統計数理(2025)
第 73 巻 第 2 号 147–166

特集「生存時間解析と信頼性解析」
[原著論文]

非一様マルコフ過程から見たソフトウェア信頼性モデリングの諸相

土肥 正[†]・岡村 寛之[†]

(受付 2024 年 11 月 30 日；改訂 2025 年 2 月 21 日；採択 2 月 27 日)

要　　旨

本稿では，ソフトウェア信頼性モデリングの進展を体系的に整理し，非一様マルコフ過程(NHMP)を基盤とした統一的なアプローチについて外観する．従来の一様マルコフ過程(Homogeneous Markov Process: HMP)および非一様ポアソン過程(Non-homogeneous Poisson Process: NHPP)モデルを一般化し，一般化 2 項過程(Generalized Binomial Process: GBP)や一般化ポリア過程(Generalized Polya Process: GPP)といった柔軟性の高い新しいモデルを紹介する．さらに，ソフトウェアのプロダクトおよびプロセスメトリクスを活用した信頼性予測モデルについても論じ，その有用性を検証する．これらの実証結果は，実際のソフトウェア開発における信頼性評価に寄与するものであり，将来のより精緻なモデル構築やソフトウェア信頼性評価へ適用可能性を広げる．

キーワード：ソフトウェア信頼性モデル，非一様マルコフ過程，一般化 2 項過程，一般化ポリア過程，ソフトウェアメトリクス，信頼性予測．

1. はじめに

過去 50 年間の長きにわたり，多くのソフトウェア信頼性モデルがソフトウェア工学の分野において開発されてきた（Lyu, 1996; Musa et al., 1987; Xie, 1991）．特に，一様マルコフ過程(Homogeneous Markov Process: HMP)に基づくソフトウェア信頼性モデルと非一様ポアソン過程(Non-homogeneous Poisson Process: NHPP)に基づくソフトウェア信頼性モデルは，テスト段階で検出されるソフトウェアフォールトの累積数の確率的挙動を記述する扱いやすさとフォールトデータへの適合性の良さから，これまでも広く利用されてきた．ソフトウェアフォールトデータは，フォールト検出時間(時間領域)データと時間間隔(グループ)データに分類される．時間領域データでは各イベントの発生時刻が記録されており，統計的完全データとみなされる．一方，グループデータは観測点ごとに発生したイベントの累積数を記録したもので，正確なイベント発生時刻が不明であるため，統計的不完全データとされる．実際，ソフトウェア開発のシステムテスト段階で観測されるフォールトデータの多くはグループデータであり，分散開発環境下でソフトウェアをテスト・デバッグすることが一般的である．

時間領域データを取り扱う際，基礎となるフォールト検出過程が広いクラスの単調非減少確率点過程に従う場合であっても，尤度関数を閉形式で求めることが比較的容易であり，最尤推

[†]広島大学 大学院先進理工系科学研究科：〒 739–8527 広島県東広島市鏡山 1–4–1; dohi@hiroshima-u.ac.jp, okamu@hiroshima-u.ac.jp

定やベイズ推定を含む統計的推論を行うことが可能となる．一方，グループデータを扱う際には，イベント累積数の確率法則を把握し，累積フォールト数に関する状態空間上で尤度関数を構築する必要がある．これにより，多くの推定可能なソフトウェア信頼性モデルは，いわゆるマルコフ性を持つモデルとして記述されてきた．例えば，Jelinski and Moranda (1972)，Moranda (1979)，Xie (1989)らによる古典的なソフトウェア信頼性モデルは，マルコフ純逆死滅過程およびマルコフ純出生過程として記述され，異なる状態依存遷移率をもつ HMP として分類される．一方，NHPP に基づくソフトウェア信頼性モデル（Abdel-Ghaly et al., 1986; Achcar et al., 1998; Erto et al., 2020; Goel and Okumoto, 1979; Littlewood, 1984; Goel, 1985; Gokhale and Trivedi, 1998; Musa and Okumoto, 1984; Ohba, 1984; Ohishi et al., 2009; Okamura et al., 2013; Xiao, 2015; Yamada et al., 1983; Zhao and Xie, 1996）では，状態とは独立で時間依存遷移率をもつマルコフ過程として分類される．

一方，ソフトウェア工学の分野では，上記ようなフォールトデータだけを扱うのではなく，ソフトウェアの開発中に観測可能な特徴量（共変量）に着目した研究も数多く行われている．ソフトウェアの特徴量としては，ソフトウェア規模，複雑性，品質，プロセス，環境などが挙げられ，それらはソフトウェアメトリクスと呼ばれる．これらのメトリクスは，ソフトウェアの信頼性に大きく影響を与える要因であり，ソフトウェアの信頼性予測において重要な役割を果たすと考えられている．例えば，ソフトウェア規模はソフトウェア信頼性に影響を与える主要な要因であり，ソフトウェアの規模が大きくなるとソフトウェアの信頼性が低下する傾向にあることは直感的に明らかである．このような外的要因を考慮したソフトウェア信頼性モデルは，ソフトウェアの信頼性予測においてより現実的な解を提供することが期待され，実証ソフトウェア工学における中心的な役割を演じると言っても過言ではない．Rinsaka et al. (2006)や Li et al. (2022)は Lawless (1987)の比例強度モデルを適用したメトリクスに基づくソフトウェア信頼性モデルを提案しており，Shibata et al. (2006)，Kuwa and Dohi (2013a)は Cox 比例ハザードモデルの枠組みを NHPP モデルに適用している．また，Okamura et al. (2010)，Okamura and Dohi (2014)，Ikemoto et al. (2013)，Kuwa and Dohi (2013b)，Okamura and Dohi (2015)はロジスティック回帰やポアソン回帰などの一般化線形モデルへの拡張を提案している．

本稿は二つの話題で構成される．一つ目は，HMP および NHPP に基づくソフトウェア信頼性モデルの一般化である非一様マルコフ過程(Non-homogeneous Markov Process: NHMP)に着目し，ソフトウェア信頼性予測および関連問題を扱うための統一的アプローチについて議論する．最近，Li et al. (2023)は NHMP の 2 つのサブクラスである一般化 2 項過程(Generalized Binomial Process: GBP)および一般化ポリア過程(Generalized Polya Process: GPP)に注目し，GBP と GPP がそれぞれ状態と時間に依存した遷移率を持つマルコフ逆死滅過程(Shanthikumar, 1981)とマルコフ出生過程 (Kendall, 1948; Shaked et al., 2002)となり，既存のソフトウェア信頼性モデルの多くを含む汎用的モデルであることを示した．過去の研究において，Shanthikumar (1981)は Goel and Okumoto (1979)による最も古典的な指数形 NHPP モデルと Jelinski and Moranda (1972)による HMP モデルによって記述されたソフトウェア信頼性モデルが GBP モデルに含まれる事実を示しているが，NHMP モデルの汎用性や柔軟性について理論的にも実証的にも十分な議論がなされていなかった．本稿の前半では，GBP および GPP に基づく統一的なモデリングの枠組みを紹介し，ソフトウェア信頼性モデルにおけるそれらの特性を概観するとともに，実際のソフトウェアフォールトデータを用いて適合性と予測性能を調査する．具体的には，NHPP，GBP，GPP によって記述されるソフトウェア信頼性モデルを比較し，それらの適合性および予測性能を評価する．特に，Okamura and Dohi (2008)で扱われた NHPP を一般化した非一様な混合ポアソン過程(Non-homogeneous Mixed Poisson Process: NHMPP)に関する新たな知見として，単一プロジェクトデータのみを用いたソフト

ウェア信頼性評価において，NHMPP は NHPP よりも常に適合性の面で劣ることが確認された．本稿の後半では，ソフトウェアメトリクスに着目し，ソフトウェアの信頼性予測におけるメトリクス活用の重要性について議論する．特に，Okamura and Dohi (2015)によって導入された一般化線形モデル(GLM)に基づくソフトウェア信頼性モデルを紹介し，文献 Okamura and Dohi (2015)で取り扱ったデータを含むより多くのデータを用いた検証を行い，ソフトウェア信頼性予測におけるメトリクスモデルの有用性を示す．

2. ソフトウェア信頼性の基本モデル

2.1 HMP によるモデリング

ソフトウェアテストの最終段階であるシステムテストにおいて，時刻 t までに検出されたフォールト数を $\{N(t), t \geq 0\}$ とする．いま，システムテストにおけるフォールト検出過程に対して，次の仮定を設ける．

仮定 A. システムテスト実施前にプログラム中に含まれる総フォールト数 N は一定かつ有限である．

仮定 B. 各々のフォールトは互いに独立に検出され，フォールト検出時間は同一の連続形確率分布関数 $F(t)$ をもつ非負の確率変数である．検出されたフォールトは直ちに修正・除去される．

仮定 A と仮定 B の下で，時刻 t までに検出される累積フォールト数は，以下の 2 項分布

$$P(N(t) = k \mid N) = \binom{N}{k} F(t)^k (1 - F(t))^{N-k} \tag{2.1}$$

によって記述される．このとき，確率過程 $N(t)$ は連続時間離散状態マルコフ過程となり，特に時刻 t においてソフトウェアに内在する残存フォールト数 $N - N(t)$ は状態数 0 で吸収状態となる死滅過程($N(t)$ は状態数 N で吸収状態となる出生過程)と等価となる．

