2007－SE－158 (8)
2007／11／2

テスト工程におけるチェンジポイントとソフトウェア信頼性評価モデルに関する一考察

井上 真二 †, 山田 茂 †

† 鳥取大学工学部社会開発システム工学科

ソフトウェア開発プロセスにおけるテスト工程では，ソフトウェア故障発生時間間隔の確率的特徴が著しく変化する現象がしばしば観測される．このような現象が発生するテスト時刻はチェンジポイント（change-point）と呼ばれ，これは，これまでに提案されているソフトウェア信頼性モデルに基づいた信頼性評価精度低下の原因の１つとして考えられる．本研究では，このような問題を解決するための１つの回答として，チェンジポイント前後におけるソフトウェア故障発生現象の確率的特徴の違いを考慮できるソフトウェア信頼性モデル構築のための枠組みを提案する．また，その枠組みに基づいて，チェンジポイントを考慮したいくつかのソフトウェア信頼性モデルを構築する．本研究では，さらに，ソフトウェア開発管理面からの興味ある応用問題として，ソフトウェアコスト最小化の観点から最適なチェンジポイントおよびソフトウェアリリース（出荷）時刻を導くための最適方策についても議論する．最後に，本研究において構築したモデルと導出された最適方策について実測データを用いた適用例を示す．

On Change-Point and Software Reliability Assessment Models in Testing-Phase

Shinji Inoue † and Shigeru Yamada †

† Department of Social Systems Engineering, Faculty of Engineering, Tottori University

We often observe a phenomenon that a characteristic of a software failure-occurrence or the fault-detection phenomenon is changed notably in an actual testing-phase of a software development process. Testing-time when such phenomenon is observed is ordinarily called change-point. Taking the notion of the change-point into consideration in software reliability modeling is one of the important issues to develop accurate software reliability models. In this paper, we discuss a framework for software reliability modeling which enables us to consider with the difference of stochastic characteristics of the software failure-occurrence phenomenon after and before the change-point, and also develop several types of software reliability models based on the framework. Additionally, we discuss an optimal policy to derive optimal software release time and change-point simultaneously in terms of the software cost minimization. Finally, we show numerical examples of our models and the derived optimal policy by using actual data.

1 はじめに

これまでに提案されているソフトウェア信頼性評価技術の中で，ソフトウェア信頼度成長モデル（software reliability growth model: SRGM）[1,2]は，定量的なソフトウェア信頼性評価のための基盤技術の1つとして知られている．SRGMは，通常，テスト実行時間上におけるソフトウェア故障発生時間（もしくはソフトウェア故障発生時間間隔）を基本的な確率量として取り扱う．このような確率量は，本質的に，テスト実行時間に対するソフトウェア故障の瞬間的な発生頻度（ソフトウェア故障率）を表すハザードレート（hazard rate）と呼ばれる特性量によって特徴付けられる．これまでに提案されているSRGMの多くは，ソフトウェア故障発生時間の確率的性質がテスト期間中を通じて同一であるという仮定に基づいて構築される場合が多くを占める．しかしながら，実際のテスト工程では，ソフトウェア信頼度成長過程に影響を与える種々の要因に起因して，ソフトウェア故障率が変化することが起こりうる．テスト期間中においてソフトウェア故障発生パターンが変化するようなテスト時刻は，チェンジポイント（change-point: CP）と呼ばれる[3].

CPを考慮したソフトウェア信頼性評価手法に関するこれまでの議論では，大別すると，CPの発生要因として以下の2つの場合を想定した議論がなされている：**(1)** 納期遅れまたは納期までに目標とする

信頼度が達成不可能と予測されるとき，テスト期間後期におけるフォールト除去率の低下を抑制するとき，または，テスト計画の変更など，ソフトウェア開発管理者がCPを意図的に発生させる場合 [4–8].

(2) フォールト発見難易度，フォールトの独立性，モジュール毎のフォールト密度の違い，およびテスト技術者の学習過程により，CPが自然現象的に発生する場合 [8,9]. これらの研究成果では，CP前後におけるソフトウェア故障発生パターンの性質は基本的に同じであることに注意しなければならない．実際のテスト工程では，CP前後のソフトウェア故障発生パターン自体が変化することも十分に考えられる．

本研究では，チェンジポイントに起因する上述したような問題を解決する1つの回答として，有限のフォールトに対するデバッギング過程を記述するための統一的な枠組み [10, 11] に基づき，CPを考慮したSRGMの構築枠組みを提案する．ここで，簡単化のため，テスト期間を通してCPは1回だけ発生するものと仮定とする．また，提案したモデリング枠組みに基づいて，いくつかのSRGMを構築する．また，本研究では，CP後における1個当りのフォールト修正コストがCP以前の修正コストよりも高い場合において，総期待ソフトウェアコストを最小にする最適リリース時刻とCPを同時に推定するための最適方策について議論する．最後に，本研究において構築されたSRGMと導出されたソフトウェアの最適リリース方策について，実測データを用いた適用例を示す．

2 モデリング枠組み

ソフトウェア内に潜在するフォールト数が有限な場合を仮定したときに構築されるSRGMは，以下のデバッギング過程に対する仮定 [10, 11] により説明可能であることが知られている [12].

