ソフトウェア工学 21－4
（1981. 11. 16）

プログラム・テストの妥当性評価に関する実験報告
―― ソフトウェア品質＝テストの質×テストの量 ――

大場 充 （日本アイ・ビー・エム 製品保証）

［概要］
人工的に作ったバグをテスト対象プログラムに移植しておき，テスト完了後移植したバグがテスト中にどの程度除去されたかによってプログラム・テストの妥当性を量的に評価しようとする方法に関する実験結果を報告する．

1．はじめに

プログラム・テスト期間中に発見される機能欠陥（バグ）の数の推移は，巨視的に次の平均値関数 $m(t)$ をもつ非定常ポアソン過程でよく記述されると言われている[1]．すなわち，テスト開始後 t 時間目における発見欠陥数 $m(t)$ は：

$$m(t) = \bar{N}(1 - e^{-bt}), \qquad (1)$$

で与えられる．ここで，パラメータ $\bar{N}$ はそのテストで発見可能な機能欠陥の総数を，パラメータ b はテストの効率を示す．(1) 式のモデルは図1に示されている曲線の性質をもつ．すなわち，

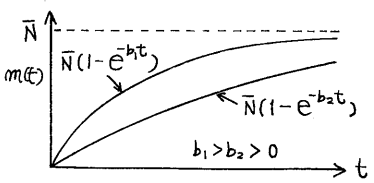

図1．平均値関数 $m(t)$ の性質

$\bar{N}$ が一定の場合，b の値が大きいほど短時間で欠陥を数多く発見できる．ところでテストで発見可能な欠陥の総数 $\bar{N}$ とは，テスト開始から完了までに行ったテストと同様のものを無限に実施したと仮定した場合に発見できると推定される欠陥の総数を言う．従って，$\bar{N}$ はテストを行う作業担当者の欠陥発見能力や，テストに必要なツール類に依存して定まるもので，対象プログラム中に潜在するバグそのものの総数を示すとは限らない．テストで発見可能な欠陥の総数 $\bar{N}$ とプログラム中に潜在するバグの総数が一致しない原因として，次の2点が考えられる：

① テストが不十分で発見の難度の高いバグが未発見のままテストが完了した，
② プログラム中に潜在したバグが少量のためパラメータ N の推定誤差が大きい．

以上のような問題のため，平均値関数 $m(t)$ をもつ非定常ポアソン過程モデルでソフトウェアの品質やテストの妥当性を評価することは実用上問題がある．すなわち，前掲の非定常ポアソン過程モデルではテストの量的側面のみしか評価することができない．従って，テストの質が良い場合や，プログラムの機能が単純なオフライン環境で動作するソフトウェアの場合には良く適合するが，プログラムの機能が複雑で高度なテスト技術を必要とするオンライン環境で動作するソフトウェアの場合，非定常ポアソン過程モデルは現実に適合しない例が観測される．

上記のような問題点を改善し，プログラム・テストの妥当性を客観的に評価する方法として，H．ミルズによって提案された Capture－Recapture 法を基礎に[2,3]，非定常ポアソン過程モデルによるテストの量的評価に対しさらにテストの質的評価を加味する尺度 SSPQL を提案し，その実験結果について報告する．

2．SSPQL

プログラム・テストの妥当性評価，

すなわちテスト完了時におけるソフトウェアの品質評価尺度，SSPQL (Shipping Software Product Quality Level) は，次式のように定義される[9]：

