ソフトウェア工学 51－5
（１９８６．１１．２６）

潜在エラー数に影響を与える開発環境要因の評価について

小室　豊
東芝エンジニアリング株式会社　技術本部

プログラムの信頼性を評価する一手段として、テスト過程でプログラムの中に残存するエラー数を推定する方法がいろいろ提案され、実用化されている。これに対し、エラー数はプログラム規模（サイズ）のg(Πm)乗に比例して作り込まれるという推定モデル式で、テスト開始前にエラー数を推定することも可能と考えられる。本報告書ではこのモデル式に開発環境の諸要因が大きく影響するものとして、この定量的な評価を試みた結果について述べている。さらに、この報告書ではこれら諸要因を信頼性ドライバーと名付けて、定量的に評価した値で推定モデル式を補正することも試みている。また評価尺度が理想的に定量化できれば、エラー数はプログラム規模に対して直線的に比例するのではないかと述べている。

"An Evaluation of Environmental Factors Affecting Errors in Programing"
(in Japan)

by Yutaka KOMURO
(Corporate Technology Division,Toshiba Engineering Co.LTD.
2-12-19 Nishi-Gotanda Shinagawa-ku Tokyo 141 Japan)

There are several evaluation methods proposed and practiced for estimating the number of residual errors in programing.It is possible to estimate the number of errors prior to test runs using a parameter estimation model incorporating the relationship of the exponential of g(Πm) of the program size.

It is assumed in this paper that the emvironmental factors would affect the model equation directly and develop a quantitive relationship to various factors named "reriability drivers". It is concluded that there is a linear relation ship between the number of error and the size of programs provided that the evaluation metrix is ideally quantified.

1．はじめに

ソフトウェアシステムの大規模化、複雑化は止まるところなく、進展し続けている。それにともないソフトウェアの品質改善や生産性向上へのニーズは強まる一方である。ソフトウェア工学分野では、このニーズに対応するため研究が続けられているが、恐らく技術的研究は永続的に続けられていくものと思われる。

当社においてもプログラムのテスト工程の進捗度を信頼性の面から定量的に評価し、管理するシステムとしてＳＳＰＱＣ(S-gibu Software Product Quality Control)と称するツールを開発し、実用に供してきた。このツールは大きく 2つのステップに分れており、まず対象プロジェクトの潜在エラー数の目安を推定し、それを基に指数関数によるエラー発生成長モデルによってテストの進捗状況をチェックし、出荷時期を判定しょうとするものである。

このモデルにおける潜在エラー数推定方式はその職場で過去に開発したプログラムから凡その潜在エラー数を推定するもので、プログラム規模の平方根にエラー数は比例するという大胆な仮定にたったものであった。この仮定が一般の開発現場に適用できるかという評価を行ったところ、このモデルを開発した当社の職場ではかなりよく適合するが、一般的には相当無理があることが判明した。その根拠はプログラムの再利用の度合によるものと推定され、この面から分析を進めているが、この問題については別の機会に報告することとする。ここではＳＳＰＱＣモデルで採用している潜在エラー数推定式からさらに一般化したモデル式を導き出し、いろいろな職場の潜在エラー数を算出する実験を行った。この実験の過程でその職場の開発環境や担当技術者のスキルの影響がかなり響いたため、この影響度を定量的に計測する方法の調査研究を試みた。その考え方と評価実験の結果を本稿で報告する。なおこの研究は当社のソフトウェア開発支援ツールＳＳＰＱＣをベースにしたものであるが、さらに協同システム開発㈱（ＪＳＤ）の「ソフトウェア信頼性モデルの実践的評価」の調査研究の成果の一部でもある。

2．潜在エラー数推定の一方法の考察

筆者達が調査した当社の開発支援ツールＳＳＰＱＣの運用結果、およびＪＳＤにおける「ソフトウェア信頼性モデルの実践的評価」における調査研究の結果、プログラム規模（サイズ）と潜在エラー数の間にはある一定の関係があることが観測された。この関係を調査分析してみるとプログラムの開発過程で作り込まれる潜在エラー数はプログラム規模との間にある数学的な関係式で表すことが考えられる。

