論　文

ソフトウェアフォールトごとの故障生起頻度の違いについての考察

島　和之[†]　松本　健一[†]　鳥居　宏次[†]

A Study on Differences between the Failure Intensity of Different Software Faults

Kazuyuki SHIMA[†], Ken'ichi MATSUMOTO[†], and Koji TORII[†]

あらまし　本論文では，ソフトウェアの信頼性をより正確に評価することを目的として，フォールトごとの故障の引き起こしやすさの違いに着目し，その指標である故障生起頻度を測定する方法と測定結果について述べる．従来のソフトウェア信頼度成長モデルには，各フォールトの故障生起頻度が同じであると仮定するモデル，テストの経過に伴って変化すると仮定するモデル，ガンマ分布等の特定の分布に従うと仮定するモデルがある．本論文では，故障生起頻度の測定結果に基づき，故障生起頻度分布を離散分布とするモデルを提案する．実測データに対して，提案するモデルと従来のモデルの累積フォールト数の予測精度について比較を行った結果，提案モデルによる予測が高い精度を示した．

キーワード　故障生起頻度，ソフトウェア信頼度成長モデル，超指数形 SRGM，Littlewood モデル，ガンマ分布，ソフトウェアの信頼性

1. まえがき

ソフトウェア信頼性を評価あるいは予測するための手段の一つとしてソフトウェア信頼度成長曲線が利用されている．ソフトウェア信頼度成長曲線は，横軸を経過時間，縦軸を検出したソフトウェアフォールト（故障の原因となるソフトウェア内の欠陥）の累積値としてプロットしたものである．経過時間としては暦時間または計算機実行時間があるが，一般には暦時間が用いられている．ソフトウェア信頼度成長モデル（以下，SRGM と略記する）は，この成長曲線特性を定式化し，検出フォールトデータから残存フォールト数とその検出に要する時間を推定するものである．

SRGM には，仮定や前提条件の違いにより，指数形 SRGM，遅延 S 字形 SRGM，習熟 S 字形 SRGM，修正指数形 SRGM，超指数形 SRGM，連結指数形 SRGM などがある．これらのモデルを実際のソフトウェア開発プロジェクトに適用する場合，その仮定や前提条件が対象プロジェクトの実態に適合していることが重要である．

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630-01 Japan

指数形 SRGM では，ソフトウェア内のどのフォールトの故障生起頻度（フォールトが実際に故障を引き起こす起こしやすさ）も同じであることを仮定している．しかし，NASA の実験でこの仮定に疑問を投げかける結果が示されている [4]．Littlewood は，ソフトウェア内のフォールトを無作為に選んだ場合に，そのフォールトの故障生起頻度が確率分布に従うと仮定するモデルを提案し，その確率分布としてガンマ分布を挙げている [1]．ガンマ分布は数学的に扱いやすいという特長をもつ．しかし，故障生起頻度がガンマ分布に従うという根拠は示されていない [3]．

本論文では，ソフトウェアのフォールトの故障生起頻度を測定する方法とその方法を小規模なプログラムに適用して得た測定結果について述べる．更に，その測定結果に基づき，フォールトの故障生起頻度分布を離散分布とするモデルを提案し，累積フォールト数の推定精度について評価を行う．故障生起頻度分布を離散分布とするモデルとしては，超指数形 SRGM がある．但し，超指数形 SRGM では，同一モジュールのフォールトの故障生起頻度は同じであると仮定している．提案するモデルでは，同一モジュールのフォールトでも異なる故障生起頻度をとり得ると仮定している．

2. では，フォールトの故障生起頻度の定義について

述べる．3. では，従来のソフトウェア信頼度成長モデルについて述べる．4. では，フォールトごとの故障生起頻度を測定する方法と測定結果について述べる．5. では，提案するモデルとそのパラメータ推定方法について述べる．6. では，結論について述べる．

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 [6]．フォールトによる故障の発生は入力データ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，あるフォールト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．この指標として，フォールト F が故障を引き起こす頻度（単位時間当りの平均回数）をフォールト F の故障生起頻度 b_F と定義する．

