バグ修正時間を考慮したソフトウェア最適リリース問題についての一考察

岡村 寛之
広島大学大学院工学研究科
okamu @ hiroshima-u.ac.jp

住田 大亮
広島大学大学院工学研究科

土肥 正
広島大学大学院工学研究科
dohi @ hiroshima-u.ac.jp

要旨

本稿では，バグ修正難易度を表す指標としてバグ修正時間を用いたソフトウェア最適リリース問題の再考を行う．特に，従来のバグ修正時間を考慮したソフトウェア信頼性モデルにおいて，バグ発見時刻と修正時間が独立であるという仮定を修正し，バグ発見時刻と修正時間の相関を表現することができる一般的なモデルフレームワークを取り上げ，そのモデルフレームワークにおけるソフトウェア最適リリース問題について考察する．

1. はじめに

ソフトウェア信頼性はソフトウェア品質における最も重要な指標の一つである．定量的なソフトウェア信頼性はある期間にバグが発見されない（システム障害が発生しない）確率として定義される．これまでに，定量的なソフトウェア信頼性を評価するための確率モデルとしてソフトウェア信頼性モデル（ソフトウェア信頼度成長モデル）が提案され，ソフトウェア開発管理において様々な側面で利用されている [6, 4, 5, 9]．

従来のソフトウェア信頼性モデルではテスト工程において発見されるバグ数に着目したモデリングとなっている．一方，現在のソフトウェア開発ではオープンソースソフトウェア（OSS: Open Source Software）で見られるように，継続的インテグレーションにより常にソフトウェアが利用できる状態を保持するスタイルが主流となっている．そのような開発形態では，各リリースでそこまでに発見したバグがすべて修正されているとは限らない．つまり，バグ発見だけではなく，バグ発見，バグ特定，バグ修正などのバグに対するライフサイクルを考慮した信頼性評価が必要となる．

バグ修正を考慮したソフトウェア信頼性モデルは，これまでにもいくつも議論されている．Schneidewind [10, 11], Xie ら [13, 12], Gokhale ら [1, 2] は累積修正バグ数を表現する確率モデルを構築している．Xie ら [12] はバグ発見とバグ修正を統合するモデルの構築をしている．これら，累積バグ発見と累積バグ修正がともに非同次ポアソン過程に従うという特徴をもつ．また，Okamura and Dohi [7] は上述のすべてのモデルを含むバグ修正モデルフレームワークを提案している．特に，文献 [7] では，バグ発見時刻とバグ修正時間の相関も考慮した一般的なモデルフレームワークを提案している．ここで言うバグ発見時刻と修正時間時間の相関は，テストの早い段階で見つかったバグほど修正時間が短い（あるいは長い），または，テスト終盤で見つかるバグほど修正時間が短い（あるいは長い）と言った関係を定量的に表しており，これは，どのモジュールを先にテストするかなどのテスト戦略と密接に関係している．文献 [7] では，現在までに見つかっていないバグ数だけではなく，現在までに見つかっているが修正されていないバグ数を考慮した信頼性指標の提案を行っている．

本稿では，上述のバグ発見時刻とバグ修正時間の相関を考慮したモデル上でのソフトウェアリリース問題を扱う．ソフトウェア開発において，経済的な観点からソフトウェアテストを終了しソフトウェアのリリースを行う適切なタイミングを決定することは非常に重要である．これはソフトウェアリリース問題と呼ばれ，古くからソフトウェア信頼性モデルに基づいた数理的なアプローチが数多く行われている．Okumoto and Goel [8] はバグ発見時刻のみを考慮した指数形ソフトウェア信頼性モデルを利用して，テスト費用，テスト期間中のバグ修正費用，リリース後のバグ修正費用の期待値を算出し，その総費用を最小にする最適なリリース時間の決定を行って

いる．Okumoto and Goel [8] 以降，様々な要因を取り入れたリリース問題の定式化がなされているが，そのほとんどがバグ発見時刻だけに着目したものである．つまり，バグ発見後，即座にバグが修正される仮定を考えている．しかしながら，現実のソフトウェア開発にでは様々なバグが存在し，その修正難易度も大きく異なる．一般的にバグ修正では，バグの原因の特定，プログラムの修正，修正箇所のテスト（回帰テスト）が行われ，再現性の低いバグでは原因特定に時間を要することがある．バグ発見だけに着目したソフトウェアリリース問題ではこのようなバグの修正難易度にかかわらず一律のバグ修正費用を課すことが多く，バグ修正がリリース時間に与える影響について明らかになっていない．そこで，本稿では，バグ修正時間を考慮したモデルにおけるソフトウェア最適リリース問題を再考する．特に，一般的なバグ修正時間を考慮したモデルフレームワーク [7] におけるソフトウェアリリース問題を再定義し，バグ修正時間ならびにバグ発見時刻とバグ修正時間の相関が最適なリリース時刻に与える影響について調べ，経済的な観点からどのようなテスト戦略が良いかについて議論する．