(仮定 1) ソフトウェア故障が発生した場合，その原因となるフォールトは，直ちにかつ完全に修正・除去される．

(仮定 2) 各ソフトウェア故障は，それぞれ，独立かつ時間に関してランダムに発生して，各ソフトウェア故障発生時刻は，それぞれ，同一の確率分布 $F(t) \equiv \Pr\{T \leq t\} = \int_0^t f(x)dx$ に従う非負の確率変数によって表される．ここで，$f(t)$ は確率密度関数を表す．

(仮定 3) テスト開始前にソフトウェア内に潜在する総フォールト数（初期潜在フォールト数）$N_0(>0)$ は，ある確率分布に従う確率変数とする．

いま，$\{N(t), t \geq 0\}$ を任意のテスト時刻 t までに発見された総フォールト数を表す確率過程であるとする．このとき，(仮定 1) ～ (仮定 3) から，初期潜在フォールト数が $N_0 = n$ である条件の下で，テスト時刻 t までに m 個のフォールトが発見される確率は，

$$\Pr\{N(t) = m \mid N_0 = n\} = \binom{n}{m} \{F(t)\}^m \{1 - F(t)\}^{n-m} \quad (1)$$

のように表される．したがって，テスト時刻 t までに m 個のフォールトが発見される確率は，次のように定式化される．

$$\Pr\{N(t) = m\} = \sum_n \binom{n}{m} \{F(t)\}^m \{1 - F(t)\}^{n-m} \Pr\{N_0 = n\} \quad (m = 0, 1, 2, \cdots). \quad (2)$$

よく知られた結果として，(仮定 3) において初期潜在フォールト数 N_0 が平均 $\omega(>0)$ のポアソン分布に従うと仮定した場合，式 (2) は，

$$\begin{aligned} \Pr\{N(t) = m\} &= \exp[-\omega] \frac{\{\omega F(t)\}^m}{m!} \sum_{n=m}^{\infty} \frac{\{w(1 - F(t))\}^{n-m}}{(n-m)!} \\ &= \frac{\{\omega F(t)\}^m}{m!} \exp[-\omega F(t)] \\ &\quad (m = 0, 1, 2, \cdots) \end{aligned} \quad (3)$$

となり，平均値関数 $\mathrm{E}[N(t)] = \omega F(t)$ をもつ非同次ポアソン過程（nonhomogeneous Poisson process: NHPP）と本質的に等価となる．ここで，$\mathrm{E}[\cdot]$ は期待値を表す．

2.1 ソフトウェア故障発生時間分布

式 (3) より，テスト工程におけるソフトウェア故障発生現象もしくはフォールト発見事象は，ソフトウェア故障発生時刻の確率的挙動を表す時間分布（ソフトウェア故障発生時間分布）$F(t)$ に対し，ある適切な確率分布関数を与えることで具体的に特徴付けられることがわかる．

ソフトウェア故障発生時間分布は，テスト時間に対するソフトウェア故障の瞬間的な発生頻度（ソフトウェア故障率）を表すハザードレートと呼ばれる特性量によって特徴付けることができる．ソフトウェア故障発生時間分布とハザードレートの関係は，一

般的に，次のように表される．

$$z(t) = \frac{f(t)}{1 - F(t)}. \tag{4}$$

式 (4) より，ソフトウェア故障発生時間分布は，

$$F(t) = 1 - \exp\left[-\int_0^t z(x)dx\right]. \tag{5}$$

と表される．したがって，ソフトウェア故障発生時間分布を与えることは，ハザードレートを与えることに帰着する．すなわち，現実のソフトウェア故障発生パターンに適したハザードレートを与えることで，様々なタイプの NHPP モデルを構築することができる．

ところで，ハザードレートは，それ自体が有する特徴によって，以下の 3 つに分類できることが知られている．

- 増加型フォールト発見率（increasing fault detection rate，以下 IFDR と略す）
- 一定型フォールト発見率（constant fault detection rate，以下 CFDR と略す）
- 減少型フォールト発見率（decreasing fault detection rate，以下 DFDR と略す）