ここで，フォールト F が単位時間に故障を引き起こす確率を ϕ_F と定義すると，フォールト F が故障を引き起こす平均時間間隔は以下のように示される．

$$\frac{1}{b_F} = \int_0^\infty (1-\phi_F)^t dt \tag{1}$$

$$= \frac{-1}{\log(1-\phi_F)} \tag{2}$$

よって，以下の関係が成り立つ．

$$b_F = -\log(1-\phi_F) \tag{3}$$

フォールトはそのソフトウェア内の位置，故障を引き起こす入力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，フォールトの故障生起頻度分布と言う．なお，以降において故障生起頻度分布とは，フォールトの故障生起頻度分布を言うものとする．

3. 従来の SRGM における故障生起頻度

ここでは，故障生起頻度に着目して従来の SRGM を以下の四つに分類して述べる．

（1） すべてのフォールトの故障生起頻度を同じとするモデル

（2） 故障生起頻度が時間的に変化するモデル

（3） 同一モジュールの故障生起頻度を同じとするモデル

（4） 故障生起頻度が特定の分布に従うモデル

3.1 すべてのフォールトの故障生起頻度を同じとするモデル

指数形 SRGM では，時刻 t において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と仮定している．この仮定は，どのフォールトでも故障を引き起こす頻度，ひいては発見される確率が同じであることを意味する．これを式で表すと，以下の微分方程式が成り立つ．

$$\frac{dH(t)}{dt} = b(a - H(t)) \tag{4}$$

但し，

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：フォールトの故障生起頻度

よって，これを解くと

$$H(t) = a(1 - e^{-bt}) \tag{5}$$

が得られる．

b は，検出率，発生率または出現率と呼ばれる場合もあるが，本論文では故障生起頻度と呼ぶことにする．

3.2 故障生起頻度が時間的に変化するモデル

遅延 S 字形 SRGM，および，習熟 S 字形 SRGM では，信頼度成長曲線が S 字形になるという共通の特徴がある．フォールトの検出率でみると，試験開始からある時間が経過した時点で最大となり，以後指数関数的に減少する．フォールトの検出率が時間的に変化すると共に，残存フォールト数も変化することから，フォールトごとの故障生起頻度が異なると仮定していると言える．

3.3 同一モジュールの故障生起頻度を同じとするモデル

Matsumoto et al. により提案された超指数形 SRGM は，指数形 SRGM の拡張であり，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ごとにフォールトの故障生起頻度が異なると仮定する．但し，同一モジュール内ではフォールトの故障生起頻度は同じであるとす

る．また，各モジュールにおける平均値関数は，指数型である．よって，ソフトウェア全体としての平均値関数が以下の式で表される．

$$H(t) = a\sum_{i=1}^{n} p_i(1-e^{-b_i t}) \tag{6}$$

但し，

$$a > 0, \quad 0 < p_i < 1, \tag{7}$$

$$\sum_{i=1}^{n} p_i = 1 \tag{8}$$

ここで，

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個当りの故障生起頻度

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

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：フォールトの故障生起頻度

3.4 故障生起頻度が特定の分布に従うモデル

Littlewood が提案したモデルは，故障が発生する時間間隔を確率変数とする時間計測モデルの一つであり，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言う．このモデルでは，$i-1$ 番目の誤りが検出されてから i 番目の誤りが検出されるまでの時間間隔を表す確率変数 T_i の確率密度関数 $pdf(T_i|\lambda_i)$ は

$$pdf(T_i|\lambda_i) = \lambda_i e^{-\lambda_i T_i}$$

であるとする．ここで，λ_i 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トレートを表し，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となると仮定している．このモデルでは，フォールトの故障生起頻度 ν_j を一般に個々のフォールトによって異なるとし，以下の確率密度関数に従う確率変数と仮定している．

$$pdf(\nu) = \frac{e^{-\nu\sum_1^{i-1} t_j} f(\nu)}{\int_0^{\infty} e^{-\nu\sum_1^{i-1} t_j} f(\nu) d\nu}$$

$f(\nu)$ はハザードレート ν の事前分布式であり，例としてガンマ分布を仮定している．ガンマ分布は，以下の式で表されるような，二つのパラメータ α, β をもつ分布である．

$$f(\nu_j) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \nu_j^{\alpha-1} e^{-\beta\nu_j} \tag{9}$$