$$SSPQL = \alpha(t_m) \times \gamma(t_m), \quad (2)$$

ここで，$\alpha(t_m)$ はテスト開始から完了までの時間を t_m とするときの，テスト完了時におけるテストの精度を示す尺度であり，$\gamma(t_m)$ はテスト完了時におけるテストの網羅度 (Test Coverage) を示す尺度である．テストの精度はテストの質的側面を評価する尺度に相等する．すなわちプログラム中に潜在するバグをテスト中にどの程度の確実さで捕捉したかを示す．またテストの網羅度はテストの量的側面を評価する尺度に相等する．すなわちプログラム中に潜在しているバグの中で，テスト中に欠陥として発見可能なもののうちどの程度の量をテスト中に発見したかを示す．(2) 式で定義される SSPQL は，テストの精度と網羅度の両方の尺度が高い (1 に近い) 値をとらない限り高い値とならない．このことは，いかにバグの捕捉率の高いテストを行っても，テストがプログラムの一部の機能のみに対して行われていれば，プログラム中に潜在するバグは十分に除去されておらず，ソフトウェアの品質も十分でないという現実に良く合致する．また同様に，いかに網羅的なテストを行ったとしても，バグを見逃していればソフトウェアの品質は十分使用に耐えるものとはなりえない．その意味で，平均値関数 $m(t)$ をもつ非定常ポアソン過程のモデルでは，テストの精度が十分に高くない場合，発見可能な欠陥数とプログラム内に潜在する欠陥数との差が大きくなるので，現実と合致しない場合がある．

3. テストの精度

テスト完了時，すなわちテスト開始後 t_m 時間経過後，におけるテストの精度 $\alpha(t_m)$ は次式で定義される：

$$\alpha(t_m) = \frac{\bar{N}_i}{N_i} = \frac{m_i(t_\infty)}{N_i}, \quad (3)$$

ただし，N_i はプログラムに潜在する欠陥の総数，$\bar{N}_i = m_i(t_\infty)$ は潜在する欠陥のうちテストで発見可能なものの総数とする．ところで，現実にプログラム中に潜在する欠陥の総数を十分な精度で推定することは困難である．そこで，本小論では人工的に作った欠陥 (制御欠陥と呼ぶ) をテスト開始直前にプログラムへ移植し，その制御欠陥の捕捉率を基礎に，$\alpha(t_m)$ を推定する方法を採用する．すなわち，N_c をプログラムに移植した制御欠陥の総数，$\bar{N}_c$ をテストで発見可能な制御欠陥の総数とするとき，$\alpha(t_m)$ は：

$$\alpha(t_m) = \frac{\bar{N}_i}{N_i} \approx \frac{\bar{N}_c}{N_c}, \quad (4)$$

で推定される．(4) 式において N_c は既知であるので，非定常ポアソン過程モデルの平均値関数 $m_c(t)$ のパラメータが推定できれば，$\bar{N}_c = m_c(t_\infty)$，なる関係から $\alpha(t_m)$ が推定できる．ここで時刻 t_i までに発見された制御欠陥の総数を y_i とすれば，テスト期間中，$(t_0, 0)$，(t_1, y_1)，…，(t_m, y_m) なる一連のデータが得られるので，最尤法により $m_c(t)$ の平均値関数のパラメータを推定できる．例として (1) 式の平均値関数を考えれば，(t_i, y_i) $(i = 0, 1, \cdots, m)$ なるデータに対して，Goel と Okumoto らの方法により尤度関数を最大にする $m_c(t)$ のパラメータを推定できる．すなわち：

$$m_c(t) = \bar{N}_c(1 - e^{-b_c t}), \quad (5)$$

なる $\bar{N}_c$ および b_c を推定できる．ところで，テストが十分に行われていれば，推定値 $\bar{N}_c$ は，$N_c \geq \bar{N}_c$ を満足するので，テストの精度 $\alpha(t_m)$ は：

$$0 \leq \alpha(t_m) \leq 1, \quad (6)$$

を満足する．

4. テストの網ら度

テスト完了時，すなわちテスト開始後t_m時間経過後，におけるテストの網ら度$\gamma(t_m)$は次式で定義される：

$$\gamma(t_m) = \frac{m_i(t_m)}{\bar{N}_i} = \frac{m_i(t_m)}{m_i(t_\infty)}, \qquad (7)$$