最も典型的な例として，ハザードレートがテスト期間中と通じて一定の $\lambda(>0)$ である場合，ソフトウェア故障発生時間分布は，平均 $1/\lambda$ をもつ指数分布に従い，テスト時間区間 $(0,t]$ において発見される総期待フォールト数を表す NHPP の平均値関数 $H(t)$ は，

$$\begin{aligned} \mathrm{E}[N(t)] &= \omega F(t) \\ &= \omega(1 - \exp[-\lambda t]) \end{aligned} \tag{6}$$

となり，Goel-Okumoto モデル [13] と本質的に等価となる．NHPP モデルに関する上述したモデリング枠組みは，任意のテスト時刻までに発見される総期待フォールト数の時間的挙動を微分方程式により記述しながら導出する従来のモデリング枠組みよりも，テスト工程におけるデバッギング過程における物理的特徴を容易に理解する上で優れているものと考えられる．

2.2 チェンジポイントを考慮した SRGM

前述したモデリング枠組みに基づいて，テスト工程において発生する CP を考慮した SRGM の構築枠組みについて議論する．ただし，本研究では，簡単化のため，テスト期間 $(0,T]$ を通じて CP が 1 回発生する場合を想定する．これより，本研究では，CP 前後のハザードレートの違いを，

$$z(t) = \begin{cases} z_1(t) & (\text{for} \quad 0 \le t \le \tau) \\ z_2(t) & (\text{for} \quad t > \tau), \end{cases} \tag{7}$$

のように表現する．ここで，$\tau(0 < \tau < T)$ は CP を表す．これより，CP 前後のソフトウェア故障発生時間分布は，式 (5) より，

$$F(t) = \begin{cases} 1 - \exp[-\displaystyle\int_0^t z_1(x)dx] \\ \qquad\qquad (\text{for} \quad 0 \le t \le \tau) \\ 1 - \exp[-\displaystyle\int_0^\tau z_1(x)dx - \int_\tau^t z_2(x)dx] \\ \qquad\qquad (\text{for} \quad t > \tau) \end{cases} \tag{8}$$

のように記述される．式 (8) において，$0 \le t \le \tau$ におけるソフトウェア故障発生時間分布を $F_1(t)$，$t > \tau$ におけるそれを $F_2(t)$ とおくと，式 (3) および式 (8) から，CP を考慮した NHPP の平均値関数 $H(t)$ は，以下のように定式化できる．

$$\begin{aligned} H(t) &= \omega F(t) \\ &= \omega\{F_1(t)U_1(\tau - t) + F_2(t)U_2(t - \tau)\}. \end{aligned} \tag{9}$$

これより，NHPP の強度関数 $h(t)$ は，

$$\begin{aligned} h(t) &\equiv \frac{dH(t)}{dt} \\ &= \omega\{f_1(t)U_1(\tau - t) + f_2(t)U_2(t - \tau)\}. \end{aligned} \tag{10}$$

と求められる．ここで，$U_1(x)$ および $U_2(x)$ はステップ関数を表し，それぞれ，

$$U_1(x) = \begin{cases} 0 & (x < 0) \\ 1 & (x \ge 0), \end{cases} \tag{11}$$

$$U_2(x) = \begin{cases} 0 & (x \le 0) \\ 1 & (x > 0) \end{cases} \tag{12}$$

である．式 (8) および式 (9) より，CP 前後のソフトウェア故障発生時間分布に対するハザードレートをそれぞれ与えることで，CP を考慮した SRGM を構築することができる．

3 信頼性評価尺度

適用したSRGMから，定量的なソフトウェア信頼性評価に有用な種々の尺度が導出できる．これらは，ソフトウェア信頼性評価尺度と呼ばれ，本研究では，式(3)に示すソフトウェア故障発生現象に関する基本的仮定に基づいて導出される以下の3つの代表的な信頼性評価尺度について議論する．

3.1 期待残存フォールト数

期待残存フォールト数（expected number of remaining faults）は，任意のテスト時刻tにおけるソフトウェア内の期待残存フォールト数を表し，

$$\begin{aligned} M(t) &\equiv \mathrm{E}[\bar{N}(t)] \\ &= \mathrm{E}[N(\infty) - N(t)] \\ &= \omega - H(t) \end{aligned} \tag{13}$$

