プログラマの性能評価

研究ノート

鳥　居　宏　次*

1．はじめに

近年の情報化社会で計算機関連の技術者の不足が議論されている。特にソフトウェアを書くプログラマの不足についてである。しかし，本質は単なる絶対数の不足が問題ではなく，高い技術力を持ったプログラマが必要なわけである。そこで問題は高い技術力を持ったプログラマとはどのようなことを意味するのかである。

技術者に対して性能という言葉を使うのは不自然な気もするが，能力と呼ぶとさらに具体的に能力の意味を議論する羽目になり，泥沼の様相がでてくる。そこで，本稿ではプログラマも一つの抽象的な機械と考え，誤り（ソフトウェアの世界ではエラーもしくはバグという）の数が少ないこと，および作ってしまった誤りに対しては早く誤りを取り除くことを性能のいいプログラマと呼ぶことにする。

この定義に従って，以下では各エラーが作り込まれてから取り除かれるまでの時間（これをエラー寿命と呼ぶ）とそれに基づくプログラマ性能の評価尺度を提案する。更に，学生実験から収集したデータをもとに，エラー寿命とプログラマ性能との関係について述べる。

2．エラー寿命とプログラマ性能

ソフトウェア中に作り込まれたエラーがソフトウェアの生産性や信頼性に悪影響を及ぼすことが知られている[6]。従って，プログラマの性能を評価する単純な方法はエラー数を数えることである。すなわち，エラー数が多いほどプログラマの性能が低いとみる。

しかし，ソフトウェアの生産性や信頼性に与える影響がどのエラーも同じであるとは考えにくいので，エラー数を数えるだけでは評価として不十分である。エラーが持つ影響度に応じた重み付けが必要であることが指摘されている[1]。

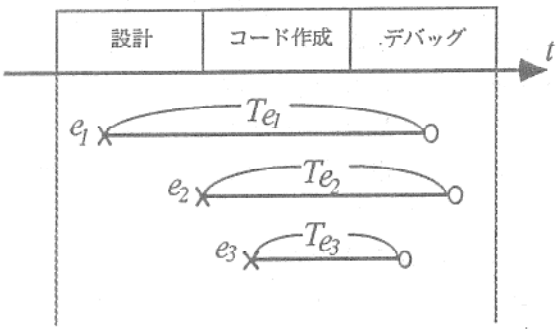

図1　エラー寿命T_e

ここではその重みとしてエラー寿命を用いることを提案する。エラー寿命T_eとは，ソフトウェア開発過程においてエラーeがソフトウェア中に具体化してから取り除かれるまでの時間とする（図1参照）。エラー寿命を重みとして用いる主な理由は次の通りである。エラー寿命が長ければ，そのエラーに関連するプログラムコード（の細部）について忘れてしまう可能性が大きい。更に，エラーが影響を及ぼすコードの量も多くなる。そのため，エラー除去に必要な労力は大きくなり，作業内容もそれだけ複雑になってしまう。

この考えに基づき，プログラマ性能の評価尺度sを次のように定義する。まず，エラー寿命T_eは与えられた問題の難しさpにも依存すると考える。すなわち，問題が難しいほど，プログラムの開発作業は複雑になり，エラーは発生しやすく，エラー寿命も長くなると仮定する。更

*鳥居宏次（Koji　TORII），大阪大学基礎工学部，情報工学科，教授，知能情報処理学

に，性能の高いプログラマほどsの値が大きくなるように逆数を用いて次式(1)のように定める。

$$s = (\text{エラー寿命}T_e\text{の総和}/f(p))^{-1} \quad (1)$$

但し，f：正規化関係

p：与えられた問題の難しさ

とする。

3．評価尺度の適用実験

次に，本学部情報工学科の学生に対して行った評価尺度の適用実験について述べる。

実験は昭和60年度第2学期に学部3年生を対象として行われた。実験ではプログラミング言語CまたはPascalを用いて，C，Pascalまたは PL/1のサブセット(ソース言語と呼ぶ)のコンパイラを作成する。

実験には約40名の学生が参加した。ここで用いるデータは，あらかじめ定めた期限までに実験を終了した9人の学生に関するものに限った。

エラー寿命T_eは，作成されたプログラムテキストの変更情報（変更時刻，変更箇所，変更理由）を人手を介して分析することにより求めた[3]。

エラー寿命T_eを測る時間の単位としては学生が端末を操作していた時間（端末使用時間）の累積を用いた。これは，端末使用時間が自動収集可能なデータであること，および，机上でよく考えた上で端末を操作するプログラマほどエラー寿命の評価が有利になること，などの理由からである。

正規化関数f(p)は，与えられた問題の難しさpがエラーの個数とエラー寿命の双方に影響を与えることより，単純に

$$f(p) = p^2 \quad (2)$$

とした。さらに，学生間でpに大きな差がないこと，各学生が用いたアルゴリズムやデータ構造がほぼ同じであることより，pを実験終了時のプログラムサイズ（行数）Lで近似的に表すことにした。従って，正規化関数は

$$f(p) \fallingdotseq L^2 \quad (3)$$

とした。

学生実験から収集したデータ，および，それらより算出したプログラマ性能に対する評価値sを表1にまとめる。

4．考　　察

ここでは，エラー寿命による重み付けがプログラマ性能の評価に有用であるか否かについて検討する。

まず，プログラマ性能に直接的に対応していると考えられる，プログラム作成に要した時間（総端末使用時間）とエラー寿命の総和との比較を行った。その結果を図2に示す。両者の間の相関係数の値は0.82であった。