2. ソフトウェア信頼性モデル

2.1. バグ発見時刻によるソフトウェア信頼性モデル

従来のソフトウェア信頼性モデル（ソフトウェア信頼度成長モデル）は，開発工程における累積バグ発見数を確率過程で表現し，現時点での残存バグ数や将来のバグ発見に関する評価・予測を行うモデルである．非同次ポアソン過程（NHPP: non-homogeneous Poisson process）に基づくソフトウェア信頼性モデルは以下の仮定から構築される．

- テスト前にソフトウェア内に潜在するバグ数 M は，平均 ω のポアソン分布に従う．
- M 個のバグの発見時刻は独立かつ同一に分布する確率変数 $\{X_1, \cdots, X_M\}$ で表され，その累積分布関数を $F(t)$ とする．

いま，時刻 $t = 0$ でテストを開始したとき，時刻 t までの累積バグ発見数を $\{N(t), t \geq 0\}$ とすると，上記の仮定から以下の確率関数が得られる．

$$\begin{aligned} P(N(t) = n) &= \sum_{m=n}^{\infty} P(N(t) = n | M = m) P(M = m) \\ &= \sum_{m=n}^{\infty} \binom{m}{n} F(t)^n (1 - F(t))^{m-n} \frac{\omega^m}{m!} e^{-\omega} \\ &= \frac{(\omega F(t))^n}{n!} e^{-\omega F(t)}. \end{aligned} \tag{1}$$

上記の確率関数より，累積バグ数の確率過程 $N(t)$ は平均値関数 $E[N(t)] = \omega F(t)$ の NHPP となる．ソフトウェア信頼性モデルが与えられた時，テスト時刻 t から $t+u$ でバグが発見される確率（ソフトウェア信頼度）は

$$R(u|t) = \exp\left(-\omega(F(t+u) - F(t))\right) \tag{2}$$

で与えられる．

2.2. バグ発見時刻と修正時間を考慮したモデル

文献 [7] では，次で紹介するバグ発見だけではなく修正時間を考慮したモデルを提案している．まず，修正時間を扱うために以下の仮定を設定する．

- テスト前にソフトウェア内に潜在するバグ数 M は，平均 ω のポアソン分布に従う．
- M 個のバグの発見時刻 X および修正時間（発見から修正完了までの時間）S は独立かつ同一に分布する二変量確率変数 $\{(X_1, S_1) \cdots, (X_M, S_M)\}$ で表され，その同時分布関数を $F(x, s)$ とする．

先のバグ発見時刻に対するモデルとの本質的な違いは，バグ発見時刻と修正時間が二変量分布で表現されている点にある．つまり，バグ修正時間がバグが発見される時刻に依存することを許容している．これは，テストの早い段階で見つかったバグほど修正時間が短い（あるいは長い），または，テスト終盤で見つかるバグほど修正時間が短い（あるいは長い）と言った相関関係を表すことになる．

文献 [7] では二変量分布 $F(x, s)$ から次の分布を考えている．

- $F_{UC}(t)$: あるバグが時刻 t までにバグが発見されたが修正されていない確率
- $F_C(t)$: あるバグが時間 t までにバグが発見・修正される確率

- $F_D(t) = F_{UC}(t) + F_C(t)$: あるバグが時間 t までに発見される確率

これらは同時密度関数 $f(x,s) = \partial^2 F(x,s)/\partial x \partial s$ を用いると

$$F_{UC}(t) = \int_0^t \int_0^{t-x} f(x,s) ds dx, \tag{3}$$

$$F_C(t) = \int_0^t \int_{t-x}^{\infty} f(x,s) ds dx, \tag{4}$$

$$F_D(t) = \int_0^t \int_0^{\infty} f(x,s) ds dx \tag{5}$$

となる．いま，$N_D(t)$ ならびに $N_C(t)$ を時刻 t までに発見されたバグの総数ならびに修正されたバグの総数とすると，初期バグ数 M がポアソン分布に従うため $N_D(t)$ と $N_C(t)$ の同時確率は

$$\begin{aligned} &P(N_D(t) = d, N_C(t) = c) \\ &= \frac{(\omega F_{UC}(t))^{d-c} (\omega F_C(t))^c}{(d-c)!c!} e^{-\omega F_D(t)} \end{aligned} \tag{6}$$

