ソフトウェアの信頼性実証試験方式

澤田　清，三道　弘明

1. はじめに

信頼性実証試験（Reliability Demonstration Testing）[1,2]は，元々ハードウェアの品質保証の一環として考案されたものであり，その目的は次のとおりである．すなわち，製品の開発段階終了後，それを受注先に納入あるいは市場に開放する前に，目標とする信頼性が十分に実現されているかどうかを実証・確認することである．従って，試験に合格したものは，そのまま納入もしくは市場開放する．しかし，試験に合格しなかった製品については，それを廃棄したり，あるいは設計にフィードバックすることにより，さらに品質向上を目指すことになる．ソフトウェアの品質保証が問題となっている今日，ソフトウェア製品に対しても上述のような信頼性実証試験を実施することは，信頼性という意味での品質向上に貢献すると考えられる[3～6]．

本稿では，ソフトウェアの信頼性実証試験の種々の方式，およびそれらの設計方法について述べる．ソフトウェアの信頼性実証試験では，試験中に発生するソフトウェア障害により，対象ソフトウェアの合否を判定する．ここで，信頼性実証試験は，試験中のソフトウェア障害回数に基づく方式と，ソフトウェア障害による損失の大きさを考慮した方式に分類される[7]．

２章で述べるソフトウェア障害回数に基づく試験は，試験中に生起するソフトウェア障害の回数を数え，それがあらかじめ定められた値を超えれば，試験に不合格とするという方式である．ここでは，ソフトウェアの利用形態に基づいて，ソフトウェアの種類を次の２つに分類している．すなわち，計算機のＯＳや生産システムの制御ソフトウェアのように時間に関して連続的に用いられるソフトウェア（連続型モデル）と数値計算ソフトウェアや通常のアプリケーションソフトウェアのように時間に関して離散的に使用されるソフトウェア（離散型モデル）である．これらの２つのモデルに対して，統計的検定論の考え方に基づいて，試験を実施したときの生産者リスクと消費者リスクを定式化し，これらの上限値を指定することにより試験を設計する方法を示す．さらに，より簡単に試験の設計ができるKullback-Leiblerの分離情報量を用いた方法についても述べる．

３章では，ソフトウェア障害による損失の大きさ（以後，ソフトウェア障害の大きさとよぶ）を考慮した信頼性実証試験方式を展開する．この方式は，ソフトウェア障害の大きさが定量的に評価可能な場合に，有効である．ここでは，試験中のソフトウェア障害回数があらかじめ設定された値を超えるか，またはソフトウェア障害の大きさの累積が既定値を超えれば，試験に不合格とし，それ以外は合格とする方式について述べる．

2. ソフトウェア障害回数に基づく信頼性実証試験

本章では，試験中に生起するソフトウェア障害回数に基づいてソフトウェアの合否を判定する試験方式およびその設計方法について述べる．

2.1 連続型モデル[8]

計算機のＯＳや生産システムの制御ソフトウェアのようなソフトウェアは，時間に関して連続的に用いられる．このようなソフトウェアの信頼性評価は，通常，平均ソフトウェア障害時間（MTTSF: Mean Time To Software Failure）や平均ソフトウェア障害時間間隔（MTBSF: Mean Time Between Software Failures）のような時間的特性，あるいはソフトウェア障害に対する障害発生率に関してなされる．ここでは，このようなソフトウェアを扱うモデルを連続型モデルとよぶこととする．

このようなソフトウェアに対して，次のような信頼

さわだ　きよし　流通科学大学情報学部
さんどう　ひろあき　流通科学大学情報学部
〒 651-2188 神戸市西区学園西町 3-1

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

性実証試験方式を考える．すなわち，対象ソフトウェアに対してt時間の試験を実施し，試験期間中に生起したソフトウェア障害回数がs以下ならばそのソフトウェアを合格とし，$s+1$以上ならば不合格とする．この試験は，$t\,(t>0)$と$s\,(s=0,1,2,\cdots)$の値を決定することにより，設計される．

ここで，ソフトウェアの生産者（開発者）がその開発を受注したときのソフトウェアのMTBSFに対する契約の値，および消費者（ユーザ）が受け入れ可能なMTBSFの下限値をそれぞれθ_0，θ_1と書くこととする（$\theta_0>\theta_1$）．さらに，t時間の試験中のソフトウェア障害回数が$s+1$以上あるという事象をE_1，E_1の余事象を$\overline{E}_1$と書くこととする．

