ソフトウェア工学 70－1
（1989．12．8）

運用段階におけるソフトウェア信頼性評価法とその応用

山田茂　谷尾嘉一　尾崎俊治
広島大学工学部（第二類）

本論文では，ソフトウェア開発の最終段階であるテスト工程とユーザに対するリリース後の運用段階とを関係づけて，ソフトウェア内の潜在エラーに起因するソフトウェア故障の発生現象を記述する．このとき，テスト工程におけるソフトウェアエラー発見事象を，非同次ポアソン過程に基づくソフトウェア信頼度成長モデルにより記述する．また，運用段階の信頼性すなわち運用信頼性の評価については，テスト終了時点におけるソフトウェア内の残存エラー数を考慮して，環境係数を導入する方法，ハザードレートによる方法，および累積故障率に基づく方法の3つについて議論する．最後に，ソフトウェアの信頼性データ解析の結果を用いて，運用信頼性に関する評価法の数値例を示す．

Software Reliability Assessment Methods during Operation Phase and Their Application

Shigeru Yamada　Yoshikazu Tanio　Shunji Osaki

Faculty of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 724 Japan

This paper discusses several methods of software reliability assessment, in which the software failure occurrence phenomenon during testing and operation phases are described by stochastic models. The failure occurrence (or software error detection) phenomenon during testing phase is formulated by software reliability growth models based on a nonhomogeneous Poisson process. The failure occurrence phenomenon during operation phase is formulated by three methods dependent on the number of remaining software errors at the initial time point of operational use : Environment factor, hazard rate, and cumulative failure rate. Then, we can obtain several operational software reliability measures. Finally,numerical examples of software reliability assessment during operation phase are presented for illustration.

1. まえがき

近年，銀行のオンラインシステムの事故にみられるように，コンピュータシステムの故障や障害が起きると社会に及ぼす影響は非常に大きく深刻である．しかしながら，銀行のシステム担当者が，大規模かつ複雑なシステムの「欠陥を完全に無くするのは無理」と述べているように，その故障を完全に推定し予測することは困難とされている．このため，コンピュータシステムの高信頼化を図ることが重要な課題となり，様々な研究がなされている．特に最近では，コンピュータシステムの主要な構成要素であるソフトウェアの高信頼化が緊急を要する問題となっている．ソフトウェアの開発過程は多くの人手に頼っているため，人為的誤りまたは欠陥いわゆるソフトウェアエラー（以下エラーと略す）が入り込むことが不可避である．ソフトウェアの信頼性問題は，このエラーに起因して発生するソフトウェア故障に関係するものである．ここで，ソフトウェア内に潜在するエラーにより，期待通りにソフトウェアが動作しない現象が起こることを，ソフトウェア故障(software failure)と定義する．したがって，ソフトウェア開発の最終段階であるテスト工程や次の運用段階におけるソフトウェアの信頼性評価が重要な問題となる．

ソフトウェアの信頼性評価は，ソフトウェア開発の最終段階であるテスト工程において観測される，エラーの発見と修正に関するデータを使って行われ，テストの進捗状況の監視とユーザに対するリリース時期の見積りのために必要不可欠である．このために，テストにおけるエラー発見事象を時系列現象として記述し，確率・統計論に基づく多くのソフトウェアの信頼性モデルが提案されている．このうち，テストの進行と共にソフトウェア内のエラーが発見・修正されていく過程を，ソフトウェアの信頼度成長過程と見なすモデルは，ソフトウェア信頼度成長モデルと呼ばれている(software reliability growth model，山田[1]およびRamamoorthy and Bastani[2]参照)．このモデルから導出される，ソフトウェア内の期待残存エラー数，ソフトウェア故障の平均発生時間間隔，ソフトウェア信頼度などの定量的尺度により，ソフトウェアの信頼性評価を行うことができる．また，ソフトウェア開発のテスト工程だけでなく，ソフトウェアがリリースされた後の運用段階における信頼性，すなわち運用信頼性の評価も重要である．実際には，過去のフィールドデータと経験により，運用段階における上記のような信頼性評価尺度を推定・予測することも可能であるが，ソフトウェアの信頼性モデルによるテスト工程の信頼性評価結果に基づく方法もよく採用されている（例えばMusa et al.[3]参照）．