として記述できる．上述の周辺分布 $P(N_D(t) = d)$, $P(N_C(t) = c)$ はともに平均 $\mathrm{E}[N_D(t)] = \omega F_D(t)$, $\mathrm{E}[N_C(t)] = \omega F_C(t)$ のポアソン分布となる．つまり，文献 [7] のモデルフレームワークはバグ発見過程，バグ修正過程がともに NHPP となるモデルをすべて包括している．実際，既存の文献 [10, 11, 13, 12] のモデルがこのモデルフレームワークのサブクラスとなっている．

文献 [7] では，ソフトウェア信頼度（ある区間でバグが見つかる確率）以外の信頼性尺度として，ある時刻までに発見されたバグがすべて修正されている確率を算出している．これは，例えば OSS を流用したソフトウェアを開発する場合に従来のソフトウェア信頼度よりも有益な尺度となることが示唆されている．また，文献 [7] ではさらに，効率的なモデルパラメータの推定アルゴリズムについて言及している．

3. ソフトウェア最適リリース問題

3.1. 従来のソフトウェア最適リリース問題

Okumoto and Goel [8] は次に示すテスト費用を考慮したソフトウェア最適リリース問題を定義している．

- c_1： テスト工程における単位時間当たりのテスト費用
- c_2： テスト工程において発見されたバグを修正するための単位個数当たりの費用
- c_3： リリース後の運用工程において発見されたバグを修正するための単位個数当たりの費用

一般的に，バグ修正費用はテスト工程よりもリリース後の運用工程の方が高いため $c_3 > c_2$ を仮定する．現在時刻 t までに発見されたバグの個数を $N(t)$ とし，ソフトウェアのサポート打ち切り時刻を T_{LC} とする．このとき，時刻 T でソフトウェアをリリースしたときの総期待費用を $C(T)$ とすると，

$$C(T) = c_1 T + c_2 E[N(T)] + c_3 (E[N(T_{LC}) - E[N(T)]). \tag{7}$$

となる． $E[N(t)]$ は時刻 t までに発見されたバグ総数の期待値であり，バグ発見時刻分布 $F(t)$ および初期バグ数の期待値 ω を用いると，式 (7) は

$$C(T) = c_1 T + c_2 \omega F(T) + c_3 \omega \{F(T_{LC}) - F_D(T)\}. \tag{8}$$

となる．このとき，ソフトウェア最適リリース問題は総期待費用 $C(T)$ を最小化するリリース時刻 T^* を求める問題として，次のように定式化される．

$$\min_{0 \le T \le T_{LC}} C(T). \tag{9}$$

3.2. バグ修正時間を考慮したソフトウェア最適リリース問題

本稿では，2.2 で紹介したバグ修正時間を考慮したモデルに基づいてソフトウェア最適リリース問題を考える．いま，Okumoto and Goel [8] と同様に以下の費用を仮定する．ただし，テスト工程ならびに運用段階におけるバグ修正費用が単一バグ当たりの費用ではなく，修正時間に比例したコストに変更している．

- c_1: テスト工程に要する単位時間当たりの費用.
- c_2': テスト工程において発見されたバグを修正するために必要な単位時間当たりの費用.
- c_3': 運用段階において発見されたバグを修正するために必要な単位時間当たりの費用.

いま，ソフトウェアの利用期間を T_{LC} とすると，時刻 T でソフトウェアをリリースするときの総期待費用 $C(T)$ は以下のように定式化される．

$$C(T) = c_1 T + c_2' W(T) + c_3' \left(W(T_L) - W(T)\right). \tag{10}$$

ここで，$W(T)$ は時刻 T までに発見されたすべてのバグを修正するのに必要な期待累積修正時間を表す．いま，$\{(X_1, S_1) \ldots, (X_{N_D(T)}, S_{N_D(T)})\}$ を時刻 T までに発見されたバグの発見時刻と修正時間の列とすると，時刻 T までに発見された $N_D(T)$ 個のバグに対する修正時間 $S_1, \ldots, S_{N_D(T)}$ の和となるため

$$
\begin{aligned}
W(T) &= \mathrm{E}\left[\sum_{i=1}^{N_D(T)} S_i\right] \\
&= \sum_{k=0}^{\infty} P(N_D(T) = k)k \int_0^T \mathrm{E}[S|X = x] f_D(x)dx \\
&= \omega \int_0^T \int_0^{\infty} s f(x, s) ds dx. \qquad (11)
\end{aligned}
$$

