工学ジャーナル　第3巻
2004年3月
指定投稿論文

ソフトウェアメトリクスの定量的検証法に関する研究*

A Study of Quantitative Method for Verifying Software Metrics

阿萬　裕久**・山田　宏之**・野田松太郎**

Hirohisa AMAN**・Hiroyuki YAMADA**・Matu-Tarow NODA**

Abstract: When some different software metrics measure a software attribute, they have to capture different aspects of the software attribute independently. In other words, if there are n metrics, they have to provide n-dimensional information about the software attribute. Otherwise, some of them would be redundant. The contribution of this paper is to provide a new method for verifying metrics using the principal component analysis and the correlation analysis. This method can verify metrics in terms of the following two points : (1) How many aspects could be measured by those metrics, and (2) Whether some metrics would be redundant or not.

Key words: metrics, principal component analysis, correlation analysis

1. 緒言

近年，Unified Modeling Language (UML) や Java 言語の普及に代表されるように，オブジェクト指向技術によるソフトウェアの設計及び開発が主流となってきている。一般にオブジェクト指向ソフトウェアは開発性・保守性・再利用性等に優れているといわれており，このことがオブジェクト指向技術普及の大きな理由である。しかしながら，実際にソフトウェアの設計・開発を行うのは現場の技術者であり，その成果物であるソフトウェアの品質は，設計・開発に携わった技術者の経験や力量に大きく依存する。つまり，必ずしもオブジェクト指向技術の利点が活かされるとは限らない。そのため，ソフトウェアの品質を客観的かつ定量的に評価及び予測し，ソフトウェアに対する品質管理を行うことが重要である。そのための尺度をソフトウェアメトリクス，あるいは単にメトリクスという [1]。例えば，最も基本的なメトリクスとしてプログラムの行数（line of code, *LOC*）がある。*LOC* はソフトウェアサイズを測定するためのメトリクスであり，それを用いた品質の評価や信頼性（ソフトウェアの場合，物理的な障害はあり得ないため，フォールトの数や障害件数が信頼性を表す。これらは日常的には "バグ" といわれるが，バグという用語はフォールトと障害の両方を含んだ曖昧な用語である。）の予測等が行われている [2],[3]。他にもソフトウェアの複雑さを測定するメトリクスとして，プログラムの制御フロー（いわゆるフローチャート）におけるサイクロマティック数（一次独立なループの数）等があり，ソフトウェアの構造評価やテスト工数の見積り等に利用されている [4]。

メトリクスは，明確な測定目標を持ち，その上で客観的かつ正確にソフトウェア属性を測定しなければならない [5]。ソフトウェア属性としては上述のサイズや複雑さ以外にも凝集度・結合度等があるが，一般にそのような属性にはさまざまな側面があり，単一のメトリクスで適切に測定することは難しい [6]。そのため，一つのソフトウェア属性を測定するためにいくつもの異なったメトリクスが考案され，それらを組み合わせることで目的の属性を正確にとらえようとしている。その際，各メトリクスは，それぞれ異なった視点から独立してソ

* 電子情報通信学会論文誌 D-I 第 85 巻 10 号（2002）pp.1000–1002 の内容に加筆・修正を加えたものである。
** 愛媛大学工学部情報工学科
** Department of Computer Science, Faculty of Engineering, Ehime University, Matsuyama, Japan.
{aman,yamada,noda}@cs.ehime-u.ac.jp
原稿受理　平成 15 年 11 月 5 日

フトウェア属性を測定していなければならない。換言すると，あるソフトウェア属性のために n 個のメトリクスが考案されているのであれば，それらによって n 次元の情報が得られなければならない。さもなくば，いくつかのメトリクスが冗長であることになる。

従来，メトリクスが新しく提案されるときは，Weyuker の性質 [7] や Briand らのフレームワーク [8] を用いた定性的な検証法——対象とするソフトウェア属性に関する数学的性質を利用——や，開発現場での測定データとの関係——例えば，複雑さメトリクスによる測定値と実際に含まれていたフォールト数との相関 [9] 等——を調べる定量的な検証法が使われてきた。しかしながら，新しく提案されるメトリクスを含めていくつかのメトリクスが与えられたとき，(1) “それらによって対象としているソフトウェア属性のどれだけの側面を表現できるのか”，(2) “冗長なメトリクスは含まれていないか” といった視点での検証は行われていない。新しいメトリクスが提案され，たとえ従来の検証法によってその有効性が示されたとしても，そのメトリクスによる測定が既存のいくつかのメトリクスを合成することで実現されるようではその有意性は低い。そのため本論文では，上述の (1)，(2) のような視点での検証を実現するため，主成分分析並びに相関分析 [10] を用いてメトリクスを定量的に検証する方法を提案する。