単純に考えると(1)式のように作り込まれるエラー数はプログラム規模に対して直線的に比例して増加していくべきものと考えることができる。

$$Y = A \cdot X \qquad (1)$$

ここで　Ｘ：プログラムサイズ
　　Ｙ：作り込まれたエラー数
　　Ａ：比例定数

しかし現実には、その関係は複雑で非直線的な関係になることが一般に知られている。この原因は開発対象プログラムの規模や複雑さ、さらに担当技術者のスキルなどの諸要因が複雑に絡み合ってくるためと思われる。本稿ではこれらの要因をＣＯＣＯＭＯのコストドライバーに対比させ信頼性ドライバーと称し、その影響度をいろいろな方面から定量的に評価することを試みたので、その考え方と実験結果を報告する。なお信頼性ドライバーは本来プログラム規模と潜在エラー数の関係に影響する全ての要因であるが、ここでは、現時点で定量的に影響度を評価できる要因のみを指すものとする。言い換えると評価値で補正可能なものを信頼性ドライバーとして取り上げているので、前に述べた「影響を与える全ての要因」ではなく、限定された狭義なドライバーとみるべきである。例えば使用言語やプログラムの適用分野などによる影響は現時点では定量的に評価不可能（何らかの方法で定量的な評価できるかも知れないが、ここでは不可能とした）であるとみて、層別分析の方法を採用している。したがって、補正分析が可能なドライバーを信頼性ドライバーとしている。

3．潜在エラー数推定モデルの考え方

3.1 推定式の考え方

プログラムの中に作り込まれたエラー数の推定方法は各種エラー発生成長モデルや再捕獲法などいろいろなものが提唱されているが、これらはテスト終了時点に残存エラー数を推定したり、推定するために特別の作業を必要とするもので、かなり重い方法論といえる。しかし、いずれの推定法ともそれを適用する目的や場所によって有用なものといえるが、テスト開始前に潜

在エラー数を推定することは困難である。

そこで筆者は精度を要求することは困難であるが、テスト開始前に凡その潜在エラー数を推定する方法として、プログラム規模と作り込まれるエラー数の間にはある一定の関係があるという仮定をたて(2)式のようなモデル式を考え出した。この式は当社の開発支援ツールSSPQCの考えを汎用化したもので、これをJSDの調査研究の中で評価した結果である。

$$Y = f(\Pi m) \cdot X^{g(\Pi m)} \qquad (2)$$

ここで　X　；プログラム規模（サイズ）

Y　；潜在エラー数

f(Πm)；諸ドライバーによって影響を受ける比例係数

g(Πm)；諸ドライバーによって影響を受ける指数

この推定式は各々のソフトウェア開発職場における過去の実績データから係数f(Πm)、指数g(Πm)を算出し、これを基に対象プログラムの規模より潜在エラー数を推定しようとするものである。さらに、この式は潜在エラー数が開発されたプログラム規模の n乗に比例するという一般にいわれている考えと一致する。nはその職場での開発環境による諸要因（ここでは広い意味の信頼性ドライバー）によって変化する指数g(Πm)と同じものである。言い換えると(2)式は一般に考えられている概念を数式化したものといえる。また、既に述べたように当社のSSPQCにおけるエラー数推定の「潜在エラー数はプログラム規模の平方根に比例する」という仮定も(2)式に包含しているとみることができる。さらに(2)式によってSSPQCのモデルがこれを開発した職場にのみよく適合する条件を調査・分析も可能であるが、ここではこの問題について言及することを割愛する。

(2)式では開発環境の諸要因によって指数g(Πm)のみが変化するのではなく、係数f(Πm)も同時に変化する。しかしこの両定数を諸要因の影響度から計算によって算出することが困難なため、層別と補正の双方を使い、回帰分析によって(2)式を求める方法をとった。本稿では諸要因のうち定量的な尺度が考えられる信頼性ドライバーに絞って、その定量的評価値の分析を試みてきた。

3.2 潜在エラー数推定式の定数の求め方

既に述べたように(2)式の両定数は回帰分析によって求めるが、諸要因によって直接計算するのではなく、それぞれのプログラム規模においてエラー数を要因毎に層別するか、または補正してから回帰分析を行う間接的な分析方法をとった。前者は同じ条件で後者は諸要因の影響を除去した後で回帰分析する考えである。

(a) 層別分析

この方法による定数の算出はそれぞれの要因の定量的な評価が困難なものに限って分析したものである。何らかの尺度で定量的に評価できるものがあれば、次の補正分析で定数を求める方法をとるようにしている。層別分析の場合層別する種類が少ない場合は有効な方法と考えられるが、層別対象の要因が多い場合は複雑になる。すなわちそれぞれの要因をクロスさせて層別させると、データ量が少なくなったり、分析結果のケースが多くなり複雑になるため評価することが困難になる。特に今回の調査研究では非常に多くの層別対象要因がある。そこでクロス層別が困難なため、それぞれの要因単独で層別して(2)式の定数がどのような値になるかを把握することに留めた。また今回の実験では使用言語別、適用システム分野別、開発職場別、プログラム再利用率別（この要因のみは定量化が可能であるが、再利用の影響度を評価するために層別分析もおこなった）の 4種類の要因に絞って層別を試みた。