のように導出される．

3.2 ソフトウェア信頼度関数

ソフトウェア信頼度関数（software reliability function）は，一般的に，任意のテスト時刻tまでテストが進行しているという条件の下で，その後の時間区間$(t, t+x](t \geq 0, x \geq 0)$において，ソフトウェア故障が発生しない条件付確率として定義される[1]．すなわち，式(3)から，ソフトウェア信頼度関数$R(x \mid t)$は，

$$R(x \mid t) = \exp[-\{H(t+x) - H(t)\}] \tag{14}$$

のように導出される．

3.3 累積MTBF

累積MTBF（cumulative mean time between software failures）は，特に，ソフトウェア故障発生現象がNHPPに従う場合，すなわち，ソフトウェア故障発生時間間隔分布が通常の確率分布関数の性質を満たさない場合，平均ソフトウェア故障発生時間間隔の代替的尺度として用いられ，

$$MTBF_C(t) = \frac{t}{H(t)} \tag{15}$$

として与えられる．

4 パラメータ推定

本研究では，提案モデルに含まれるパラメータを最ゆう法に基づいて推定する．ここで，一定のテスト時間間隔$(0,\ t_k]$において発見された総フォールト数y_kに関するK組のフォールト発見数データ$(t_k,\ y_k)(k = 0, 1, 2, \cdots, K)$が観測されたものとする．このとき，チェンジポイントτが所与の下で，確率過程$\{N(t), t \geq 0\}$に関する対数ゆう度関数$\ln L(\boldsymbol{\theta} \mid \tau)$は，

$$\begin{aligned} &\ln L(\boldsymbol{\theta} \mid \tau) \\ &= \sum_{k=1}^{K} (y_k - y_{k-1}) \\ &\quad \cdot \ln[H(t_k; \boldsymbol{\theta} \mid \tau) - H(t_{k-1}; \boldsymbol{\theta} \mid \tau)] \\ &\quad - H(t_K; \boldsymbol{\theta} \mid \tau) - \sum_{k=1}^{K} \ln[y_k - y_{k-1}], \end{aligned} \tag{16}$$

のように求められる．ここで，$L(\boldsymbol{\theta} \mid \tau)$は確率過程$\{N(t), t \geq 0\}$に関する同時確率関数もしくは同時ゆう度関数を表しており，$\boldsymbol{\theta}$はモデル内部に含まれるパラメータを表す．これより，式(16)から導出できる同次対数ゆう度方程式

$$\frac{\partial \ln L(\boldsymbol{\theta} \mid \tau)}{\partial \boldsymbol{\theta}} = 0 \tag{17}$$

を，それぞれ，モデル内部に含まれるパラメータに対して数値的に解くことによって，パラメータ推定値を得ることができる．

5 ソフトウェア最適リリース問題

本研究では，CPが総期待ソフトウェアコストに与える影響を考慮しながら，総期待ソフトウェアコスト最小化の観点[16]から最適リリース政策を導出する．特に，テスト期間中にソフトウェア開発管理者がテスト計画の変更へテスト労力量を追加投入することなどによって意図的にCPを発生させる場合を想定して議論を行う．この場合，ソフトウェア開発管理者は，テスト期間中のどの時刻にCPを発生させるかという問題に直面する．すなわち，CP前後におけるフォールト1個当りの修正コストが異なる場合，CPが信頼性向上およびテスト実施期間に与える効果とテストコストに与える効果のトレードオフの関係が存在する．

上述した状況下におけるCPを考慮した総期待ソフトウェアコストを定式化するために，以下のようなコストパラメータを設定する．

- c_1：CP以前におけるテスト工程において発見されたフォールト1個当りの修正コスト，
- c_2：CP後におけるテスト工程において発見されたフォールト1個当りの修正コスト（$c_1 < c_2$），
- c_3：運用段階において発見されるフォールト1個当りの修正コスト（$c_2 < c_3$），
- c_4：単位テスト時間当りのテストコスト．

これより，テスト工程および運用段階において費やされる総期待ソフトウェアコスト $C(T,s)$ は，

$$C(T,s) = c_1 H_1(T-s) + c_2\{H_2(T) - H_1(T-s)\} + c_3\{a - H_2(T)\} + c_4 T \tag{18}$$

として定式化される．ここで，T はテスト終了時刻，s はテスト終了時刻から遡ったチェンジポイントまでのテスト期間を表す．コスト評価基準に基づいた場合の最適リリース時刻およびテスト終了時刻からチェンジポイントまでのテスト期間は，式 (18) を最小にする T^* および s^* を求めればよいことになる．すなわち，

$$\frac{\partial C(T,s)}{\partial T} = \frac{\partial C(T,s)}{\partial s} = 0 \tag{19}$$