ここで，$\bar{N}_i$はテストで発見可能な真の欠陥の総数，そして$m_i(t_m)$は時刻t_mまでに発見された真の欠陥の総数とする．(7)式において$m_i(t_m)$はテスト期間中に発見された真の欠陥の数であり，また$\bar{N}_i$は非定常ポアソン過程モデルの平均値関数$m_i(t)$によって与えられる．すなわち，テスト中時刻t_iまでに発見された真の欠陥の総数をz_iとすれば，テスト期間中，$(t_0, 0), (t_1, z_1), \cdots, (t_m, z_m)$，なる一連のデータが得られるから，最尤法により$m_i(t)$なる平均値関数のパラメータを推定することができる．例として(1)式の平均値関数を考えれば，(t_i, z_i) $(i = 0, 1, \cdots, n)$なるデータに対して，尤度関数を最大にする$m_i(t)$のパラメータ：

$$m_i(t) = \bar{N}_i(1 - e^{-b_i t}), \qquad (8)$$

なる$\bar{N}_i$およびb_iを推定できる．

ところで，一般に中小規模ソフトウェアのテストにおける欠陥発見過程を追跡すると，図2の(A)または(B)の2種類の曲線のどちらかに分類できる[5]：

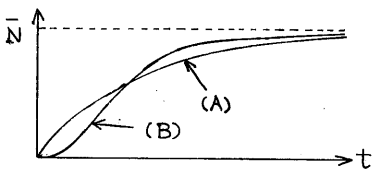

図2．欠陥発見過程の追跡曲線

このことから，われわれは現実のテスト結果でソフトウェアの品質評価やテストの妥当性評価を行う場合，(1)式の平均値関数だけでは不十分なことを知ることができる．現実のテスト結果が図2(B)のような曲線となる原因として，以下の2点が考えられる：

① テスト開始直後はテスト担当者がプログラムやテスト環境（テストシステム，テスト・ツール等）に習熟していないため，有効なテストを実施していない空白時間（遅れ）が生じる，

② テスト担当者が欠陥らしき現象を発見してから欠陥と確認するまでに再試行等現象を再現させるための努力をするため，テスト全体にわたり時間遅れを生じる．

上記①の現象は，(1)式のモデルを以下のように変形することで記述される：

$$m_i(t) = \bar{N}_i(1 - e^{-b_i(t-d)}). \qquad (9)$$

また，上記②の現象は次の平均値関数モデルによって記述される[6]：

$$m_i(t) = \bar{N}_i(1 - (1 + b_i t)e^{-b_i t}). \qquad (10)$$

(9)式のモデルは，(1)式の平均値関数を時間軸上でシフトさせたもので，基本的に(1)式と同じ性質をもち，自動制御で言うtime-lagをもったモデルである．(10)式のモデルは，(1)式のモデルが自動制御で言う1次遅れの系であるのに対し，2次遅れの系の性質をもっている．著者らの実験結果では，3,000 LOC (Line of Code)から20,000 LOC程度の規模のモジュールの欠陥発見過程を分析する場合には，(9)式または(10)式のモデルのどちらかが良く適合した．また，10,000 LOC以上の規模のプログラム全体の欠陥発見過程を1つの平均値関数で表現することは誤差のため実用上問題があった．モデルの適合性については，一般的に機能の単純なモジュール，小規模なモジュールに対しては(9)式のtime-lagをもつモデルが良く適合するのに対し，機能の複雑な制御プログラム等のモジュールなどでは(10)式のDelayをもつモデルが良く適合した．また，80,000 LOC以上のプログラムでは，(1)式のモデルが巨視的には適合していた．

5. 実験例 I

実際に制御欠陥を3,000 LOC程度の小さなプログラムへ移植し，オフライン環境での機能試験を完了した直後SSPQLを評価した例について述べる．

実験対象のプログラムは構造化プログラミング用マクロ命令を用いてアセンブラ言語で書かれた，オンライン端末装置を制御するための入出力制御用プログラムである．実験データは通信回線や端末装置を接続しないオフラインの環境で，通信回線および端末装置の状態や応答をシミュレートするデバッグ・ツールを用いて，プログラムの機能が正しく動作することを検査する目的の試験において得られたものである．

本実験において，制御欠陥はプログラムの基本設計を担当した評価者の立場の人間と，実際にプログラムの詳細設計，コード化，および機能試験を担当した被験者の立場の人間とにより，ほぼ半数づつ移植された．移植された欠陥は次の3種類に分類される：

①欠陥を通過した場合常に異常終了となるもの，

②欠陥を通過しても通常は機能異常を生じないが，特定の条件またはその後の処理で機能異常の原因となるもの，

③上記①または②に該当し，事前のテストまたは精査で欠陥と判断されたバグを残留させたもの．[7]