(b) 補正分析

この考えは推定式(2)の定数を求める回帰分析のときに信頼性ドライバーの影響を除外して計算しようとするものである。したがって全ての要因の影響力を除去することができればプログラム規模と潜在エラー数との関係は(1)式で表した理想的な関係式になると考えられる。しかし残念ながら(2)式の両定数に影響する全ての要因を洗い出すことは不可能に近い。そこで筆者達がこの調査研究で選んだ信頼性ドライバーとして対象としたものは17個を挙げたに過ぎなかった。

本稿ではこの17個の信頼性ドライバーについて、どのような補正値が最適かという考え方とその補正値による(2)式の定数の算定結果を報告する。

4．採用した信頼性ドライバーとその定量的評価

信頼性ドライバーとして採用した項目はCOCOMO

のコストモデルに使われているコストドライバー15個と仕様の変更度合、プログラム再利用度の17個であった。特に馴染み深いCOCOMOのコストドライバーを採用したことはデータを収集していく過程で、データの評価レベルを合せ易いことと、当面考えれるドライバーとしてほとんど網羅されていると考えれたためである。さらに定量的尺度で評価できるものとしてコストドライバー以外のものについてはさしあたり 2種類のドライバーを採用した。

4.1 採用した信頼性ドライバー

潜在エラー数推定式(2)を評価するため次の17個の諸要因を信頼性ドライバーとして採用した。

〈プロダクト要因〉

1) ソフトウェアの信頼性要求 (RELY)
2) データベースの規模 (DATA)
3) ソフトウェアの複雑さ (CPLX)

〈コンピュータ要因〉

4) 実行時間の制約 (TIME)
5) メモリの制約 (STOR)
6) 開発環境の安定性 (VIRT)
7) 応答時間 (TURN)

〈人間的要因〉

8) 分析チームの能力 (ACAP)
9) 対象アプリケーション・システムの経験(AEXP)
10) プログラマの能力 (PCAP)
11) 開発環境での作業経験 (VEXP)
12) プログラミング言語の経験 (LEXP)

〈プロジェクト要因〉

13) 近代的プログラミング手法の使用と経験(MODP)
14) ソフトウェアツールの使用レベル (TOOL)
15) スケジュールの制約 (SCED)
16) 仕様変更の度合 (SPCD)
17) プログラムの再利用の度合 (REUS)

ここで1)～ 15)までの15個はCOCOMOのコストドライバーと同じ項目のドライバーを採用し、かつ、データ収集のときはCOCOMOの評価基準をそのまま活用した。その理由はデータ収集のとき評価レベルを合せれることと、評価方法は同じでも設定する補正値で調整することができるためである。 16)と 17)は筆者達が追加したもので定量的な尺度で評価でき、かつ相当強く影響するドライバーと推定されたので採用した。ただ、今回の調査では 17)のREUSは別の実験で再利用の影響度を調査したので、この実験から分離ししている。そのため補正値の設定調査から除外されている。

4.2 信頼性ドライバーによる補正の考え方

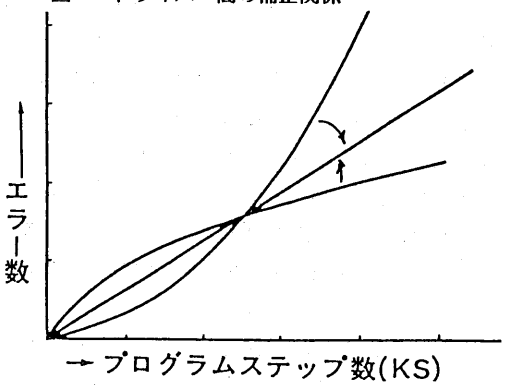
(2)式の両定数を回帰分析で求める前に信頼性ドライバーでエラー数を補正し、可能な限りバラツキの少ないデータに補正し、相関係数γの高い回帰曲線を求める必要がある。しかし相互のドライバー間の補正量の重み付けをする基準がないので、いろいろな補正値で補正することによって、潜在エラー数推定式(2)の両定数の変化と求め、同式の相関係数γの大きさで試行錯誤的に補正量を求める方法をとった。