を満たす解を求めればよい．

本研究では，最も単純な場合として，チェンジポイント $\tau(\equiv T-s)$ 前後におけるハザードレートがCFDRであり，パラメータがそれぞれ b_1 および $b_2(>b_1)$ の場合，すなわち，

$$\left.\begin{aligned} H_1(T) &\equiv \omega F_1(T) \\ &= \omega\{1-\exp[-b_1]T\} \qquad (0 \le T \le \tau) \\ H_2(T) &\equiv \omega F_2(T) \\ &= \omega\{1-\exp[-b_1(T-s)-b_2 T]\} \\ &\qquad (T > \tau) \end{aligned}\right\} \tag{20}$$

に対して，チェンジポイントを考慮したソフトウェアの最適リリース政策を導出する．まず，式 (19) より，式 (19) を満たす T は，次のように導出される．

$$T = T(s) = -\frac{1}{b_1}\log\left[\frac{c_4\exp[(b_2-b_1)s]}{ab_2(c_3-c_2)}\right]. \tag{21}$$

これより，s の存在範囲は，

$$s < \frac{\ln\left[\frac{ab_2(c_3-c_2)}{c_4}\right]}{b_2-b_1} (\equiv A) \tag{22}$$

となることがわかる．次に，式 (21) を式 (19) に代入して整理すると，s に関する方程式：

$$Z(s) \equiv \frac{b_1 c_4}{b_2}\left\{\left(\frac{b_2}{b_1}-1\right) - \frac{c_2-c_1}{c_3-c_2}\exp[b_2 s]\right\} = 0 \tag{23}$$

が得られる．$Z(s)$ は，s に関して単調減少関数であるため．式 (23) を満たす唯一の解が存在し，

$$s = \frac{1}{b_2}\ln\left[\frac{c_3-c_2}{c_2-c_1}\left(\frac{b_2}{b_1}-1\right)\right] \tag{24}$$

と求められる．以上をまとめると，コスト評価基準に基づいた以下のようなソフトウェア最適リリース政策を得る．

[最適政策]

$c_1 < c_2 < c_3$，$b_1 < b_2$，および

$A = \ln[ab_2(c_3-c_2)/c_4]/(b_2-b_1)$ とする．

(1) $Z(A) < 0$ ならば，$Z(s) = 0$ を満たす唯一の解 $s^*(<A)$ が存在して，s^* は，

$$s^* = \frac{b_1}{b_2}\ln\left[\frac{c_3-c_2}{c_2-c_1}\left(\frac{b_2}{b_1}+c_4\right)\right] \tag{25}$$

である．ただし，$Z(0) > 0$ ならば最適なチェンジポイントはテスト期間内に存在し，最適リリース時刻 T^* は，

$$T^* = -\frac{1}{b_1}\ln\left[\frac{c_4\exp[(b_2-b_1)s^*]}{ab_2(c_3-c_2)}\right] \tag{26}$$

である．$Z(0) \le 0$ ならば最適なチェンジポイントはテスト期間中に存在しない．

(2) $Z(A) \ge 0$ ならば $s < A$ 上において $Z(s) = 0$ を満たす解 s^* は存在せず，最適リリース時刻 T^* も存在しない．

6 適用例

実測データを用いて，提案したSRGMと導出したソフトウェア最適リリース政策に関する適用例を示す．本研究において用いた実測データは，DS1と称する27組のフォールト発見数データ $(t_k, y_k)(k = 0,1,2,\cdots,27;\ t_{27} = 27\ (\text{日}), y_{27} = 142)$ [1] およびDS2と称する19組のフォールト発見数データ $(t_k, y_k)(k = 0,1,2,\cdots,19;\ t_{19} = 19\ (\text{週}), y_{19} = 328)$ [15] である．

まず，**2** において議論したモデリング枠組みに基づき，チェンジポイント前後のハザードレートをそれぞれ仮定することで，チェンジポイントを考慮したSRGMを具体的に構築する．本研究では，基本的に，CFDRを表すハザードレートを b，IFDRを表すハザードレートを bt として，それらの組合せによって構築される4つのSRGMについて議論する．表1および表2に，DS1およびDS2に関して，本研究において取り上げたチェンジポイント前後のハザードレートの組合せ，それらのモデル含まれるパラメータの推定値，およびチェンジポイント τ をそれぞれ示す．また，表1および表2では，実用に最も多く供されている既存のSRGMとして，Goel-Okumotoモデルと呼ばれる指数形 (exponential) SRGM [13] および遅延S字形 (Delayed S-shaped) SRGM [16] について，MSEに基づいた適合性評価結果も示している．