上記①の欠陥は，発見の比較的容易なバグである．例としては，GETMAINマクロで獲得したバッファをFREEMAINマクロで開放する際に，開放する記憶域の長さ指定を獲得時の指定と矛盾させておくというような常時異常終了となるバグがある．上記②のバグは，発見の困難度の高い制御欠陥である．例として，制御ブロックの参照・更新時に参照・更新フィールドの相対番地をズラせておくなどのバグがある．制御ブロックのポインタを更新するルーチンにこの種の制御欠陥を移植した場合，この欠陥の通過時には何の機能異常も発生しないが，その後の処理で機能異常を発生する．上記③の欠陥は，プログラムに本来潜在していたもので，そのプログラムに潜在しているバグがどの程度除去されたかを知る目安となる．例としては，制御ブロック内に定義されているフラグ（1ビットのスイッチ）をリセットするために排他的論理和を実行する命令を使用していたものが事前のコード精査で確認され，それを残留させたものなどがある．この例の場合，この命令実行時に当該フラグのオン状態が保証される場合は問題を生じないが，当該フラグがオフ状態である場合には，逆にオン状態に反転しその後の誤動作の原因となる．以上のことに加えて，制御欠陥の移植に際しては，コード精査で誤りの多く発見されたモジュールや，論理の複雑なモジュールなどには多くの制御欠陥を移植するよう配慮した．

図3に本実験例での真の欠陥および制御欠陥の発見過程を示す．実験期間は10労働日（約200人時）であった．

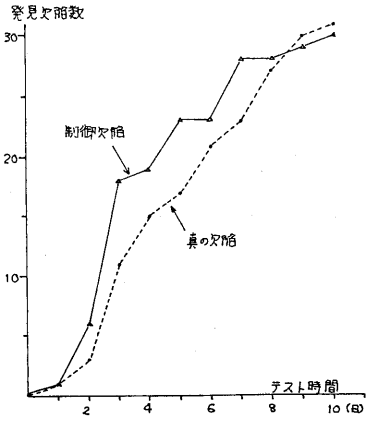

図3. 実験結果

図3から明らかなように，本実験例の場合，(1)式の平均値函数モデルは適合していない．本例の真の欠陥の場合について(9)式および(10)式のモデルを仮定した平均値函数のパラメータの推定結果を図4および図5に示す．ちなみに，実測値とモデルによる推定値との2乗誤差E^2：

$$E^2 = \sum_{i=1}^{m} [Z_i - m_i(t_i)]^2, \qquad (11)$$

の比較では，(9)式の(Tim-lag)モデルが(10)式のモデルに勝っていた．しかし，推定されたパラメータ$\bar{N}_i$を比較すると，(9)式のモデルの場合が52，(10)式のモデルの場合が38となっており，大きな差があった．ここで，制御欠陥の網ら度が90％に達していたため，真の欠陥の網ら度が60％以下

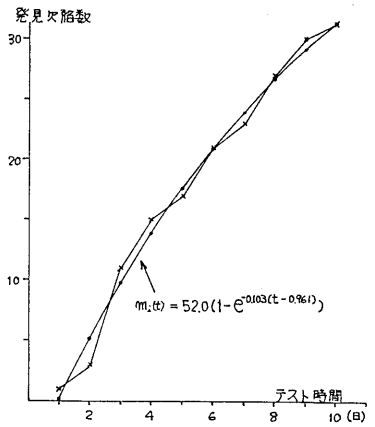

図4.

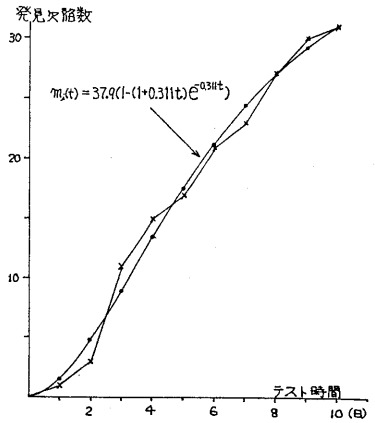

図5.