いま，マルコフ過程の定常推移確率を

$$P_k(t) = \Pr\{N(t) = k | N(0) = 0\}, \quad k = 0, 1, 2, \ldots \tag{2.2}$$

とし，各状態への推移率を λ_k $(k = 0, 1, 2, \ldots)$ のように表す．ソフトウェア信頼性で最も初期のモデルである Jelinski and Moranda モデル（Jelinski and Moranda, 1972)では，フォールト検出時間分布にパラメータ b (> 0) の指数分布 $F(t) = 1 - \exp(-bt)$ を仮定しており，確率過程 $N(t)$ は時間に関して一定の推移率をもつ連続時間マルコフ連鎖に帰着される．この連続時間マルコフ連鎖の推移率は $\lambda_k = (N - k)b$ $(k = 0, 1, \ldots, N - 1)$ によって与えられる．この事実は，定常推移確率に関するコルモゴロフの前進方程式(微分差分方程式)

$$\frac{\mathrm{d}}{\mathrm{d}t} P_0(t) = -\lambda_0 P_0(t), \tag{2.3}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} P_k(t) = \lambda_{k-1} P_{k-1}(t) - \lambda_k P_k(t), \; k = 1, 2, \ldots, N - 1, \tag{2.4}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} P_N(t) = \lambda_{N-1} P_{N-1}(t) \tag{2.5}$$

を初期条件 $P_0(0) = 1$, $P_k(0) = 0$ $(k = 1, \ldots, N)$ の下で解くことにより容易に確認できる．すなわち，

$$P_k(t) = P(N(t) = k \mid N) = \binom{N}{k} \{1 - e^{-bt}\}^k e^{-b(N-k)t}, \quad k = 0, 1, \ldots, N \tag{2.6}$$

となる．

仮定 A と仮定 B とは離れて，オープンソースソフトウェアのようにシステムの運用期間中においてもなお，フォールトの作りこみが発生するような状況を考える．一般の推移率 λ_k が $\sum_{k=0}^{\infty}\lambda_k^{-1}=\infty$ を満たすとき，定常推移確率は

$$P_k(t)=\lambda_{k-1}e^{-\lambda_k t}\int_0^t e^{\lambda_k x}P_{k-1}(x)dx \tag{2.7}$$

を満足する純出生過程となる．Moranda（1979）は推移率を $\lambda_k=ab^k$ $(k=0,1,\ldots)$ のように仮定し，geometric de-eutrophication モデルと呼ばれる異なる HMP モデルを提案しており，Boland and Singh（2003）はその定常推移確率 $P_k(t)$ の無限級数表現を与えている．

2.2 NHPP によるモデリング

システムテストの長さ如何にかかわらず，テスト前にソフトウェア内に残存する初期フォールト数 N は未知であるので，N を非負の整数値確率変数とみなすことが可能である．すなわち，N の（混合比分布と呼ばれる）確率関数を $P(N=n)=\pi_N(n|\cdot)$ とすれば，$N(t)$ の確率関数は

$$P(N(t)=k)=\sum_{n=k}^{\infty}P(N(t)=k|n)\pi_N(n|\cdot) \tag{2.8}$$

のような混合分布によって与えられる．特別な場合として，

$$\pi_N(n|\omega)=\frac{\omega^n e^{-\omega}}{n!} \tag{2.9}$$

のようなパラメータ ω (>0) のポアソン分布を仮定すれば，確率過程 $N(t)$ の確率関数は

$$P(N(t)=k)=\frac{(\omega F(t))^k}{k!}e^{-\omega F(t)} \tag{2.10}$$

となり，平均値関数 $\mathrm{E}[N(t)]=\Lambda(t)=\omega F(t)$ の NHPP となる．ここで，確率分布 $F(t)$ は t に関して非減少関数であるので，$\lim_{t\to\infty}\Lambda(t)=\omega<\infty$ となり，平均値関数 $\Lambda(t)$ は上に有界となる．

Moranda（1979）による geometric de-eutrophication モデルと同様に，$\lim_{t\to\infty}\Lambda(t)\to\infty$ を満たす非有界な平均値関数をもつ NHPP モデルはいくつか知られており，Cretois and Gaudoin（1998），Duane（1964），Littlewood（1984）による冪法則モデルや Musa and Okumoto（1984），Musa et al.（1987）による対数ポアソン実行時間モデルがソフトウェア信頼性評価においてもよく使われている．平均値関数の有界性に関するこれら二つのモデルの違いは，無限長のテスト期間や運用期間を想定したとき，前者では累積フォールト数の確率関数がパラメータ ω のポアソン分布 $\lim_{t\to\infty}P_n(t)=\omega^n\exp(-\omega)/n!$ となるのに対し，後者は確率 1 で発散する $(\lim_{t\to\infty}N(t)\to\infty)$ ことである．実際，$P(N(t)\le k)=P(T_k\ge t)$ を満たす各イベントの発生時間を表す確率変数 T_k $(k=0,1,2,\ldots;T_0=0)$ を k 番目のフォールトが検出される時間（到着時間）とすれば，有界な平均値関数をもつ NHPP の到着時間分布は defective となり，任意の k に対して確率変数 T_k の有限次元モーメントは発散する．これより，平均値関数 $\Lambda(t)=\omega F(t)$ をもつ NHPP では，平均フォールト検出時間 $\mathrm{E}[T_k]$ を評価することは意味をなさないと言える．

表 1 において，代表的なフォールト検出時間分布に対応した NHPP モデルを整理する．Okamura and Dohi（2013）では，表 1 の 11 種類の NHPP モデルを実装したソフトウェア信頼性評価ツール SRATS（Software Reliability Assessment Tool on the Spreadsheet）を開発している．このように，NHPP モデルの表現能力はフォールト検出時間分布 $F(t)$ の表現能力に依存

表 1．代表的な NHPP モデル．

Models	$\Lambda(t)$ $(\omega>0, a>0, b>0, c>0)$
Exponential dist. (Exp) Goel and Okumoto (1979)	$\Lambda(t)=\omega F(t)$ $F(t)=1-e^{-bt}$
Gamma dist. (Gamma) Yamada et al. (1983), Zhao and Xie (1996)	$\Lambda(t)=\omega F(t)$ $F(t)=\int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto dist. (Pareto) Abdel-Ghaly et al. (1986)	$\Lambda(t)=\omega F(t)$ $F(t)=1-(\frac{b}{t+b})^c$
Truncated normal dist. (Tnorm) Okamura et al. (2013)	$\Lambda(t)=\omega\frac{F(t)-F(0)}{1-F(0)}$ $F(t)=\frac{1}{\sqrt{2\pi}b}\int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Log-normal dist. (Lnorm) Achcar et al. (1998), Okamura et al. (2013)	$\Lambda(t)=\omega F(\ln t)$ $F(t)=\frac{1}{\sqrt{2\pi}b}\int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
Truncated logistic dist. (Tlogist) Ohba (1984)	$\Lambda(t)=\omega F(t)$ $F(t)=\frac{1-e^{-bt}}{1+ce^{-bt}}$
Log-logistic dist. (Llogist) Gokhale and Trivedi (1998)	$\Lambda(t)=\omega F(\ln t)$ $F(t)=\frac{(bt)^c}{1+(bt)^c}$
Truncated extreme-value max dist. (Txvmax) Ohishi et al. (2009)	$\Lambda(t)=\omega\frac{F(t)-F(0)}{1-F(0)}$ $F(t)=e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value max dist. (Lxvmax) Ohishi et al. (2009)	$\Lambda(t)=\omega F(\ln t)$ $F(t)=e^{-(\frac{t}{b})^{-c}}$
Truncated extreme-value min dist. (Txvmin) Ohishi et al. (2009)	$\Lambda(t)=\omega\frac{F(0)-F(t)}{F(0)}$ $F(t)=e^{-e^{-\frac{t-c}{b}}}$
Log-extreme-value min dist. (Lxvmin) Goel (1985)	$\Lambda(t)=\omega\left(1-F(-\ln t)\right)$ $F(t)=e^{-e^{-\frac{t-c}{b}}}$
Logarithmic Poisson (Log) Musa and Okumoto (1984), Musa et al. (1987)	$\Lambda(t)=a\ln(1+bt)$
Power-law (Plaw) Cretois and Gaudoin (1998), Duane (1964), Littlewood (1984)	$\Lambda(t)=at^b$
Generalized truncated logist. dist. (Gtlogist) Erto et al. (2020), Xiao (2015)	$\Lambda(t)=\omega\frac{1-e^{-at^c}}{1+be^{-at^c}}$ $F(t)=\frac{1-e^{-at^c}}{1+be^{-at^c}}$

するため，より広いクラスの確率分布を考えることは有効である．Okamura and Dohi (2016) は $F(t)$ に連続形相型分布を導入することで，汎用的な NHPP モデルの適用可能性について言及している．Dohi et al. (2024) は多項式近似を導入することで，異なる観点から NHPP モデルを一般化している．

式(2.8)の混合分布表現において，初期フォールト数 N の分布関数 $\pi_N(n|\cdot)$ にポアソン分布以外の確率分布を仮定することで，NHPP モデルとは異なる時間依存遷移率をもつマルコフ過程を構成することが可能である．例えば，混合比分布の確率関数が

$$\pi_N(n|\gamma_1,\gamma_2)=\binom{n+\gamma_1-1}{n}\left(\frac{\gamma_2}{\gamma_2+1}\right)^{\gamma_1}\left(\frac{1}{\gamma_2+1}\right)^n \tag{2.11}$$

