デバッギング難易度を考慮したソフトウェア信頼性モデルに関する一考察

On Software Reliability Modeling with Fault Debugging Difficulties

関西大学・総合情報学部　井上 真二
鳥取大学・大学院工学研究科　山田　茂

Shinji Inoue (Faculty of Informatics, Kansai University)
Shigeru Yamada (Graduate School of Engineering, Tottori University)

1　はじめに

ソフトウェア開発工程の最終工程であるテスト工程では，事前に準備されたテストケースを入力・実行することで，ソフトウェアシステムが期待通りに動作するか否かを確認する作業が行われる．ソフトウェア故障を観測した場合，一般的に，その原因解析が行われ，その結果に基づいてフォールト（バグ）の修正作業が行われることになる．この一連の過程はデバッギングプロセスと呼ばれ，テスト作業の進捗に伴い観測される信頼度成長過程に影響を与えることが考えられる．本研究では，このデバッギングプロセスと観測される信頼度成長過程との関係性を，位相型確率分布 [1] を用いながら表現し，デバッギングプロセスを意識したソフトウェア信頼性評価モデルを構築するための手法について議論する．

位相型確率分布は，吸収マルコフ連鎖における初期状態から吸収状態に至るまでの時間分布を表現する確率分布として知られ，指数分布，ガンマ分布，アーラン分布などの代表的な確率分布を一般的に包含する確率分布である．この確率分布は，ソフトウェア故障発生時間分布を一般的に表現する確率分布として，ソフトウェア信頼性モデルを構築する際に活用され，モデルパラメータの推定手法も含めソフトウェア信頼性モデリングの一般的な構築枠組みを与える有用な確率分布としても知られている [3]．

本研究では，先に簡単に述べた位相型確率分布の考え方をデバッギングプロセスにおける不確実性を表現するアプローチとして応用し，デバッギングプロセス考慮したソフトウェア信頼性モデリングについて議論する．また，この枠組みに基づいて，デバッギング難易度によるデバッギングプロセスの違いを意識した新たなソフトウェア信頼性モデルを提案し，議論したアプローチの有用性についても議論する．

2　デバッギングプロセス指向型モデル

ソフトウェア信頼性評価のための位相型確率分布を適用したデバッギングプロセス指向型モデルについて議論する．まず，モデリングのための下記の基本的仮定を設ける：

(A1)　テスト開始前までにソフトウェア内に作り込まれた総フォールト数は，非負の確率変数 $N_0(>0)$ で与え，ある一定の離散型確率分布に従う．

(A2)　1 つのソフトウェア故障の観測からその原因となるフォールトの修正に至るまでの時間 $T(T \geq 0)$ は，独立かつ同一の位相型確率分布 $F_{PH}(t) = \Pr\{T \leq t\}$ に従う．

(A3)　ソフトウェア故障の原因となるフォールトはデバッギング作業により完全に修正・除去され，デバッギング作業による新たなフォールトの作り込みは考えない．

ここで，$N_0 = n$ と与えたとき，テスト時刻 t までに発見されたフォールト数を表す計数過程 $\{N(t), t \geq 0\}$ に対する条件付き確率関数は，上記の仮定に基づいて，

$$\Pr\{N(t) = m \mid N_0 = n\} = \binom{n}{m}\{F_{PH}(t)\}^m\{1 - F_{PH}(t)\}^{n-m}, \tag{1}$$

と求められる．式 (1) から，$\{N(t), t \geq 0\}$ に対する確率関数は，

$$\Pr\{N(t) = m\} = \sum_n \binom{n}{m}\{F_{PH}(t)\}^m\{1 - F_{PH}(t)\}^{n-m}\Pr\{N_0 = n\}$$
$$(m = 0, 1, 2, \cdots, n), \tag{2}$$

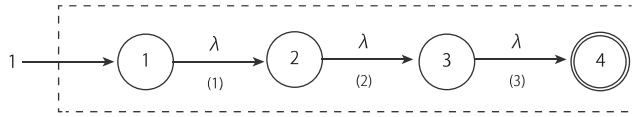

図 1： モデル 1 におけるデバッギングプロセス．

となる．このとき，確率変数 N_0 がパラメータ ω のポアソン分布に従うと仮定した場合，式 (2) は，

$$\Pr\{N(t)=m\}=\frac{\{\omega F_{PH}(t)\}^m}{m!}\exp[-\omega F_{PH}(t)] \qquad (m=0,1,2,\cdots), \tag{3}$$

