数理解析研究所講究録
第1912巻 2014年 53-58

フォールト数の変化を考慮したソフトウェア信頼性モデルにおける最適リリース問題

大阪府立大学大学院　理学系研究科　情報数理科学専攻
大西 健司　北條 仁志
Kenji Onishi　Hitoshi Hohjo
Department of Mathematics and Information Sciences,
Graduate School of Science,
Osaka Prefecture University

1　はじめに

ソフトウェア開発の最終工程であるテスト工程は、ソフトウェアシステムの品質・信頼性を確認する工程である。したがって高い信頼性を持つソフトウェアを開発するうえで、品質・信頼性を評価を正確に行うことが重要である。ソフトウェア信頼性モデル [11] は定量的にソフトウェア信頼性評価を行う基盤技術であることが知られている。特にフォールト発生事象やソフトウェア故障発生現象を確率や統計を用いてモデル化した動的モデルは、ソフトウェア信頼度成長モデル (software reliability growth model, 以下 SRGM と略す) [3] [6] [7] と呼ばれ、数多く実用されているソフトウェア信頼性モデルの一つである。

これまでに提案されてきた多くの SRGM では、ソフトウェア故障発生時間の確率的性質が、テストの期間中において同一であるという仮定の下で提案されてきた。しかし様々な要因によって、ソフトウェア故障発生時間間隔の確率的性質は変化すると考えられる。そこで、テスト期間中にソフトウェア故障発生パターンが変化する時刻をチェンジポイントとして、チェンジポイントの影響を考慮したモデルが提案されてきた。チェンジポイントが発生する要因として、以下の 2 つのように大別されて研究が行われてきた。(1) ソフトウェア開発管理者が納期までに間に合わないと判断して、意図的にチェンジポイントを発生させる [1] [5] [8] 。(2) フォールトの発見難易度の変化や独立性、テスト技術者の学習能力によって、自然発生的にチェンジポイントが発生する [8] [9] 。これらのチェンジポイントを用いた研究では、テスト工程と他の工程が同時進行することは許されていなかった。しかしながら実際のソフトウェア開発においては、時間短縮のために開発が最後まで終わらないうちにテストを開始すると考えられる。

本研究では、最後までソフトウェアの開発が終わらない状態で、テストを開始したときのモデルについて議論する。特にテストが開始された時に、同時にコーディング工程も行われているモデルについて考える。コーディング工程で書かれるプログラムの量は時間に対して一定の割合で増えると考えられ、書かれたプログラムの量に対して一定の割合でソフトウェア内部のフォールト数は増えると考えられる。この時、ソフトウェア内部のフォールト数は時間に対して一定の割合で増える。2 章では、以上のような環境を考慮したモデルの仮定について説明を行い、 3 章でモデルの数学的定式化を行う。4 章では、3 章で定式化したモデルを用いて最適リリース問題の解を求める。5 章では、実際に測定されたデータを用いて、本研究のモデルに適用した数値例と最適リリース問題の数値例を示す。最後に、 6 章で本研究についてのまとめを述べる。

2 モデル

本研究では、ソフトウェア信頼性モデルを非同次ポアソン過程 (nonhomogeneous Poisson process, 以下 NHPP と略す) モデルを用いて定式化する。時間 t までに発見される総期待フォールト数あるいは発生するソフトウェア故障数を表す確率過程 $\{N(t), t \geq 0\}$ は、

$$Pr\{N(t) = n\} = \frac{\{m(t)\}^n}{n!} e^{-m(t)} \quad (n = 0, 1, 2, ...)$$

$$m(t) = \int_0^t f(x)dx \quad (t \geq 0)$$

に従うとする。

ここで、$m(t) \equiv E[N(t)]$ は NHPP の平均値関数と呼ばれ、時間区間 $(0, t]$ において発見されるフォールト数あるいは発生するソフトウェア故障数の期待値を表す。また、$f(t)$ は NHPP の強度関数と呼ばれ、時刻 t における瞬間フォールト発見率あるいはソフトウェア故障率を表す。

本研究では次のような仮定を持つ NHPP モデルを扱う。

1. 時間 0 にテストが開始されてから時間 τ までは開発が同時に行われる期間とし、時間 τ 以降は開発は行われないとする。
2. テスト開始時にソフトウェア内部には c 個のフォールトが存在し、開発が同時に行われている期間は一定の割合でフォールト数が増える。
3. 時間 τ 以降はテストのみが行われるため、ソフトウェア内部のフォールト数は変化しない。このときのフォールト数を a 個とする。
4. 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する。
5. 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瞬時にかつ完全に修正・除去される。
6. 単位時間当りに発見されるフォールト数は、その時点でソフトウェア内に残存するフォールト数に比例する。
7. 1 個当りのフォールト発見率は時間に対して一定である。