本論文では，ソフトウェア開発の最終段階であるテスト工程とリリース後の運用段階とを関係づけて，ソフトウェア内の潜在エラーに起因するソフトウェア故障の発生現象をモデル化し，運用信頼性の評価法について議論する．このとき，テスト工程におけるエラー発見事象を非同次ポアソン過程(nonhomogeneous Poisson process，以下NHPPと略す)[4]に基づくソフトウェア信頼度成長モデルにより記述する．また，運用信頼性については，テスト終了時点におけるソフトウェア内の残存エラー数を考慮して，環境係数を導入する方法，ハザードレートによる方法，および累積故障率に基づく方法の3つの評価技法を提案する．最後に，ソフトウェアの信頼性データ解析の結果を用いて，運用信頼性に関する評価法の数値例を示す．

2. ソフトウェア信頼度成長モデル[1],[5]

2.1 モデルの記述

知的生産物であるソフトウェアは，その開発過程のほとんどが人的作業から成り立っているので，エラーが入り込むことは避けられない．そこで，ソフトウェア開発の最終段階であるテスト工程においては，所定の手順に従ってソフトウェア内に潜在するエラーを発見し修正することが反復して行われる．このようなエラー発見事象あるいはソフトウェア故障発生事象を記述して，テスト工程におけるソフトウェアの信頼性評価を行うために，テスト時刻tまでに発見された総エラー数を表す計数過程$\{N(t),\ t\geq 0\}$を導入する．この計数過程に対してNHPPを仮定し，$N(t)$の平均値を$H(t)$とすれば，ソフトウェア信頼度成長モデルは，

$$\Pr\{N(t)=n\}=\frac{\{H(t)\}^n}{n!}\exp[-H(t)] \qquad (n=0,1,2,\cdots), \tag{1}$$

と定式化される．ここで$H(t)$は，NHPPの平均値関数と呼ばれ，テスト時刻tまでに発見される総期待エラー数を表す．テスト開始前に潜在する総期待エラー数をaとすれば，一般に$H(\infty)=a$であるから，式(1)より$N(t)$の極限分布は，

$$\lim_{t\to\infty}\Pr\{N(t)=n\}=\frac{a^n}{n!}e^{-a} \qquad (n=0,1,2,\cdots), \tag{2}$$

となるポアソン分布であることがわかる．

平均値関数$H(t)$をもつNHPPに基づくソフトウェア信頼度成長モデルから，ソフトウェアの信頼性評価に

有用な定量的尺度が導出できる．テスト時刻tにおいてソフトウェア内に残存する総期待エラー数は，

$$n(t) = a - H(t), \tag{3}$$

により与えられる．また，テスト時刻tにおいてエラーが発見されたという条件の下で，時間区間$(t, t+x]$ $(x \geq 0)$でエラーが発見されない確率は，

$$R(x|t) = \exp[-\{H(t+x) - H(t)\}] \quad (t \geq 0,\ x \geq 0), \tag{4}$$

により与えられる．式(4)は，ソフウェア信頼度(software reliability)と呼ばれる．

実際に信頼性評価を実施するときには，平均値関数$H(t)$に具体的な関数形を与え，モデルを特定化する必要がある．例えば，Goel and Okumoto[6]は，単位テスト時間当りに発見されるエラー数は，その時点において残存するエラー数に比例するものと仮定して，

$$H(t) \equiv m(t) = a(1 - e^{-bt}) \quad (b > 0), \tag{5}$$

とするNHPPにより，ソフトウェア故障発生現象を記述した．ここで，パラメータbは，任意のテスト時刻における1個当りのエラー発見率を表す．このモデルは，テスト時間に関して発見された総エラー数が指数形成長曲線を示す場合に適切であり，指数形ソフトウェア信頼度成長モデルと呼ばれている．これに対して，テスト工程では，発見された総エラー数がS字形成長曲線を示す場合もよく観測され，このエラー発見事象をYamada et al.[7]は，

$$H(t) \equiv M(t) = a[1 - (1+bt)e^{-bt}] \quad (b > 0), \tag{6}$$

とするNHPPにより記述した．ここで，パラメータbは，定常状態における1個当りのエラー発見率を表す．このモデルは，テストにおけるエラーの発見・除去過程を説明するのに有用であり，遅延S字形ソフトウェア信頼度成長モデルとよばれている．

2.2 パラメータの推定

NHPPに基づくソフトウェア信頼度成長モデルを，実際のデータに適用してデータ解析を行うとき，そのモデルパラメータは最尤法により推定できる．

テスト工程においてk番目のソフトウェア故障発生時刻s_k $(k = 1,2,\cdots,n;\ 0 \leq s_1 \leq s_2 \leq \cdots \leq s_n)$に関するデータが観測されたものとする．このとき，式(1)の平均値関数$H(t)$をもつNHPPモデルに対して，n個の実測データs_k $(k = 1,2,\cdots,n)$に対する尤度関数は，その同時確率密度関数$f(s_1, s_2, \cdots, s_n)$により与えられ，

$$L \equiv f(s_1, s_2, \cdots, s_n) = \prod_{k=1}^{n} h(s_k) \cdot \exp[-\int_{s_{k-1}}^{s_k} h(x)dx], \tag{7}$$