と導かれ，式 (3) は，平均値関数 $\omega F_{PH}(t)$ をもつ非同次ポアソン過程（NHPP）と本質的に等価となる．したがって，平均値関数に含まれる位相型確率分布 $F_{PH}(t)$ に対して様々なデバッギングプロセスを背景にした連続時間吸収マルコフ連鎖を構成することで，ソフトウェア信頼性評価のための種々の NHPP モデル [4,6] を構築できる．

一般的に，連続時間マルコフ連鎖は，無限小生成行列 $\boldsymbol{Q}$ によって各状態間の瞬間的な状態推移率が表現される．つまり，過渡状態 $\mathcal{S}_T=\{1,2,\cdots,n\}$ および吸収状態 $\mathcal{S}_A=\{n+1\}$ から構成される吸収マルコフ連鎖を考えたとき，この吸収マルコフ連鎖の無限小生成行列は，

$$\boldsymbol{Q}=\begin{pmatrix}\boldsymbol{D_0} & \boldsymbol{d_1}\\ \boldsymbol{0} & 0\end{pmatrix}, \tag{4}$$

のように表現できる．ここで，$\boldsymbol{D_0}$ は過渡状態間における瞬間的な推移率を表す $n\times n$ の小行列，$\boldsymbol{d_1}$ は過渡状態から吸収状態への瞬間的な推移率を表した $n\times 1$ ベクトルであり，$\boldsymbol{D_0 1}+\boldsymbol{d_1}=\boldsymbol{0}$ を満たす．吸収状態から過渡状態に推移することはないため，それらは零ベクトル $\boldsymbol{0}$ もしくは 0 によって表現している．式 (4) から，一般的なマルコフ解析によって，初期状態から吸収状態に至るまでの時間分布は，最終的に，

$$F_{PH}(t)\equiv\Pr\{T\le t\}=1-\boldsymbol{\pi}\exp[\boldsymbol{D_0}t]\boldsymbol{1} \qquad (t\ge 0), \tag{5}$$

と求められる．

既知の内容として，NHPP の確率的性質からソフトウェア信頼性評価に有用な種々のソフトウェア信頼性評価尺度も導出できる．例えば，ソフトウェア信頼度関数は，テスト時刻 t までテスト作業が行われた条件の下で，その後の時間区間 $(t,t+x](x\ge 0)$ においてソフトウェア故障が発生しない確率を表現する確率確率関数として定義され，

$$\begin{aligned}R(x\mid t)&=\sum_k \Pr\{N(t+x)=k\mid N(t)=k\}\Pr\{N(t)=k\}\\ &=\exp[-\{H(t+x)-H(t)\}],\end{aligned} \tag{6}$$

のように求めれれる．ここで，$H(t)$ は NHPP の平均値関数を表し，ここでは，$H(t)\equiv\omega F_{PH}(t)$ となる．また，平均ソフトウェア故障発生時間間隔（MTBF）の代替的尺度である累積 MTBF は，

$$MTBF_C(t)=\frac{t}{H(t)}. \tag{7}$$

として求められる．

3 モデルの構築

前述したモデリング枠組みに基づいて，様々なデバッギングプロセスを想定しながら，新たなソフトウェア信頼性モデルを構築する．まず，図 1 のようなデバッギングプロセスを想定したモデルを開発する．図 1 では，デバッギングプロセスが，(1) ソフトウェア故障発生過程，(2) 原因解析過程，(3) フォールト修正・

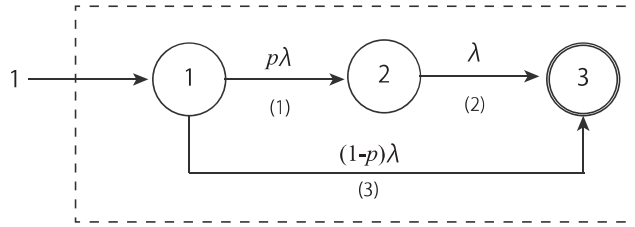

図 2： モデル 2 におけるデバッギングプロセス．

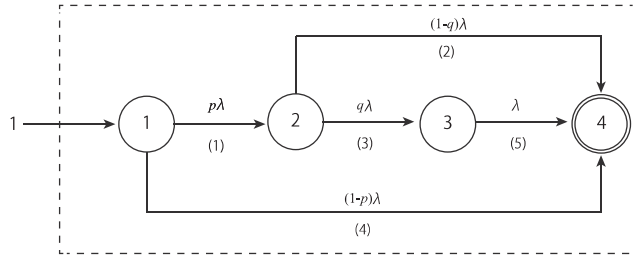

図 3： モデル 3 におけるデバッギングプロセス．

除去過程から構成されることを示しており，各過程への推移率は同一の λ として与えている．このとき，無限小生成行列 $\boldsymbol{Q}$ は，

$$\boldsymbol{Q} = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -\lambda & \lambda & 0 \\ 0 & 0 & -\lambda & \lambda \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{8}$$