仮定から、開発が同時に行われる期間 $(0 \leq t < \tau)$ において、ソフトウェア内部のフォールト数は $\frac{a-c}{\tau}t + c$ で表され、テストのみが行われる期間 $(t \geq \tau)$ のフォールト数は a で表される。

3 数学的定式化

2 章で述べた仮定の下で、ソフトウェア信頼度成長モデルの定式化を行う。開発とテストが同時に行われている期間 $(0 \leq t < \tau)$ について、時間 t までに修正・除去されたフォールト数を $m_1(t)$ とすると、

$$\frac{dm_1(t)}{dt} = b((\frac{a-c}{\tau}t + c) - m_1(t))$$

が成り立つ。ここで、b は 1 個当りのフォールト発見率を表し、$(\frac{a-c}{\tau}t + c) - m_1(t)$ は時間 t での残存フォールト数を表す。初期条件 $m_1(0) = 0$ の下でこの微分方程式を解くと

$$m_1(t) = \frac{a-c}{\tau}t + (c - \frac{a-c}{b\tau})(1 - e^{-bt})$$

となる。

次に、開発が終わり、テストのみが行われる期間 $(t \geq \tau)$ について、時間 t までに修正・除去されたフォールト数を $m_2(t)$ とすると、

$$\frac{dm_2(t)}{dt} = b(a - m_2(t))$$

が成り立つ。発見されたフォールト数は開発が終わった瞬間に連続であると考えられるため、条件 $m_1(\tau - 0) = m_2(\tau)$ の下でこの微分方程式を解くと

$$m_2(t) = a(1 - e^{-b(t-\tau)}) + (a - c + (c - \frac{a-c}{b\tau})(1 - e^{-b\tau}))e^{-b(t-\tau)}$$

となる。したがって、平均値関数 $m(t)$ は

$$m(t) = \begin{cases} \frac{a-c}{\tau}t + (c - \frac{a-c}{b\tau})(1 - e^{-bt}) & (0 \leq t < \tau) \\ a(1 - e^{-b(t-\tau)}) + (a - c + (c - \frac{a-c}{b\tau})(1 - e^{-b\tau}))e^{-b(t-\tau)} & (t \geq \tau) \end{cases}$$

と表される。

4 最適リリース問題

総期待ソフトウェアコストを定式化するために、次のようなコストパラメータを設定する。テスト工程において発見されるフォールト1個当りの修正コストを c_1、保守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を $c_2(0 < c_1 < c_2)$ とする。また、時間当りのテストコストを $c_3 > 0$ とする。本研究のモデルでは、開発が途中で打ち切られてリリースされるケースが存在する。その際には満足する商品をリリースできなかったため、$c_4 > 0$ のコストがかかる。c_4 の値はソフトウェアの完成度に関係なく、一定であるとする。この時、総期待ソフトウェアコスト $C(T)$ は

$$C(T) = \begin{cases} C_1(T) = m(T) \times c_1 + (a - m(T)) \times c_2 + \int_0^T c_3 dt + c_4 & (0 \leq T < \tau) \\ C_2(T) = m(T) \times c_1 + (a - m(T)) \times c_2 + \int_0^T c_3 dt & (T \geq \tau) \end{cases}$$

で定式化できる。ここで、T はテスト終了時刻を表す。我々の目的は総期待ソフトウェアコスト $C(T)$ を最小にする最適リリース時刻 T^* を求めることである。

4.1 開発が終わっていない状態でのリリース時刻

この節では、開発が終わっていない状態での総期待ソフトウェアコスト $C_1(T)$ を最小にする最適リリース時刻 $T_1^*(0 \leq T_1^* < \tau)$ を求める。$C_1(T)$ の1階および2階導関数は

$$\frac{dC_1(T)}{dT} = -(bc - \frac{a-c}{\tau})(c_2 - c_1)e^{-bT} - \frac{a-c}{\tau}(c_2 - c_1) + c_3$$