表 1　チェンジポイントを考慮した SRGM と適合性比較結果. (DS1)

	$z_1(t)$	$z_2(t)$	ω	b_1	b_2	τ	MSE
Model 1	b_1	b_2	203.31	1.1339×10^{-2}	5.0151×10^{-2}	4	**10.1074**
Model 2	b_1	b_2t	155.33	3.8819×10^{-2}	6.4147×10^{-3}	6	60.0533
Model 3	b_1t	b_2	232.09	7.6885×10^{-3}	3.7898×10^{-2}	7	10.2076
Model 4	b_1t	b_2t	150.48	1.4857×10^{-2}	7.8027×10^{-3}	3	30.2325
Exponential	b_1		434.31	1.4664×10^{-2}	—	—	39.9071
Delayed S-shaped	$b_1^2t/(1+b_1t)$		171.14	1.1878×10^{-1}	—	—	12.1935

表 2　チェンジポイントを考慮した SRGM と適合性比較結果. (DS2)

	$z_1(t)$	$z_2(t)$	ω	b_1	b_2	τ	MSE
Model 1	b_1	b_2	348.60	7.8415×10^{-2}	2.1229×10^{-1}	9	**58.1106**
Model 2	b_1	b_2t	339.88	7.8403×10^{-2}	1.7983×10^{-2}	7	59.9907
Model 3	b_1t	b_2	466.55	6.5359×10^{-2}	6.5638×10^{-2}	1	188.754
Model 4	b_1t	b_2t	334.66	1.8565×10^{-2}	2.1707×10^{-2}	1	43.6099
Exponential	b_1		513.15	5.3653×10^{-2}	—	—	222.094
Delayed S-shaped	$b_1^2t/(1+b_1t)$		359.92	2.1263×10^{-1}	—	—	188.852

ここで，チェンジポイント τ は，**4** において議論したパラメータ推定手法に基づいて推定された SRGM の平均偏差平方和 (mean squared error: MSE) が最小となるテスト時刻として取り扱っている．MSE は，K 組のフォールト発見数データが観測されている場合，ある一定のテスト時刻 $t_k(k=1,2,\cdots,K)$ までに発見された総フォールト数の実測値 y_k と推定値 $\widehat{y}(t_k)$ との偏差 2 乗和をデータ数で平均化したものであり，

$$\mathrm{MSE}=\frac{1}{K}\sum_{k=1}^{K}\{y_k-\widehat{y}(t_k)\}^2 \qquad (27)$$

として算出される．

表 1 および表 2 から，SRGM の構築においてチェンジポイントを考慮することが，SRGM の適合性向上に寄与していることが伺えられ，特に，表 2 では，その効果を顕著に伺うことができる．また，これらの表から，たとえチェンジポイントを考慮した SRGM を構築してもチェンジポイント前後のハザードレートをどのように仮定しているかによって，実測データに対する適合性が大きく左右することがわかる．

表 1 において，適用した実測データに対する適合性が最も高かった "Model 1" を用いてソフトウェア信頼性解析例を示す．図 1 に，推定された NHPP の

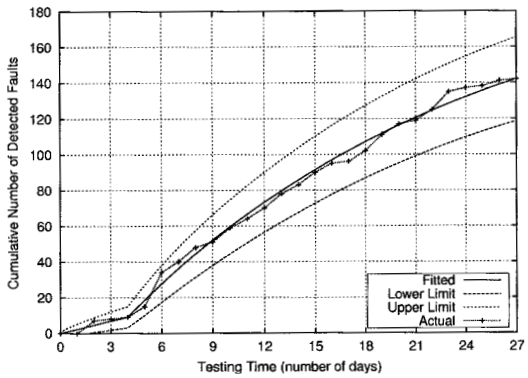

図 1　推定された NHPP の平均値関数とその 95% 信頼限界. (DS1, Model 1; $\tau=4$(日))

平均値関数とその 95% 信頼限界を示す．ただし，推定された発見フォールト数の期待値 $\widehat{\mathrm{E}}[N(t)]\equiv\widehat{H}(t)$ の 100γ% 信頼限界は，$\{N(t),t\geq0\}$ が NHPP に従う場合，

$$\widehat{H}(t)\pm K_\gamma\sqrt{\widehat{H}[(t)]} \qquad (28)$$

として計算される [16]．ここで，K_γ は標準正規分布の $100(1+\gamma)/2$ パーセント点を表す．このような信頼限界を求めておくことは，テスト工程管理上，極めて有効である．図 2 に，推定された式 (13) の期待残