また補正することによって推定式(2)に表れる影響を検討してみると次の二通りの側面が考えられる。

(a) 信頼性ドライバー間の補正
(b) 信頼性ドライバー内の補正

この二通りの補正は全く別の効果を持たらしている。すなわち前者は各ドライバーの影響を排除するもので、それぞれのドライバーの各評価点の補正値にバイアスを加え、理想的な推定式(1)に近ずけようとする補正である。この関係を図に示すと図-1のようになる。

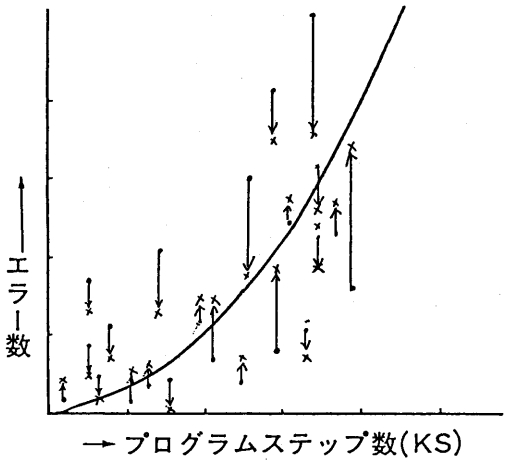
図-1 ドライバー間の補正関係



一方、後者はそれぞれドライバー内のバラツキを補正し、そのドライバー内の標準的環境でプログラムを開発した場合に推定式(2)を求めたものである。例えば「プログラマの能力(PCAP)」において、スキルの低いプログラマは当然エラー数の多く作り込んだプログラムを作成する筈であるから、標準的スキルのプログラマが作り込むであろうエラー数まで少なくするように補正する。逆に高度なスキルを持つプログラマはエ

ラー数の少ないプログラムを作成する筈であるから、標準的プログラマが作り込むエラー数まで増加させる方向に補正し、個々の信頼性ドライバー内の環境のレベルを平準化することを意味している。この場合補正効果を判定する尺度は相関係数γが大きくすることであり、指数g(Πm)が 1に近ずくとは限らない。図で表わすと図-2のようになり、それぞれのプログラム規模のエラー数が矢印の方向に補正され、全体として求める(2)式の相関係数を大きくする働きをするものである。

図-2　ドライバー内の補正効果の考え



4.3 信頼性ドライバーの補正値の求め方

前項で述べたように信頼性ドライバーによる補正は二通りあり、それぞれの補正値も補正後の結果も異なるものである。今回の調査・研究における実験では、補正値の相互間の基準が判然としない前者の補正値算出は保留し、後者のドライバー内の補正量の算出に的を絞って実験を試みた。したがって前者の補正値の算出については今後の研究課題として、その基準を求め、その基準値に従って補正値を算出することになるものと思われる。

後者の補正値の算出には次のような手順で行った。まず各ソフトウェア開発職場で、開発中の個々のプロジェクトに対し、各信頼性ドライバーがどのようなレベルにあるかをデータ収集者に評価してもらった。評価方法はそれぞれのトライバーの平均的中間値を標準点（これをN とする）とし、これに対して「非常に低い」と思われるものを“VL”、「低い」と思われるものを“ L”、同等と思われるものを“ N”、以下高い(H)、非常に高い“VH”、特別に高い“EH”と 4～ 6段階で評価してもらった。全体的に直感的で定性的な評価になるため、データ収集者によって評価レベルがバラツク恐れがあった。そこで一般に馴染みのあるCOCOMOのコストドライバーの評価基準を採用し、極力バランスをとらせるようにした。

このようにして集めた各信頼性ドライバーの評価点に対し、第 6節に示す表-1(a) ～(f) のような補正値を対応させた。次に各ドライバー毎にこの補正値でエラー値を補正した後に回帰分析を行い、潜在エラー数推定式(2)を求め、そのときの相関係数γが極大になったときの補正値をそのドライバーの適正補正値とすることにした。