となる．ここで，$h(t) \equiv dH(t)/dt$，$s_0 = 0$とする．式(7)の尤度関数の自然対数をとると，

$$\ln L = -H(s_n) + \sum_{k=1}^{n} \ln h(s_k), \tag{8}$$

を得る．

したがって，NHPPに基づく指数形ソフトウェア信頼度成長モデルでは，式(8)において$H(t) \equiv m(t)$とし，モデルパラメータaおよびbに関する偏微分をとって，

$$\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial b} = 0, \tag{9}$$

とおいて得られる同時尤度方程式を，数値的に解けば各パラメータの最尤推定値が求まる．また，NHPPに基づく遅延S字形ソフトウェア信頼度成長モデルでは，式(8)において$H(t) \equiv M(t)$とし，同様にすればモデルパラメータの最尤推定値が求まる．

3. 運用段階の信頼性評価

前節では，テスト工程において発見・修正されるエラー数の時間的傾向や変動に着目して，計数過程を導入するソフトウェア信頼性評価を議論した．ここでは，ソフトウェア開発の最終段階であるテスト工程を経て，ユーザにリリースされた後の運用信頼性についてその評価方法を提案する．この運用信頼性は，テストを実施したことによりどの程度ソフトウェアからエラーが除去されたかということに関係しているといえる．すなわち，テスト終了時点におけるソフトウェア内の残存エラー数に関係すると考えることができる．

3.1 環境係数の導入

まず，テスト工程におけるエラー発見の環境とリリースされた後の運用段階におけるソフトウェアの使用環境の違いに着目する．例えば，製品としてリリースされたソフトウェアが不特定多数のユーザに対して，様々なハードウェアあるいはシステム上において運用される場合，運用段階における使用環境は，テスト段階のエラー発見の環境に比べて厳しいかもしれない．この場合，真の残存エラー数は，テスト終了時刻Tで推定されたソフトウェア内の期待残存エラー数に比べると，運用段階ではより多いものと予想される．逆に，ユーザが特定化されており，運用段階の使用環境もそれ程厳しくなければ，テスト終了時刻Tで推定されたソフトウェア内の期待残存エラー数に比べると，真の残存エラー数は，運用段階ではより少ないものと予想される．したがって，このようなソフトウェアの使用状況を

考慮するために，式(3)から運用段階における期待残存エラー数を次式のように定式化する．

$$\bar{n}(t) = c \cdot n(t) \qquad (c > 0,\ t \geq T). \tag{10}$$

ここで，cを環境係数と呼び，テスト工程のエラー発見の環境と運用段階の使用環境の違いを示す係数とする．また，$n(T)$はテスト終了時刻Tにおいて未発見となった期待残存エラー数を表す．上式において，$c > 1$では運用段階の使用環境が厳しく，残存エラー数が増加している場合，逆に$c < 1$では運用段階の使用環境がそれ程厳しくなく，見かけ上残存エラー数が減少している場合を表す．なお，期待残存エラー数を表す$n(t)$の推定値は，テスト工程において観測されたソフトウェアエラーデータを2.2の方法により解析し，推定されたモデルパラメータを式(3)に代入すれば求められる．また，パラメータcについては，以前の実績データおよび経験から与えるものとする．

式(10)から，信頼性評価尺度となる式(4)のソフトウェア信頼度は，

$$\bar{R}(x \mid t) = \exp[-\{\bar{n}(t) - \bar{n}(t+x)\}] \qquad (t \geq T), \tag{11}$$

により与えられる．

3.2 ハザードレートによる方法

運用段階におけるソフトウェアの信頼性評価法として，ソフトウェア故障の発生頻度を運用時間に対するハザードレート(hazard rate)により捉えることが可能である．ここで，運用の開始時点から測って時刻yにおけるハザードレート$z(y)\Delta y$は，時刻yまでにソフトウェア故障が発生しなかったという条件のもとで，微小時間区間$(y,\ y+\Delta y]$において発生する確率を意味する．したがって，時刻yにおけるハザードレート$z(y)$は，式(3)からテストをT時間実施して終了した時点でソフトウェア内に残存するエラー数$n(T)$および運用時間yに比例するものと仮定すれば，

