ソフトウェア工学 107－3
（1996．1．18）

ソフトウェアフォールトの故障生起頻度分布について

杉元 秀人[†‡]　島 和之[†]　松本 健一[†]　鳥居 宏次[†]

[†] 奈良先端科学技術大学院大学 情報科学研究科

[‡] (株) 日立製作所　情報システム事業部

E-mail:{hideto-s, shima, matumoto, torii}@is.aist-nara.ac.jp

本稿では，ソフトウェア内に含まれるフォールトの故障生起頻度分布を測定した実験について報告し，故障生起頻度分布の型に依存しないモデルを提案する．また，提案モデルと従来のモデルとの実測データへの適合度を調査し，比較した結果を報告する．ソフトウェア信頼度成長モデルは，ソフトウェアの信頼性を定量的に評価する技術の１つである．これまで数多くのモデルが提案されているが，そのほとんどは各フォールトの故障生起頻度を一定と仮定するか，または，ガンマ分布等の特定の分布関数で近似できるとしている．

A Distribution of Failure Intensity of Software Faults

E-mail:{hideto-s, shima, matumoto, torii}@is.aist-nara.ac.jp

Hideto Sugimoto[†‡]　Kazuyuki Shima[†]　Ken-ichi Matsumoto[†]　Koji Torii[†]

[†] Graduate School of Information Science, Nara Institute of Science and Technology

[‡] Information Systems Division, Hitachi,Ltd

[†] 8916-5 Takayama, Ikoma, Nara 630-01, Japan

TEL：(+81)7437-9-9211(EX.5296)　FAX：(+81)7437-2-5299

[‡] 3-5-29 Kitahama Chuo, Osaka541, Japan

TEL：(+81)06-203-6151　FAX：(+81)06-271-7411

E-mail:{hideto-s, shima, matumoto, torii}@is.aist-nara.ac.jp

This paper presents the experiment on distribution of failure intensity of the faults in a software and proposes a model using distribution of failure intensity. We also investigated fitness between the proposed model and the conventional model. Software reliability growth model is one of the methods to assess the reliability of a software product quantitatively. There are many models that has been proposed, but most of them assumed that the failure intensity of each fault was constant, or could be approximated by gamma distributions.

1 はじめに

近年のコンピュータシステムの役割増大に伴い，その障害による経済的損失，社会的混乱も増大している．このため，ソフトウェアの信頼性を定量的，かつ，客観的に評価する技術に関心が高まっている [2][7]．その技術の一つに，開発におけるテスト工程や運用段階における故障発生時刻等のデータに基づいて，ソフトウェア中の残存フォールト数や最適リリース時刻を推定するための数学的モデルとしてソフトウェア信頼度成長モデル (Software Reliability Growth Model. 以下，SRGM と略す．) がある [1][5].

SRGM には，仮定や前提条件の違いにより様々なモデルがあるが，実際のソフトウェア開発プロジェクトに適用する場合，それらの仮定，前提条件が対象プロジェクトの実態に適合しているモデルを選択することが重要である．多くの SRGM では，ソフトウェアフォールトの故障生起頻度については，フォールトによらず一定としている．しかし，少なくともこの仮定については，否定的な結果を示した実験がある [6].

フォールト毎の違いについて考慮しているモデルとしては，超指数型 NHPP モデル [4] と，Littlewood モデル [3] があげられる．超指数型 NHPP モデルでは，いくつかのモジュール毎で一定であると仮定し，Littlewood モデルでは故障生起頻度分布として，ガンマ分布を仮定している．しかし，これらはあくまで仮定であり，実際の故障生起頻度がどのようになっているかは不明である．一般に，故障生起頻度の小さいフォールトはソフトウェア内に残存する確率が高いので，フォールト毎の故障生起頻度を調べることにより，よりよい信頼性予測が可能となる．