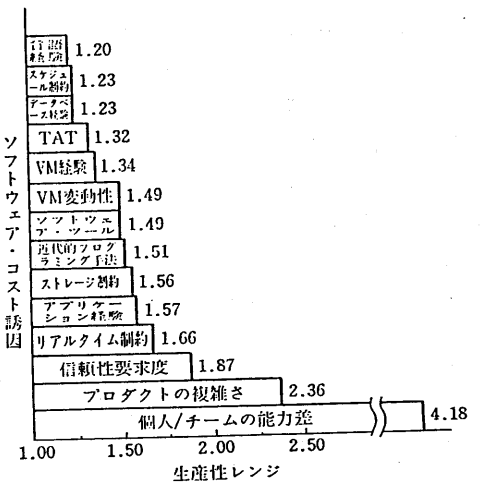
次に信頼性に影響する信頼性ドライバーの大きさを比較するため、生産性に寄与するコストドライバーの大きさを示す図-3のように、補正値の大きいものから順に山積みすることによって、エラーの作り込みに大きな影響を与える（信頼性に影響を与える）信頼性ドライバーの影響度を評価する試みを行った。

この補正値の算出計算では、 17)の「プログラムの再利用の度合(REUS)」についての検討は他のドライバーと異り、理論的に算出可能なためこの項では除外した。このドライバーの補正値算出方法は再利用率から次の式のように求めることができる。

補正値＝１／（１－再利用率）　　(3)

ただし、単に再利用率のみに着目すると上式の算出法

図-3　COCOMOのコストドライバーの生産性効果



でよいが、実際には再利用率が高くなるに従いインタフェースの煩雑さが増加したり、流用プログラムの加工等により新しくエラーを作り込む可能性が出てくるので、(3)式はさらに再補正する必要がある。

5．収集データ

信頼性ドライバーの補正値を算出し、その値を評価するために数多くのデータを必要とした。そこで今回の実験ではＪＳＤの「ソフトウェア信頼性モデルの実践的評価」のプロジェクトに参加した企業 6社より97件のデータを提供してもらった。この中から計測値の欠落や異常計測値のデータを取り除いた63件を実験に使用した。

6．補正値の算出と各ドライバーの影響度の評価

まず各信頼性ドライバーの適正補正値を算出するため、表-1(a) ～(f) に示すようにVL～ L、 L～ N、 N～ H、 H～VH、VH～EH間のそれぞれの値を0.05ステップでメッシュ切りした 6ケースの補正値を仮に設定した。その後個々のドライバー毎に各ケースでエラー数を補正し、推定式(2)を回帰分析によって求めた。その結果求まった回帰曲線の相関係数γの集計表が表-2である。各ドライバーの補正値を変化させていけば、相関係数γはいずれかの補正値で極大になる筈である。なぜならば、補正値を増加させるに従い相関係数γも大きくなっていく場合、ある点でγは飽和するかまたは減少し始めなければならない。この考えが成立しなければγは無限に大きくなっていかなければならなくなる。これはγの最大値が1.00と有限値をもっていることと矛盾することになるので、補正値を変化させていけばγは必ず極大値を持つといえる。この相関係数γが極大値になる補正値をそれぞれのドライバーの適正補正値とみなした。その結果が表-2の中で*印を付けたものである。

また、信頼性ドライバーの補正値の大きさを比較したものが図-4である。これは補正値

表-1(a) 信頼性ドライバー補正値（ケース１）

項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.10	1.05	1.00	0.95	0.90	–
2	DATA	–	1.05	1.00	0.95	0.90	–
3	CPLX	1.10	1.05	1.00	0.95	0.90	0.85
4	TIME	–	–	1.00	0.95	0.90	0.85
5	STOR	–	–	1.00	0.95	0.90	0.85
6	VIRT	–	1.05	1.00	0.95	0.90	–
7	TURN	–	1.05	1.00	0.95	0.90	–
8	ACAP	0.90	0.95	1.00	1.05	1.10	–
9	AEXP	0.90	0.95	1.00	1.05	1.10	–
10	PCAP	0.90	0.95	1.00	1.05	1.10	–
11	VEXP	0.90	0.95	1.00	1.05	1.10	–
12	LEXP	0.90	0.95	1.00	1.05	–	–
13	MODP	0.90	0.95	1.00	1.05	1.10	–
14	TOOL	0.90	0.95	1.00	1.05	1.10	–
15	SCED	0.90	0.95	1.00	0.95	0.90	–
16	SPCD	–	–	1.00	0.95	0.90	0.85