$$z(y;\ n(T)) = pn(T)y^{m-1} \qquad (p > 0,\ m > 0,\ y \geq 0), \tag{12}$$

により与えられる．ここで，パラメータpは比例定数を表し，パラメータmは形状パラメータを表す．このように，ソフトウェア故障発生現象をハザードレートにより取り扱う研究には，たとえばJelinski and Moranda[8]，Moranda[9]，Shooman[10]などに見られる．式(12)は，ソフトウェア故障の発生時間分布がワイブル(Weibull)分布[11]であることを意味する．すなわち，尺度パラメータおよび形状パラメータがそれぞれt_0およびmとするワイブル分布のハザードレートは，

$$z(y) = mt_0y^{m-1} \qquad (m > 0,\ t_0 > 0,\ y \geq 0), \tag{13}$$

により表されるので，式(12)および式(13)を比較すれば$pn(T)$がmt_0に対応していることがわかる．式(12)から，$m = 1$のときは，運用時間yに無関係となり，ソフトウェア故障の発生時間分布は指数分布に従うことになる．これは，Jelinski and Moranda[8]やShooman[10]の信頼性モデルに対応している．また，$m = 2$のときは，ソフトウェア故障の発生時間分布はレイリー(Rayleigh)分布に従い，Schick and Wolverton[12]の信頼性モデルに対応していることがわかる．このように，式(12)のハザードレートは，運用段階における各種のソフトウェア故障の発生パターンを表現できるので適用性の高い指標となっている．

さらに，式(12)から，ソフトウェアの運用開始後，時間区間$(0,\ y]$に対する確率密度関数および信頼度関数は，それぞれ，

$$f(y) = pn(T)y^{m-1} \cdot \exp[-\frac{pn(T)}{m}y^m], \tag{14}$$

$$R(y) = \exp[-\frac{pn(T)}{m}y^m], \tag{15}$$

により与えられる．さらに，運用開始後に最初のソフトウェア故障が発生するまでの時間Yの平均値，すなわち平均故障時間(mean time to failure，以下MTTFと略す)とその分散は，それぞれ，

$$MTTF = [\frac{m}{pn(T)}]^{\frac{1}{m}}\Gamma(1+\frac{1}{m}), \tag{16}$$

$$Var[Y] = (\frac{m}{pn(T)})^{\frac{2}{m}}\{\Gamma(1+\frac{2}{m}) - \Gamma^2(1+\frac{1}{m})\}, \tag{17}$$

となる．ここで，$\Gamma(x)$はガンマ関数を表し，

$$\Gamma(x) = \int_0^\infty u^{x-1}e^{-u}du,$$

である．

実際に上述の信頼性評価を行うために，式(12)におけるハザードレートのパラメータを推定する必要がある．まず，テスト終了時点において未発見となったソフトウェア内の期待残存エラー数$n(T)$の推定値は，テスト工程において観測されたソフトウェアエラーデータを2.2の方法により解析して，推定されたモデルパラメータを式(3)に代入すれば求まる．次に，パラメータpおよびmは，一般にはソフトウェアの運用開始前にこれらの推定値が必要となるので，以前の同種のソフトウェア開発プロジェクトの実績データおよび経験から

最尤法により推定することにする．そこで，以前の実績データからl個のソフトウェア故障発生時間間隔のデータy_i $(i=1,2,\ldots,l)$が利用可能であるとき，この観測データ$\{y_1,y_2,\ldots,y_l\}$に対する同時確率密度関数すなわち尤度関数は，式(14)を使って

$$L_H \equiv \prod_{i=1}^{l} f(y_i) = (pn(T))^l \prod_{i=1}^{l} y_i^{m-1} \exp[-\frac{pn(T)}{m} y_i^m], \quad (18)$$

により与えられる．式(18)から対数尤度関数$\ln L_H$を求め，同時尤度方程式$\partial \ln L_H/\partial p = \partial \ln L_H/\partial m = 0$よりパラメータ$p$および$m$の最尤推定値を求めることができる．

3.3 累積故障率に基づく方法

前節では，ソフトウェアの運用開始後，最初にソフトウェア故障が発生するまでの時間を問題にした．実際には，3.1でも議論したように，同一のソフトウェアを不特定多数のユーザが使用するため，単一のソフトウェア故障の発生時間分布ではその故障発生現象を取り扱うことができない場合もある．本節では，ハードウェアの新製品開発時に実施される信頼性試験において，その信頼性評価のために使われている信頼度成長モデル(1)を適用して，ソフトウェア故障の発生現象を記述することにする．このモデルは，開発試験において連続的に信頼度改善の効果があるとき，累積故障率(cumulative failure rate)に対する開発試験に費やされた累積時間は両対数グラフ上で直線関係にあるという経験則から出発して，これをNHPP(4)により理論的裏付けを行ったものである．

