSE	AIC	BIC
Exponential	-359.9	990.1	723.8	729.2
S-shaped	-320.0	340.6	644.0	649.4
Rayleigh	-350.7	725.0	705.5	710.9
CP	LLF	MSE	AIC	BIC
Exponential	-359.9	990.1	739.8	773.8
S-shaped	-320.0	340.6	660.0	694.1
Rayleigh	-352.9	2131.8	725.8	759.8

表 3　適合性評価結果 (DS-SS)

DMALT	LLF	MSE	AIC	BIC
Exponential	**-85.3**	**22.7**	**176.6**	**183.6**
S-shaped	-122.1	148.9	250.3	257.3
Rayleigh	-147.0	349.7	300.0	307.0
NHPP	LLF	MSE	AIC	BIC
Exponential	-98.4	43.6	200.8	204.0
S-shaped	-137.9	228.0	279.7	283.0
Rayleigh	-161.5	450.4	327.1	330.3

開発された 4 種類の異なるリーリースの障害データとその実行時間であり，Wood [47] によって報告された．なお，DS-R3 に関しては実行時間だけでなく，投入テストケース数もメトリクスデータとして記録されている．

DS-T1 では 111 日間のテスト期間中にテスト作業者数が 1 日目の 4 から 5, 6, 8, 4, 3, 4, 2, 1 へとそれぞれ 5, 27, 36, 47, 65, 72, 77, 100 日目終了時点で 8 回にわたって変更されている．このテスト作業者数の変更時点を CP モデルのチェンジポイントとして仮定し，α_i の制約として追加した．したがって，DS-T1 に対する CP モデルのパラメータ数は合計で 10 個となる．これに対して，NHPP モデルのパラメータ数は 2 個，DMALT モデルでは 3 個となる．

表 2から表 7には各データに対する適合性評価結果を示す．表 2より，DS-T1 に対しては指数分布を仮定した DMALT モデルが LLF, MSE, AIC, BIC のすべての評価基準において NHPP モデルや CP モデルよりも適合していることが読み取れる．図 1には DS-T1 に対する平均値関

表 4　適合性評価結果 (DS-R1)

DMALT	LLF	MSE	AIC	BIC
Exponential	**-41.0**	**10.3**	**87.9**	93.0
S-shaped	-53.2	27.9	112.4	117.5
Rayleigh	-61.7	39.8	129.3	134.4
NHPP	LLF	MSE	AIC	BIC
Exponential	-42.9	20.2	89.7	**91.7**
S-shaped	-53.3	28.4	110.5	112.5
Rayleigh	-62.9	53.2	129.8	131.8

表 5　適合性評価結果 (DS-R2)

DMALT	LLF	MSE	AIC	BIC
Exponential	**-41.2**	**12.1**	**88.4**	**93.3**
S-shaped	-48.2	13.6	102.3	107.2
Rayleigh	-55.5	34.1	117.1	122.0
NHPP	LLF	MSE	AIC	BIC
Exponential	-44.0	31.2	91.9	93.8
S-shaped	-48.2	14.1	100.5	102.3
Rayleigh	-55.8	36.8	115.5	117.4

表 6　適合性評価結果 (DS-R3)

DMALT(E&T)	LLF	MSE	AIC	BIC
Exponential	**-24.7**	6.2	57.4	62.1
S-shaped	-25.7	5.2	59.3	64.0
Rayleigh	-25.5	4.1	59.0	63.7
DMALT(E)	LLF	MSE	AIC	BIC
Exponential	**-24.7**	6.2	55.4	58.9
S-shaped	-25.7	5.2	57.3	60.8
Rayleigh	-25.5	4.1	57.0	60.5
DMALT(T)	LLF	MSE	AIC	BIC
Exponential	-26.8	15.7	59.7	63.2
S-shaped	-25.5	5.6	57.0	60.5
Rayleigh	-25.2	**3.2**	56.4	60.0
NHPP	LLF	MSE	AIC	BIC
Exponential	-29.8	27.2	63.6	64.5
S-shaped	-26.9	10.9	57.8	58.7
Rayleigh	-25.5	4.6	**55.1**	**56.0**

表 7　適合性評価結果 (DS-R4)