本稿では，比較的小規模なソフトウェアについて，実際にフォールトの故障生起頻度の測定を行い，実際の分布が従来のモデルの仮定に適合していないことを確かめるとともに，より正確性の高い信頼性予測が行えるように，特定の分布型を仮定しない新しい SRGM について考察する．2 節では，フォールトの故障生起頻度について説明する．3 節では，超指数型 NHPP モデル，Littlewood モデルの概要を述べる．4 節ではフォールトの故障生起頻度測定実験とその結果を，5,6 節では故障生起頻度測分布に依存しないモデルの提案，および従来のモデルとの比較について述べる．

2 フォールトの故障生起頻度

ソフトウェアの故障とは，ソフトウェアが仕様どおりに動作せず正しく機能しないことである．また，フォールトとは，故障の原因となったプログラム内の欠陥と定義される．フォールトによる故障の発生は入力などの条件によって左右される．また，フォールトが存在するからといって，故障を引き起こすとは限らない．そのため，フォールトの故障がどの程度故障を引き起こしやすいのかという指標が考えられる．それがフォールトの故障生起頻度であり，以下の式で定義される．

$$b_F = -\log(1-\phi_F) \tag{1}$$

ただし，

$$\phi_F = \frac{F\text{により故障を引き起こす入力数}}{\text{可能な全入力数}} \tag{2}$$

フォールトはそのソフトウェア内の位置，故障を引き起こすデータの種類がそれぞれ違う．よって一般的に，

- 各フォールトの故障生起頻度は異なる

また，

- ソフトウェア中にどのようなフォールトが含まれているかは未知である

これらより，ソフトウェア内のフォールトを無作為に選び出した場合，選ばれたフォールトの故障故障生起頻度は確率変数とみなすことができる．この確率分布を，故障生起頻度分布という．

3 従来のモデルにおける故障生起頻度分布

従来の SRGM はそのほとんどがフォールトの故障生起頻度頻度について考慮していない (フォールトによる故障生起頻度の違いはなく一定)．ここでは，フォールトの故障生起頻度頻度の違いを考慮している従来のモデルについて説明する．

3.1 超指数型 NHPP モデル

Matsumoto et al. により提案されたこのモデルは，一定時間内に発見されるフォールト数を確率変数とする個数計測モデルの中でも代表的な NHPP(Non-Homogenius Poisson Process) モデルの 1 つであり，以下の基本式で表される．

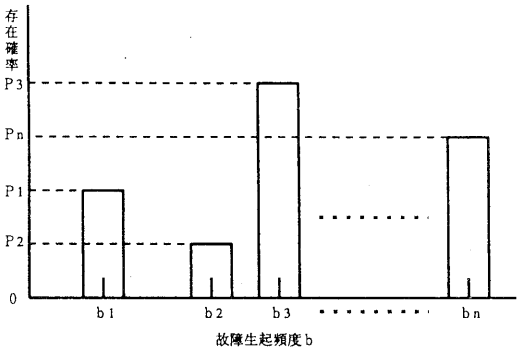

図 1: 超指数型 NHPP モデルにおけるフォールトの故障生起頻度

$$H(t) = a\sum_{i=1}^{n} p_i(1-e^{-b_i t}) \quad (3)$$

ただし,

$$a > 0 \quad , \quad 0 < p_i < 1, \quad (4)$$

$$\sum_{i=1}^{n} p_i = 1 \quad (5)$$

ここで,

- $H(t)$ = 時刻 t までに発見される総フォールト数の期待値 (平均値関数)
- a = テスト開始前にソフトウェア内に存在する総フォールト数の期待値
- b_i = i 番目のモジュールに含まれるフォールト 1 個あたりの故障生起頻度
- p_i = i 番目のモジュールに含まれるフォールトの含有率

このモデルでは, ソフトウェアを新規部分や再利用部分などの基準で複数個のモジュールに分割し, それらのモジュール毎にフォールトの故障生起頻度が異なると仮定する. (図 2参照) しかし, この仮定については否定的な結果を示す実験結果が存在する.