と表現され，初期状態ベクトルは $\boldsymbol{\pi} = (1 \quad 0 \quad 0)$ である．したがって，図 1 のようなデバッギングプロセスに対する位相型確率分布関数は，式 (5) より，

$$\begin{aligned} F_{PH}(t) = 1 - \boldsymbol{\pi} \exp[\boldsymbol{D_0} t]\mathbf{1} &= 1 - \boldsymbol{\pi} \exp\left[\begin{pmatrix} -\lambda & \lambda & 0 \\ 0 & -\lambda & \lambda \\ 0 & 0 & -\lambda \end{pmatrix} t\right] \mathbf{1} \\ &= 1 - \left\{1 + \lambda t + \frac{1}{2}(\lambda t)^2\right\} \exp[-\lambda t], \end{aligned} \tag{9}$$

と求められる．よって，NHPP の平均値関数は，式 (3) の枠組みに従い，

$$H_1(t) = \omega \left[1 - \left\{1 + \lambda t + \frac{1}{2}(\lambda t)^2\right\} \exp[-\lambda t]\right], \tag{10}$$

のように得られる．本研究では，式 (10) の平均値関数をモデル 1 と呼ぶことにする．

次に，図 2 のようなデバッギングプロセスを想定したモデルを構築する．図 2 では，デバッギング難易度によるデバッギングプロセスの違いを表現している．具体的には，確率 $p(0 \leq p \leq 1)$ でデバッギング難易度が高いフォールトに起因するソフトウェア故障がプロセス (1) のソフトウェア故障発見過程で観測され，プロセス (2) の原因解析およびフォールト修正・除去過程を経てフォールトが完全に修正・除去される．一方，デバッギング難易度が低いフォールトに起因するソフトウェア故障は，プロセス (3) においてソフト

ウェア故障を観測後直ちにフォールトが修正・除去される．図 2 に示したデバッギングプロセスに対する無限小生成行列および初期状態ベクトルは，それぞれ，

$$\boldsymbol{Q} = \begin{pmatrix} -\lambda & p\lambda & (1-p)\lambda \\ 0 & -\lambda & \lambda \\ 0 & 0 & 0 \end{pmatrix}, \tag{11}$$

および $\boldsymbol{\pi} = (1 \quad 0)$ と与えられる．これより，前述のモデルと同様な操作によって，図 2 に示したデバッギングプロセスを反映した平均値関数は，最終的に，

$$H_2(t) = \omega \left[1 - (1 + p\lambda t)\exp\{-\lambda t\}\right], \tag{12}$$

と求められる．本研究では，式 (12) の平均値関数をモデル 2 と呼ぶことにする．

さらに，図 2 を拡張した図 3 のようなデバッギングプロセスについても議論する．図 3 では，デバッギング難易度を低，中，および高に分けてこれらのデバッギングプロセスの違いを表現している．具体的には，確率 $p(0 \leq p \leq 1)$ に従ってデバッギング難易度が低のものと，中もしくは高のものに分かれる．デバッギング難易度が低であるデバッギングプロセスは，ソフトウェア故障観測後直ちにその原因となるフォールトが修正・除去されるようなプロセス (4) のみを経てデバッギング作業が完了する．一方，デバッギング難易度が中または高であるフォールトに対しては，その後，さらに確率 $q(0 \leq q \leq 1)$ でデバッギング難易度が中であるプロセスと高であるプロセスに分かれる．つまり，デバッギング難易度が中であるフォールトは，プロセス (1) のソフトウェア故障発見過程を経た後，原因解析およびフォールト修正・除去のプロセス (2) によって，フォールトが完全に修正・除去される．一方，デバッギング難易度が高であるフォールトは，プロセス (1) のソフトウェア故障発見過程を経た後，原因解析のプロセス (3) およびフォールト修正・除去のプロセス (5) によって最終的にフォールトが完全に修正・除去される．図 2 および図 3 にそれぞれ示したデバッギングプロセスの記述は，デバッギング難易度に基づいて，これらのデバッギングプロセスの違いを表現した記述となっている．図 3 に示した連続時間吸収マルコフ連鎖に対する無限小生成行列は，

$$\boldsymbol{Q} = \begin{pmatrix} -\lambda & p\lambda & 0 & (1-p)\lambda \\ 0 & -\lambda & q\lambda & (1-q)\lambda \\ 0 & 0 & -\lambda & \lambda \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{13}$$