DMALT	LLF	MSE	AIC	BIC
Exponential	-33.2	3.2	72.5	77.4
S-shaped	-31.0	1.2	68.0	72.9
Rayleigh	**-30.7**	**0.9**	67.3	72.3
NHPP	LLF	MSE	AIC	BIC
Exponential	-33.9	6.0	71.7	73.6
S-shaped	-31.0	1.1	**66.0**	**67.9**
Rayleigh	-31.7	2.1	67.3	69.2

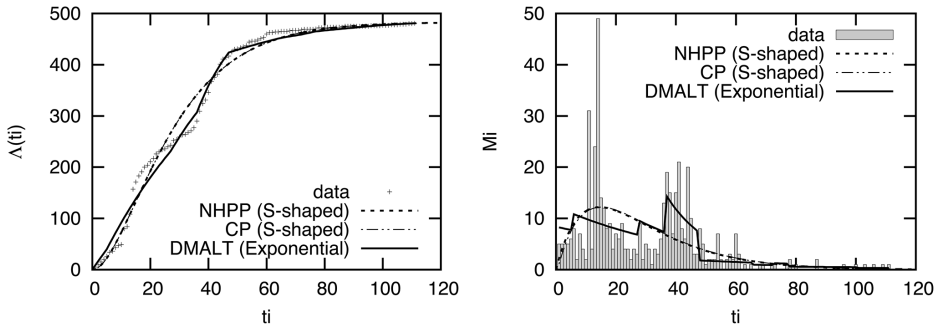

図 1　平均値関数と強度関数の振る舞い (DS-T1)

数と強度関数の振る舞いを示す．図 1に示すとおり，CP モデルは NHPP モデルと大きくは異ならない結果となっている．一方で DMALT モデルでは特にテスト作業者数に最大の 8 が投入された 37 日目から 47 日目に検出フォールト数が増加している様子を捉えることができている．表 3から表 5でも，表 4の BIC を除くすべての場合において指数分布を仮定した DMALT モデルが NHPP モデルよりもよく適合していることが読み取れる．

表 6の DS-R3 については 2 種類のメトリクスデータ (E, T) が観測されているため，DMALT モデルにおいては推定時にどのメトリクスの組み合わせを採用すべきかというモデル選択も必要となる．これには AIC または BIC を基準にすればよい．表 6より，Rayleigh 分布を仮定し，Test Cases (T) のメトリクスのみを採用した DMALT モデルの AIC, BIC が最も小さくなるため，このモデルを採用すればよい．このとき，AIC, BIC では NHPP モデルが，LLF と MSE では DMALT モデルの適合性が高いことがわかる．

さらに表 7の DS-R4 についても AIC, BIC 基準では NHPP モデルが，LLF と MSE 基準では DMALT モデルがよく適合していることが読みとれる．

以上のことから，多くのデータおよび評価基準において本論文で提案した DMALT モデルが高い適合度を有していることが示された．

5.　むすび

本論文では，加速寿命試験に基づいたソフトウェア信頼性評価モデルを定式化し，動的メトリクスデータを有機的に利用可能な動的メトリクス加速寿命試験 (DMALT) モデルを提案した．DMALT モデルではフォールト検出個数データだけでなく，テスト作業者数やテスト実行時間などの動的メトリクスデータも利用して最尤法によってモデルパラメータを推定可能である．これによって AIC や BIC などの情報量基準に基づいた適合性の評価が可能となった．また，実際のフォールト検出データを利用した適合性評価によって本論文で提案した DMALT モデルが従来の NHPP モデルや Inoue and Yamada [20,21] のチェンジポイントモデルよりも高い適合性を有していることを示した．

なお，テスト労力依存型モデルに基づいたチェンジポイントモデルはある時点までに得られたメトリクスデータからテスト労力関数を推定することで将来のテスト労力をも規定するのに対し，DMALT モデルはテスト期間中に観測されたメトリクスデータを直接利用するため，将来のフォールト検出事象を予測するためには将来に投入されるであろうテスト作業者数やテスト実行時間などのメトリクスデータの予測値が必要となる．将来のメトリクスデータとフォールト検出に関する予測精度を検証することは今後の課題である．

参考文献

[1] Goel, A. L. and Okumoto, K.: Time-dependent error-detection rate model for software reliability and other performance measures, *Reliability, IEEE Transactions on*, Vol. R-28, No. 3, pp. 206–211 (1979).

[2] Yamada, S., Ohba, M. and Osaki, S.: S-shaped reliability growth modeling for software error detection, *Reliability, IEEE Transactions on*, Vol. R-32, No. 5, pp. 475–484 (1983).