$$\frac{d^2C_1(T)}{dT^2} = b(bc - \frac{a-c}{\tau})(c_2 - c_1)e^{-bT}$$

であり、最適リリース時刻 T_1^* は以下のようになる。

(Ⅰ) $bc - \frac{a-c}{\tau} \leq 0$ ならば $T_1^* = argmin\{C_1(0), C_1(\tau - 0)\}$

(Ⅱ) $bc - \frac{a-c}{\tau} > 0$ ならば

(ⅰ) $0 \leq -\frac{1}{b}logA_1 < \tau$ の時、$T_1^* = -\frac{1}{b}logA_1$

(ⅱ) $-\frac{1}{b}logA_1 < 0$ の時、$T_1^* = 0$

(ⅲ) $-\frac{1}{b}logA_1 \geq \tau$ の時、$T_1^* = \tau - 0$

但し、$A_1 = \frac{-\frac{a-c}{\tau}(c_2-c_1)+c_3}{(bc-\frac{a-c}{\tau})(c_2-c_1)}$ とする。

4.2 開発が終わった状態でのリリース時刻

この節では、開発が終わった状態の総期待ソフトウェアコスト $C_2(T)$ を最小にする最適リリース時刻 $T_2^*(\tau \leq T_2^*)$ を求める。$C_2(T)$ の 1 階および 2 階導関数は

$$\frac{dC_2(T)}{dT} = \{-bc - \frac{a-c}{\tau}(e^{b\tau}-1)\}(c_2-c_1)e^{-bT} + c_3$$

$$\frac{d^2C_2(T)}{dT^2} = b\{bc + \frac{a-c}{\tau}(e^{b\tau}-1)\}(c_2-c_1)e^{-bT}$$

となる。すべての $T \geq \tau$ に対して $\frac{d^2C_2(T)}{dT^2} > 0$ であるので、$C_2(T)$ は T に関して狭義凸関数である。従って、最適リリース時刻 T_2^* は以下のようになる。

(I)$-\frac{1}{b}logA_2 \geq \tau$ ならば $T_2^* = -\frac{1}{b}logA_2$
(II)$-\frac{1}{b}logA_2 < \tau$ ならば $T_2^* = \tau$
但し、$A_2 = \frac{c_3}{(c_2-c_1)\{bc+\frac{a-c}{\tau}(e^{b\tau}-1)\}}$ とする。

4.3 最適リリース時刻

4.1 節および 4.2 節において各条件下での最適リリース時刻を求めた。それらの結果から、最適リリース時刻 T^* は

$$T^* = argmin\{C_1(T_1^*), C_2(T_2^*)\}$$

で表される。

5 数値例

この章では、実際のテスト工程において観測されたデータを適用した数値例を示す。本論文では、Pham [7] の *Data Set* ♯8 を用いて最小二乗法によりパラメータ推定を行う。その結果、パラメータの値は $a = 476.32$、$c = 100.09$、$b = 0.098$、$\tau = 44.41$ となる。実際のデータでは、ソフトウェア内のフォールトの数は 481 個であるので、モデルと実際のデータの値が近い値となっている。

パラメータの値をもとにして最適リリース問題を解く。$c_1 = 50$、$c_2 = 100$、$c_3 = 10$、$c_4 = 10000$ に対するコスト関数は、図 1 のようになる。図において横軸は時間、縦軸は総期待ソフトウェアコストを表す。この時の最適リリース時間とそれに対応する総期待ソフトウェアコストは $T^* = 82.49$、$C(T^*) = 24742.07$ となる。これは開発が終わった後にリリースしている。

次に指数形ソフトウェア信頼度成長モデルと遅延 S 字形ソフトウェア信頼度成長モデルとの比較を行う。先ほどと同様のデータ、手法を用いてパラメータ推定を行い、最適リリース問題を解くと、指数形ソフトウェア信頼度成長モデルの最適リリース問題の結果は、$T^* = 164.51$、$C(T^*) = 28933.6$ となる。遅延 S 字形ソフトウェア信頼度成長モデルの最適リリース問題の結果は、$T^* = 106.12$、$C(T^*) = 25638.4$ となる。本研究で提案したモデルでは、指数形ソフトウェア信頼度成長モデルと遅延 S 字形ソフトウェア信頼度成長モデルより総期待ソフトウェアコストが下がっている。

最後に開発が終わる前にリリースするケースの数値例を示す。Pham [7] の *Table* 6.9 を用いて最小二乗法によりパラメータ推定を行うと、$a = 93.48$、$c = 55.32$、$b = 0.57$、$\tau = 5.98$ となる。このデータから最適リリース問題を解く。$c_1 = 110$、$c_2 = 120$、$c_3 = 100$、$c_4 = 20$ の時のコスト関数は、図 2 のようになる。この時の最適リリース時間とそれに対応する総期待ソフトウェアコストは $T^* = 3.41$、$C(T^*) = 10983.6$ となる。