となる．ここで $f_D(x)$ はバグ発見時刻に対する周辺密度関数であり，さらに，

$$
\mathrm{E}[S|X = x] = \frac{\int_0^{\infty} s f(x, s) ds}{f_D(x)} \qquad (12)
$$

である．

特別な場合として，バグ発見時刻 X とバグ修正時間 S が独立な場合，つまり，テストの初期と終盤で発見されたバグの修正時間が同じ場合は

$$
W(T) = \mathrm{E}[N_D(T)]\mathrm{E}[S] = \omega F_D(T)\mathrm{E}[S] \qquad (13)
$$

となる．つまり，単一のバグ修正費用を $c_2'\mathrm{E}[S]$ ならびに $c_3'\mathrm{E}[S]$ とした Okumoto and Goel [8] によるソフトウェアリリース問題に帰着される．換言すると，Okumoto and Goel [8] によるリリース問題の定式化では「テストの初期と終盤で発見されたバグの修正時間が同じ」であることを暗に仮定している．また，ここで提案した上記の定式化は Okumoto and Goel [8] による問題の数理的な一般化となっている．

以上の準備のもと，バグ修正時間を考慮したソフトウェア最適リリース問題は総期待費用 $C(T)$ を最小化する時刻 T を見つける問題として定式化される．

$$
\min_{0 \le T \le T_{LC}} C(T). \qquad (14)
$$

4. 数値例

ここでは，バグ発見時刻と修正時間を相関を表現するために以下に示す FGM コピュラ [3] を用いる．

$$
F(x, s) = F_D(x)F_C(x)\left(1 + \alpha \overline{F}_D(x)\overline{F}_C(s)\right). \qquad (15)
$$

表 1. 最適リリース時刻と最小総期待費用.

α	T^*	$C(T^*)$
1.0	495.4	1095.6
0.5	477.3	1077.4
0.0	455.3	1055.3
-0.5	427.5	1027.0
-1.0	389.9	988.0

ここで，$F_D(x) = P(X \le x)$, $F_C(x) = P(S \le s)$ はバグ発見時刻ならびに修正時間の周辺分布関数，$F(x, s) = P(X \le x, S \le s)$ は同時分布関数を表す．また，$\overline{F}_D(t) = 1 - F_D(t)$, $\overline{F}_C(t) = 1 - F_C(t)$ である．さらに $-1 \le \alpha \le 1$ は相関の強さを表すパラメータであり，$\alpha < 0$ の時は負の相関（早い段階で見つかったバグほど修正時間が長い），$\alpha = 0$ の時は無相関（テストの初期と終盤で発見されたバグの修正時間が同じ），$\alpha > 0$ の時は正の相関（早い段階で見つかったバグほど修正時間が短い）となる．FGM コピュラによりバグ発見時刻と修正時間の相関を表現するとき期待累積修正時間は

$$
\begin{aligned}
W(T) &= \omega F_D(T)\mathrm{E}[S] \\
&\quad \times \left(1 - \frac{\alpha \overline{F}_D(T)}{\mathrm{E}[S]} \int_0^{\infty} F_C(s)\overline{F}_C(s) ds\right) \qquad (16)
\end{aligned}
$$

となる．さらに，簡単のためバグ発見時刻と修正時間をともにパラメータ β_D, β_C の指数分布とすると

$$
W(T) = \frac{\omega}{\beta_C}(1 - e^{-\beta_D T})\left(1 - \frac{\alpha}{2}e^{-\beta_D T}\right) \qquad (17)
$$

となる．

表 1 は $c_1 = 1.0$, $c_2' = 0.01$, $c_3' = 0.2$, $\omega = 100.0$, $\beta_d = 0.01$, $\beta_c = 0.002$, $T_{LC} = 720.0$ とし，パラメータ α を -1 から 1 まで変化させた時の最適リリース時刻 T^* と最小総期待費用 $C(T^*)$ を示している．パラメータ c_1, c_2', c_3' の設定では基本的なテスト費用 c_1 に対する比率が重要であるため，$c_1 = 1$ と設定し，c_2', c_3' は基本的なテスト費用の 1%, 20% として相対的に設定している．