表-1(b) 信頼性ドライバー補正値（ケース２）

項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.20	1.10	1.00	0.90	0.80	–
2	DATA	–	1.10	1.00	0.90	0.80	–
3	CPLX	1.20	1.10	1.00	0.90	0.80	0.70
4	TIME	–	–	1.00	0.90	0.80	0.70
5	STOR	–	–	1.00	0.90	0.80	0.70
6	VIRT	–	1.10	1.00	0.90	0.80	0.70
7	TURN	–	1.10	1.00	0.90	0.80	–
8	ACAP	0.80	0.90	1.00	1.10	1.20	–
9	AEXP	0.80	0.90	1.00	1.10	1.20	–
10	PCAP	0.80	0.90	1.00	1.10	1.20	–
11	VEXP	0.80	0.90	1.00	1.10	–	–
12	LEXP	0.80	0.90	1.00	1.10	–	–
13	MODP	0.80	0.90	1.00	1.10	1.20	–
14	TOOL	0.80	0.90	1.00	1.10	1.20	–
15	SCED	0.80	0.90	1.00	0.90	0.80	–
16	SPCD	–	–	1.00	0.90	0.80	0.70

表-1(c) 信頼性ドライバー補正値（ケース３）

項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.30	1.15	1.00	0.85	0.70	–
2	DATA	–	1.15	1.00	0.85	0.70	–
3	CPLX	1.30	1.15	1.00	0.85	0.70	0.55
4	TIME	–	–	1.00	0.85	0.70	0.55
5	STOR	–	–	1.00	0.85	0.70	0.55
6	VIRT	–	1.15	1.00	0.85	0.70	–
7	TURN	–	1.15	1.00	0.85	0.70	–
8	ACAP	0.70	0.85	1.00	1.15	1.30	–
9	AEXP	0.70	0.85	1.00	1.15	1.30	–
10	PCAP	0.70	0.85	1.00	1.15	1.30	–
11	VEXP	0.70	0.85	1.00	1.15	–	–
12	LEXP	0.70	0.85	1.00	1.15	–	–
13	MODP	0.70	0.85	1.00	1.15	1.30	–
14	TOOL	0.70	0.85	1.00	1.15	1.30	–
15	SCED	0.70	0.85	1.00	0.85	0.70	–
16	SPCD	–	–	1.00	0.85	0.70	0.55

表-1(d) 信頼性ドライバー補正値（ケース４）

項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.40	1.20	1.00	0.80	0.60	–
2	DATA	–	1.20	1.00	0.80	0.60	–
3	CPLX	1.40	1.20	1.00	0.80	0.60	0.40
4	TIME	–	–	1.00	0.80	0.60	0.40
5	STOR	–	–	1.00	0.80	0.60	0.40
6	VIRT	–	1.20	1.00	0.80	0.60	–
7	TURN	–	1.20	1.00	0.80	0.60	–
8	ACAP	0.60	0.80	1.00	1.20	1.40	–
9	AEXP	0.60	0.80	1.00	1.20	1.40	–
10	PCAP	0.60	0.80	1.00	1.20	1.40	–
11	VEXP	0.60	0.80	1.00	1.20	1.40	–
12	LEXP	0.60	0.80	1.00	1.20	1.40	–
13	MODP	0.60	0.80	1.00	1.20	1.40	–
14	TOOL	0.60	0.80	1.00	1.20	1.40	–
15	SCED	0.60	0.80	1.00	0.80	0.60	–
16	SPCD	–	–	1.00	0.80	0.60	0.40

の大きなものから順に積みあげたもので、生産性を示す図-3に相当するものである。すなわちプログラム作成過程でエラー数を多く作り込むドライバーの重み付けを表したものである。この考えが正しければ信頼性に大きな影響を与えるものとして

「プログラマの能力(PCAP)」

「ソフトウェアツールの使用レベル(TOOL)」

が挙げられる。ただ時間の制約上実験ケースが 6組と少なかったため、この両ドライバーの補正値はまだ収れんしておらず、さらに大きな補正値になる可能性がある。特にPCAPが大きく影響する傾向があることは注目される。逆にほとんど影響のないものとして

「スケジュールの制約(SCED)」

「近代的プログラミング手法の使用と経験(MODP)」

「データベースの規模(DATA)」

「対象アプリケーション・システムの経験(AEXP)」

「開発環境での作業経験(VEXP)」

「プログラミング言語の経験(LEXP)」

の 6つのドライバーが挙げられた。特に各種経験はこの手法による補正では信頼性にあまり影響がないことを示している。この評価は今回の分析 1回だけの結果であるため結論付けることは危険であるが、面白い傾向がでたといえよう。