3.2 Littlewood モデル

Littlewood が提案したこのモデルは, 故障の発生する時間間隔を確率変数とする時間計測モデルの 1 つで

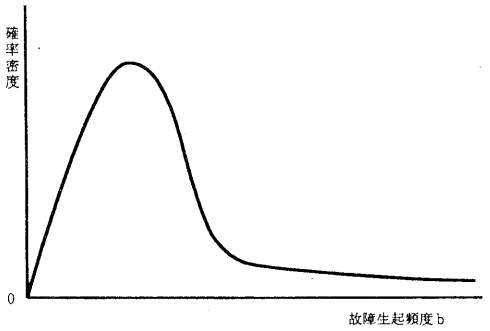

図 2: ガンマ分布 ($\alpha \geq 2$)

あり, ハザードレートに着目したモデルである. ハザードレートとは, ソフトウェアがある時刻までに故障せず, かつ次の瞬間に故障する確率密度をいう. このモデルでは, 各故障の発生時間間隔におけるソフトウェアのハザードレートは, その時点でソフトウェアに残存するフォールトの故障生起頻度の和で表されると仮定している (10).

このモデルでは, 故障生起頻度が一般に個々のフォールト毎に異なっていることを考慮している. しかし, それらは確率的にガンマ分布 (図 1参照) に従うと仮定している.

ガンマ分布は, 以下の式で表されるような, 2 つのパラメータ α, β を持つ分布である.

$$f(\nu_j) = \frac{\beta^{\alpha}}{\Gamma(\alpha)}\nu_j^{\alpha-1}e^{-\beta x} \quad (6)$$

ここで, $\Gamma(\alpha)$ は, ガンマ関数であり,

$$\Gamma(\alpha) = \lim_{n\to\infty}\frac{n^{\alpha}n!}{\alpha(\alpha+1)\cdots(\alpha+n)} \quad (7)$$

で定義される.

このモデルの基本式は, 以下の通りである.

$$F_i(x) = 1 - \left(\frac{\beta + \sum_{j=1}^{i-1} t_j}{\beta + \sum_{j=1}^{i-1} t_j + x}\right)^{(N-i+1)\alpha} \quad (8)$$

$$f_i(x) = \frac{(N-i+1)\alpha\left(\beta+\sum_{j=1}^{i-1}t_j\right)^{(N-i+\alpha)}}{\left(\beta+\sum_{j=1}^{i-1}t_j+x\right)^{(N-i+\alpha)+1}} \quad (9)$$

$$\lambda_i = \sum_{j=1}^{N-i+1}\nu_j \quad (10)$$

ここで,

$F_i(x)$ = $(i-1)$ 番目の故障が発生した時刻から x 時間内にソフトウェアが故障する確率

$f_i(x)$ = $F_i(x)$ の確率密度関数

N = テスト開始前にソフトウェア内に存在する総フォールト数

t_j = $(j-1)$ 番目の故障が発生してから, j番目の故障が発生するまでの時間間隔の実測値

λ_i = $(i-1)$ 番目の故障が発生した時刻でのハザードレート

ν_j = ソフトウェア中に残存している j番目のフォールトの故障生起頻度

α, β = 故障生起頻度分布 (ガンマ分布) のパラメータ

故障生起頻度がガンマ分布であるという仮定は，数式上は便利な点も多いが，仮定自体にあまり明確な理由はない．

4 故障生起頻度の計測

故障生起頻度がガンマ分布に従うという仮定については，明確な理由はないが，逆にそれを否定する根拠もなく，実際のフォールトの故障生起頻度は不明である．よって，本章では，実際にフォールトの故障生起頻度を計測した実験について説明する．