この表からバグ発見時刻と修正時間に正の相関がある場合，独立な場合と比較して最適なリリース時刻が長くなり総費用も高くなることがわかる．反対に負の相関がある場合，最適リリース時刻が短くなり総費用が低くなる．この結果は，バグ発見時刻と修正時間に正の相関がある場合，テスト終盤で修正時間の長いバグが発見され

る傾向があるため，リリース後のバグ修正費用を抑えるためにリリースの延長を考慮する必要があることを示唆している．また，正の相関がある場合はリリースを延長したとしても全体の費用を抑えることにはつながらないこともわかる．つまり，全体のテスト関係の費用を抑えるためにはテスト初期において修正に時間のかかるバグを発見する方が良いと言うこともわかる．ここでは，c_1, c_2', c_3' に相対的な費用を設定しているため，$\alpha = 1$ と $\alpha = -1$ の場合の費用の差は 1095.6 - 988.0 = 107.6 であり，全体費用の 10 % 程度ではあるが，プロジェクトの規模が大きければなるほど基本的なテスト費用として設定した単位時間あたりのテスト費用（1 日当たりのテスト費用） c_1 が大きくなるため，負の相関を示すようなテスト戦略を行うことによって，絶対的な費用としてはかなり多くの費用が低減できることもわかる．

5. まとめと今後の課題

本稿では，バグ修正時間を考慮したソフトウェア信頼性モデルの一般的なモデルフレームワークを利用し，ソフトウェアリリース問題の定式化ならびに，バグ発見時刻とバグ修正時間の相関が費用とリリースに与える影響について分析した．文献 [7] ではいくつかのオープンソースプロジェクトのデータからバグ発見時刻とバグ修正時間における相関を分析し，いくつかのプロジェクトで（強い相関ではないが）相関があることを確認している．

本稿の数値例における結果では，テスト初期において修正時間のかかるバグを発見することが経済的な観点から重要であることがわかった．また，リリースに対する意思決定においてはバグ発見時刻とバグ修正時間の傾向を見ることも重要であることがあることも示された．この指標は恐らくこれまでリリースに対して，あまり意識されていない指標であると考えられる．

今後は，ソフトウェア品質評価やテスト進捗などから得られる指標に加えて，ここで議論したバグ発見時刻とバグ修正時間の相関も考慮したリリース判定に対する意思決定スキームについて議論する予定である．

参考文献

[1] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi. Analysis of software fault removal policies using a non-homogeneous continuous time markov chain. *Software Quality Journal*, Vol. 12, No. 3, pp. 211–230, 2004.

[2] S. S. Gokhale, M. R. Lyu, and K. S. Trivedi. Incorporating fault debugging activities into software reliability models: A simulation approach. *IEEE Transactions on Reliability*, Vol. 55, pp. 281–292, 2006.

[3] H. Joe. *Dependence Modeling with Copulas*. CRC Press, 2014.

[4] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1996.

[5] J. D. Musa. *Software Reliability Engineering*. McGraw-Hill, New York, 1999.

[6] J. D. Musa, A. Iannino, and K. Okumoto. *Software Reliability, Measurement, Prediction, Application*. McGraw-Hill, New York, 1987.

[7] H. Okamura and T. Dohi. A generalized bivariate modeling framework of fault detection and correction processes. In *Proceedings of the 26th International Symposium on Software Reliability Engineering (ISSRE 2017)*, pp. 35–45. IEEE CPS, 2017.

[8] K. Okumoto and L. Goel. Optimum release time for software systems based on reliability and cost criteria. *Journal of Systems and Software*, Vol. 1, pp. 315–318, 1980.

[9] H. Pham. *Software Reliability*. Springer, Singapore, 2000.

[10] N. F. Schneidewind. Analysis of error processes in computer software. *Proceedings of the International Conference on Reliable Software*, Vol. 10, pp. 337–346, 1975.

[11] N. F. Schneidewind. Modelling the fault correction process. In *Proceedings of The 12th International Symposium on Software Reliability Engineering*, pp. 185–190. IEEE Computer Society Press, 2001.

[12] M. Xie, Q. P. Hu, Y. P. Wu, and S. H. Ng. A study of the modeling and analysis of software fault-detection and fault-correction processes. *Quality and Reliability Engineering International*, Vol. 23, pp. 459–470, 2007.

[13] M. Xie and M. Zhao. The schneidewind software reliability model revisited. In *Proceedngs of the 3rd International Symposium on Software Reliability Engineering*, pp. 184–192. IEEE Computer Society Press, 1992.