のような負の 2 項分布に従うものとすれば，$N(t)$ の確率関数は

$$P(N(t)=k)=\binom{k+\gamma_1-1}{k}\left(\frac{\gamma_2}{\gamma_2+F(t)}\right)^{\gamma_1}\left(\frac{F(t)}{\gamma_2+F(t)}\right)^k \tag{2.12}$$

となり，ポリア-ランドバーグ過程の一般化でもある非一様な混合ポアソン過程(Non-homogeneous Mixed Poisson Process: NHMPP)となる．このように，異なる混合比分布を仮定することで様々な非一様点過程を導出することができるが，次節で明らかになるように，結果として統計的に重要な意味をなさないモデルになることが理論的に示される．上述の結果は，式(2.10)で与えられるポアソン分布のパラメータ ω が確率密度関数 $dG(t \mid \boldsymbol{\theta}_G)/dt = \gamma_2^{\gamma_1}\omega^{\gamma_1-1}\exp(-\gamma_2\omega)/(\gamma_1-1)!$ $(\boldsymbol{\theta}_G = (\gamma_1, \gamma_2))$ のガンマ分布に従う場合に相当する．

2.3 NHMP によるモデリング

次に，一般化されたソフトウェア信頼性モデルについて紹介する．マルコフ過程の推移率が状態 k と時刻 t の両方に依存する場合，すなわち，$\lambda_k(t)$ $(k = 0, 1, 2, \ldots)$ の場合について考える．このような計数過程は非一様マルコフ過程(Non-homogeneous Markov process: NHMP)と呼ばれ，HMP モデルと NHPP モデルを特殊な場合として包含することに注意する．先述のモデルと同様に，Li et al. (2023)によって示されたように，NHMP モデルにおいても吸収状態をもつ場合ともたない場合を考え，以下のような 2 種類の推移率を考える．

$$\lambda_k(t) = (\beta - \alpha k)\kappa(t),\ k = 0, 1, 2, \ldots, \boldsymbol{\gamma}_{\alpha,\beta}, \tag{2.13}$$

$$\lambda_k(t) = (\alpha k + \beta)\kappa(t),\ k = 0, 1, 2, \ldots. \tag{2.14}$$

ここで，α (≥ 0) と β (≥ 0) は非負の実数パラメータであり，$\kappa(t)$ は t に関する任意の連続関数，$\boldsymbol{\gamma}_{\alpha,\beta} = \lceil \beta/\alpha \rceil$ であり，$\lceil x \rceil$ は x を超えない整数値を示す．本稿では，$\kappa(t)$ と $\Lambda(t) = \int_0^t \kappa(x)dx$ をそれぞれベースライン強度関数 (baseline intensity function)，累積ベースライン強度関数 (cumulative baseline intensity function)と呼ぶことにする．式 (2.13)と式 (2.14)で与えられる線形推移率は，それぞれ Kendall (1948)と Konno (2010)によって導入され，対応する NHMP は一般化 2 項過程 (generalized binomial process: GBP)並びに一般化ポリア過程(generalized Polya process: GPP)と呼ばれる．Shanthikumar (1981, 1983)は $\lambda_k(t) = (\beta - \alpha k)\kappa(t)$ $(n = 0, 1, 2, \ldots, \boldsymbol{\gamma}_{\alpha,\beta})$ および $\kappa(t) = \omega b \exp(-bt)$ となる逆死滅過程タイプの NHMP のコルモゴロフ前進方程式

$$\frac{\mathrm{d}}{\mathrm{d}t}P_0(t) = -\beta\kappa(t)P_0(t), \tag{2.15}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_k(t) = \big\{\beta - \alpha(k-1)\big\}\kappa(t)P_{k-1}(t) - \big\{\beta - \alpha k\big\}\kappa(t)P_k(t), \quad k = 0, 1, \ldots, \boldsymbol{\gamma}_{\alpha,\beta} \tag{2.16}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_{\boldsymbol{\gamma}_{\alpha,\beta}}(t) = \big\{\beta - \alpha(\boldsymbol{\gamma}_{\alpha,\beta} - 1)\big\}\kappa(t)P_{\boldsymbol{\gamma}_{\alpha,\beta}-1}(t) \tag{2.17}$$

を初期条件 $P_0(0) = 1, P_k(0) = 0 (k = 1, 2, \ldots, \boldsymbol{\gamma}_{\alpha,\beta})$ の下で解き，定常推移確率が

$$P_k(t) = \binom{\boldsymbol{\gamma}_{\alpha,\beta}}{k} A(t)^{\boldsymbol{\gamma}_{\alpha,\beta}-k}(1 - A(t))^k, \quad k = 0, 1, \ldots, \boldsymbol{\gamma}_{\alpha,\beta} \tag{2.18}$$

によって与えられることを示した．ここで，

$$A(t) = e^{-\alpha\Lambda(t)}, \tag{2.19}$$

$$\Lambda(t) = \int_0^t \kappa(x)dx \tag{2.20}$$

であり，式(2.18)の確率分布が平均と分散

$$\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{1 - e^{-\alpha\Lambda(t)}\}, \tag{2.21}$$

$$\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{1 - e^{-\alpha\Lambda(t)}\}e^{-\alpha\Lambda(t)} \tag{2.22}$$

をもつ 2 項分布となることを示した．式 (2.21) と式 (2.22) から，$\mathrm{E}[N(t)] > \mathrm{Var}[N(t)]$ となることがわかり，GBP モデルは常に under-dispersion となる．式 (2.22) の分散は時間 t の減少関数となるため，テスト時間が経過するとともに不確実性が減少することが理解できる．特に，$\alpha = 0$ および $\kappa(t) = 1$ のとき，式 (2.18) の確率関数は強度関数 $\kappa(t)$ をもつ NHPP モデルと Jelinski and Moranda (1972) による HMP モデルに帰着される．表 1 の $\Lambda(t)$ を式 (2.20) に代入することで，NHPP モデルに対応した GBP モデルを構築することが可能となる．

NHPP モデルと同様に，累積ベースライン強度関数 $\Lambda(t)$ が有界，すなわち $\lim_{t\to\infty}\Lambda(t) = \omega$ のとき，$\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{1 - \exp(-\alpha\omega)\}$ および $\lim_{t\to\infty}\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\exp(-\alpha\omega)\{1 - \exp(-\alpha\omega)\}$ を確認することができる．逆に，$\lim_{t\to\infty}\Lambda(t) \to \infty$ ならば，$\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}$，$\lim_{t\to\infty}\mathrm{Var}[N(t)] = 0$ となる．GBP モデルがサンプルパスの意味で $N(t) = \boldsymbol{\gamma}_{\alpha,\beta}$ において吸収状態をもつことに注意すれば，有界な累積ベースライン強度関数 $\Lambda(t)$ において $\lim_{t\to\infty}P_{\boldsymbol{\gamma}_{\alpha,\beta}}(t) = \{1 - \exp(-\alpha\omega)\}^{\boldsymbol{\gamma}_{\alpha,\beta}}$ が成立し，無限長のシステムテスト期間においてソフトウェアに残存するすべてのフォールトが検出され得ない正の確率が存在することになる．さらに，GBP モデルを NHMP として眺めたとき，任意の状態 k から $k+1$ $(k = 0, 1, \ldots, \boldsymbol{\gamma}_{\alpha,\beta} - 1)$ に推移するまでの時間分布は defective となり，確率過程 $N(t)$ は漸近的に $\boldsymbol{\gamma}_{\alpha,\beta}$ に収束する．よって，フォールト検出の観点から $\boldsymbol{\gamma}_{\alpha,\beta}\{1 - \exp(-\alpha\omega)\}$ は検出可能な正味累積フォールト数として解釈することができる．一方，累積ベースライン強度関数 $\Lambda(t)$ が非有界のとき，大数の法則により，$N(t)$ は $\boldsymbol{\gamma}_{\alpha,\beta}$ に収束することがわかる．

GBP によるモデル化の枠組を理解すると，これまでは詳細に調べられることのなかった既存のソフトウェア信頼性モデルの構造を容易に理解することが可能となる．例えば，表 1 の冪法則タイプの平均値関数 $\Lambda(t) = at^b$ を考えたとき，$\kappa(t) = (a/(1+b))t^{b+1}$ および $\lambda_k(t) = (\beta - \alpha k)(a/(1+b))t^{b+1}$ となるので，$b' = 1 + b$,$\beta' = (a/b')\beta$, $\alpha' = (a/b')\alpha$ のような変換を行うと $\lambda_k(t) = (\beta' - \alpha' k)t^{b'}$ を得る．特に $b' = 2$ $(b = 1)$,$b' > 2$ $(b > 1)$ のとき，この NHMP モデルは，それぞれ Schick and Wolverton (1978) と Wagoner (1973) によるソフトウェア信頼性モデルと一致する．また，Littlewood (1980, 1981) はベイズ推論の枠組において，パレートタイプのフォールト検出時間分布を扱い，推移率が $\lambda_k(t) = (\beta - \alpha k)/(\gamma + t)$ $(k = 0, 1, 2, \ldots, \boldsymbol{\gamma}_{\alpha,\beta})$ になるモデルを提案しており，$\kappa(t) = (t + \gamma)^{-1}$ および $\Lambda(t) = \ln(1 + t/\gamma)$ の GBP となる．このモデルは Musa and Okumoto (1984) による対数ポアソン実行時間モデルの拡張になっており，期待累積フォールト数は $\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{1 - \gamma(\gamma + t)^{\alpha}\}$ によって与えられる．