4.1 実験の目的

実際のフォールトの故障生起頻度を測定し，分布を調べる，特に，ガンマ分布に従うという仮定について検定を行う．

4.2 実験方法

- 被験者にプログラムの仕様を与え，設計からテスト，デバッグまでの一連の作業を行ってもらう．
- ここでは，最初にコンパイルエラーがなくなった時点を，テストの開始とする．
- テスト中に行われたソースプログラムの変更は，すべてフォールトの修正であると考える
- 観測者は，テスト中の被験者の作業を側で観察し，フォールトの位置，および，変更内容を全て記録する．
- 完成したプログラムに対して，上記で特定したフォールトを，1つだけ埋め込みなおしあらかじめ作成しておいたテストデータを入力して故障の発生の有無を確かめる．
- 全テストデータを入力し終えた時点で，故障が発生した個数を数え，そのフォールトの故障生起頻度を測定する．
- 以上の作業を，全フォールトについて繰り返す．

4.3 実験条件

- 被験者は，実際の開発プロジェクトを想定して，5年以上のプログラミング経験のある人とする．
- プログラム記述言語は，一般的な普及度を考慮して，C言語とする．
- テストデータは，意図して故障を発生させないようにランダムに作成する．

4.4 結果の検証

実験により取得した各フォールトの故障生起頻度データより，最尤推定法によってガンマ分布のパラメータを推定する．ただし，全ての入力に対して故障を引き起こすようなフォールトについては故障生起頻度が計算上無限大となるため，そのようなフォールトが存在する時は，故障生起頻度を十分大きい数で近似した場合，およびそのデータを除外した場合の2通りでそれぞれ推定を行う．

具体的には，故障生起頻の実測データを $x_1, x_2, \cdots, x_n$ とすると，パラメータ αが

表 1: 実験結果

フォールト	故障数	故障生起頻度
1	100	∞
2	100	∞
3	100	∞
4	100	∞
5	100	∞
6	48	0.6539
7	1	0.0101
8	39	0.5108
9	47	0.6349
10	60	0.9163
11	1	0.0101
12	0	0.0000
13	3	0.0306
14	5	0.0513
15	1	0.0101
16	3	0.0305
17	7	0.0726
18	59	0.8916
19	1	0.0101
20	1	0.0101
21	59	0.8916

$$\frac{\Gamma(\alpha)'}{\Gamma(\alpha)} - \log\alpha = \frac{1}{n}\log\prod_{k=1}^{n} x_k - \log\frac{\sum_{k=1}^{n} x_k}{n} \quad (11)$$

の解を求めることによって得られ，パラメータβが

$$\beta = \frac{n\alpha}{x_1 + x_2 + \cdots + x_n} \quad (12)$$

によって得られる．

次に，上記で推定されたパラメータをもつガンマ分布が実際のデータに良く適合しているかどうかを検定する．検定法には，Kolmogorov-Smirnov 適合度検定，およびχ^2 適合度検定の 2 つを用いる．

表 2: パラメータの推定結果

区分	α	β
故障生起頻度∞を含む	0.1036	$4.351\times e^{-5}$
故障生起頻度∞を含まない	0.3888	1.31233

表 3: 検定結果

区分	Kolmogorv 距離	χ^2統計量
故障生起頻度∞を含む	0.5152	43.11
故障生起頻度∞を含まない	0.1470	38.79

4.5 実験結果

与えたソフトウェア仕様は，技術計算ソフトウェアのサブルーチンの機能を実現するもので，プログラムの規模は，約 300 行となり，21 個のフォールトが検出された．実験結果を表 (1) に示す．

故障生起頻度が無限大のフォールトを含む場合，および含まない場合について最尤推定法によるパラメータ推定の結果を表 2に示す．また，それぞれの場合について Kolmogorv 距離，χ^2統計量を求めた結果を表 3に示す．

故障生起頻度が無限大のフォールトを含む場合は，Kolmogorv 適合度検定，χ^2適合度検定で有意水準 0.5 %で棄却され，また含まない場合にもχ^2適合度検定で有意水準 0.5 %で棄却された．よって，この実験に用いたソフトウェア内のフォールトについて，ガンマ分布はあまり適合しないと言える．