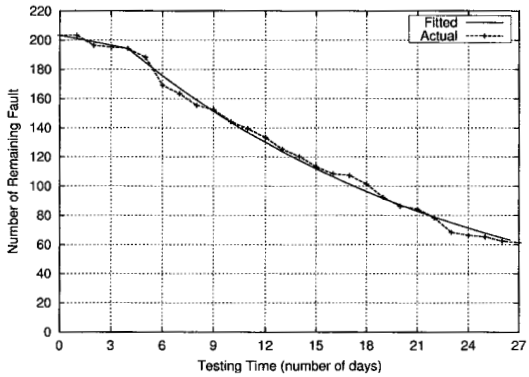

図 2　推定された期待残存フォールト数．(DS1, Model 1; $\tau = 4$(日))

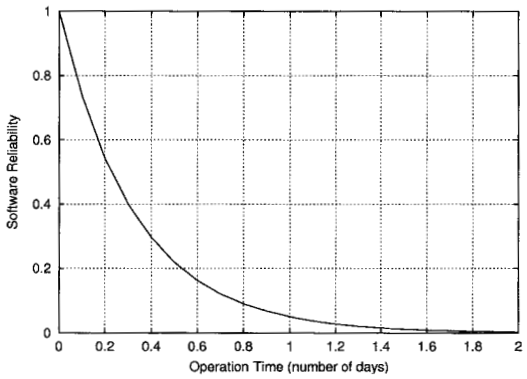

図 3　推定されたソフトウェア信頼度関数．(DS1, Model 1)

存フォールト数 $\widehat{M}(t)$ を表す．図 2 から，テスト終了時刻 ($t = 27$(日)) においてソフトウェア内に残存している総期待フォールト数 $\widehat{M}(27)$ は約 $61.307 \approx 62$ 個と推定される．また，図 3 に，推定された式 (14) のソフトウェア信頼度関数 $\widehat{R}(x \mid 27)$ を示す．図 3 から，テスト終了時刻 $t = 27$（日）に当該ソフトウェアがリリースされ，そのソフトウェアがテスト工程と同様の環境で運用された場合，運用 1 日目におけるソフトウェア信頼度 $\widehat{R}(1.0 \mid 27)$ は，約 0.0498 と推定される．さらに，図 4 に，推定された式 (15) の累積 MTBF を示す．図 4 より，テスト終了時刻における累積 MTBF は約 0.1901 と推定される．

次に，**5** において導出されたソフトウェアの最適リリース政策に関する適用例を示す．ここでは，ソフトウェアコストの相対的大きさについて議論するため，c_1 を標準値 ($c_1 = 1$) とする．また，適用するモデルは，チェンジポイント前後のハザードレートが共に CFDR である “Model 1”であり，DS1 を用いて推定されたパラメータ推定値 $\widehat{\omega} = 203.31$, $\widehat{b}_1 = 1.1339 \times 10^{-2}$，および $b_2 = 5.0151 \times 10^{-2}$ を用いる（表 1 を参照）．図 4 に，$c_1 = 1$，$c_2 = 5$，$c_3 = 25$，および $c_4 = 10$ のときの総期待ソフトウェアコスト $C(T, s)$ の挙動を示す．このとき，$A = 77.687$ であり $Z(A) = Z(77.687) \approx -14.512 (< 0)$ より，**最適政策** (1) が適用されることがわかる．ここで，$Z(0) \approx 7.2868 (> 0)$ より，テスト終了時刻から遡ったチェンジポイントまでの最適テスト期間 s^* は約 56.628（日），最適リリース時刻 T^* は約 72.082 (週)，およびこのときの総期待ソフトウェアコスト $C(T^*, s^*)$ は約 1806.03 と推定される．

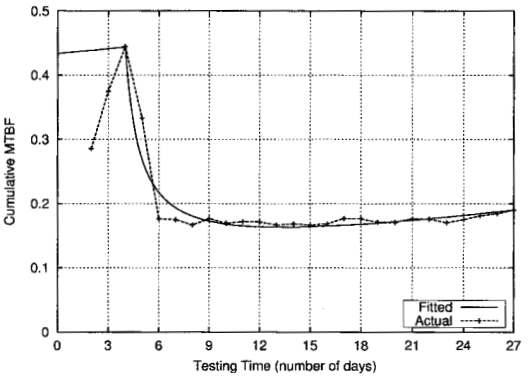

図 4　推定された累積 MTBF．(DS1, Model 1; $\tau = 4$(日))