この累積故障率による信頼度成長モデルを使うと，運用開始後の時間区間$(0, y]$におけるソフトウェア故障に対する累積故障率は，運用時間yに対する総故障数$N(y)$の比として$N(y)/y$と定義される．この累積故障率の期待値から，単位時間当りに発生するソフトウェア故障数を表す運用段階の瞬間故障率は，

$$r(y) \equiv \frac{d}{dy}\{E[N(y)]\} = \lambda n(T) y^{m-1}, \quad (19)$$

により与えられる．式(19)は，テストをT時間実施して終了した時点でソフトウェア内に残存するエラー数$n(T)$および運用時間yに比例するものと仮定している．ここで，$r(y)$はNHPPにおける強度関数であり，これよりNHPPの平均値関数$M(y)$は，

$$M(y) = E[N(y)] = \frac{\lambda n(T) y^m}{m}, \quad (20)$$

で表される．平均値関数$M(y)$は運用時間区間$(0, y]$において発生する総期待エラー数を表している．また，累積故障率$r(y)$はワイブル分布と関数形が同一であり，ワイブル型信頼度成長モデル(1)となる．

式(20)の平均値関数$M(y)$をもつNHPPモデルは，式(1)と同様にして運用段階におけるソフトウェア故障発生現象が記述できる．このとき，運用段階におけるl個のソフトウェア故障発生時刻u_k $(k=1,2,\cdots,l;\ 0 \le u_1 \le u_2 \le \cdots \le u_l)$の同時確率密度関数は式(7)と同様にして，

$$L_C \equiv f_{U_1,U_2,\cdots,U_l}(u_1,u_2,\cdots,u_l) = \prod_{k=1}^{l} r(u_k) \cdot \exp[-\int_{u_{k-1}}^{u_k} r(x)dx]$$

$$= \exp[-\frac{\lambda n(T)}{m} u_l^m] \prod_{k=1}^{l} \lambda n(T) u_k^{m-1}, \quad (21)$$

により与えられる．式(21)から，ソフトウェアの運用開始後の時間区間$(0, y]$に対する確率変数U_kの確率密度関数および分布関数は，それぞれ

$$f_{U_k}(y) = \frac{\lambda n(T) y^{m-1} \{\frac{\lambda n(T)}{m} y^m\}^{k-1}}{\Gamma(k)} \exp[-\frac{\lambda n(T)}{m} y^m], \quad (22)$$

$$F_{U_k}(y) = \int_0^{\frac{\lambda n(T)}{m} y^m} \frac{x^{k-1} e^{-x}}{\Gamma(k)} dx, \quad (23)$$

により与えられる．さらに，$(k-1)$番目とk番目の間のソフトウェア故障時間間隔を示す確率変数X_kについて，その分布関数，期待値すなわち平均故障時間間隔(mean time between failures，以下MTBFと略す)，およびその分散は，それぞれ以下の式のようになる．

$$F_{X_k}(t) = 1 - \int_0^{\infty} \frac{\lambda n(T) y^{m-1} \{\frac{\lambda n(T)}{m} y^m\}^{k-2}}{\Gamma(k-1)} \times \exp[-\frac{\lambda n(T)}{m}(y+t)^m] dy, \quad (24)$$

$$E[X_k] = (\frac{1}{m})(\frac{m}{\lambda n(T)})^{\frac{1}{m}} \frac{\Gamma(k+\frac{1}{m}-1)}{\Gamma(k)}, \quad (25)$$

$$Var[X_k] = \{\frac{2\Gamma(k+\frac{2}{m}-1)}{m\Gamma(k)} - \frac{\Gamma^2(k+\frac{1}{m}-1)}{m\Gamma^2(k)}(\frac{1}{m}+2k-2)\} \times (\frac{m}{\lambda n(T)})^{\frac{2}{m}}. \quad (26)$$

20pt ここで，運用時間間隔$(0, y]$における瞬間故障率$r(y)$の逆数も，任意のソフトウェアの運用時刻における平均故障時間間隔すなわちMTBFを表し，特に瞬間MTBF(以下，$(MTBF)_I$と表す．)と呼ばれる(1)．また，累積故障率$N(y)/y$の期待値の逆数も，ソフトウェアの運用開始時点から考えたときのMTBFを意味し，累積MTBF(以下，$(MTBF)_C$と表す．)と呼ばれる(1)．