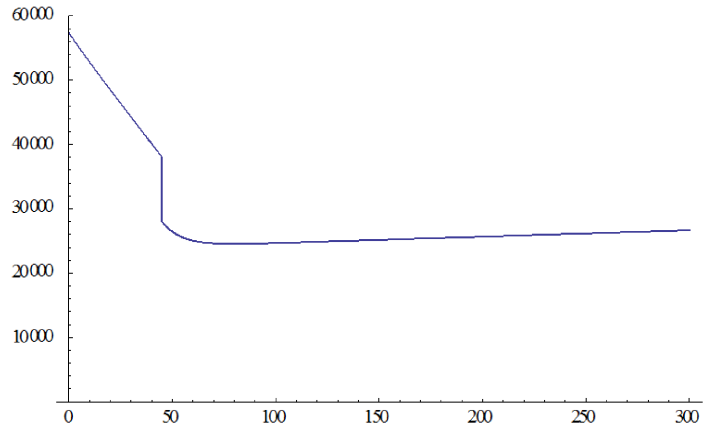

図 1: 開発が終わった後に最適解がくるケース

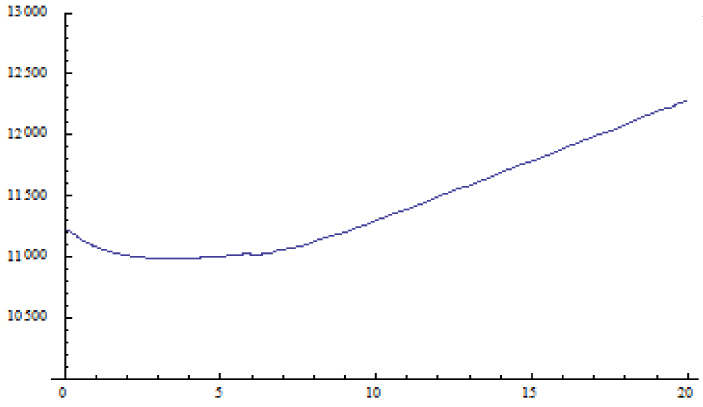

図 2: 開発が終わる前に最適解がくるケース

6 まとめ

本研究では、テストと開発が同時に行われる期間があるソフトウェア信頼性モデルの提案を行った。NHPP モデルによる定式化を行うため、テスト時間に対する発見フォールト期待数を導出した。また、このモデルを用いた最適リリース問題を提案し、総期待ソフトウェアコストが最小となる最適リリース時刻を求めた。数値例では、最適解が開発が終わる前と開発が終わった後の 2 つのケースを与え、2 つの代表的な既存モデルとの比較を行った。今後の拡張としては、テスト労力を考慮した場合の研究などが考えられる。

参考文献

[1] Huang,C.Y., Performance analysis of software reliability growth models with testing-effort and change-point, The Journal of Systems and Software, Vol.76(2005)181-194.

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

[3] Huang,C.Y., M.R.Lyu., S.Y.Kuo, A unified scheme of some nonhomogenous poisson process models for software reliability estimation, IEEE Transactions on Software Engineering, Vol.29, No.3(2003)261-269.

[4] Lin,C.T., Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency, The Journal of Systems and Software, Vol.77(2005)139-155.

[5] Lin,C.T., C.Y.Huang., Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models, The Journal of Systems and Software, Vol.81(2008)1025-1038.

[6] Mohd,R., M.Nazir., Software reliability growth models: overview and applications, Journal of Emerging Trends in Computing and Information Sciences, Vol.3, No.9(2012)1309-1320.

[7] Pham,H., System Software Reliability, Springer, 2007.

[8] Zhao,J., H.W.Lin., G.Cui., X.Z.Yang., Software reliability growth model with change-point and environmental function, The Journal of Systems and Software, Vol.79(2006)1578-1587.

[9] Zou,F.Z., A change-point perspective on the software failure process, Software Testing, Verification and Reliability, Vol.13(2003)85-93.

[10] 井上　真二, 山田　茂, チェンジポイントを考慮したソフトウェア信頼性評価法に関する一考察, 電子情報通信学会論文誌, Vol.107, No.43(2007)25-30.

[11] 山田　茂, ソフトウェア信頼性の基礎—モデリングアプローチ, 共立出版, 2011.