2. メトリクスの定量的検証法

あるソフトウェア属性に対し，n 個のメトリクス μ_k $(k = 1, \ldots, n)$ が存在しているものとする。このとき，これらのメトリクスの検証法を以下に提案する：

(1) N 個のソフトウェアを無作為に抽出し，各メトリクスによる測定を行う。その結果，各メトリクス値を成分とした n 次元ベクトルが N 個得られる。その各ベクトルを

$$\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in})^T$$

と表す $(i = 1, \ldots, N)$。ただし，^{T} は転置を意味する。つまり，$\boldsymbol{x}_i$ は列ベクトルである。

(2) $\boldsymbol{x}_i$ $(i = 1, \ldots, N)$ に対し，各成分間の相関係数を算出し，相関行列 R を求める。すなわち，次式で定義される r_{jk} を j 行 k 列要素とした n 次正方行列 R を求める：

$$r_{jk} = \frac{\sum_{i=1}^{N}(x_{ij} - \overline{x_j})(x_{ik} - \overline{x_k})}{\sqrt{\sum_{i=1}^{N}(x_{ij} - \overline{x_j})^2 \sum_{i=1}^{N}(x_{ik} - \overline{x_k})^2}}, \qquad (j, k = 1, \ldots, n)$$

ただし，

$$\overline{x_j} = \frac{1}{N}\sum_{i=1}^{N} x_{ij}, \qquad \overline{x_k} = \frac{1}{N}\sum_{i=1}^{N} x_{ik},$$

である。

(3) 相関行列 R の固有値を算出する。ここでは，各固有値を λ_j $(j = 1, \ldots, n)$ とする。ただし，$\lambda_1 > \lambda_2 > \cdots > \lambda_n$ である。λ_j に対応する固有ベクトルは第 j 主成分と呼ばれている‡。

‡ 通常，主成分分析では共分散行列の固有値・固有ベクトルを用いるが，各データの単位が異なっている場合はそれらを正規化しなければならない。これは相関行列を用いることを意味する。詳しくは文献 [10] を参照されたい。

(4) 各主成分の累積寄与率 α_j $(j = 1, \ldots, n)$ を算出する：

$$\alpha_j = \frac{\sum_{t=1}^{j} \lambda_t}{\sum_{t=1}^{n} \lambda_t}.$$

ただし，λ_j は相関行列 R の固有値であるため，$\sum_{t=1}^{n} \lambda_t = n$ である。なお，累積寄与率 α_j とは，第 1 主成分から第 j 主成分までの j 個の固有ベクトルを基底とした j 次元空間でもって，元の n 次元ベクトルの $\alpha_j \times 100$ (%) の情報を表現できることを意味する。

(5) いま，n 次元の情報をより低次元で表現するとしたときに，望ましい累積寄与率の下限を τ とする。すなわち，全データの $\tau \times 100$ (%) の情報を表現できれば十分であるとする。そこで $\alpha_j \geqq \tau$ となるような最小の j を求める。もし $j < n$ ならば，測定されたメトリクス値には現実には n 次元の広がりはないことになる。すなわち，n 個のメトリクス μ_k $(k = 1, \ldots, n)$ のうち，いくつかは冗長であると考えることができる。逆に $j = n$ ならば，n 個のメトリクスはいずれを欠くこともできないことが分かる。
なお，τ の値について，本論文では文献 [11] に従い $\tau = 0.95$ とする。
冗長なメトリクスの選定には，各メトリクス間の相関係数（相関行列 R の各成分；対角成分を除く）を用いればよい。つまり，m $(\leqq n - j)$ 個のメトリクスを冗長とする場合，まず最も相関の強いメトリクス対を選び，どちらか一方を冗長なメトリクスと見なす。そして，そのメトリクスの存在を無視し，同じ操作を $m - 1$ 回繰り返せばよい。ただし，本論文では相関係数の絶対値が 0.8 以上であるときを “強い相関” とみなし [12]，相関係数の絶対値が 0.8 未満であるようなメトリクス対についてはこの選定の対象外とする。