今回の実験では、各ドライバー内の補正値の算出に限ったことと、原データの各評価ポイントに対し等間隔に補正値を設定したなど特定の条件で実験したため多くの問題をのこしている。また収集データの精度と量にも多くの問題を残しているので、今後この点についても留意して分析を繰り返す予定である。しかし、補正値の設定方法や信頼性への各ドライバーの影響の度合を評価する糸口を把握できたといえる。すなわち、今後ドライバー間の補正値の算出方法、ドライバー内の補正値の改善方法等を検討する方向付けができたといえる。

表-1(e) 信頼性ドライバー補正値（ケース5）

項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.50	1.25	1.00	0.75	0.50	–
2	DATA	–	1.25	1.00	0.75	0.50	–
3	CPLX	1.50	1.25	1.00	0.75	0.50	0.25
4	TIME	–	–	1.00	0.75	0.50	0.25
5	STOR	–	–	1.00	0.75	0.50	0.25
6	VIRT	–	1.25	1.00	0.75	0.50	–
7	TURN	–	1.25	1.00	0.75	0.50	–
8	ACAP	0.50	0.75	1.00	1.25	1.50	–
9	AEXP	0.50	0.75	1.00	1.25	1.50	–
10	PCAP	0.50	0.75	1.00	1.25	1.50	–
11	VEXP	0.50	0.75	1.00	1.25	1.50	–
12	LEXP	0.50	0.75	1.00	1.25	1.50	–
13	MODP	0.50	0.75	1.00	1.25	1.50	–
14	TOOL	0.50	0.75	1.00	1.25	1.50	–
15	SCED	0.60	0.75	1.00	0.75	0.50	–
16	SPCD	–	–	1.00	0.75	0.50	0.25

表-1(f) 信頼性ドライバー補正値（ケース6）

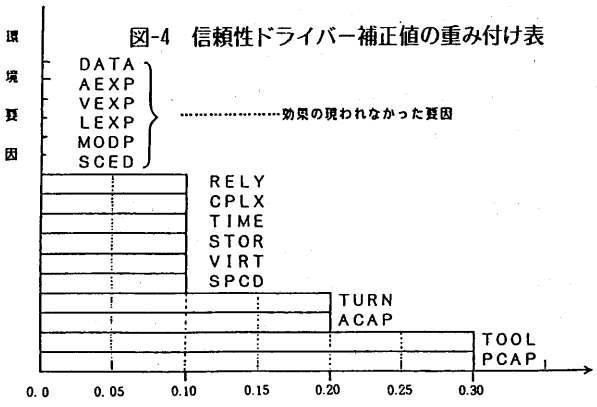
項番	要因名	VL	L	N	H	VH	EH
1	RELY	1.60	1.30	1.00	0.70	0.40	–
2	DATA	–	1.30	1.00	0.70	0.40	–
3	CPLX	1.60	1.30	1.00	0.70	0.40	0.10
4	TIME	–	–	1.00	0.70	0.40	0.10
5	STOR	–	–	1.00	0.70	0.40	0.10
6	VIRT	–	1.30	1.00	0.70	0.40	–
7	TURN	–	1.30	1.00	0.70	0.40	–
8	ACAP	0.40	0.70	1.00	1.30	1.60	–
9	AEXP	0.40	0.70	1.00	1.30	1.60	–
10	PCAP	0.40	0.70	1.00	1.30	1.60	–
11	VEXP	0.40	0.70	1.00	1.30	1.60	–
12	LEXP	0.40	0.70	1.00	1.30	1.60	–
13	MODP	0.40	0.70	1.00	1.30	1.60	–
14	TOOL	0.40	0.70	1.00	1.30	1.60	–
15	SCED	0.40	0.70	1.00	0.70	0.40	–
16	SPCD	–	–	1.00	0.70	0.40	0.10

表-2 信頼性ドライバーの補正後の相関係数表

項番	要因名	補正値(1)	補正値(2)	補正値(3)	補正値(4)	補正値(5)	補正値(6)
1	RELY	0.74008	※ 0.80630	0.80333	0.79936	0.79413	0.78714
2	DATA	0.80972	0.80972	0.80955	0.80956	0.80947	0.80937
3	CPLX	0.71355	※ 0.80702	0.80444	0.80057	0.79415	0.77964
4	TIME	0.61145	※ 0.81213	0.80956	0.80087	0.77743	0.70365
5	STOR	0.66091	※ 0.81458	0.81408	0.80916	0.79446	0.74834
6	VIRT	0.69943	※ 0.80682	0.80472	0.80211	0.79884	0.79462
7	TURN	0.66208	0.81131	0.81168	※ 0.81179	0.81162	0.81109
8	ACAP	0.67831	0.81193	0.81252	※ 0.81278	0.81266	0.81203
9	AEXP	0.81425	0.81425	0.81489	0.81428	0.81222	0.80823
10	PCAP	0.67339	0.81346	0.81478	0.81580	0.81654	※ 0.81699
11	VEXP	0.80947	0.80947	0.80783	0.80493	0.80041	0.79365
12	LEXP	0.81697	0.81697	0.81956	0.82136	0.82217	0.82165
13	MODP	0.81136	0.81136	0.81152	0.81112	0.80908	0.80773
14	TOOL	0.78383	0.82054	0.82514	0.82912	0.83222	※ 0.83401
15	SCED	0.81078	0.81146	0.81217	0.81274	0.81307	0.81295
16	SPCD	0.79285	※ 0.80594	0.80268	0.79716	0.78854	0.77063