ここでは，次のように仮定する．

(i) ソフトウェア障害時間間隔は平均θの指数分布に従う．

(ii) 試験中に発生したソフトウェア障害に対するフォールト（ソフトウェア障害の原因となるプログラム内の誤り）の検出・修正は，試験終了後にまとめて実施する．

(iii) θ_0，θ_1は試験開始時点でのMTBSFに対する値を表す．

ソフトウェア開発段階に実施されるテスト工程では，ソフトウェア障害が発生すると，その原因となるフォールトの検出・修正が即座に行われる．したがって，テスト工程中にMTBSFが変化する（一般に大きくなる）．しかし，信頼性実証試験では，上の仮定(ii)のように，試験中にフォールトの検出・修正を実施しないのが普通である．これは，MTBSFが試験中には変化しないことを意味している．また，仮定(iii)は，信頼性実証試験が試験開始時点，すなわち開発段階終了時点のソフトウェアを評価することを意味している．

以上の仮定により，ハードウェアの信頼性実証試験同様，上述したソフトウェアの信頼性実証試験についても，以下のように統計的検定論を適用できる．

ここで，ソフトウェアのMTBSFθを検定対象の未知パラメータとし，$\theta=\theta_0$を帰無仮説，$\theta=\theta_1$を対立仮説として考えることとする．このとき，前述した信頼性実証試験を実施した結果，試験に不合格となる，すなわち事象E_1が生起した場合，帰無仮説を棄却する（対立仮説を採択する）ことに対応する．また，試験に合格する，すなわち事象$\overline{E}_1$が生起した場合は，帰無仮説を採択することに対応する．

このとき，第１種の過誤および第２種の過誤の確率，すなわち生産者リスク，消費者リスクはそれぞれ

$$Pr[E_1|\theta_0] = 1-\sum_{i=0}^{s}\frac{(t/\theta_0)^i}{i!}e^{-t/\theta_0} \tag{1}$$

$$Pr[\overline{E}_1|\theta_1] = \sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} \tag{2}$$

となる．したがって，生産者リスク，消費者リスクの上限をそれぞれ，α，βと指定することによって得られる次の連立不等式を(t,s)に関して解くことにより設計可能である．

$$1-\sum_{i=0}^{s}\frac{(t/\theta_0)^i}{i!}e^{-t/\theta_0} \leq \alpha \tag{3}$$

$$\sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} \leq \beta \tag{4}$$

ただし，上の連立不等式の解は一般には一意に定まらないので，sが非負の整数であることと，試験時間tをできる限り小さくするということを考慮して，(t,s)の値を決定することとなる．

2.2 離散型モデル[4,9]

数値計算ソフトウェアや通常のアプリケーションソフトウェアのようなソフトウェアは，時間に関して離散的に用いられる．このようなソフトウェアは，ここでは離散型モデルとして扱うこととする．離散型モデルの場合，MTTSFやMTBSFのような時間的特性を用いて信頼性を評価することは適切ではない．このため，ソフトウェアの離散型モデルの評価基準として，次のようなソフトウェアの不信頼度を定義する．

ソフトウェアが必要とする一組の入力変数の値の集合を単に入力とよぶこととすると，ソフトウェアの不信頼度は，次のように定量的に定義することができる．すなわち，対象ソフトウェアに対する可能な入力の数をN，そのうちソフトウェア障害を生起させる入力数をN_0と表すこととする．このとき，N個の可能な入力から任意に選んだ１つの入力に対して，ソフトウェア障害が生起する確率はN_0/Nで与えられる．なお，通常N，N_0の値は非常に大きく，天文学的な数値となる．前述の確率をpで表す，すなわち$p=N_0/N$とおくと，pは対象ソフトウェアの不信頼度を表すと解釈することができる．

上述したような性質を有するソフトウェアに対して，次のような信頼性実証試験を考える．すなわち，

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

対象ソフトウェアに対する可能な入力N個の中から任意に選んだn個の入力を用いて対象ソフトウェアを試験し，ソフトウェア障害を引き起こした入力の数がc以下ならそのソフトウェアを合格とし，$c+1$以上ならば不合格とする．この試験の設計変数は，$n\,(n=1,2,\cdots)$と$c\,(c=0,1,2,\cdots,n-1)$である．