$$(MTBF)_I = \frac{1}{r(y)} = \frac{1}{\lambda n(T) y^{m-1}}, \quad (27)$$

$$(MTBF)_C = \frac{1}{E[N(y)/y]} = \frac{m}{\lambda n(T) y^{m-1}}. \quad (28)$$

実際に運用段階の信頼性評価を行うためには，式(19)の瞬間故障率を推定する必要がある．まず，総テスト時間Tでの残存エラー数$n(T)$は，テスト工程における信頼性評価結果から求められる．次に，パラメータλおよびmは，3.2と同様に以前の同種のソフトウェア開発プロジェクトの実績データおよび経験から最尤法により推定できる．そこで，以前の実績データとしてl個のソフトウェア故障発生時間データ$u_i\,(i=1,2,\cdots,l)$が利用可能であるものとする．この観測データ$\{u_1,u_2,\cdots,u_l\}$に対する尤度関数は，式(21)により与えられ，式(21)から対数尤度関数$\ln L_C$を求め，同時尤度方程式$\partial \ln L_C/\partial\lambda = \partial \ln L_C/\partial m = 0$よりパラメータ$\lambda$および$m$の最尤推定値を求めることができる．

4. 数値例

実際に観測されたデータを使って，本論文で議論した運用信頼性に関する評価法の数値例を示す．適用対象としたデータは，Goel and Okumoto[6]が引用した31個のソフトウェア故障発生時間$s_k\,(k=1,2,\cdots,31$; 単位は日数)に関するものである(表1参照)．彼らは，最初の

表1．ソフトウェア故障発生時間データ[6]

Failure No. n	Time Between Failures x_n (days)	Cumulative Time $s_n=\sum x_k$ (days)
Phase 1		
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250
Phase 2		
27	87	337
28	47	384
29	12	396
30	9	405
31	135	540

26個のデータ$s_k\,(k=1,2,\cdots,26)$を用いて，式(5)の平均値関数$m(t)$をもつ指数形ソフトウェア信頼度成長モデルによりテスト工程における信頼性評価を行い，残りの5個のデータ$s_k\,(k=27,28,\cdots,31)$をモデルの予測精度の検証のために使っている．本論文では，最初の26個のデータ$s_k\,(k=1,2,\cdots,26)$をテスト工程の実測データとし，残りの5個のデータ$s_k\,(k=27,28,\cdots,31)$を運用段階の実測データとして解析する．

テスト工程において観測されたデータ$s_k\,(k=1,2,\cdots,26)$を，指数形ソフトウェア信頼度成長モデルにより解析する．まず，式(9)の尤度方程式を解くと，モデルパラメータの最尤推定値として$\hat{a}=33.99$および$\hat{b}=5.79\times10^{-3}$を得る．したがって，テスト終了時刻を$T=s_{26}=250$として，テスト工程において未発見となったソフトウェア内の期待残存エラー数は式(3)より，

$$\hat{n}(T)=33.99\cdot e^{-5.79\times10^{-3}\times250}=7.99, \tag{29}$$

となる．この推定結果をもとに，実際の運用段階の信頼性評価を行う．

まず，3.1で述べた環境係数を導入する方法について考える．ここで，環境係数cを，0.5, 1.0, 2.0と変化させると，式(11)のソフトウェア信頼度の推定値は図1となった．図1から，運用段階における時間の経過とともに

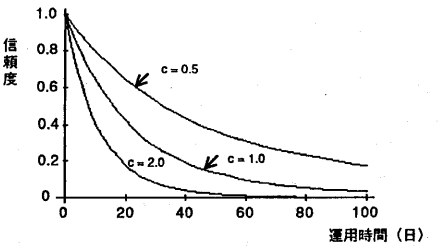

図1．環境係数の導入による運用ソフトウェア信頼度

にソフトウェア故障が発生する確率が高くなっていることがわかる．また，環境係数cの値が大きいほど，すなわち運用段階での使用環境が厳しいほど信頼度の減少率が大きくなっている様子がわかる．

次に，ハザードレートによる運用信頼性の評価を行う．まず，データ$s_k\,(k=27,28,\cdots,31)$を使って，式(12)のハザードレートのパラメータpおよびmを推定する．運用段階におけるソフトウェア故障発生時間間隔データを$y_i=s_{26+i}-s_{26+i-1}\,(i=1,2,\cdots,5)$として，式(18)の尤度関数により同時尤度方程式を導き，2分法(bisection method)により解くとパラメータの推定値，

$$\hat{m}=1.12,\qquad \hat{p}=1.41\times10^{-3}, \tag{30}$$

を得る．ここで，式(29)および式(30)の推定結果より，式(16)のMTTFの推定値は$\widehat{MTTF}=57.99$(日)となり，式(17)の分散の推定値は$\widehat{Var}[Y]=2678$(日)となる．さらに，式(15)のソフトウェアの信頼度の推定値$\hat{R}(y)$は図2に示した．テスト終了後の運用時間$y=100$(日)