次に，$\lambda_k(t) = (\alpha k + \beta)\kappa(t)$ $(k = 0, 1, 2, \ldots)$ の GPP について考えよう．Konno (2010) は出生過程タイプの NHMP として以下のようなコルモゴロフの前進方程式

$$\frac{\mathrm{d}}{\mathrm{d}t}P_0(t) = -\beta\kappa(t)P_0(t), \tag{2.23}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_k(t) = \big\{\alpha(k-1) + \beta\big\}\kappa(t)P_{k-1}(t) - \big\{\alpha k + \beta\big\}\kappa(t)P_k(t), \quad k = 1, 2, \ldots \tag{2.24}$$

を初期条件 $P_0(0) = 1$,$P_n(0) = 0$ の下で解き，定常推移確率

$$P_n(t) = \binom{n + \frac{\beta}{\alpha} - 1}{n} A(t)^{\frac{\beta}{\alpha}}(1 - A(t))^n, \quad n = 0, 1, \ldots \tag{2.25}$$

を導出し，平均と分散がそれぞれ

$$\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{e^{\alpha\Lambda(t)} - 1\}, \tag{2.26}$$

$$\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta} e^{\alpha\Lambda(t)}\{e^{\alpha\Lambda(t)} - 1\} \tag{2.27}$$

となる負の 2 項分布に従うことを示した．Konno (2010) と Gat (2016) はこの確率過程をそれぞれ一般化ポリア過程，一般化ユール過程と呼んでいるが，本稿では統一的に GPP と略記する．式 (2.26) と式 (2.27) から，$\mathrm{E}[N(t)] < \mathrm{Var}[N(t)]$ となり，GBP は over-dispersion となる．累積ベースライン強度関数 $\Lambda(t)$ について $\lim_{t\to\infty}\Lambda(t) = \omega$ が成立すれば，$\lim_{t\to\infty}\mathrm{E}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\{\exp(\alpha\omega) - 1\}$ および $\lim_{t\to\infty}\mathrm{Var}[N(t)] = \boldsymbol{\gamma}_{\alpha,\beta}\exp(\alpha\omega)\{\exp(\alpha\omega) - 1\}$ となり，累積ベースライン強度関数が時刻無限大で発散すれば $\lim_{t\to\infty}\mathrm{E}[N(t)] = \lim_{t\to\infty}\mathrm{Var}[N(t)] \to \infty$ となる．GPP モデルが吸収状態をもたない NHMP であるのに対し，有界な累積ベースライン強度関数であっても累積期待フォールト数の期待値は $\mathrm{E}[N(t)] \to \boldsymbol{\gamma}_{\alpha,\beta}\{\exp(\alpha\omega) - 1\}(t \to \infty)$ となる．GPP モデルは Okamura and Dohi (2008) による非一様混合ポアソンモデルの一般化としてもみなすことができ，式 (2.18) と式 (2.25) の確率関数が $\alpha = 0$ で連続であるので，$\kappa(t) = 1$ と $\alpha = 0$ のとき HMP モデルと NHPP モデルは，それぞれ GBP モデルと GPP モデルの特別な場合に帰着される (Gat, 2016)．

3. 最尤推定法

3.1 HMP モデルと NHPP モデル

HMP および NHPP に基づくソフトウェア信頼性モデルでは，推移率(強度関数)が決定された後，モデルパラメータを推定するための一般的な手法として，対数尤度関数を最大化する最尤推定法が用いられる．先にも述べたように，ソフトウェアフォールトデータは時間領域データとグループデータの 2 種類に分類される．$t_0 = 0$ として，m 個のフォールト検出時刻 $t_i (i = 1, 2, \ldots, m)$ が観測され，このとき $t_e (\geq t_m)$ は観測(または打ち切り)時刻を示す．一方，グループデータは観測時刻とソフトウェアの累積フォールト検出数 (τ_i, n_i) $(i = 1, 2, \ldots, m)$ のペアで構成される．この場合，m はテストの実施日数，週数，または月数を示し，初期状態として $(\tau_0, n_0) = (0, 0)$ が設定される．

まず最初に，打切り時刻 t_e までに観測される m 個の時間領域データ $(t_1, t_2, \ldots, t_m; t_e)$ が与えられているとする．このとき，推移率 λ_k をもつ HMP モデルと NHPP モデルに対する対数尤度関数は

$$\ln L_{HMP}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \ln\lambda_i(\boldsymbol{\theta}) - \lambda_m(\boldsymbol{\theta})t_e, \tag{3.1}$$

$$\begin{aligned}\ln L_{NHPP}(\omega, \boldsymbol{\theta}) &= \sum_{i=1}^{m} \ln\lambda(t_i; \omega, \boldsymbol{\theta}) - \Lambda(t_e; \omega, \boldsymbol{\theta}) \\ &= \sum_{i=1}^{m} \ln f(t_i; \boldsymbol{\theta}) + \ln\omega^m - \omega F(t_e; \boldsymbol{\theta})\end{aligned} \tag{3.2}$$

によって与えられる．ここで，$\boldsymbol{\theta}$ は各モデルに含まれる未知パラメータベクトルであり，HMP モデルの推移率を $\lambda_k(\boldsymbol{\theta})$ $(k = 0, 1, 2, \ldots)$, $f(t; \boldsymbol{\theta}) = dF(t; \boldsymbol{\theta})/dt$, NHPP モデルの平均値関数と強度関数をそれぞれ $\Lambda(t; \omega, \boldsymbol{\theta})$, $\lambda(t; \omega, \boldsymbol{\theta}) = d\Lambda(t; \boldsymbol{\theta})/dt$ のように表記する．

次に，グループデータ (τ_i, n_i) $(i = 1, 2, \ldots, m)$ が与えられているものとする．時間領域データにおいて m が観測された検出フォールト数を表していたのに対し，グループデータでは m は観測データ区間数を表しており，$(\tau_{i-1}, \tau_i]$ を i 番目の観測区間，n_i を i 番目の観測区間で検出された累積フォールト数を示す．HMP モデルの対数尤度関数は，マルコフ性から

$$\ln L_{HMP}(\boldsymbol{\theta}) = \prod_{i=1}^{m} \Pr\{N(\tau_i) = n_i \mid N(\tau_{i-1}) = n_{i-1}\} \tag{3.3}$$

のように与えられるが，任意の推移率に対して常に陽な表現が与えられる訳ではない．例えば，Jelinski and Moranda モデル では，$\boldsymbol{\theta} = (N, b)$ に対する対数尤度関数は

$$\ln L_{HMP}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \ln \binom{N - n_{i-1}}{n_i - n_{i-1}} \{1 - e^{-b\tau_i}\}^{n_i - n_{i-1}} e^{-b(N-n_i)\tau_i} \tag{3.4}$$

となる．一方，NHPP モデルでは，ポアソン分布に従うことから，対数尤度関数は

$$\begin{aligned} \ln L_{NHPP}(\omega, \boldsymbol{\theta}) = & \sum_{i=1}^{m} \{(n_i - n_{i-1}) \ln[\Lambda(\tau_i; \omega, \boldsymbol{\theta}) - \Lambda(\tau_{i-1}; \omega, \boldsymbol{\theta})] \\ & - \ln[(n_i - n_{i-1})!]\} - \Lambda(\tau_m; \omega, \boldsymbol{\theta}) \\ = & \sum_{i=1}^{m} \{(n_i - n_{i-1}) \ln[F(\tau_i; \boldsymbol{\theta}) - F(\tau_{i-1}; \boldsymbol{\theta})] \\ & - \ln[(n_i - n_{i-1})!]\} + \ln \omega^{n_m} - \omega F(\tau_m; \boldsymbol{\theta}) \end{aligned} \tag{3.5}$$

となる．

より一般に，式 (2.8)で与えられる混合分布表現を考える．NHPP モデルと同様に，NHMPP モデルにおける対数尤度関数は，時間領域データとグループデータに対して，それぞれ

$$\ln L_{NHMPP}(\boldsymbol{\theta}_G) = \sum_{i=1}^{m} \ln f(t_i; \boldsymbol{\theta}) + \ln \int_0^\infty \ln \omega^m e^{-\omega F(t_e; \boldsymbol{\theta})} dG_\omega(\omega | \boldsymbol{\theta}_G), \tag{3.6}$$

$$\begin{aligned} \ln L_{NHMPP}(\boldsymbol{\theta}_G) = & \sum_{i=1}^{m} \{(n_i - n_{i-1}) \ln[F(\tau_i; \boldsymbol{\theta}) - F(\tau_{i-1}; \boldsymbol{\theta})] \\ & - \ln[(n_i - n_{i-1})!]\} + \ln \int_0^\infty (\omega^{n_m} e^{-\omega F(\tau_m; \boldsymbol{\theta})}) dG_\omega(\omega | \boldsymbol{\theta}_G) \end{aligned} \tag{3.7}$$

のように求められる．次に示す一連の結果は，NHPP モデルと NHMPP モデルに対する対数尤度関数の大小関係を規定するものである．

性質 1. 任意の $\hat{\omega}$ に対して以下を満たすパラメータ $\boldsymbol{\theta}_G$ が存在する．

$$G(\omega; \boldsymbol{\theta}_G) = \begin{cases} 0 & (0 < \omega < \hat{\omega}), \\ 1 & (\hat{\omega} \leq \omega < \infty). \end{cases} \tag{3.8}$$