5 測定された故障生起頻度分布を用いたSRGMの提案

実験より，ソフトウェアの故障生起頻度について，一般的にはガンマ分布が適合しないことは確かめられた．

ここでは，故障生起頻度の分布型依存しないSRGMについて提案する．まず，故障データの観測のしやすさ，式の扱いやすさの点から，

- 個数計測モデルを提案する．
- フォールトの発見過程には，NHPPを仮定する．

今，フォールトの故障生起頻度分布を$[0,\infty)$で連続な確率変数bとし，その分布関数を$F(b)$とおく．すなわち，

$$F(b) = Pr\{\text{故障生起頻度} \leq b \text{となるフォールトの割合}\} \quad (13)$$

となる．すると，テスト開始時における$[b, b+db]$の範囲の故障生起頻度を示すフォールトの総数は，

$$a\{F(b+db) - F(b)\} \quad (14)$$

とおける．

ここで，従来のNHPPにおける平均値関数$H(t)$に対し，フォールトの故障生起頻度bを新たに変数としてもつ関数$H(t,b)$を考える．$H(t,b)$は，故障生起頻度bのフォールトが，時刻tまでに発見される期待値を表している．よって，ソフトウェア全体の総発見フォールト数の期待値$H(t)$とは，

$$H(t) = \int_0^\infty H(t,b)db \quad (15)$$

の関係がある．

指数型，超指数型NHPPモデルでは，故障生起頻度bが一定のモジュールにおいては，瞬間の故障発生率は，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と仮定している．すなわち，

$$\frac{dH(t)}{dt} = b(a - H(t)) \quad (16)$$

が成り立つ．ここでaはテスト開始時にソフトウェアに含まれている総フォールト数の期待値，bはそのモジュールにおけるフォールトの故障生起頻度である．上記$H(t,b)$にもこの仮定が成り立つとすると，(16)のaに(14)を，$H(t)$に$H(t,b)db$を，それぞれ代入すると，

$$\frac{d\{H(t,b)db\}}{dt} = b[a\{F(b+db) - F(b)\} - H(t,b)db] \quad (17)$$

a:テスト開始時の総フォールト数

b: 故障生起頻度

F(b): 故障生起頻度分布関数

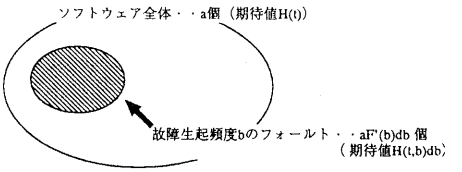

図 3: フォールトの故障生起頻度

この方程式を，$H(0,b) = 0$の条件の下で解くと，

$$H(t,b)db = a\{F(b+db) - F(b)\}(1 - e^{-bt}) \quad (18)$$

$$H(t,b) = a\frac{\{F(b+db) - F(b)\}}{db}(1 - e^{-bt}) \quad (19)$$

$$= aF'(b)(1 - e^{-bt}) \quad (20)$$

を得る．

$F'(b)$は故障生起頻度分布の密度関数であり，この式は，$aF'(b)db$がソフトウェア全体における故障生起頻度bのフォールトの総数の期待値（図3参照）であることを示している．

(15)の関係から，

$$H(t) = \int_0^\infty aF'(b)(1 - e^{-bt})db \quad (21)$$

が導かれ，$F(b)$は確率分布関数なので，

$$\int_0^\infty F'(b)db = 1 \quad (22)$$

が成立し，

$$H(t) = a - a\int_0^\infty F'(b)e^{-bt}db \quad (23)$$

を得る．

(23)は，一般にフォールトの故障生起頻度分布を仮定しない平均値関数の式であり，故障生起頻度分布を用いて平均値関数が表されることを示している．また，

この式は，e^{-bt}の Maclaurin 展開

$$e^{-bt} = \sum_{k=0}^{\infty} \frac{(-bt)^n}{k!} \quad (24)$$

を用いると，

$$H(t) = a - a\sum_{k=0}^{\infty} \frac{a_k(-t)^k}{k!} \quad (25)$$

$$a_k = \int_0^{\infty} F'(b)b^k db \quad (26)$$

となる．

a_kは故障生起頻度の nk次のモーメントであり，これは，平均値関数が故障生起頻度の各次のモーメントを用いた式で表されることを示している．このことから，十分にフォールトが発見された状況から，故障生起頻度を測定し，各次のモーメントを求めると，任意の故障生起頻度分布に対応した平均値関数が求められる．