ここで，ソフトウェアの生産者（開発者）がその開発を受注したときのソフトウェアの不信頼度に対する契約の値，および消費者（ユーザ）が受け入れ可能な不信頼度の上限値をそれぞれp_0，p_1と書くこととする（$p_0<p_1$）．さらに，n個の入力を用いた試験中に，ソフトウェア障害を引き起こす入力が$c+1$以上あるという事象をE_2，E_2の余事象を$\overline{E}_2$と書くこととする．

また，2.1の連続型モデルと同様の理由により，次のように仮定する．

(i) 試験中に発生したソフトウェア障害に対するフォールトの検出・修正は，試験終了後にまとめて実施する．

(ii) p_0，p_1は試験開始時点での不信頼度に対する値を表す．

これらの仮定の下で，連続型モデルの場合と同様に統計的検定論に基づいて，生産者リスク$Pr[E_2|p_0]$と消費者リスク$Pr[\overline{E}_2|p_1]$が定義される．この場合も，生産者リスク，消費者リスクの上限をα，βと設定することによって得られる次の連立不等式を解けばよい．

$$\sum_{i=c+1}^{n}\binom{n}{i}{p_0}^i(1-p_0)^{n-i} \leq \alpha \tag{5}$$

$$\sum_{i=0}^{c}\binom{n}{i}{p_1}^i(1-p_1)^{n-i} \leq \beta \tag{6}$$

ただし，nは自然数，cはn未満の非負の整数であることと，入力数nをできる限り小さくするということを考慮して，(n,c)の値を決定することとなる．

2.3 Kullback-Leiblerの分離情報量に基づく設計方法

ここでは，2.1で述べた連続型モデルの信頼性実証試験方式に対して，Kullback-Leiblerの分離情報量を用いた設計方法を示す．なお，2.2の離散型モデルについてはここでは述べないが，同様の手順で設計可能である[10]．

2つの確率分布$F_1(x)$，$F_2(x)$の密度関数をそれぞれ$f_1(x)$, $f_2(x)$とするとき

$$J(F_1,F_2) \equiv \int_0^\infty [f_1(x)-f_2(x)]\log\frac{f_1(x)}{f_2(x)}dx \tag{7}$$

をKullback-Leiblerの分離情報量(Divergence)と呼ぶ[11,12]．分離情報量が大きいほど，分布$F_1(x)$と$F_2(x)$は弁別されやすくなる．すなわち，分離情報量は二つの分布間の距離を表す一つの尺度としてとらえることができる．

2.1の記号と仮定の下で，対象ソフトウェアの真のMTBSFが契約の値θ_0であるときに，事象E_1，$\overline{E}_1$が起こる確率は，それぞれ

$$Pr[E_1|\theta_0] = 1-\sum_{i=0}^{s}\frac{(t/\theta_0)^i}{i!}e^{-t/\theta_0} \tag{8}$$

$$Pr[\overline{E}_1|\theta_0] = \sum_{i=0}^{s}\frac{(t/\theta_0)^i}{i!}e^{-t/\theta_0} \tag{9}$$

となる．また，対象ソフトウェアの真のMTBSFが受け入れ可能な下限値θ_1であるときに，事象E_1，$\overline{E}_1$が起こる確率は，それぞれ

$$Pr[E_1|\theta_1] = 1-\sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} \tag{10}$$

$$Pr[\overline{E}_1|\theta_1] = \sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} \tag{11}$$

となる．

このとき，式(8)，(9)および式(10)，(11)は，それぞれ，E_1，$\overline{E}_1$という2つの事象からなる離散型確率分布となるので，これらの分布の分離情報量が最も大きくなる（2つの確率分布が最も弁別される）ような(t,s)を選べば，真のMTBSFに対して誤った試験結果を下す可能性が最も小さいこととなる．

上述した2つの分布の分離情報量は

$$J(F_1,F_2) = [A(t,s)-B(t,s)]\log\frac{A(t,s)[1-B(t,s)]}{B(t,s)[1-A(t,s)]} \tag{12}$$

であるから，これを最大にするような(t,s)を求めればよい．ただし

$$A(t,s) = \sum_{i=0}^{s}\frac{(t/\theta_0)^i}{i!}e^{-t/\theta_0} \tag{13}$$

$$B(t,s) = \sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} \tag{14}$$

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

である．

Kullback-Leiblerの分離情報量を用いた設計方法は，統計的検定論に基づく設計方法と比べて，生産者リスクや消費者リスクの上限値を指定する必要がないため，パラメータが少なく設計変数の値をより簡単に求めることができる．