補題 1. 任意の $n\ (> 0)$ と $0 \leq F(\tau; \boldsymbol{\theta}) \leq 1$ に対して $\omega^n e^{-\omega F(\tau; \boldsymbol{\theta})}$ を最大にする $\hat{\omega}\ (0 < \hat{\omega} < \infty)$ が唯一存在する．

補題 2. 混合比分布 $G(\omega; \boldsymbol{\theta}_G)$ が性質 1 を満たすとき，NHMPP に基づくソフトウェア信頼性モデルの対数尤度関数を最大にする混合比分布は

$$G(\omega) = \begin{cases} 0 & (0 < \omega < \hat{\omega}), \\ 1 & (\hat{\omega} \leq \omega < \infty) \end{cases} \tag{3.9}$$

となる．補題 1 と 補題 2 から，直ちに以下の結果を得る．

定理 1. (1)NHPP モデルの最尤推定値が存在し，NHMPP モデルの混合比分布が性質 1 を満たすものとする．このとき，NHPP モデルと NHMPP モデルの対数尤度の値は等しくなる．

(2)NHMPP モデルの混合比分布が性質 1 を満たさないとき，NHPP モデルの対数尤度関数は NHMPP モデルの対数尤度関数よりも常に大きい値をもつ．

以上の結果から，混合比分布が性質 1 を満たすステップ関数であれば，NHMPP モデルにおける対数尤度関数は NHPP モデルの対数尤度より常に小さくなるため，最尤推定の意味で NHMPP モデルを考える必然性はないことが示される．

3.2 NHMP モデル

次に GBP と GPP に対する対数尤度関数を導出する．NHPP における ω の存在は仮定せず，$\boldsymbol{\theta}$ を累積ベースライン強度関数のパラメータ $\kappa(t)=\kappa(t;\boldsymbol{\theta})$ および $\Lambda(t)=\Lambda(t;\boldsymbol{\theta})$ とし，時間領域データ が観測されているとする．GBP に対する尤度関数は

$$L_{GBP}(\alpha,\beta,\boldsymbol{\theta})=\prod_{i=1}^{m}\{\beta-\alpha(i-1)\}\kappa(t_i;\boldsymbol{\theta})e^{-\int_0^{t_e}\{\beta-\alpha m\}\kappa(x;\boldsymbol{\theta})dx} \tag{3.10}$$

のように表現され，特に

$$\begin{aligned}\int_0^{t_e}\{\beta-\alpha m\}\kappa(x;\boldsymbol{\theta})dx &= \sum_{i=1}^{m}\int_{t_{i-1}}^{t_i}\{\beta-\alpha(i-1)\}\kappa(x;\boldsymbol{\theta})dx+\int_{t_m}^{t_e}\{\beta-\alpha m\}\kappa(x;\boldsymbol{\theta})dx\\ &=\beta\Lambda(t_e;\boldsymbol{\theta})-\alpha\left[m\Lambda(t_e;\boldsymbol{\theta})-\sum_{i=1}^{m}\Lambda(t_i;\boldsymbol{\theta})\right]\end{aligned} \tag{3.11}$$

のような表現を得る．式 (3.10)の階乗部分は

$$\prod_{i=1}^{m}\{\beta-\alpha(i-1)\}\kappa(t_i;\boldsymbol{\theta})=\frac{\Gamma(\boldsymbol{\gamma}_{\alpha,\beta}+1)}{\Gamma(\boldsymbol{\gamma}_{\alpha,\beta}-m)}\alpha^m\prod_{i=1}^{m}\kappa(t_i;\boldsymbol{\theta}) \tag{3.12}$$

のように書けるので，$\Gamma(k+1)=k\Gamma(k)$ に注意すると，対数尤度関数は

$$\begin{aligned}\ln L_{GBP}(\alpha,\beta,\boldsymbol{\theta})=&\left[\sum_{i=1}^{m}\ln\kappa(t_i;\boldsymbol{\theta})+\sum_{i=1}^{m}\alpha\Lambda(t_i;\boldsymbol{\theta})\right]+\ln[(\boldsymbol{\gamma}_{\alpha,\beta})!-(\boldsymbol{\gamma}_{\alpha,\beta}-m-1)!]\\ &+m\ln(\alpha)+[(\alpha m-\beta)\Lambda(t_e;\boldsymbol{\theta})]\end{aligned} \tag{3.13}$$

となる．グループデータ (τ_i,n_i) $(i=0,1,\ldots,m)$ の場合，式 (2.18) から，対数尤度関数を

$$\begin{aligned}\ln L_{GBP}(\alpha,\beta,\boldsymbol{\theta})=&\sum_{i=1}^{m}\ln\{(\boldsymbol{\gamma}_{\alpha,\beta}-n_{i-1})!-(n_i-n_{i-1})!-(\boldsymbol{\gamma}_{\alpha,\beta}-n_i)!\}\\ &-\sum_{i=1}^{m}\{\boldsymbol{\gamma}_{\alpha,\beta}-n_i\}\times\alpha\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}\\ &+\sum_{i=1}^{m}(n_i-n_{i-1})\times\ln[1-e^{-\alpha\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}]\end{aligned} \tag{3.14}$$

のように求めることができる．

次に GPP に対する対数尤度関数について考える．時間領域データが与えられたとき，式 (3.13)から，直ちに

$$\begin{aligned}\ln L_{GPP}(\alpha,\beta,\boldsymbol{\theta})=&\left[\sum_{i=1}^{m}\ln\kappa(t_i;\boldsymbol{\theta})+\sum_{i=1}^{m}\alpha\Lambda(t_i;\boldsymbol{\theta})\right]+\ln[(\boldsymbol{\gamma}_{\alpha,\beta})!-(\boldsymbol{\gamma}_{\alpha,\beta}-m-1)!]\\ &+(m\ln(\alpha)-[(\alpha m+\beta)\Lambda(t_e;\boldsymbol{\theta})])\end{aligned} \tag{3.15}$$

を得る．グループデータ (τ_i,n_i) $(i=0,1,\ldots,m)$ の場合，GPP の対数尤度関数は

$$\ln L_{GPP}(\alpha,\beta,\boldsymbol{\theta}) = \sum_{i=1}^{m} \ln\{(n_i+\boldsymbol{\gamma}_{\alpha,\beta}-1)! - (n_i-n_{i-1})! - (n_{i-1}+\boldsymbol{\gamma}_{\alpha,\beta}-1)!\} - \sum_{i=1}^{m}\{\boldsymbol{\gamma}_{\alpha,\beta}+n_{i-1}\}\alpha\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\} + \sum_{i=1}^{m}(n_i-n_{i-1})\ln[1-e^{-\alpha\{\Lambda(\tau_i;\boldsymbol{\theta})-\Lambda(\tau_{i-1};\boldsymbol{\theta})\}}] \tag{3.16}$$

のように求めることが可能である．

4. ソフトウェアメトリクスの利用

4.1 一般化線形ソフトウェア信頼性モデル

ここまでのソフトウェア信頼性モデルでは，フォールト検出時間やフォールト検出数のみを扱ってきた．しかし，ソフトウェアの信頼性は，ソフトウェアの設計やテスト工程における情報に大きく影響すると考えられ，信頼性評価の結果に基づいて設計や開発プロセスの改善を行うことも必要とされる．このように，ソフトウェアの設計やテスト工程における情報をソフトウェアメトリクスと呼び，これらをソフトウェア信頼性モデルに組み込む拡張モデルが提案されている．ソフトウェアメトリクスには，コード行数や複雑性のようなソフトウェアそのものの特徴を表すプロダクトメトリクス(あるいはデザインメトリクス)と，開発工程において計測されるレビュー回数，テストケース数，カバレッジのような開発プロセスの情報を表すプロセスメトリクスの 2 種類がある．これらのメトリクスはそれぞれ，ソフトウェアの信頼性に影響を与えるが，影響を与える範囲が異なる．Okamura and Dohi (2014)では，プロダクトメトリクスをソフトウェア信頼性モデルに組み込むことで，ソフトウェアの信頼性評価を行っている．また，Okamura et al. (2010)は，プロセスメトリクスを組み込んだソフトウェア信頼性モデルを提案している．いずれのモデルにおいても，基本的なアイデアは一般化線形モデル(Generalized Linear Model: GLM)と呼ばれる回帰モデルの活用である．

一般化線形モデルとは，正規分布による線形回帰(重回帰)モデルを一般化したモデルの枠組みであり，正規分布以外の分布や，多様なリンク関数を用いて独立変数と従属変数の因果関係を表すモデルである．いま，確率変数(従属変数)$Y_1,\ldots,Y_N$ とそれらに対応する確率変数(独立変数)ベクトル $X_1,\ldots,X_N$ の観測値が与えられるとする．ここで，$X_i=(X_{i,1},\ldots,X_{i,L})$ であり，$X_{i,l}(l=1,2,\ldots,L)$ は Y_i に対する要因 l の値を表す．このとき，Y と X の関係を表すモデルとして次の仮定を考える．

- $Y_1,\ldots,Y_N$ は指数分布族に属する同一の分布に従う．ただし，パラメータが異なる．
- 従属変数の平均は $g(\mathrm{E}[Y_i])=\boldsymbol{\beta}\boldsymbol{x}_i$ で与えられる．$\boldsymbol{x}_i$ は X_i の観測値であり，$g(\cdot)$ はリンク関数と呼ばれる．