このモデルの基本式は，以下のとおりである．

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

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

$$\lambda_i = \sum_{j=1}^{N-i+1} \nu_j$$

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

4. 故障生起頻度の計測

Littlewood モデルで故障生起頻度分布として用いられているガンマ分布は Bayes 統計では事前分布の連続型分布としてよく仮定するものであり，ソフトウェア信頼性モデルのベイジアンモデルでもよく適用される分布である．しかし，現実の故障生起頻度がガンマ分布に従っているという測定結果が得られているわけではない．本論文では，フォールトの故障生起頻度を測定する方法と実際に測定した結果について述べる．

4.1 計測方法

以下の三つの手順により計測を行う．

（1） 対象プログラムのテスト・デバッグ

テストデータ T を用いて対象プログラムをテスト・デバッグする．対象プログラムは，最初にコンパイルエラーがなくなった時点のものを用いる．テスト・デバッグを行った際，フォールトの位置，および，変更内容をすべて記録する．これにより，ほぼフォールトが含まれていないと考えられる完成プログラム P およびフォールトの報告 $F_1, F_2, \cdots F_n$ を得る．

（2） フォールトの埋込み

各々のフォールトの故障生起頻度を調べるため，手順1で得た完成プログラムおよびフォールトの報告より，フォールトを，それぞれ一つだけ戻す．これにより，各フォールト $F_1, F_2, \cdots, F_n$ をそれぞれ一つだけ含むプログラムを得る．

（3） 故障生起頻度の測定

手順2で得た各フォールト $F_1, F_2, \cdots, F_n$ をそれぞれ一つだけ含むプログラムに対しテストを行い，単位時間に故障が発生した回数を調べ，各フォールトの故障生起頻度 $b_{F_1}, b_{F_2}, \cdots, b_{F_n}$ を求める．

故障生起頻度は，入力の頻度（単位時間当りの入力の回数）にも影響を受ける．但し，本論文では簡単のため入力の頻度を1と仮定する．このとき，フォールト F が単位時間に故障を発生する確率は，以下のように示される．

$$\phi_F = \frac{F\text{が故障を起こす入力数}}{\text{可能な全入力数}} \tag{10}$$

実際には，可能な全入力数が大きすぎて，可能な全入力についてテストを行うことが困難である場合が多い．このような場合は，できるだけ多くの入力についてテストを行い，以下の式で近似する．

$$\phi_F = \frac{F\text{が故障を起こすテスト入力数}}{\text{全テスト入力数}} \tag{11}$$

このようにして，各フォールトが単位時間に故障を発生する確率 $\phi_{F_1}, \phi_{F_2}, \cdots \phi_{F_n}$ を求め，式(3)より故障生起頻度 $b_{F_1}, b_{F_2}, \cdots b_{F_n}$ を求める．

4.2 計測結果

プログラム記述言語は，一般的な普及度を考慮してC言語とした．二つの異なる仕様A（酒屋の在庫管理），B（技術計算ソフトウェアのサブルーチン）のソフトウェア合わせて4本（A: 3本，B: 1本）について故障生起頻度を測定した．テストデータは，データ生成プログラムを作成し，ランダムに1000件生成した．プログラムの規模は約300行であり，それぞれ18個，9個，12個，16個のフォールトが検出された．各ソフトウェアのフォールトの故障生起頻度分布を図1〜4に示す．

実験により取得した各フォールトの故障生起頻度はフォールトよって明らかに異なっていた．これは，フォールトの故障生起頻度を一つのパラメータで示すことに問題があることを意味する．しかし，一方，複数のフォールトの故障生起頻度が一致している場合も多く，故障生起頻度の分布は離散分布とみなすことができる．この結果は，対象プログラムが小規模であったことが，その傾向を強めているとは言え，コンピュータソフトウェアの制御構造に起因するものであるので，多くのソフトウェアにも当てはまると考えられる．