と与えられ，初期状態ベクトルは $\boldsymbol{\pi} = (1 \quad 0 \quad 0)$ と与えられる．これより，図 3 のデバッギングプロセスを反映した平均値関数は，先ほど議論したモデルの導出過程と同様にして，最終的に，

$$H_3(t) = \omega \left[1 - \left\{1 + p\lambda t + \frac{1}{2}pq(\lambda t)^2\right\}\exp\{-\lambda t\}\right], \tag{14}$$

のように求められる．本研究では，式 (14) の平均値関数をモデル 3 と呼ぶことにする．

4　適合性比較

本研究において新たに構築した 3 つのモデルおよびデバッギングプロセスを考慮した既存の遅延 S 字形ソフトウェア信頼性モデル [5] を含め，実際のテスト工程において観測されたフォールト発見数データを適用した適合性比較を行う．遅延 S 字形ソフトウェア信頼性モデルは，テスト工程におけるソフトウェア故障発見過程と発見されたソフトウェア故障の原因となるフォールトを修正・除去するフォールト認知過程から成るデバッギングプロセスを想定しながら構築されたモデルである．このモデルは，これらの 2 つのデバッギングプロセスを想定した発見フォールト数の平均的挙動を，下記の微分方程式を用いて記述したモデルである．

$$\frac{\mathrm{d}H_d(t)}{\mathrm{d}t} = b[h_d(t) - H_d(t)], \tag{15}$$

$$\frac{\mathrm{d}h_d(t)}{\mathrm{d}t} = b[\omega - h_d(t)]. \tag{16}$$

ここで，$h_d(t)$ はソフトウェア故障発見過程においてテスト時刻 t までに発見されたソフトウェア故障数の期待値，$H_d(t)$ はフォールト認知過程においてテスト時刻 t までに修正・除去されたフォールト数の期待値，$b(>0)$ はフォールト 1 個あたりのソフトウェア故障発見率もしくはフォールト認知率，および $\omega(>0)$ はテスト開始前に残存する総期待フォールト数を表す．式 (15) および式 (16) の連立微分方程式を $H_d(t)$ について解くと，NHPP の平均値関数は，

$$H_d(t) = \omega\left[1 - (1+bt)\exp[-bt]\right], \tag{17}$$

のように導出される．

また，適用する実測データは，次の 3 つのフォールト発見数データ [2] を適用する：DS1: $(t_k, y_k)(k = 1, 2, \cdots, 22; t_{22} = 22(\text{週}), y_{22} = 212)$，DS2: $(t_k, y_k)(k = 1, 2, \cdots, 25; t_{25} = 25(\text{CPU 時間}), y_{25} = 136)$，DS3: $(t_k, y_k)(k = 1, 2, \cdots, 19; t_{19} = 19(\text{週}), y_{19} = 328)$．ここで，$y_k$ はテスト時刻 t_k までに修正・除去されたフォールト数の累積値を示す．これらのデータに対して，横軸にテスト時間，縦軸に累積修正フォールト数を表す 2 次元平面上にこれら 3 つのデータをそれぞれプロットしたとき，DS1 および DS2 は S 字形信頼度成長曲線を示すデータであり，DS3 は指数形信頼度成長曲線を示すデータをあることも付記しておく．また，モデルに含まれるパラメータは，NHPP の性質を利用しながら最尤法により推定した．

適合性比較を行う際に利用する適合性評価尺度として，以下に示す平均偏差平方和（MSE : mean squared error），赤池情報量規準（AIC : Akaike's information criterion），および最大対数尤度（MLL : Maximum log-likelihood）を用いる．MSE は，実測値とモデルによる推定値の二乗誤差をデータ観測数で平均化した尺度であり，

$$\text{MSE} = \frac{1}{K}\sum_{k=1}^{K}\{y_k - \widehat{y}(t_k)\}^2, \tag{18}$$

に基づいて算出される．ここで，$\widehat{y}(t_k)$ はテスト時刻 t_k においてモデルを用いた推定された値を示す．したがって，より小さい MSE を示すモデルほど実測値を良く表現しているモデルと言える．AIC は，パラメータの数を考慮したモデルの適合性評価尺度であり，

$$\text{AIC} = -2(\text{最大対数尤度}) + 2(\text{モデルパラメータの数}), \tag{19}$$

に従い算出され，モデル間の AIC の差が 1 以上であるとき AIC が小さいモデルの方が有意に良いモデルとして判定される．さらに，MLL は，パラメータの最尤推定値を対数尤度関数に代入したときの尤度の値であり，実測データに対する尤度を示す尺度である．