上記の枠組みは $Y_1,\ldots,Y_N$ の分布とリンク関数を選ぶことにより，既存の回帰モデルに帰着させることができる．分布を正規分布，リンク関数を $g(\mu)=\mu$ とすると，重回帰モデルに帰着される．また，分布をポアソン分布，リンク関数を $g(\mu)=\log(\mu)$ とするとポアソン回帰，分布をベルヌーイ分布，リンク関数をロジット関数 $g(\mu)=\log\mu/(1+\mu)$ とすると，ロジスティック回帰へそれぞれ帰着される．

上記の枠組みを利用してソフトウェアメトリクスと NHPP に基づいたソフトウェア信頼性モデルの融合を考える．いま，j 個のモジュールからなるソフトウェアを考え，以下の仮定を設ける．

- ソフトウェアテストは逐次的かつ離散的に実施され，各テスト毎で検出されたフォールトは次のテストまでに修正される．
- モジュール i において 単一のフォールトが k 番目のテスト区間で検出されるフォールト検出確率 $p_{i,k}$ $(0 \leq p_{i,k} \leq 1)$ は，モジュール i において k 番目のテスト区間に関するプロセスメトリクスベクトル $\boldsymbol{x}_{i,k} = (x_{i,k,1}, \ldots, x_{i,k,r_i})$ を用いて

$$l_p(p_{i,k}) = \boldsymbol{\alpha}_{0,i} + \boldsymbol{\alpha}_i \boldsymbol{x}_{i,k} \tag{4.1}$$

のように表現される．ここで $l_p(\cdot)$ はフォールト検出確率のリンク関数である．
- 各モジュール $i = 1, \ldots, j$ に含まれる総フォールト数(初期フォールト数) $M_1, \ldots, M_j$ はモジュール固有のプロダクトメトリクス $\boldsymbol{s}_i$ に依存した平均 ξ_i のポアソン分布に従い，その平均は以下で与えられる．

$$l_\xi(\xi_i) = \beta_0 + \boldsymbol{\beta}\boldsymbol{s}_i. \tag{4.2}$$

ここで，$l_\xi(\cdot)$ は総フォールト数のリンク関数である．

フォールト検出確率のリンク関数としては，ロジット関数，プロビット関数および complementary log-log 関数などが適用できる．また，総フォールト数のリンク関数としては対数関数や一次関数が適用できる．いま，

$$l_p(p) = \mathrm{logit}(p) = \log \frac{p}{1-p}, \quad l_\xi(\xi) = \log(\xi) \tag{4.3}$$

とすると，モジュール i において k 番目のテスト区間のフォールト検出確率は

$$p_{i,k} = \frac{\exp(\boldsymbol{\alpha}_{0,i} + \boldsymbol{\alpha}_i \boldsymbol{x}_{i,k})}{1 + \exp(\boldsymbol{\alpha}_{0,i} + \boldsymbol{\alpha}_i \boldsymbol{x}_{i,k})} \tag{4.4}$$

となる．さらに，モジュール i の初期フォールト数は平均 $\exp(\xi_i)$ のポアソン分布に従うため，モジュール i の k 番目のテスト区間までに検出されるフォールト数 $Y_{i,k}$ は以下の確率関数で与えられる．

$$\begin{aligned} P(Y_{i,k_i} = y_{i,k}) &= \sum_{m_i = y_{i,k}}^{\infty} P(Y_{i,k_i} = y_{i,k} | M_i = m_i) P(M_i = m_i) \\ &= \exp(-\xi_i \lambda_{i,k}) \frac{(\xi_i \lambda_{i,k})^{y_{i,k}}}{y_{i,k}!}. \end{aligned} \tag{4.5}$$

ここで，$\lambda_{i,n_i} = 1 - \prod_{k=1}^{n_i}(1 - p_{i,k})$ である．このモデルは，モジュール i に含まれる総フォールト数に対するポアソン回帰と k 番目のテスト区間におけるフォールト検出確率に対するロジスティック回帰を同時に適用している．

4.2 EM アルゴリズム

次に効率的な推定を行う目的で EM(Expectation-Maximization)アルゴリズムの適用を考える．ここで考える一般化線形モデルの場合，未観測データを各モジュールの総フォールト数 $N_1, \ldots, N_j$ とし，モジュール m のフォールト検出時刻列を $T_{m,1} < T_{m,2} < \cdots < T_{m,N_m}$ とする．このとき，完全対数尤度は

$$\begin{aligned} \log \mathcal{L}(\beta_0, \boldsymbol{\beta}, \alpha_{0,1}, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m) = &\sum_{m=1}^{j} N_m(\beta_0 + \boldsymbol{\beta}\boldsymbol{s}_j) \\ &- \sum_{m=1}^{j} \exp(\beta_0 + \boldsymbol{\beta}\boldsymbol{s}_j) + \sum_{m=1}^{j} \sum_{k=1}^{N_m} \log(\lambda_{m,k} - \lambda_{m,k-1}) \end{aligned} \tag{4.6}$$

となる．つまり，最初の二項が $N_1,\ldots,N_m$ を従属変数としたポアソン回帰となり，$\lambda_{m,k}$ に関する項は $N_m-\sum_{u=1}^{k-1}y_u$ を従属変数としたロジスティック回帰となる．一方で，総フォールト数の期待値は与えられたパラメータ $\beta_0,\boldsymbol{\beta},\alpha_{0,m},\boldsymbol{\alpha}_m$ に対して

$$\mathrm{E}[N_m]=\sum_{k=1}^{K_m}y_{m,k}+\exp(\beta_0+\boldsymbol{\beta}\boldsymbol{s}_m)\prod_{k=1}^{K_m}\frac{\exp(\alpha_{0,m}+\boldsymbol{\alpha}_m\boldsymbol{x}_{k,m})}{1+\exp(\alpha_{0,m}+\boldsymbol{\alpha}_m\boldsymbol{x}_{k,m})} \tag{4.7}$$

として計算できる．最終的に，EM 原理に基づいた推定アルゴリズムは次のようになる．

- Step 1: Compute$\omega_m\leftarrow\exp(\beta_0+\boldsymbol{\beta}\boldsymbol{s}_m)$,$m=1,\ldots,j$.
- Step 2: Compute

$$p_{m,k}\leftarrow\frac{\exp(\alpha_{0,m}+\boldsymbol{\alpha}_m\boldsymbol{x}_{k,m})}{1+\exp(\alpha_{0,m}+\boldsymbol{\alpha}_m\boldsymbol{x}_{k,m})},\quad m=1,\ldots,j,\quad k=1,\ldots,K_m. \tag{4.8}$$

- Step 3: For$i=1,\ldots,j$, compute

$$Z_m\leftarrow\sum_{k=1}^{K_m}y_{m,k}+\omega_m\prod_{k=1}^{K_m}(1-p_{m,k}). \tag{4.9}$$

- Step 4: $(\beta_0,\boldsymbol{\beta})\leftarrow\mathrm{PR}(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_j;Z_1,\ldots,Z_j)$.
- Step 5: $(\alpha_{0,m},\boldsymbol{\alpha}_m)\leftarrow\mathrm{LR}(\boldsymbol{x}_{m,1},\ldots,\boldsymbol{x}_{m,K_m};Z_m,\ldots,Z_m-\sum_{u=1}^{K_m-1}y_u)$.

ここで，PR() と LR() は，それぞれ通常のポアソン回帰およびロジスティック回帰による回帰係数の推定を表す．上記の Step 1 から Step 5 を繰り返すことで，回帰係数に対する最尤推定値が得られる．

一般化線形ソフトウェア信頼性モデルでは要因数に応じた回帰係数が必要となる．一般的に，パラメータ数を多くすればするほどデータへの適合は良くなるがモデルの汎化能力を高めることはできないことが知られている．このような問題に対処するためには不要な要因を削除して要因を選別する必要がある．これは要因選択と呼ばれ，回帰をもとにしたモデルでは一般的に行われる手続きである．

要因選択手法には，Deviance を使う手法や検定による手法があるが，ここでは AIC（Akaike Information Criterion）を用いた手法を用いる．AIC の定義式を以下のように与える．

$$\mathrm{AIC}=-2\,\text{最大対数尤度}+2\,\text{パラメータ数}. \tag{4.10}$$

これは最尤法における真のモデルとのバイアスをパラメータ数によって近似した指標であり，AIC を最小にするモデルが最良のモデルとなる．このことから，AIC を最小にする要因の組み合わせを求めることで，データへの過適合を解消し汎化能力の高いモデルを得ることができる．具体的な手続きでは，すべての要因の組み合わせを総当たり的に調べることは不可能であるため変数増減法が適用される．変数増減法では，変数を加えることによって AIC を最も低くする要因から優先的に選択する手続きを AIC が増加するまで行う．これによって，AIC を最も小さくする要因の組み合わせを近似的に求める．

4.3 有効性検証

実際の開発で計測されたテストデータおよびオープンソースプロジェクトにおけるデータを用いてモデルの有効性に関する分析を行う．ここでは，従来の手法である単純な NHPP モデルと，プロダクトメトリクスだけをポアソン回帰として利用するソフトウェア信頼性モデル，プロセスメトリクスをロジスティック回帰として利用するソフトウェア信頼性モデルとの比較を行う．特に，AIC を用いた評価を行うことで，単純なデータとの誤差比較ではなく，背後にあ

る(真の)モデルを適切に表現できるかどうかの検証を行う.

実験で用いるデータは Apache Software Foundation (ASF)で管理されている Tomcat プロジェクトを対象とした.また,バージョンの違いによる影響も見るため,Tomcat については幾つかのバージョンに対するデータを収集する.さらに,それぞれのプロジェクト毎のバグトラッキングシステム(ASF Bugzilla)からバグレポートを収集した.Tomcat プロジェクトのモジュール構成,バグレポート収集期間は次の通りである.