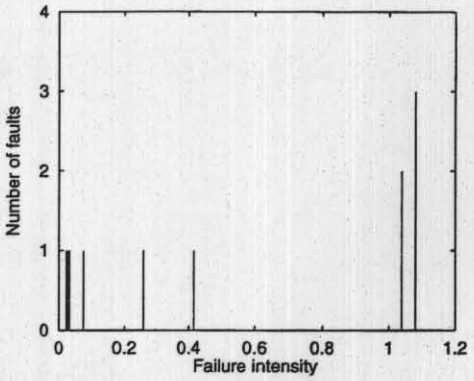


図1 フォールトの故障生起頻度（プログラム1）
Fig. 1 Failure intensity of faults. (Program 1)

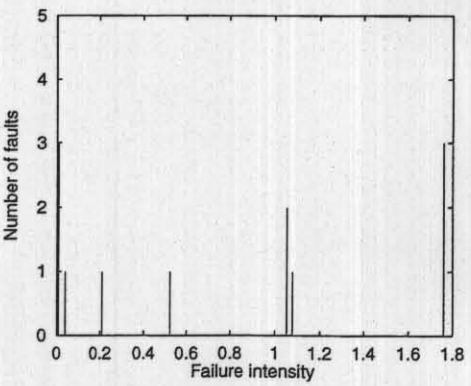


図2 フォールトの故障生起頻度（プログラム2）
Fig. 2 Failure intensity of faults. (Program 2)

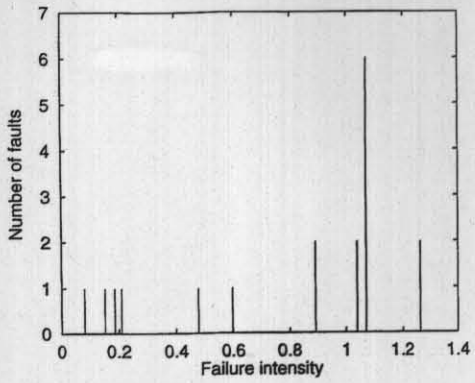


図3 フォールトの故障生起頻度（プログラム3）
Fig. 3 Failure intensity of faults. (Program 3)

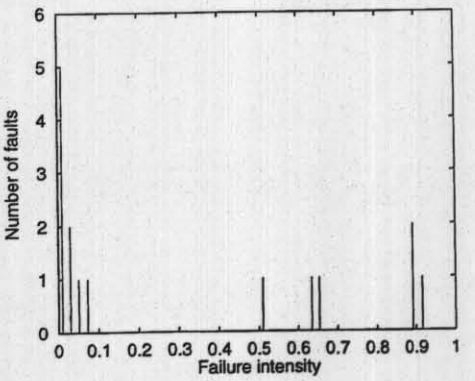


図4 フォールトの故障生起頻度（プログラム4）
Fig. 4 Failure intensity of faults. (Program 4)

4.3 分析方法

ここでは，フォールトの故障生起頻度がガンマ分布に従うという仮定について調べる．具体的には，まず最ゆう推定法によってガンマ分布のパラメータを推定する．故障生起頻度の実測データを $x_1, x_2, \cdots, x_n$ とすると，パラメータ α が

$$\frac{\Gamma(\alpha)'}{\Gamma(\alpha)} - \log\alpha = \frac{1}{n}\log\prod_{k=1}^{n} x_k - \log\frac{\sum_{k=1}^{n} x_k}{n} \quad (12)$$

の解を求めることによって得られ，パラメータ β が

$$\beta = \frac{n\alpha}{x_1 + x_2 + \cdots + x_n} \quad (13)$$

によって得られる．

但し，すべ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り，分析を行う上で問題となる．このようなフォールトは以下の2通りの方式で扱った．

（1） 故障生起頻度を無限大とする代わりに十分大きい値（10000）を用いる．これにより，故障生起頻度が無限大のフォールトの影響を見ることができる．

（2） 故障生起頻度が無限大のフォールトを無視する．故障生起頻度の高いフォールトはテストで発見されやすく，残存する可能性が低い．よって，そのようなフォールトが発見されても無視し，累積フォールト数も増やさないものとする．

次に，上記の方法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よく適合しているかどうかを検定する．検定法には，Kolmogorov-Smirnov 適合度検定を用いる．