[3] Goel, A. L.: Software reliability models: Assumptions, limitations, and applicability, *Software Engineering, IEEE Transactions on*, Vol. SE-11, No. 12, pp. 1411–1423 (1985).

[4] Musa, J. D., Iannino, A. and Okumoto, K.: *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill (1987).

[5] Singpurwalla, N. D. and Wilson, S. P.: Software reliability modeling, *International Statistical Review*, Vol. 62, No. 3, pp. 289–317 (1994).

[6] Lyu, M. R.: *Handbook of Software Reliability Engineering*, Vol. 222, IEEE Computer Society Press (1996).

[7] Pham, H.: *Software Reliability*, Springer-Verlag (2000).

[8] Pham, H.: *Springer Handbook of Engineering Statistics*, Springer-Verlag (2006).

[9] Kumar, A.: Software reliability growth models, tools and data sets : A review, in *Proceedings of the 9th India Software Engineering Conference*, pp. 80–88, ACM (2016).

[10] Yamada, S., Ohtera, H. and Narihisa, H.: Software reliability growth models with testing-effort, *Reliability, IEEE Transactions on*, Vol. R-35, No. 1, pp. 19–23 (1986).

[11] Yamada, S., Ohtera, H. and Narihisa, H.: A testing-effort dependent reliability model for computer programs, *IEICE Transactions*, Vol. E69, No. 11, pp. 1217–1224 (1986).

[12] 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 D, Vol. J70-D, No. 12, pp. 2438–2445 (1987).

[13] Yamada, S., Hishitani, J. and Osaki, S.: Software-reliability growth with a Weibull test-effort: A model & application, *Reliability, IEEE Transactions on*, Vol. 42, No. 1, pp. 100–106 (1993).

[14] Chang, Y.-P.: Estimation of parameters for nonhomogeneous Poisson process: Soft-

ware reliability with change-point model, *Communications in Statistics – Simulation and Computation*, Vol. 30, No. 3, pp. 623–635 (2001).

[15] Shyur, H.-J.: A stochastic software reliability model with imperfect-debugging and change-point, *Journal of Systems and Software*, Vol. 66, No. 2, pp. 135–141 (2003).

[16] Zhao, J. and Wang, J.: Testing the existence of change-point in NHPP software reliability models, *Communications in Statistics – Simulation and Computation*, Vol. 36, No. 3, pp. 607–619 (2007).

[17] Zou, F.-Z.: A change-point perspective on the software failure process, *Software Testing, Verification and Reliability*, Vol. 13, No. 2, pp. 85–93 (2003).

[18] Kapur, P. K., Singh, V. B., Anand, S. and Yadavalli, V. S. S.: Software reliability growth model with change-point and effort control using a power function of the testing time, *International Journal of Production Research*, Vol. 46, No. 3, pp. 771–787 (2008).

[19] Kapur, P. K., Kumar, A., Yadav, K. and Khatri, S. K.: Software reliability growth modelling for errors of different severity using change point, *International Journal of Reliability, Quality and Safety Engineering*, Vol. 14, No. 4, pp. 311–326 (2007).

[20] Inoue, S. and Yamada, S.: Software reliability growth modeling frameworks with change of testing-environment, *International Journal of Reliability, Quality and Safety Engineering*, Vol. 18, No. 4, pp. 365–376 (2011).

[21] Inoue, S. and Yamada, S.: Software reliability assessment with multiple changes of testing-environment, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 98, No. 10, pp. 2031–2041 (2015).

[22] Kapur, P. K., Gupta, A., Shatnawi, O. and Yadavalli, V. S. S.: Testing effort control using flexible software reliability growth model with change point, *International Journal of Performability Engineering*, Vol. 2, No. 3, pp. 245–262 (2006).

[23] Lin, C.-T. and Huang, C.-Y.: Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models, *Journal of Systems and Software*, Vol. 81, No. 6, pp. 1025–1038 (2008).

[24] Gupta, A., Kapur, R. and Jha, P.: Considering testing efficiency in estimating software reliability based on testing variation dependent SRGM, *International Journal of Reliability, Quality and Safety Engineering*, Vol. 15, No. 2, pp. 77–81 (2008).

[25] Huang, C.-Y.: Performance analysis of software reliability growth models with testing-

effort and change-point, *Journal of Systems and Software*, Vol. 76, No. 2, pp. 181–194 (2005).