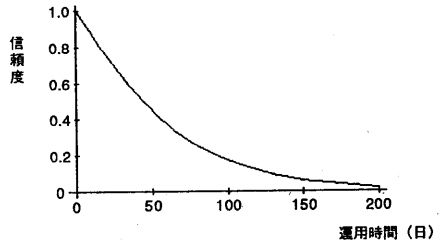

図2．ハザードレートによる運用信頼度

に対する信頼度は，図2から $\hat{R}(100)=0.172$ であることがわかり，ソフトウェア故障の発生する確率が高いことを示している．表1から，実際に運用後最初のソフトウェア故障が $y=87$（日）において発生しており，この推定結果によく一致していることがわかる．

最後に，累積故障率に基づく運用信頼性の評価について考える．まず，ハザードレートによる方法と同様に，データ s_k $(k=27,28,\cdots,31)$ から式(19)の累積故障率のパラメータ λ および m を推定しなければならない．ソフトウェア故障発生時間データ $u_i=s_{26+i}-s_{26}$ $(i=1,2,\cdots,5)$ として，式(21)の尤度関数より同時尤度方程式を導き，2分法により解くとパラメータの推定値，

$$\hat{m}=0.932, \qquad \hat{\lambda}=2.96\times10^{-3}, \tag{31}$$

を得る．式(29)および式(31)の推定結果から，式(25)のMTBFの推定値 $\hat{E}[X_k]$ は図3のようになる．図3より，運用段階において発見されるエラーの時間間隔は発見されるエラー数に対して増加している様子がわかり，ソフトウェアの信頼性が高くなっている．また図4においては，式(27)および式(28)で求めた瞬間MTBFおよび累積MTBFが，運用時間の経過に対して増加している様子を示した．さらに，式(23)の分布関数と補完関係にある信頼度，すなわち運用時刻 y までに k 個目の故障が発生しない確率の推定結果を図5に示した．ここで，運用段階における1個目のソフトウェア故障に関して，運用時間 $y=100$（日）に対する信頼度は，図5より $\hat{R}_{U_1}(100)=1-\hat{F}_{U_1}(100)=0.157$ であり，ソフトウェア故障が発生する確率が高くなっている．実際にも1個目のソフトウェア故障が87（日）で起こっているので，よく一致した結果といえる．また，運用時間100（日）において5個以上のソフトウェア故障が発生する確率は非常に小さいことも図5から明かである．

5. むすび

本論文では，ソフトウェア開発の最終段階であるテスト工程とユーザに対するリリース後の運用段階とを関係づけて運用信頼性の評価法について議論した．このとき，テスト工程におけるエラー発見事象をNHPPに基づくソフトウェア信頼度成長モデルにより記述し，テスト工程における実際データによるパラメータ推定法を示した．また運用信頼性の評価については，テスト終了時点におけるソフトウェア内の残存エラー数を考慮して，3種類の評価技法を提案した．まず第1には，テスト環境と運用環境の違いを表す環境係数を導入することにより，運用開始時点の残存エラー数を見積り，信頼性評価尺度であるソフトウェア信頼度を導出した．第2には，運用段階におけるソフトウェア故障に対するハザードレートが，テスト終了後の残存エラー