4.4 分析結果

プログラム1〜4について，ガンマ分布のパラメータおよびKolmogorov距離を求め，有意水準2.5%で適合度検定を行った．プログラム1および3は，故障生起頻度が無限大となるフォールトを含んでいなかったので，前述の方式1や方式2を用いる必要はなかった．プログラム1および3についての検定結果を表1に示す．

プログラム2および4は，故障生起頻度が無限大となるフォールトを含んでいたので方式1あるいは方式2を用いて，パラメータの推定と適合度の検定を行った．方式1を用いた検定結果を表2に，方式2を用いた検定結果を表3に示す．

超指数形SRGMでは，モジュールごとに故障生起頻度が異なると仮定しているが，一つのモジュール内の複数のフォールトの故障生起頻度は同一と仮定している．この測定結果は，一つのモジュールのみからなる小規模なプログラムについての結果であるが，各フォールトの故障生起頻度は同一とはみなせなかった．プログラムの規模が大きくなれば，更に多くのフォールトが含まれ，それらの故障生起頻度が同一であると仮定することは難しくなる．

以下の三つのことが言える．

- 故障生起頻度の分布は離散分布とみなせる．なぜならば，複数のフォールトが同じ故障生起頻度をもつ場合が多いからである．
- 測定で得た分布の形状からは，ガンマ分布かどうかを判断することはできなかった．統計的検定ではガンマ分布の適合性が採択されることもあれば，棄却されることもあった．

表1 故障生起頻度に対するガンマ分布の適合度（プログラム1および3）
Table 1 Goodness of fit of gamma distribution to failure intensity. (Programs 1 and 3)

プログラム	prog1	prog3
パラメータ α	0.7419	2.2477
パラメータ β	1.3279	2.7870
Kolmogorov 距離	0.2902	0.3169
検定結果	採択	棄却

表2 方式1における故障生起頻度に対するガンマ分布の適合度（プログラム2および4）
Table 2 Goodness of fit of gamma distribution to failure intensity using method 1. (Programs 2 and 4)

プログラム	prog2	prog4
パラメータ α	0.1096	0.0991
パラメータ β	1.423×10^{-4}	1.585×10^{-4}
Kolmogorov 距離	0.4974	0.5152
検定結果	棄却	棄却

表3 方式2における故障生起頻度に対するガンマ分布の適合度（プログラム2および4）
Table 3 Goodness of fit of gamma distribution to failure intensity using method 2. (Programs 2 and 4)

プログラム	prog2	prog4
パラメータ α	0.5093	0.4765
パラメータ β	0.6599	1.5098
Kolmogorov 距離	0.2580	0.2191
検定結果	採択	採択

• たとえ小規模なプログラム（モジュール）の中のフォールトであっても，すべてのフォールトが同じ故障生起頻度をもつことはまれであると考えられる．

5. 提案するモデル

超指数形 SRGM は，故障生起頻度の違いをモジュールごとに見るという特徴はあるものの，故障生起頻度の異なるフォールトの存在を仮定している．また，故障生起頻度の分布を連続分布ではなく，離散分布とする点でも測定結果に合致している．そこで，超指数形 SRGM を変更して，フォールトごとに故障生起頻度の異なるモデルとする．

$$H(t) = \sum_{i=1}^{n} a_i(1 - e^{-b_i y_i}) \tag{14}$$

$$y_i = \begin{cases} 0, & t < t_i \\ t - t_i, & t \geq t_i \end{cases} \tag{15}$$

ここで，以下のようにおく．

$H(t)$：時間 t までに検出される期待フォールト数

a_i：時刻 t_i に混入した故障生起頻度 b_i のフォールトの数

b_i：フォールトの故障生起頻度

t_i：フォールトの混入時刻

このモデルは，超指数形 SRGM とほぼ同じである．但し，超指数形 SRGM では，モジュールごとにフォールトの混入時刻，フォールトの故障生起頻度，初期フォールト数などを定義している．ここで提案するモデルでは，同一モジュール内のフォールト間でも故障生起頻度は異なると仮定する．このため，超指数形 SRGM で用いられるような，モジュールごとにフォールトを識別してパラメータを決定する方法は使用できない．また，本論文の測定実験で用いたような，フォールトの埋込みとテストによる方法を，実際の開発現場において適用することは非現実的である．