6 モデルの比較

本章では，提案モデルと従来モデルとで実測データへの適合度を比較した結果について報告する．

6.1 比較方法

テストデータの実行順序によって，フォールトが検出される時刻は変わる．このため，提案モデルおよび従来モデルの適合度もテストデータの実行順序に依存している．そこで，100 件のテストケースを並べ替えた 100 通りのデータセットを作成し，その各々について提案モデルと従来モデルのパラメータを推定し，モデルの適合度を調べた．

モデルの適合度は Kolmogorov 距離を求めて，そのモデルを棄却する最小の有意水準を 0.005, 0.01, 0.025, 0.05, 0.1 から選んだ値で示した．

6.2 比較結果

この比較検証においては，比較対象の従来のモデルとして，Littlewood モデルを用いた．また，対象のソフトウェア，テストデータは，故障生起頻度を測定した時のものを流用した．テストデータセット数は，100 であった．

それぞれのモデルについて，棄却される有意水準と，棄却されたデータセット数を表 4に示す．

表 4: 各モデルの実験結果

有意水準	Littlewood	提案モデル
.005	8	0
.01	13	3
.025	27	10
.05	39	17
.1	55	31

表から，各有意水準において棄却されているデータセット数は提案モデルの方小さい．すなわち，どの有意水準においても，提案モデルの方が棄却される確率は低いので，提案モデルの方が，実測値により適合しているといえる．

7 むすび

本稿では，ソフトウェア内に含まれるフォールトの故障生起頻度を測定する実験について報告した．そして，その結果としてフォールトの故障生起頻度分布をガンマ分布とする仮定は，必ずしも正しくないことを示した．

更に，NHPP モデルの平均値関数に故障生起頻度を変数として導入することにより，故障生起頻度分布の型に依存しないモデルを提案した．そして，提案モデルの従来のモデルに対する優位性を，実測データへの適合度を検定することにより確かめた．

これらより，フォールトの故障生起頻度分布を考慮する必要性，および，考慮した提案モデルの利用が，信頼性評価の精度を向上させることが分かった．

フォールトの故障生起頻度を測定することは，小規模なソフトウェアでは比較的簡単であるが，大規模ソフトウェアについて行うのは大変な労力を要する．今後は，大規模ソフトウェアについて故障生起頻度を測定する手法の開発が必要となる．

参考文献

[1] Brocklehurst, S., Chan, P. Y., Littlewood, B. and Snell, J.:"Recalibrating software reliability models", *IEEE Trans. Software Eng.*, SE-16, 4, pp.

458-470, Apr.1990

[2] 菊野, 松本, 鳥居：ソフトウェア信頼性の実現技術 - 現状と今後の展望 - , 電子情報 通信学会誌, J73, 5, pp. 454-460 (1990).

[3] Littlewood, B. and Verall, J. L.："How good are they and how can they be improved ?", *IEEE Trans. Software Eng.*, SE-6, 5, pp. 489-500, Sept.1980

[4] Matsumoto, K., Inoue, K., Kikuno, T. and Torii, K.："Experimental evaluation of software reliability growth models", *Proc. of 18th FTCS*, pp. 148-153, 1988

[5] Schneidewind N. F.: "Software reliability model with optimal selection of failure data", *IEEE Trans. Software Eng.*, SE-19, 11, pp 1095-1104, Nov. 1993

[6] 当麻, 南谷, 藤原：フォールトトレラントシステムの構成と設計, 槙書店, pp. 251-254, 1991

[7] 山田, 大場,："エラー発見率に基づくS字形ソフトウェア信頼度成長モデルの考察", 情報処理学会論文誌, 27, 8, pp. 821-828, Aug. 1986