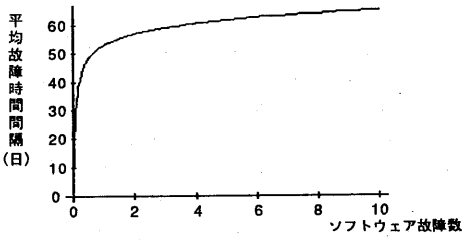

図3．累積故障率に基づく平均故障発生時間間隔

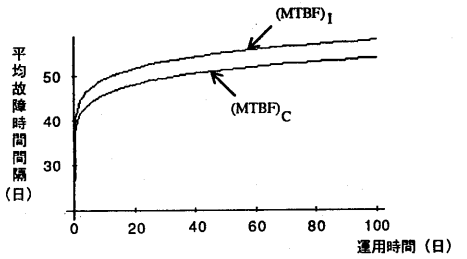

図4．瞬間MTBFと累積MTBF

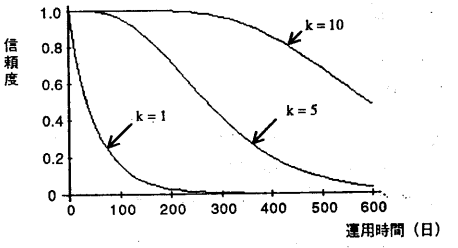

図5．累積故障率に基づく運用信頼度

数と運用時間に比例するものと仮定して，その結果運用段階におけるソフトウェア故障の発生現象をワイブル分布で記述できた．この方法により，運用段階における信頼度やMTTFなどの信頼性評価尺度を導出できた．第3には，運用段階で発生するソフトウェア故障に対する累積故障率に基づいて信頼性評価法を議論した．ここで，NHPPにより記述される瞬間故障率をテスト終了後の残存エラー数と運用時間に比例するものと仮定し，ワイブル型信頼度成長モデルによりソフトウェア故障の発生現象を記述できた．この方法により，運用段階における信頼度やMTBFなどの運用信頼性の評価尺度を導出できた．さらに，第2および第3の方法においては運用段階でのソフトウェア故障発生時間に関する実績データが利用できるとき，仮定したモデルパラメータの推定をするためのデータ解析方法を示した．以上の運用信頼性の評価法について，実際データを適用した結果，引用データに対しては運用段階におけるソフトウェア故障の発生現象がよく記述されていることがわかった．

本論文では，実際問題としても重要なソフトウェアの運用信頼性の評価に対して3種類の考え方を示した．実際には，これらの信頼性評価法を適用するにあたり，運用段階のソフトウェア故障発生現象を記述するのに必要なパラメータを推定するための実績データが必要不可欠である．本論文では，これを実績データが利用可能な場合について考察したが，より実際的見地からのデータ収集に関する手順を確立することは今後の課題となろう．また，多くの実際データに対する適用結果から，本論文で提案した信頼性評価法の妥当性を検証していくことが必要である．さらに，運用信頼性の評価法の応用として，例えば，運用信頼性の各種の評価尺度に対して目標値を与えて，それを達成するのに必要な総テスト時間を求める最適リリース問題[13]–[15]について研究を進めていく予定である．

参考文献

(1) 山田茂：ソフトウェア信頼性評価技術，HBJ出版局 (1989)．

(2) C. V. Ramamoorthy and F. B. Bastani : " Software reliability - Status and perspectives ", *IEEE Trans. Software Eng.*, Vol. SE-8, No. 4, pp. 354 – 371 (July 1982).

(3) J. D. Musa, A. Iannino and K. Okumoto : *Software Reliability : Measurement, Prediction, Application* , McGraw - Hill, New York (1987).

(4) H. Ascher and H. Feingold : *Repairable Systems Reliability : Modeling, Inference, Misconceptions and Their Causes* , Marcel Dekker, New York (1984).

(5) S. Yamada and S. Osaki : " Software reliability growth modeling : Models and applications ", *IEEE Trans. Software Eng.*, Vol. SE - 11, No. 12, pp. 1431 – 1437 (Dec.1985).

(6) A. L. Goel and K. Okumoto : " Time - dependent error - detection rate model for software reliability and other performance measures ", *IEEE Trans. Reliability*, Vol. R - 28, No. 3, pp. 206 – 211 (Aug. 1979).

(7) S. Yamada, M. Ohba and S. Osaki : " S - shaped reliability growth modeling for software error detection ", *IEEE Trans. Reliability*, Vol. R - 32, No. 5, pp. 475 – 478, 484 (Dec. 1983).

(8) Z. Jelinski and P. B. Moranda : " Software reliability research ", in *Statistical Computer Performance Evaluation*, W. Freiberger (ed.), pp. 465 – 484, Academic Press, New York (1972).

(9) P. B. Moranda : " Event - altered rate models for general reliability analysis ", *IEEE Trans. Reliability*, Vol. R - 28, No. 5, pp. 376 – 381 (Dec. 1979).

(10) M. L. Shooman : " Probabilistic models for software reliability prediction ", in *Statistical Computer Performance Evaluation*, W.Freiberger (ed.), pp. 485 – 502, Academic Press, New York (1972).

(11) N. R. Mann, R. E. Schafer and N. D. Singpurwalla : *Methods for Statistical Analysis of Reliability and Life Data* , John Wiley & Sons, New York (1974).

(12) G. J. Schick and R. W. Wolverton : " Analysis of competing software reliability models ", *IEEE Trans. Software Eng.*, Vol. SE - 4, No.2, pp. 104 – 120 (Mar. 1978).

(13) H. S. Koch and P. Kubat : " Optimal release time for computer software ", *IEEE Trans. Software. Eng.*, Vol. SE - 9, No. 3, pp. 323 – 327 (Mar. 1983).

(14) K. Okumoto and A. L. Goel : " Optimal time for software systems based on reliability and cost criteria ", *J. System and Software*, Vol. 1, pp. 315 – 318 (1980).

(15) S. Yamada and S. Osaki : " Cost - reliability optimal release policies for software systems ", *IEEE Trans. Reliability*, Vol. R - 34, No. 5, pp. 422 – 424 (Dec. 1985).