そこで，フォールトの検出時間と検出数のデータから，本モデルに必要なパラメータを推定する方法について考える．但し，ここで求めるパラメータの値はデータの適合を目的として定めるものであり，実際のフォールトの混入時刻および故障生起頻度を示すものではない．しかし，このことは従来の SRGM におけるパラメータについても同様のことが言え，モデルを実用化する上では必要であると考える．

本モデルはパラメータの数が多いため，これを推定するには数値解析が必要である．しかし，最ゆう推定法や最小2乗法は計算コストが大きくなるので用いない．ここでは，より単純で計算量の少ないアルゴリズムによってパラメータを推定する方法を示す．この方法では，混入時刻の早いフォールトから順に，故障生起頻度と初期フォールト数を決定していく．まず，一つのフォールトについて混入時刻を定め，その時刻の検出フォールトの数だけ検出されるものとする．また，それ以後のそのフォールトの累積数が全体の累積フォールト数を超えないという制約のもとで，故障生起頻度を決定する．これを各混入時刻について繰り返し，パラメータを段階的に定めていく．

図5は，この方法によってフォールトの混入時刻と故障生起頻度を求めていった結果である．横軸は時間（単位は月），縦軸は検出フォールト数である．“Observed” の点は実測値を示し，“Estimated” は提案モデルによる推定値を示す．“First” はパラメータ推定方法の第1段階の結果を示す．曲線が実測値を超えてはならないという制約があるため，実測値よりもかなり低い値をとっている．“Second” は第2段階の

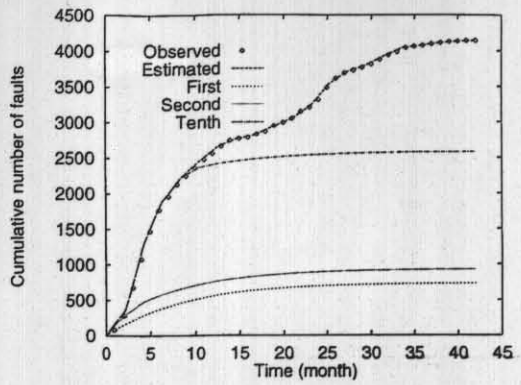


図5 提案モデルによる信頼度成長曲線の推定
Fig. 5 Estimation of reliability growth curve based on the proposed model.

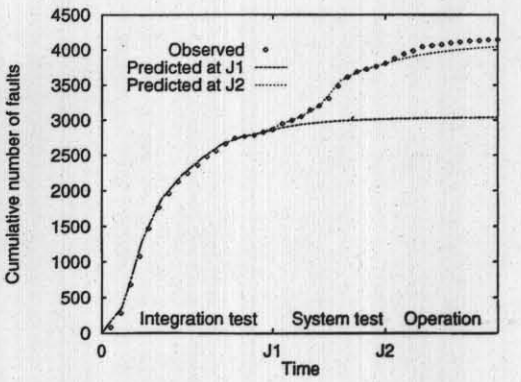


図6 提案モデルによる累積フォールト数の予測
Fig. 6 Prediction of the cumulative number of faults based on the proposed model.

結果を示す．2段階目で定めたフォールトを第1段階で求めた曲線に上乗せする．“Tenth” は10段階目で，かなり実測値に近づいている．このようにして，20段階まで行った結果が “Estimated” である．

図6は，提案モデルを用いて実際のフォールトデータ[2]の予測を行った結果を示している．横軸は時間，縦軸は検出フォールト数である．横軸の J1 は統合試験終了時点を，J2 はシステム試験終了時点を示している．0から J1 までは統合試験期間，J1 から J2 まではシステム試験期間，J2 以降は運用期間を意味する．“Observed” の点は実測値を示す．“Predicted at J1” および “Predicted at J2” は，それぞれ統合試験終了時点およびシステム試験終了時点における予測結果を示している．統合試験終了時点における予測の精度はあまり良くないが，システム試験終了時点における予測の精度はかなり良いことがわかる．

表4 累積フォールト数の予測精度の比較（実測値 4142）
Table 4 Comparison of the estimation precision of the number of cumulative faults. (Actual value = 4142)