3. ソフトウェア障害の大きさを考慮した信頼性実証試験

本章では，試験中に生起するソフトウェア障害の大きさを考慮した信頼性実証試験方式について述べ，その後で統計的検定論に基づく設計方法を示す．なお，ここでは，時間に関して連続的に使用されるソフトウェアを対象とした信頼性実証試験（連続型モデル）[13]について述べるが，時間に関して離散的に使用されるソフトウェアの場合（離散型モデル）[14]についても，同様の考え方で設計可能である．

ここで，対象ソフトウェアに対してt時間の試験を実施し，試験期間中に生起したソフトウェア障害の回数およびソフトウェア障害の大きさの累積に基づき次のような判定を行う．ソフトウェア障害回数がs以下で，かつソフトウェア障害の大きさの累積がd以下ならば合格とする．ソフトウェア障害回数が$s+1$以上，またはソフトウェア障害の大きさの累積がdより大きければ不合格とする．この場合の設計変数は，$t\,(t>0)$，$s\,(s=1,2,\cdots)$，$d\,(d\geq 0)$である．

ここで，θ_0，θ_1については2.1の定義をそのまま用い，それ以外の記号については以下に定義する．

X_k　k番目に生起するソフトウェア障害の大きさ（$k=1,2,\cdots$）

$F(x|\mu)$　ソフトウェア障害の大きさの平均（以後，平均ソフトウェア障害サイズとよぶ）がμの場合のソフトウェア障害の大きさの分布関数

$f(x|\mu)$　$F(x|\mu)$の密度関数

μ_0　ソフトウェアの生産者がその開発を受注したときの，平均ソフトウェア障害サイズに対する契約の値

μ_1　消費者が受け入れ可能な平均ソフトウェア障害サイズの上限値（$\mu_1>\mu_0$）

E_3　t時間の試験中のソフトウェア障害回数が$s+1$以上，またはソフトウェア障害の大きさの累積がdより大きいという事象

$\overline{E}_3$　E_3の余事象

ここでは，2.1の仮定に加えて，次のような仮定を設ける．

(i) k番目に生起するソフトウェア障害の大きさX_k $(k=1,2,\cdots)$は，それぞれ独立で同一の確率分布$F(x|\mu)$に従う．

(ii) μ_0，μ_1は試験開始時点での平均ソフトウェア障害サイズに対する値を表す．

このとき，生産者リスク，消費者リスクを次のように定義する．

$$
\begin{aligned}
&Pr[E_3|\theta_0,\mu_0]\\
&= \sum_{i=1}^{s} Pr\left[\sum_{k=1}^{i} X_k > d \,\middle|\, \mu_0\right] \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0}\\
&\quad + \sum_{i=s+1}^{\infty} \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0}\\
&= \sum_{i=1}^{s} \overline{F}^{(i)}(d|\mu_0) \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0}\\
&\quad + \sum_{i=s+1}^{\infty} \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0} \qquad (15)
\end{aligned}
$$

$$
\begin{aligned}
&Pr[\overline{E}_3|\theta_1,\mu_1]\\
&= \sum_{i=1}^{s} Pr\left[\sum_{k=1}^{i} X_k \leq d \,\middle|\, \mu_1\right] \frac{(t/\theta_1)^i}{i!} e^{-t/\theta_1}\\
&\quad + e^{-t/\theta_1}\\
&= \sum_{i=1}^{s} F^{(i)}(d|\mu_1) \frac{(t/\theta_1)^i}{i!} e^{-t/\theta_1} + e^{-t/\theta_1} \qquad (16)
\end{aligned}
$$

但し，

$$F^{(i)}(d|\cdot) = \int_0^d F^{(i-1)}(d-x|\cdot) f(x|\cdot)dx \qquad (17)$$

$$F^{(1)}(d|\cdot) = F(d|\cdot) \qquad (18)$$

$$\overline{F}^{(i)}(d|\cdot) = 1 - F^{(i)}(d|\cdot) \qquad (19)$$

である．

この場合も，生産者リスク，消費者リスクの上限をそれぞれ，α，βと指定することによって，以下のように連立不等式の形で定式化できる．ただし，ここでは，障害の大きさとして指数分布を仮定した．

$$
\begin{aligned}
&\sum_{i=1}^{s}\left[\sum_{k=0}^{i-1} \frac{(d/\mu_0)^k}{k!} e^{-d/\mu_0}\right] \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0}\\
&\qquad + \sum_{i=s+1}^{\infty} \frac{(t/\theta_0)^i}{i!} e^{-t/\theta_0} \leq \alpha \qquad (20)
\end{aligned}
$$