[26] Huang, C.-Y. and Lin, C.-T.: Reliability prediction and assessment of fielded software based on multiple change-point models, in *Dependable Computing, 2005. 11th Pacific Rim International Symposium on*, pp. 379–386, IEEE (2005).

[27] Huang, C.-Y., Kuo, S.-Y. and Lyu, M. R.: An assessment of testing-effort dependent software reliability growth models, *Reliability, IEEE Transactions on*, Vol. 56, No. 2, pp. 198–211 (2007).

[28] Huang, C.-Y. and Lyu, M. R.: Estimation and analysis of some generalized multiple change-point software reliability models, *Reliability, IEEE Transactions on*, Vol. 60, No. 2, pp. 498–514 (2011).

[29] 山田茂, 谷尾嘉一, 尾崎俊治：運用段階におけるソフトウェア信頼性評価に関する考察, 電子情報通信学会論文誌 D-I, Vol. J72-D-I, No. 11, pp. 797–803 (1989).

[30] Kenny, G. Q.: Estimating defects in commercial software during operational use, *Reliability, IEEE Transactions on*, Vol. 42, No. 1, pp. 107–115 (1993).

[31] Chillarege, R., Iyer, R. K., Laprie, J. and Musa, J. D.: Field failures and reliability in operation, in *Software Reliability Engineering, 1993. 4th International Symposium on*, pp. 122–126, IEEE (1993).

[32] Musa, J. D.: Sensitivity of field failure intensity to operational profile errors, in *Software Reliability Engineering, 1994. 5th International Symposium on*, pp. 334–337, IEEE (1994).

[33] Philip, T., Marinos, P. N., Trivedi, K. S. and Lala, J.: A multiphase software reliability model: from testing to operational phase, Technical Report TR-96-01, Center for Advanced Computing and Communication, Duke University (1996).

[34] Chen, Y.: Modelling software operational reliability via input domain-based reliability growth model, in *Fault-Tolerant Computing, 1998. 28th Annual International Symposium on*, pp. 314–323, IEEE (1998).

[35] Yang, B. and Xie, M.: A study of operational and testing reliability in software reliability analysis, *Reliability Engineering & System Safety*, Vol. 70, No. 3, pp. 323–329 (2000).

[36] 岡村寛之, 土肥正, 尾崎俊治：運用段階におけるソフトウェア製品の信頼性評価法：加速寿命試験モデルの提案, 電子情報通信学会論文誌 A, Vol. J83-A, No. 3, pp. 294–301 (2000).

[37] Barlow, R. E. and Scheuer, E. M.: Estimation from accelerated life tests, *Technometrics*, Vol. 13, No. 1, pp. 145–159 (1971).

[38] Bagdonavičius, V. and Nikulin, M.: *Accelerated Life Models: Modeling and Statistical Analysis*, Chapman and Hall/CRC (2001).

[39] Nakagawa, T.: *Stochastic Processes: with Applications to Reliability Theory*, Springer-Verlag (2011).

[40] Lawless, J. F.: Regression methods for Poisson process data, *Journal of the American Statistical Association*, Vol. 82, No. 399, pp. 808–815 (1987).

[41] Rinsaka, K., Shibata, K. and Dohi, T.: Proportional intensity-based software reliability modeling with time-dependent metrics, in *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, Vol. 1, pp. 369–376, IEEE (2006).

[42] 林坂弘一郎：比例強度ソフトウェア信頼性モデルに対するカーネル回帰, 神戸学院大学経営学論集, Vol. 9, No. 1, pp. 105–125 (2012).

[43] 茨木俊秀：最適化の数学, 共立出版 (2011).

[44] Nelder, J. A. and Mead, R.: A simplex method for function minimization, *Computer Journal*, Vol. 7, No. 4, pp. 308–313 (1965).

[45] Tohma, Y., Jacoby, R., Murata, Y. and Yamamoto, M.: Hyper-geometric distribution model to estimate the number of residual software faults, in *Computer Software and Applications Conference, 1989. COMPSAC 89. Annual International*, pp. 610–617, IEEE (1989).

[46] Misra, P. N.: Software reliability analysis, *IBM Systems Journal*, Vol. 22, No. 3, pp. 262–270 (1983).

[47] Wood, A.: Predicting software reliability, *Computer*, Vol. 29, No. 11, pp. 69–77 (1996).