表 1 に，上述した適合性評価尺度に基づいたモデルの適合性比較結果を示す．表 1 から，まず，提案したモデル 1 は，従来のデバッギング指向型モデルである遅延 S 字形モデルよりも評価精度の向上が認められないことがわかる．モデル 1 は，原因解析過程とフォールト修正・除去過程をそれぞれ別に設けることで，遅延 S 字形モデルで想定しているデバッギングプロセスをさらに詳細化したモデルである．一方，フォールト修正難易度を考慮したデバッギングプロセスを反映した提案モデルであるモデル 2 とモデル 3 は，遅延 S 字形モデルよりも優れた適合性を有していることが確認できる．特に，モデル 3 は，ほとんどの実測データにおいて最も優れた適合性を有していることがわかる．つまり，デバッギングプロセスにおけるフォールト修正難易度は，ソフトウェアの信頼度成長過程を記述する上で効果的な要因のひとつであることが伺える．また，このようなデバッギングプロセスを比較的容易に記述できる位相型確率分布の有用性も無視できないものと考えられる．

5　おわりに

本研究では，テスト工程におけるデバッギングプロセスがソフトウェア信頼度成長過程に与える影響を考慮したソフトウェア信頼性評価モデルについて議論した．特に，ソフトウェア故障の観測からその原因となるフォールトの修正・除去に至るまでのプロセスを連続時間吸収マルコフ連鎖で表現しながら，初期状態からプロセス完了までに要する時間の不確実性を位相型確率分布を用いて表現した．このアプローチは，各

表 1： モデルの適合性比較結果

データ	モデル	MSE	MLL	AIC
DS1	遅延 S 字形モデル	36.650	-68.191	140.38
	提案モデル 1	99.354	-82.431	168.86
	提案モデル 2	26.332	-66.326	**136.65**
	提案モデル 3	**22.952**	**-66.116**	**136.23**
DS2	遅延 S 字形モデル	167.77	-94.604	193.21
	提案モデル 1	288.12	-144.11	292.21
	提案モデル 2	35.547	-57.219	118.44
	提案モデル 3	**3.9092**	**-50.874**	**105.75**
DS3	遅延 S 字形モデル	188.75	-109.19	222.38
	提案モデル 1	394.08	-138.73	281.46
	提案モデル 2	130.58	-102.20	208.39
	提案モデル 3	**103.89**	**-100.85**	**205.71**

プロセス毎に微分方程式を段階的に構築することで NHPP の平均値関数を導出する従来の方法とは大きく異なり，デバッギングプロセス全体を吸収マルコフ連鎖として捉えるため，現実的に想定される様々なデバッギングプロセスを容易に捉えることができ，その影響を反映したソフトウェア信頼性評価が可能となる．特に，今回の議論では，フォールト修正難易度に基づいたデバッギングプロセスを違いを吸収マルコフ連鎖を用いて表現しながら様々な位相型ソフトウェア信頼性モデルを構築した．また，デバッギングプロセスを意識した既存モデルとして遅延 S 字形ソフトウェア信頼性モデルを取り上げ，本研究で新たに構築したモデルとの適合性比較も行い，フォールト修正難易度を考慮することで従来モデルよりもより高精度なソフトウェア信頼性評価が可能であることが伺える検証結果を得た．本研究において構築したモデルは，今回のモデリング枠組みに基づいて構築できる一部のモデルであり，今後，より現時的なデバッギングプロセスを反映したモデルを開発することで，議論したアプローチの有用性やソフトウェア開発管理を支援するための応用的問題についても議論したい．

参考文献

[1] P. Buchholz, J. Kriege and I. Felko, *Input Modeling with Phase-Type Distributions and Markov Models — Theory and Applications —*. Springer, Cham, 2014.

[2] S. Inoue and S. Yamada, “Generalized discrete software reliability modeling with effect of program size,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 37, No. 2, pp. 170–179, 2007.

[3] H. Okamura and T. Dohi, “Phase type software reliability model: parameter estimation algorithms with grouped data,” *Annals of Operations Research*, Vol. 244, No. 1, pp. 177–208, 2016.

[4] H. Pham, *Software Reliability*. Springer-Verlag, Singapore, 2000.

[5] S. Yamada and S. Osaki, “Software reliability growth modeling : Models and applications,” *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 12, pp. 1431–1437, 1985.

[6] S. Yamada. Software Reliability Modeling — Fundamentals and Applications —. Springer, Tokyo, 2013.