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

$$\sum_{i=1}^{s}\left[\sum_{k=i}^{\infty}\frac{(d/\mu_1)^k}{k!}e^{-d/\mu_1}\right]\frac{(t/\theta_1)^i}{i!}e^{-t/\theta_1} + e^{-t/\theta_1} \leq \beta \qquad (21)$$

4. おわりに

本稿では，ソフトウェアの信頼性実証試験の種々の方式，およびそれらの設計方法について述べた．ここで示した設計方法では，設計変数の値は解析的には得られていないが，簡単な数値計算により求めることができる．また，本稿で述べた試験方式以外に，試験中に生起した障害回数が0の場合のみ対象ソフトウェアを合格とするゼロ障害型信頼性実証試験方式がある．ゼロ障害型信頼性実証試験方式については，統計的検定論に基づく設計方法[15]，Kullback-Leiblerの分離情報量を用いた設計方法[15]以外に，テスト工程で得られる事前情報を有効利用することを意図したベイズ論を用いた設計方法もある[16]．これらの試験方式および設計方法を，ソフトウェアの種類やユーザの要求によりうまく使い分けることが重要である．

参考文献

[1] 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).

[2] H. F. Martz and R. A. Waller: "Bayesian Reliability Analysis", John Wiley & Sons, New York (1982).

[3] 日本生産管理学会(編): "生産管理ハンドブック", 日刊工業新聞社, pp.450–460 (1999).

[4] C. K. Cho: "An Introduction to Software Quality Control", John Wiley & Sons, New York (1980). / 後藤公雄(監訳): "高品質ソフトウェア — ソフトウェア品質管理概論", 近代科学社 (1982).

[5] 山田茂: "ソフトウェア信頼性モデル — 基礎と応用", 日科技連 (1994).

[6] 三道弘明: "ハードウェアとの対比によるソフトウェアの信頼性実証試験方法に関する研究", 電子情報通信学会論文誌, Vol.J72–A, No.11, pp.1881–1886 (1989).

[7] K. Sawada and H. Sandoh: "A summary of software reliability demonstration testing models", *International Journal of Reliability, Quality and Safety Engineering*, Vol.6, No.1 (1999) (to appear).

[8] H. Sandoh: "Reliability demonstration testing for software", *IEEE Trans. Reliability*, Vol.R–40, No.1, pp.117–119 (1991).

[9] H. Sandoh and K. Sawada: "Reliability demonstration testing for discrete-type software products", *Proc. of 1991 Annual Reliability and Maintainability Symposium*, Orlando, U. S. A., pp.428–432 (1991).

[10] 澤田清, 三道弘明, 藤井進: "Kullback-Leiblerの情報量に基づくソフトウェアの信頼性実証試験に関する離散型モデル", 電子情報通信学会論文誌, Vol.J79–A, No.3, pp.830–833 (1996).

[11] S. Kullback: "Information Theory and Statistics", John Wiley & Sons, New York (1959).

[12] 宮沢光一: "情報・決定理論序説", 岩波書店 (1976).

[13] K. Sawada and H. Sandoh: "Continuous model for software reliability demonstration testing considering damage size of software failures", *Mathematical and Computer Modelling* (1999) (to appear).

[14] 澤田清, 三道弘明: "障害の大きさを考慮したソフトウェアの信頼性実証試験", 電子情報通信学会論文誌, Vol.J81–A, No.1, pp.98–109 (1998).

[15] K. Sawada and H. Sandoh: "Zero-failure software reliability demonstration testing based on Kullback-Leibler information", *Stochastic Modelling in Innovative Manufacturing*, Lecture Notes in Economics and Mathematical Systems Vol.445 (ed. by A. H. Christer, S. Osaki and L. C. Thomas), pp.272-280, Springer-Verlag, Berlin (1997).

[16] 三道弘明, 澤田清: "ソフトウェアに対するゼロ障害型信頼性実証試験に関する研究", 電子情報通信学会論文誌, Vol.J73–A, No.3, pp.564–569 (1990).

© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.