であるとは考えにくいので，本例については(10)式の(Delay)モデルが適合していると判断した．

次に，制御欠陥のデータに関する解析では，(9)式および(10)式のモデルで試算した結果，2乗誤差E^2およびパラメータ$\bar{N}_c$の推定値の両方ともほとんど差がなかった．図6に(10)式の(Delay)モデルによる推定結果を示す．

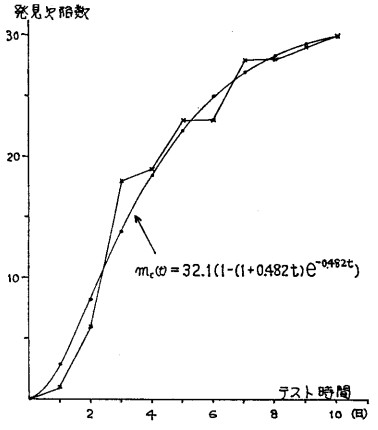

図6. 制御欠陥

以上のデータ解析結果より，本例におけるテストの網ら度γは：

$$\gamma(t_{10}) = \frac{31}{m_i(t_\infty)}$$

$$= \frac{31}{37.9(1-(1+0.32t_\infty)e^{-0.312t_\infty}}$$

$$= 0.818, \qquad (12)$$

となる．また，テストの精度αは：

$$\alpha(t_{10}) = \frac{m_c(t_\infty)}{37}$$

$$= \frac{31.3(1-(1+0.498t_\infty)e^{-0.498t_\infty})}{37}$$

$$= 0.846, \qquad (13)$$

となる．従って，SSPQLは：

$$SSPQL = 0.818 \times 0.846$$

$$= 0.692, \qquad (14)$$

となる．

本例においてテスト完了時における

SSPQLが約0.7と低いのは，テスト担当者が制御プログラムのテストに未経験であったためであり，現象として発生している欠陥を異常として認識できていない点や，テスト・ケースが正常処理の流れのテストに偏っている点から考えると，妥当な値であろう．ちなみに，本実験の後，現実のオンライン環境におけるシステム統合テスト（5労働日，延30人時）において同一テスト担当者により，5種類の新しい真の欠陥が発見された．また，それに続く約3週間の実使用期間中に確認された真の欠陥が2種類（機能上影響の小さなもの），他に確認されていない異常現象が数種類報告された．以上のことから，テストの網羅度，精度ともほぼ妥当値であったと言えよう．

6．実験例Ⅱ

本実験例は，全体で約10,000 LOCの3つのモジュールから構成されるプログラムについてSSPQLを実験的に適用した例の一部分である．ここでは，約4,000 LOCの入出力実行制御モジュールの評価例を紹介する．プログラムはアセンブラ言語で書かれており，簡単な構造のIF文，CASE文，GOTO文，BEGIN文，END文，およびDEBUG文等のマクロ命令が準備されている以外は全てアセンブラ命令でコード化されている．

本例の入出力実行制御モジュールは，プログラムの基本設計を担当した当該アセンブラ言語でのプログラミング経験者が，詳細設計，コード化，およびデバッグを担当した．制御欠陥の移植は，他の関連モジュールの詳細設計，コード化の担当者が，事前の詳細設計精査やコード精査のデータを基に行った．特に本例の制御欠陥は，品質の要求水準が高かったため，発見の困難度の高いバグを多く移植した．