※ ：本研究で極大値を与えた相関係数である。

図-4 信頼性ドライバー補正値の重み付け表



7．今後の課題

今回の実験では信頼性に与える開発環境の諸要因がどの程度影響するかという定量的評価の糸口を把握することができたのに過ぎなかった。しかし、ドライバー内、ドライバー間いずれの場合も定量的に評価し、潜在エラー数推定式(2)を補正することが可能であることをつきとめた。その結果定量的尺度の取り方を次のように考えながら再度実験を繰り返すこととした。

(1) ドライバー内の補正値の設定の考え方

今回の実験結果をベースに、再度評価分析をする必要がある。そのため、現在新しいデータを収集し実験をすすめている。現在進めている実験は単に相関係数γの大きさにのみ頼らず、図-2に示すように分散図で補正前と補正後の変化を検討しながら補正値を設定するようにしている。この方法は表-1に示すような各評価点の間の補正値が均等ではなく、それぞれに差があるこも考えられるのでその検証をするためである。いずれにしても繰り返しながら値を模索する方法をとらねばならない。

(2) ドライバー間の補正値設定の考え方

今回の実験ではこの補正方法の基準値が把握できなかったため、実験を割愛した。しかし、実験の結果図-1に示すような変化があることが認められたので、 1個1個のドライバーについて全体の補正値を平行に増大または減少させ、推定式 (2) の指数g(Πm)が１に近ずくようなバイアスを加える。このときg(Πm)→１になるとともに相関係数γが減少しないようしなければならない。このときの値がドライバー間の補正値と考えられる。ここで注意することはこの補正値を設定する前にドライバー内の補正値設定は終わらせておく必要がある。

(3) 信頼性ドライバー全体の補正値設定

上記(1) (2) で算出した補正値を組み合せたものである。従って評価点 Nの補正値は標準値であるといっても、1.00になるとは限らない。

以上が信頼性ドライバーの定量的な評価尺度の設定の仕方の考え方であるが、この種の研究がより多く行われ、繰り返し検証が行われることを期待したい。なお筆者達は現在(1) の実験を新しいデータで繰り返しており、次の機会にこの考え方の正当性をいくらかでも実証した報告ができればと思っている。

8．あとがき

信頼性ドライバーの定量的尺度の設定という得体の知れないテーマの研究に取り組んだが、ようやくその実体を把握できるところまで到達することができたといえよう。今後ともこの検証実験は続けていく予定でいるので、できるだけ多くの人のご批判とご指導を仰ぎたい。

最後に今回の研究の機会と援助を戴いたＪＳＤのソフトウェア工学グループの方々および職場の上司の方に深く感謝いたします。また絶えずこの研究を支援してくれた当社の中山、冨田の両君にも厚く謝意を表します。

参考文献

1) 情報処理工学に関する調査研究「ソフトウェア信頼性モデルの実践的評価」情報サービス産業協会・協同システム開発株式会社、昭和60年

2) 上杉勝；日本科学技術連盟第 4回ソフトウェア生産性における品質管理シンポジウム「ソフトウェア品質管理支援システムの開発と適用」昭和59年、pp107-114

3) 真野俊樹；日本科学技術連盟第 5回ソフトウェア生産性における品質管理シンポジウム「ソフトウェアにおける生産と品質の科学的管理の実践」昭和60年、pp79-86

4) 小室豊；日本科学技術連盟第 6回ソフトウェア生産性における品質管理シンポジウム「潜在エラー数の一推定方法」昭和61年、pp119-126

5) Barry.W.Boehm;「Software Engineering Economics」

6) 宮本勲；「ソフトウェアエンジニアリング現状と展望」ＴＢＳ出版会、pp229-288