一方，エラー数と総端末使用時間の比較を行ったところ，両者の間の相関係数の値は0.45であった。

以上の2つの結果より，単純にエラー数を評価するよりも，エラー寿命による重み付けを行って評価した方が，プログラム作成の効率を良く表現していると考えられる。

表1　実　験　デ　ー　タ

学生	ソース言語	プログラミング言語	プログラムサイズ(行)	総端末使用時間(分)	エラー数	エラー寿命の総和(分)	エラー寿命の平均(分)	評価値 s
#1	PL/I	C	2098	7955	77	93750	1218	46.9
#2	PL/I	C	1685	6202	101	29715	294	95.5
#3	PL/I	C	1530	5906	61	28620	469	81.8
#4	PL/I	C	1789	8021	78	67020	859	47.8
#5	C	C	1094	4754	55	32145	584	37.2
#6	C	C	1661	3463	35	11550	330	238.9
#7	PASCAL	PASCAL	2111	5838	26	24045	923	185.3
#8	C	PASCAL	1084	8651	49	66765	1363	17.6
#9	PASCAL	PASCAL	2420	5139	33	49170	1490	119.1

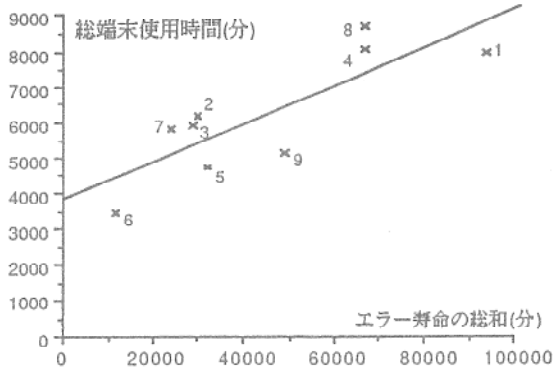

図2　エラー寿命の総和と総端末使用時間の比較

Moher と Schneider の実験結果[4] より，プログラマの“経験（experience）”と“素質（aptitude）”がプログラマ性能を決定すると考えられる。特に，プログラマが学生の場合，“経験”は大学で修得したコンピュータサイエンスおよびプログラミングに関する講義の総数で近似的に測れると考えている。一方，“素質”はそれらの講義の成績の平均点で測れると考えている。

今回の実験の場合，各学生の経験（すなわち修得したコンピュータサイエンス及びプログラミングに関する講義の総数）はほぼ同じである。従って，もしプログラマ性能に差があるなら，それは各学生の素質の差によって説明される。

そこで，各学生のコンピュータサイエンス及びプログラミングに関する講義の成績の平均点と評価値 s の比較を行った。その結果を図3に示す。両者の間の相関係数は0.75であった。これより，提案した尺度 s はプログラマ性能をかなり良く表現していると考えられる。

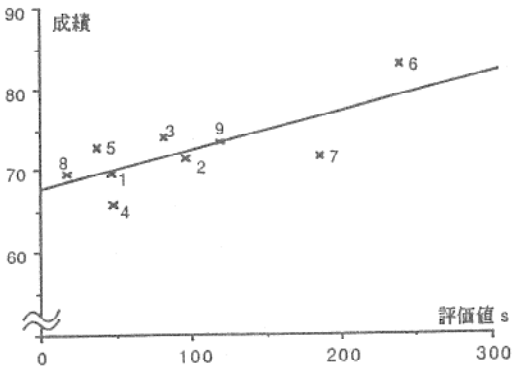

図3　評価値と成績の比較

5．おわりに

本稿では，エラーがソフトウェアに作り込まれてから除去されるまでの時間に注目したエラー寿命を新しく導入し，エラー寿命に基づくプログラマ性能の定式化を提案した。

更に，提案した尺度の有効性についての具体的な検証を行うため，学生実験から収集したデータを用いて考察した。

大学の教育の現場で実験データの収集を行ったため，考察に用いた実験データの総数は小さくなっている。提案した評価尺度の妥当性を十分に示すためにも，多人数のプログラマ（例えば，企業におけるプログラマ）を対象にした適用実験を行うことは重要な今後の課題の一つである。

エラー寿命の算出は，人手を介して行った。そのため，多くの時間と労力が必要であった。プログラム中のエラーを特定する技術など，今のところ自動化が困難な問題がエラー寿命の算出には含まれている。幸い，自動収集が容易なデータにのみ基づいてエラー寿命を近似的に求める方法について検討を終えた[2]。これを用いれば，エラー寿命に基づく情報をプログラマに実時間でフィードバックすることも可能になる[5]。

参考文献

1）福田昌弘：“バグ重大度の定量化の試み”，情報処理学会研究報告，SW-32-2 (1983).

2）K. Matsumoto, K. Inoue, H. Kudo, Y. Sugiyama, and K. Torii, : “Error life span and programmer performance”, Proc. of 11th COMPSAC, pp. 259-265 (1987).

3）松本，大西，井上，工藤，杉山，鳥居：“プログラム作成能力の評価尺度とデータ収集ツール”，情報処理学会研究報告，SW-50-6 (1986).

4）Moher, T. and G. M. Schneider,: “Methods for improving controlled experimentation in software engineering”, Proc. 5th ICSE, pp. 224–233 (1981).

5）大西，松本，杉山，鳥居：“ソフトウェア開発におけるメトリックス環境”，情報処理学会全国大会，1G–1 (1986).

6）Weiss, D.: “Evaluating software development by error analysis: The data from the architecture research facility”, J. Syst. & Software, 1, pp. 57–70 (1979).