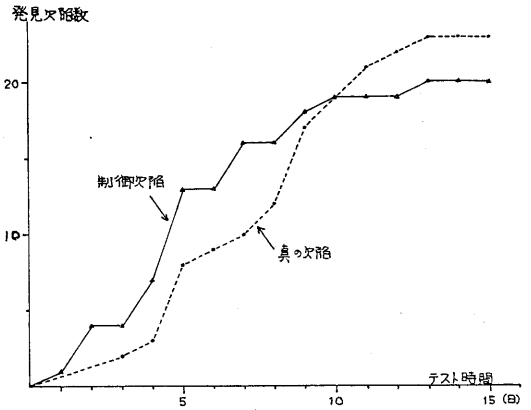

図7．実験結果

図7に本例での真の欠陥および制御欠陥の発見過程を示す．テスト期間は14日間，発見した真の欠陥の総数は23，発見した制御欠陥の総数は20であった．図7から明らかなように，本例の場合(1)式の平均値函数モデルは適合していない．真の欠陥について(9)式および(10)式のモデルを仮定し，それぞれの平均値函数のパラメータを推定した結果，実測値とモデルによる推定値との2乗誤差の比較，およびパラメータN_0の推定値の妥当性の比較とも(10)式の(Delay)モデルがよくデータに適合していた．図8に(10)式のモデルによる推定曲線と実測値を示す．

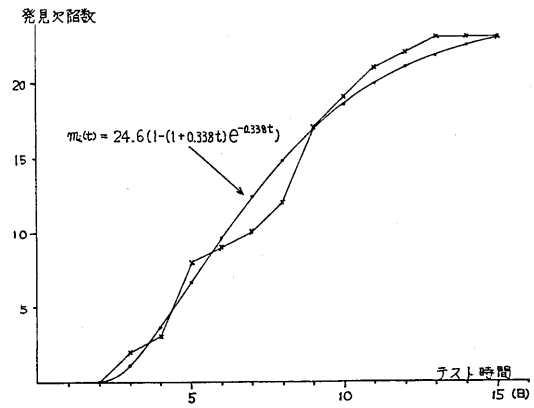

図8．真の欠陥

次に，制御欠陥のデータについては全部で26個の制御欠陥が移植されたが，テスト中に確認されたものは全部で20個であった．本例のテストはモジュールの特定機能を検定する目的で行われモジュールの他の機能には注目していないため，テスト対象外の下位モジュールがあった．テスト対象となった下位モジュールに移植された制御欠陥の数は，26個中21個であった．従って，テスト対象であった下位モジュール中に未発見の制御欠陥が1個残存している．制御欠陥の発見データについて，(9)式および(10)式のモデルを仮定し，平均値函数のパラメータを推定した結果，実測値とモデルによる推定値との2乗誤差の比較では(10)式のモデルが約1.5倍勝っていたが，推定したパラメータ $\bar{N}_c$ の値に差はなかった．図9に(10)式の(Delay)モデルによる推定値と実測値との関係を示す．

以上のデータ解析の結果よりテストの網ら度 γ は：

$$\gamma(t_{15}) = \frac{23}{m_s(t_\infty)} = \frac{23}{24.6(1-(1+0.338t_\infty)e^{-0.338t_\infty})} = 0.935, \quad (15)$$

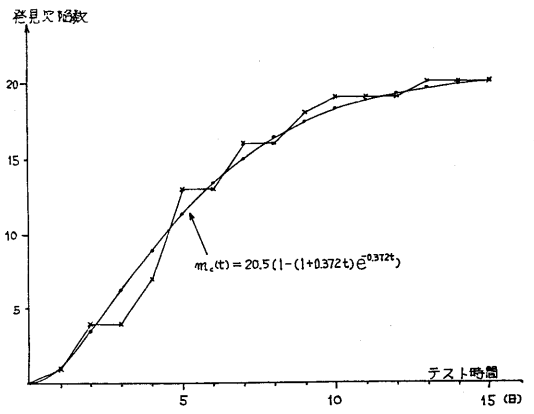

図9．制御欠陥

となる．また，テストの精度 α は，テスト対象となった下位モジュールに移植された制御欠陥が21個であったことから：

$$\alpha(t_{15}) = \frac{m_c(t_\infty)}{21} = \frac{20.5(1-(1+0.372t_\infty)e^{0.372t_\infty})}{21} = 0.976, \quad (16)$$

となる．従って，本例の場合SSPQLは：

$$SSPQL = 0.935 \times 0.976 = 0.913, \quad (17)$$

となる．

本例の場合，SSPQLが約0.9とかなり高い値であった原因として，プログラムの設計・コード化および本例のテスト担当者が，テスト・ツールおよびプログラミング言語の経験がある点が指摘できる．ちなみに，プログラミング言語とテスト・ツールに未経験のプログラマが担当したモジュールについては，SSPQLは約0.6であった．本例のテスト完了後，新たに発見されたテスト対象下位モジュールの真の欠陥は，1個のみであった．