SRGM	予測値	誤差
指数形	4449	307
遅延S字形	3929	213
連結指数形	4276	134
提案モデル	4204	62

表4は，システム試験終了時点における予測精度について提案モデルと従来の SRGM（指数形，遅延S字形，連結指数形）とを比較した結果を示している．予測値は，各モデルを用いて計測終了時点における累積フォールト数を予測した結果を示している．誤差は，各モデルによる予測値と実測値 4142 との差を示している．この表より，提案モデルによる予測精度が高いことがわかる．

6. むすび

本論文では，ソフトウェアフォールトごとの故障生起頻度の違いについて考察し，故障生起頻度分布として離散分布を仮定する SRGM を提案し，そのモデルのパラメータの推定方法を示した．従来の SRGM では，同じソフトウェアまたは同じモジュールに混入しているフォールトの故障生起頻度は同じであると仮定する場合や故障生起頻度の分布としてガンマ分布を仮定する場合などがあった．提案するモデルでは，フォールトによって故障生起頻度は異なり，その分布が離散分布に従うと仮定する．

一つのモジュールからなる小規模なプログラムついて，フォールトごとの故障生起頻度を調べた結果，故障生起頻度が同じであるとはみなせず，また，その分布がガンマ分布に従うとは限らないことが示された．提案するモデルでは，これらの結果に基づいて，故障生起頻度分布を離散分布としている．また，本論文では，実際のフォールトデータの予測精度について，従来の SRGM（指数形，遅延S字形，連結指数形）と提案モデルとを比較した．その結果，提案モデルによる累積フォールト数の予測値の誤差が最も小さいことが示された．

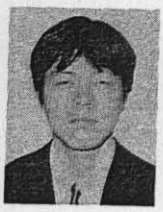
本論文では，超指数形 SRGM の解釈を変更することによって，故障生起頻度分布を考慮したモデルを構築した．今後は，他の SRGM の特長を取り入れることにより，より予測精度の高い SRGM を構築するこ

とが課題である．

文　献

[1] B. Littlewood, "How good are they and how can they be improved ?," IEEE Trans. Software Eng., vol.SE-6, no.5, pp.489–500, Sept. 1980.
[2] 中川　豊，"ソフトウェア信頼度成長曲線の分析に基づく連結指数形ソフトウェア信頼度成長モデル，" 信学論（D-I），vol.J77D-I, no.6, pp.433–442, June 1994.
[3] 杉元秀人，島　和之，松本健一，鳥居宏次，"ソフトウェアフォールトの故障生起頻度について，" 情処学研報，vol.96, no.6, pp.17–24, Jan. 1996.
[4] 当麻喜弘，南谷　崇，藤原秀雄，"フォールトトレラントシステムの構成と設計，" pp.251–254, 槙書店，1991.
[5] 山田　茂，大場　充，"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，" 情処学論，vol.27, no.8, pp.821–828, Aug. 1986.
[6] 山田　茂，木村光宏，"制約のないテスト労力投入量を考慮したソフトウェア信頼度成長モデル，" 信学論（D-I），vol.J78D-I, no.6, pp.532–538, June 1995.

（平成8年3月11日受付，7月15日再受付）



島　和之　（正員）

平3阪大・基礎工・情報卒．平6同大大学院博士課程中退．同年奈良先端科学技術大学院大学・助手．工修．高信頼性ソフトウェアの研究に従事．

松本　健一　（正員）

昭60阪大・基礎工・情報卒．平1同大大学院博士課程中退．同年同大・基礎工・情報・助手．平5奈良先端科学技術大学院大学・助教授．工博．ソフトウェア開発における計測環境，ソフトウェア品質保証の枠組みに関する研究に従事．情報処理学会，IEEE各会員．

鳥居　宏次　（正員）

昭37阪大・工・通信卒．昭42同大大学院博士課程了．同年電気試験所（現電子技術総会研究所）入所．昭50ソフトウェア部言語処理研究室室長．昭59阪大・基礎工・情報教授．平4奈良先端科学技術大学院大学・教授．工博．ソフトウェア工学の研究に従事．情報処理学会，日本ソフトウェア学会，人工知能学会，ACM，IEEE各会員．