Tomcat: Apache Software Foundation が開発しているオープンソースの Web サーバであり,Java サーブレットを提供する.主として Java 言語で作成されており,Apache Web Server と組み合わせて利用されることが多い.Bugzilla の管理では以下のプログラム群(モジュール)で構成される.

- webapps (manager): 管理用ウェブアプリケーション
- catalina: Servlet container のコア
- connectors: Tomcat と Apache の連携
- jasper: JSP ページコンパイラとランタイムエンジン
- servlets: Servlet プログラム群

対象とする Tomcat のバージョンは 5, 6, 7 であり,メジャーバージョンアップについては別アプリケーション(別プロジェクト)と見なして分析する.それぞれのバージョンにおけるバグデータの収集期間は Tomcat5 が 2004/5 から 2009/1,Tomcat6 が 2006/1 から 2013/11,Tomcat7 が 2010/6 から 2013/11 となっている.

一般化線形ソフトウェア信頼性モデルでは,プロダクトメトリクスと総フォールト数の因果関係,プロセスメトリクスとフォールト検出確率の因果関係をそれぞれポアソン回帰モデル,ロジスティック回帰モデルで関連づけている.そのため,実データの分析においてもこれらのメトリクスの収集が必要となる.実際に,各プロジェクトのモジュールにおけるソースコードから,直接以下のプロダクトメトリクスを算出した.

- Files (Fl): ファイル数
- Lines (Ln): コード行数
- Statements (St): ステートメント数
- % Branches (Br): 分岐の出現率
- Calls (Ca): メソッドの呼び出し回数
- % Comments (Cm): コメントの出現率
- Classes (Cl): クラス数
- Methods/Class (Me/Cl): 単一クラスあたりのメソッド数
- Avg Stmts/Method (St/Me): 単一メソッドあたりのステートメント数
- Max Complexity (MCx): 最大複雑度
- Max Depth (MDp): 最大のネストの深さ
- Avg Depth (ADp): 平均のネストの深さ
- Avg Complexity (ACx): 平均複雑度

一方,プロジェクトのアクティビティを表す指標として考えられる以下のプロセスメトリクスをそれぞれのモジュールに対して計測した.

- time: 日数
- ctime: 累積日数

- comments: Bugzilla 上での該当モジュールに対するコメント数
- votes: Bugzilla 上での該当モジュールに対する投票回数
- wokers: コメントを投稿した人の数(Tomcat5 のみ)

まずモジュール毎で独立に(1)NHPP モデルの推定，(2)ロジスティック回帰によるソフトウェア信頼性モデルの推定を行った．(1)では標準的に使われる 11 種類のモデルすべてに対してパラメータを推定し最小の AIC を示すモデルを選択した．また，ロジスティック回帰によるソフトウェア信頼性モデルでは，変数増減法による要因選択を行って最小の AIC を示す要因の組み合わせを求めている．さらに，各プロジェクトのモジュールに対するプロダクトメトリクスを用いて，(3)ポアソン回帰によるソフトウェア信頼性モデルの推定と(4)一般化線形ソフトウェア信頼性モデルの推定を行った．ポアソン回帰によるソフトウェア信頼性モデルの推定では，事前に各モジュールに適用する NHPP モデルを選ぶ必要がある．これには，(1)の手順で選ばれたモデルと同じモデルを採用した．同様に(4)では，各モジュールに適用するロジスティック回帰によるソフトウェア信頼性モデルで使用する要因を選ぶ必要がある．これも(2)で単一モジュールに対して選択された要因を用いることとした．加えて，ポアソン回帰によるソフトウェア信頼性モデルおよび一般化線形ソフトウェア信頼性モデルでは，プロダクトメトリクスに対する要因選択を行う必要がある．これも，変数増減法により最小の AIC を示す要因の組み合わせを選んだ．

表 2 から表 4 は各プロジェクトに対して NHPP モデルのみ(NHPP)，ロジスティック回帰に

表 2. Tomcat5 に対する AIC.

Module	**NHPP**	**Logit**	**Poi**	**GLSRM**
catalina	268.96	271.04	-	-
connectors	145.10	128.20	-	-
jasper	159.96	147.70	-	-
servlets	116.99	102.08	-	-
webapps	150.51	137.69	-	-
Total (project)	841.52	786.71	838.87	784.58

表 3. Tomcat6 に対する AIC.

Module	**NHPP**	**Logit**	**Poi**	**GLSRM**
catalina	489.30	388.66	-	-
connectors	277.74	201.74	-	-
jasper	293.51	247.27	-	-
manager	163.15	125.58	-	-
servlets	181.54	181.58	-	-
Total (project)	1405.24	1144.83	1403.82	1144.84

表 4. Tomcat7 に対する AIC.

Module	**NHPP**	**Logit**	**Poi**	**GLSRM**
catalina	247.91	210.87	-	-
connectors	168.82	129.44	-	-
jasper	167.34	137.57	-	-
manager	117.07	90.49	-	-
servlets	148.33	110.57	-	-
Total (project)	849.47	678.94	847.40	675.86

よるソフトウェア信頼性モデル(Logit)，ポアソン回帰によるソフトウェア信頼性モデル(Poi)，一般化線形ソフトウェア信頼性モデル(GLSRM)それぞれを適用したときの AIC を示している．表からいずれの場合でも

$$\text{NHPP} \geq \text{Poi} \geq \text{Logit} \geq \text{GLSRM} \tag{4.11}$$

の関係が得られる．つまり，適合性の観点からは GLSRM が最も良いことがわかった．また，プロダクトメトリクスを扱うポアソン回帰によるソフトウェア信頼性モデルと，プロセスメトリクスを扱うロジスティック回帰によるソフトウェア信頼性モデルを比較すると，プロセスメトリクスを扱うロジスティック回帰によるソフトウェア信頼性モデルの方が適合性が高い．これは初期の総フォールト数を決めるための情報量よりも，観測されたバグ報告数から得られる情報量の方が多いことを示唆している．ここでは，モジュール数が比較的少ないためこのような結果になったものと考えられる．つまり，モジュール数が多い状況では，プロダクトメトリクスからの情報による影響がより大きくなるものと予想される．

また表 5 から表 7 は，各プロジェクトにおけるモジュールに対して AIC により選択された動的メトリクス，静的メトリクスを表している．選択されたメトリクスをみると，プロダクトメトリクスでは最大複雑度(MCx)が比較的多く，プロセスメトリクスではコメント数が多くのデータで選択される結果となった．これは，プロダクトメトリクスでは従来から重要とされてきた，コード行数や複雑度が大きく影響することを意味している．また，コメント数などのプロジェクトのアクティビティを表す指標がフォールト検出確率を決定するのに利用可能であることがわかる．

表 5．Tomcat5 において選択されたメトリクス．

Module	**Logit**	**Poi**	**GLSRM**
catalina	comments, ctime	St, MCx	St, MCx
connectors	worker, ctime		
jasper	worker, ctime		
servlets	comments, ctime		
webapps	comments, ctime		

表 6．Tomcat6 において選択されたメトリクス．

Module	**Logit**	**Poi**	**GLSRM**
catalina	comments, votes, ctime	Fl, Cl, MCx	Fl, Ln, Cm, Cl
connectors	comments, votes, ctime		
jasper	time, comments		
manager	comments, ctime		
servlets	comments, ctime		

表 7．Tomcat7 において選択されたメトリクス．

Module	**Logit**	**Poi**	**GLSRM**
catalina	comments, votes	Fl, Ca, Cm	St, Cm
connectors	comments, votes		
jasper	comments		
manager	comments, votes		
servlets	comments, votes, ctime		

5. まとめと今後の課題

本稿では，ソフトウェア信頼性モデルに関する主要な進展を体系的に整理し，特に非一様マルコフ過程(NHMP)を中心にした統一的なアプローチを提案した．従来の HMP および NHPP に基づくソフトウェア信頼性モデルを一般化し，GBP や GPP といった柔軟性の高い新たなモデルの特性を分析した．また，ソフトウェアメトリクスの利用による信頼性予測モデルの有用性を実証的に検証し，プロダクトおよびプロセスメトリクスの重要性を明らかにした．これらの成果は，実際のソフトウェア開発プロジェクトにおける信頼性向上に寄与することが期待できる．

今後の課題として，提案モデルをさらに広範なデータセットや産業分野で検証し，特に商用ソフトウェアへの適用によってモデルの汎用性を評価する必要がある．また，ソフトウェアメトリクスの収集には手間がかかるため，自動化や標準化された計測手法を開発し，実務での利用可能性を高める必要がある．これらの課題への取り組みによって，ソフトウェア信頼性モデリングの理論的および実践的貢献がさらに進展することが期待できる．

参 考 文 献

Abdel-Ghaly, A. A., Chan, P. Y. and Littlewood, B. (1986). Evaluation of competing software reliability predictions, *IEEE Transactions on Software Engineering*, **SE-12**(9), 950–967.

Achcar, J. A., Dey, D. K. and Niverthi, M. (1998). A Bayesian approach using nonhomogeneous Poisson processes for software reliability models, *Frontiers in Reliability* (eds. A. P. Basu, K. S. Basu and S. Mukhopadhyay), 1–18, World Scientific, Singapore.

Boland, P. J. and Singh, H. (2003). A birth-process approach to Moranda's geometric software-reliability model, *IEEE Transactions on Reliability*, **52**(2), 168–174.