7．まとめ

以上，ソフトウェアの品質評価尺度およびテストの妥当性評価尺度としてのSSPQLについて述べた．従来の信頼度成長曲線によるソフトウェアの品質評価は前述のように，テストの質に評価が依存するという問題点があった．また，Capture-Recapture法による評価では，制御欠陥の移植の方法（分布のさせ方）によって評価結果が左右されるという難点があった．SSPQLでは，これらの問題を改善するために評価をテストの質と量の独立した2つの評価に分け，テストの量的評価を従来の信頼度成長曲線のモデルによって行い，よ

たテストの質的評価をCapture-Recapture法によって行うこととした．このことにより，SSPQLでの評価値は信頼度成長曲線のモデルに比較してテストの担当者の能力やテスト・ツールの能力等の影響を小さくするとともに，制御欠陥の移植法の影響をも少さくした．

SSPQLの推定には信頼度成長曲線モデルの現実のデータへの適合性が重要となる．このため，本小論では信頼度成長曲線モデルとして従来から知られている(1)式のモデルとそれを拡張した(9)式の(Time-lag)モデルに加えて，新たに(10)式の(Delay)モデルを採用した．実験例からも明らかように，小規模プログラムやそのモジュールに対しては(10)式の(Delay)モデルが一般的に適合性が良く，実用的であった．すなわち(1)式や(9)式のモデルでは，発見可能な欠陥数を示すパラメータNの値が非現実的な(大きな)値となる場合が多く，特に絶対数の未知である真の欠陥については有効でない場合が多い．ただし，制御欠陥についてはその分布が真の欠陥に比較して各モジュールへ一様に分散しているため，時間軸上のズレを考慮した(9)式の(Time-lag)モデルも(10)式の(Delay)モデルとほぼ同程度に適合していた．

SSPQLの評価値の妥当性については，さらに中規模ソフトウェアでの実験が必要であろう．ただ現在までの実験結果から，定性的にはSSPQLの評価値の高いプログラムのテスト後の信頼度が高いことが確認されている．従来の信頼度成長曲線モデルのみの評価に比較して，信頼度の高いプログラムの品質評価の確度が向上している．しかしながら，逆に信頼度の低いプログラムの品質評価が，従来の評価法に比較して厳しくなる傾向がある．すなわち，SSPQLの評価値が0.6から0.7程度のプログラムでも基本的な機能については不安定ながら使用可能である場合がある．これらは，SSPQLの評価が網ら度と精度とのカケ算で定義され，現実には精度を評価する時に網ら度の影響があることなどが原因と考えられる．今後，この点についての検討が重要であろう．

謝辞

本研究を行うに際し，ソフトウェアの欠陥に関するデータ収集について御助力を頂きました日本アイ・ビー・エム(株)製品保証担当佐藤武義氏ほか，同竹屋藤岡義彦氏，および同武田健二氏に感謝します．

[参考文献]

1) Goel A., Okumoto K.: "Time-dependent error-detection rate model for software reliability & other performance measures," IEEE Trans. Reliability, Vol.R-28, 1979.

2) Duran J, et al.: "Capture-Recapture Sampling for Estimating Software Error Content," IEEE Trans. Software Eng., Vol. SE-7, 1981.

3) 伊土誠一他："Capture & Recapture法による潜在バグの推定法とその応用"，情処学会 ソフトウェア工学研究会資料 19，1981年7月．

4) 篠田康吾他："制御欠陥と機能試験の妥当性評価"，情処学会 22回全国大会 5C-7，1981年3月．

5) 酒巻恒一他："ソフトウェアの信頼性：品質管理のためのソフトウェアの信頼性予測，"信学会 信学技報，(信頼性) Vol. 81，1981年5月．

6) 山田茂他："ソフトウェアエラー発見過程の確率モデルとその応用"，情処学会 ソフトウェア工学研究会資料，1981年11月．

7) 坂本恵市他："バグ残留法によるプログラムの品質測定，"情処学会 21回全国大会 7C-8，1980年5月．