7　おわりに

今回議論したモデリング手法は，チェンジポイント前後のハザードレートを適切に仮定することによって，今回取り扱った SRGM 以外の様々な SRGM を構築することができる．また，同様なモデリング枠組みに基づいて，テスト期間中に発生する複数のチェンジポイントを考慮した SRGM へと拡張することができる．今後は，統計的手法に基づいたチェンジポイントの同定手法の開発と共に，より多くの実測データを用いた提案モデルの有効性および適合性の検証を行う必要がある．また，ソフトウェア最適リリース問題に対しては，実際のテスト工程において設定されるソフトウェア信頼度目標や納期を考慮しながら，現実の状況をより反映した状況下での問題についても議論を行う必要がある．

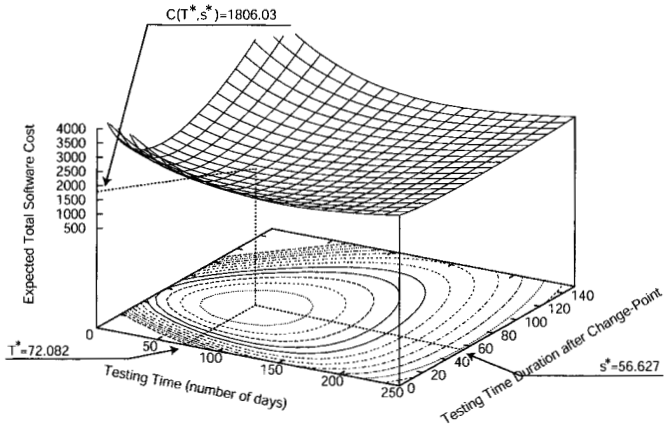

図 5　総期待ソフトウェアコスト. (DS1, Model 1; $c_1 = 1$, $c_2 = 5$, $c_3 = 25$, および $c_4 = 10$)

参考文献

1) 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, 東京, 1994.

2) 山田茂, 福島利彦：品質指向ソフトウェアマネジメント, 朝倉書店, 東京, 2007.

3) M. Zhao： Change-point problems in software and hardware reliability, *Commun. Statist. — Theory Meth.*, Vol. 22, No. 3, pp. 757–768, 1993.

4) 大寺浩志, 山田茂, 成久洋之: ソフトウェア信頼度成長モデルによるテスト工程管理, 電子情報通信学会論文誌, Vol. J70-D, No. 5, pp. 889–895, 1987.

5) H. Ohtera and S. Yamada: Optimal allocation & control problems for software-testing resources, *IEEE Trans. Reliab.*, Vol. 39, No. 2, pp. 171–176, 1990.

6) C.Y. Huang: Performance analysis of software reliability growth models with testing-effort and change-point, *J. Sys. Softw.*, Vol. 76, No. 2, pp. 181–194, 2005.

7) C.Y. Huang: Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency, *J. Sys. Softw.*, Vol. 77, No. 2, pp. 139–155, 2005.

8) J. Zhao, H.W. Liu, G. Cui, and X.Z. Yang: Software reliability growth model with change-point and environmental function, *J. Sys. Soft.*, Vol. 79, No. 11, pp. 1578–1587, 2006.

9) F.Z. Zou: A change-point perspective on the software failure process, *Softw. Test. Verf. Reliab.*, Vol. 13, No. 2, pp. 85–93, 2003.

10) N. Langberg and N.D. Singpurwalla: A unification of some software reliability models, *SIAM J. Scient. Comput.*, Vol. 6, No. 3, pp. 781–790, 1985.

11) H. Joe: Statistical inference for general-order-statistics and nonhomogeneous-Poisson-process software reliability models, *IEEE Trans. Softw. Eng.*, Vol. 15, No. 11, pp. 1485–1490, 1989.

12) 岡村寛之, 渡部保博, 土肥正, 尾崎俊治：EMアルゴリズムに基づいたソフトウェア信頼性モデルの推定, 電子情報通信学会論文誌, Vol. J85-A, No. 4, pp. 442–450, 2002.

13) A.L. Goel and K. Okumoto: Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliab.*, Vol. R-28, No. 3, pp. 206–211, 1979.

14) S. Yamada and S. Osaki: Cost-reliability optimal release policies for software systems, *IEEE Trans. Reliab.*, Vol. R-34, No. 5, pp. 422–424, 1985.

15) M. Ohba: Software reliability analysis models, *IBM J. Res. Dev.*, Vol. 28, No. 4, pp. 428–443, 1984.

16) S. Yamada, M. Ohba, and S. Osaki: S-shaped reliability growth modeling for software error detection, *IEEE Trans. Reliab.*, Vol. R-32, No. 5, pp. 475–478,485, 1983.