Cretois, E. and Gaudoin, O. (1998). New results on goodness-of-fit tests for the power-law process and application to software reliability, *International Journal of Reliability, Quality and Safety Engineering*, **5**(3), 249–267.

Dohi, T., Li, S. and Okamura, H. (2024). Local polynomial software reliability models and their application, *Information and Software Technology*, **166**, 107366.

Duane, J. T. (1964). Learning curve approach to reliability monitoring, *IEEE Transactions on Aerospace*, **2**(2), 563–566.

Erto, P., Giorgio, M. and Lepore, A. (2020). The generalized inflection S-shaped software reliability growth model, *IEEE Transactions on Reliability*, **69**(1), 228–244.

Gat, Y. L. (2016). *Recurrent Event Modeling based on the Yule Process*: *Application to Water Network Asset Management*, **2**, John Wiley & Sons, London.

Goel, A. L. (1985). Software reliability models: assumptions, limitations and applicability, *IEEE Transactions on Software Engineering*, **SE-11**(12), 1411–1423.

Goel, A. L. and Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability*, **R-28**(3), 206–211.

Gokhale, S. S. and Trivedi, K. S. (1998). Log-logistic software reliability growth model, *Proceedings of The 3rd IEEE International Symposium on High-Assurance Systems Engineering* (*HASE 1998*), 34–41, IEEE CPS, Los Alamitos, California, USA.

Ikemoto, S., Dohi, T. and Okamura, H. (2013). Quantifying software test process and product reliability simultaneously, *Proceedings of The 24th International Symposium on Software Reliability Engineering* (*ISSRE 2013*), 108–117, IEEE CPS, Piscataway, New Jersey, USA.

Jelinski, Z. and Moranda, P. B. (1972). Software reliability research, *Statistical Computer Performance Evaluation* (ed. W. Freiberger), 465–484, Academic Press, New York.

Kendall, D. G. (1948). On the generalized "birth-and-death" process, *Annals of Mathematical Statistics*, **19**, 1–5.

Konno, H. (2010). On the exact solution of a generalized Polya process, *Advances in Mathematical Physics*, **2010**, 504267.

Kuwa, D. and Dohi, T. (2013a). Generalized Cox proportional hazards regression-based software reliability modeling with metrics data, *Proceedings of The 19th IEEE Pacific Rim International Symposium on Dependable Computing* (*PRDC 2013*), 328–337, IEEE CPS, Piscataway, New Jersey, USA.

Kuwa, D. and Dohi, T. (2013b). Generalized logit-based software reliability modeling with metrics data, *Proceedings of The 37th Annual International Computer Software and Applications Conference* (*COMPSAC 2013*), 246–255, IEEE CPS, Piscataway, New Jersey, USA.

Lawless, J. F. (1987). Regression methods for Poisson process data, *Journal of the American Statistical Association*, **82**(399), 808–815.

Li, S., Dohi, T. and Okamura, H. (2022). A comprehensive analysis of proportional intensity-based software reliability models with covariates, *Electronics*, **11**, 2353.

Li, S., Dohi, T. and Okamura, H. (2023). Non-homogeneous Markov process modeling for software reliability assessment, *IEEE Transactions on Reliability*, **72**(4), 1540–1555.

Littlewood, B. (1980). Theories of software reliability: How good are they and how can they be improved?, *IEEE Transactions on Software Engineering*, **SE-6**(5), 489–500.

Littlewood, B. (1981). Stochastic reliability growth: A model for fault removal in computer-program and hardware-designs, *IEEE Transactions on Reliability*, **R-30**(4), 313–320.

Littlewood, B. (1984). Rationale for a modified Duane model, *IEEE Transactions on Reliability*, **R-33**(2), 157–159.

Lyu, M. R. (ed.) (1996). *Handbook of Software Reliability Engineering*, McGraw-Hill, New York.

Moranda, P. B. (1979). Event-altered rate models for general reliability analysis, *IEEE Transactions on Reliability*, **R-28**(5), 376–381.

Musa, J. D. and Okumoto, K. (1984). A logarithmic Poisson execution time model for software reliability measurement, *Proceedings of The 7th International Conference on Software Engineering* (*ICSE 1984*), 230–238, ACM, New York, USA.

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

Ohba, M. (1984). Inflection S-shaped software reliability growth model, *Stochastic Models in Reliability Theory* (eds. S. Osaki and Y. Hatoyama), 144–165, Springer-Verlag, Berlin.

Ohishi, K., Okamura, H. and Dohi, T. (2009). Gompertz software reliability model: Estimation algorithm and empirical validation, *Journal of Systems and Software*, **82**(3), 535–543.

Okamura, H. and Dohi, T. (2008). Software reliability modeling based on mixed Poisson distributions, *International Journal of Reliability, Quality and Safety Engineering*, **15**(1), 9–32.

Okamura, H. and Dohi, T. (2013). SRATS: Software reliability assessment tool on spreadsheet, *Proceedings of The 24th International Symposium on Software Reliability Engineering* (*ISSRE 2013*), 100–117, IEEE CPS, Piscataway, New Jersey, USA.

Okamura, H. and Dohi, T. (2014). A novel framework of software reliability evaluation with software reliability growth models and software metrics, *Proceedings of The 15th IEEE International Symposium on High Assurance Systems Engineering* (*HASE 2014*), 97–104, IEEE CPS, Piscataway, New Jersey, USA.

Okamura, H. and Dohi, T. (2015). Towards comprehensive software reliability evaluation in open source software, *Proceedings of The 26th International Symposium on Software Reliability Engineering* (*ISSRE 2015*), 121–129, IEEE CPS, Piscataway, New Jersey, USA.

Okamura, H. and Dohi, T. (2016). Phase-type software reliability model: Parameter estimation algorithms with grouped data, *Annals of Operations Research*, **244**(1), 177–208.

Okamura, H., Etani, Y. and Dohi, T. (2010). A multi-factor software reliability model based on logistic regression, *Proceedings of The 21st International Symposium on Software Reliability Engineering* (*ISSRE 2010*), 31–40, IEEE CPS, Los Alamitos, California, USA.

Okamura, H., Dohi, T. and Osaki, S. (2013). Software reliability growth models with normal failure time distributions, *Reliability Engineering and System Safety*, **116**, 135–141.

Rinsaka, K., Shibata, K. and Dohi, T. (2006). Proportional intensity-based software reliability modeling with time-dependent metrics, *Proceedings of The 30th Annual International Computer Software and Applications Conference* (*COMPSAC 2006*), 405–410, IEEE CPS, Los Alamitos, California, USA.

Schick, G. J. and Wolverton, R. W. (1978). An analysis of competing software reliability models, *IEEE Transactions on Software Engineering*, **SE-4**(2), 104–120.

Shaked, M., Spizzichino, F. and Suter, F. (2002). Nonhomogeneous birth processes and l_∞ spherical densities, with applications in reliability theory, *Probability in the Engineering and Informational Sciences*, **16**(3), 271–288.

Shanthikumar, J. G. (1981). A general software reliability model for performance prediction, *Microelectronics and Reliability*, **21**(5), 671–682.

Shanthikumar, J. G. (1983). Software reliability models: A review, *Microelectronics and Reliability*, **23**(5), 903–943.

Shibata, K., Rinsaka, K. and Dohi, T. (2006). Metrics-based software reliability models using non-homogeneous Poisson processes, *Proceedings of The 17th International Symposium on Software Reliability Engineering* (*ISSRE 2006*), 52–61, IEEE CPS, Los Alamitos, California, USA.

Wagoner, W. L. (1973). The final report on a software reliability measurement study, Aerospace Corporation Report, TOR-0074 (4112)-1, Aerospace Corporation.

Xiao, X. (2015). NHPP-based software reliability model with Marshall-Olkin failure time distribution, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E98-A**(10), 2060–2068.

Xie, M. (1989). On a generalization of the J-M model, *Proceedings of Reliability 1989*, 5 Ba/3/1–5 Ba/3/7.

Xie, M. (1991). *Software Reliability Modeling*, World Scientific, Singapore.

Yamada, S., Ohba, M. and Osaki, S. (1983). S-shaped reliability growth modeling for software error detection, *IEEE Transactions on Reliability*, **R-32**(5), 475–478.

Zhao, M. and Xie, M. (1996). On maximum likelihood estimation for a general non-homogeneous Poisson process, *Scandinavian Journal of Statistics*, **23**(4), 597–607.

166 Proceedings of the Institute of Statistical Mathematics Vol. 73, No. 2, 147–166 (2025)

Some Aspects in Software Reliability Modeling from the Viewpoint of Non-homogeneous Markov Processes

Tadashi Dohi and Hiroyuki Okamura

Graduate School of Advanced Science and Engineering, Hiroshima University

This paper systematically summarizes the advancement of software reliability modeling and proposes a unified approach based on non-homogeneous Markov processes (NHMP). We generalize the conventional homogeneous Markov process (HMP) and non-homogeneous Poisson process (NHPP) based models, and introduces more flexible models such as generalized binomial process (GBP) and generalized Poisson process (GPP). Furthermore, we intyroduce a reliability prediction model using software product and process metrics, and verify its usefulness through an empirical study. These results contribute to improving software reliability assessment in actual software development, and expand the possibilities for future model development and applicability in industry.

Key words: Software reliability model, non-homogeneous Markov process, generalized binomial process, generalized Polya process, software metrics, reliability prediction.