□

3. 適用例

前節で述べた検証法の適用例として，本節ではクラス凝集度メトリクスに対する検証実験の結果を示す。

凝集度は，モジュール内での要素間の機能的な関係付けの強さの度合として定義されている [1]。高い凝集度のモジュールでは，全要素が単一の機能を実現することに関係しており，開発・保守・再利用が容易で，フォールトも含まれにくいといわれている [13]。オブジェクト指向ソフトウェアでは，メソッドや属性をクラスの要素とみなすことで，クラスをモジュールに見立てることができる。つまり，クラスに対しても凝集度の概念を適用できる。

これまでにクラス凝集度メトリクスとしては以下の 8 つが知られている：

(1) lack of cohesion in methods-1 (*LCOM1*)
属性を共有していないメソッド対の数
(2) *LCOM2*
LCOM1 -（属性を共有しているメソッド対の数）
(3) *LCOM3*
属性参照に基づいてグループ化されたメソッド集合の数
(4) *LCOM4*
LCOM3 にメソッド呼出しを加味したもの
(5) *LCOM5*
メソッド・属性間結合のクラス内での密度
(6) tight class cohesion (*TCC*)
メソッドによる属性共有の比率

(7) loose class cohesion (*LCC*)
　TCC に間接アクセスを加味したもの
(8) information flow-based cohesion (*ICH*)
　パラメータ数で重み付けされたメソッド呼出し回数

このうち，*LCOM1* 及び *LCOM2* では，“メソッドどうしが同一の属性にアクセスしているか否か” に着目し，そのような組合わせの数等を用いることでクラス凝集度を測定している。また，*LCOM3* ～ *LCOM5* 及び *TCC*, *LCC* では，“メソッド間での属性の共通利用” や “メソッド呼出し” をメソッド間での結合関係ととらえ，そのような関係の広がりや密度等を用いてクラス凝集度を測定している。*ICH* では，メソッド呼出しの回数並びにメソッドのパラメータ数を利用してクラス凝集度を測定している。これらの詳細については文献 [13] を参照されたい。

今回，Sun Java 2 SDK 1.3.1 付属のクラスから 67 個を無作為に抽出し，これら 8 つのメトリクスを使って測定を行った。その結果に対して主成分分析を施し，その累積寄与率を算出すると表 1 に示す結果が得られた。

表 1. 主成分分析の結 果：累積寄与率

主成分	累積寄与率
第 1 主成分	0.508
第 2 主成分	0.733
第 3 主成分	0.845
第 4 主成分	0.945
第 5 主成分	0.977
第 6 主成分	0.995
第 7 主成分	0.999
第 8 主成分	1.000

累積寄与率は第 5 主成分で 0.95 を越えている。つまり今回の実験データは，現実的には 8 次元の広がりは持っていない。このことから，いくつかのメトリクスは冗長であると推察される。そこで，各メトリクス値間の相関を調べたところ表 2 に示す結果が得られた。

表 2. 各メトリクス間の相関係数

	LCOM1	*LCOM2*	*LCOM3*	*LCOM4*	*LCOM5*	*TCC*	*LCC*	*ICH*
LCOM1	1	0.999	0.882	0.896	0.0321	−0.0739	−0.118	0.437
LCOM2		1	0.885	0.887	0.0329	−0.0746	−0.119	0.448
LCOM3			1	0.860	0.0676	−0.0752	−0.169	0.692
LCOM4				1	0.0612	−0.161	−0.186	0.316
LCOM5					1	−0.171	−0.283	0.0372
TCC						1	0.727	−0.0145
LCC							1	−0.122
ICH								1

LCOM1, *LCOM2*, *LCOM3*, *LCOM4* それぞれの間の相関係数はいずれも 0.8 を上回っており，強い正の相関が確認できる。そのため，これらのうちのいずれか 1 つと *LCOM5*, *TCC*, *LCC*, *ICH* の 5 つのメトリクスを用いて測定を行うことも可能である。このことは，既存のクラス凝集度メトリクスで測定できる情報が

現実的には 5 次元であることに対応し，表 1 の累積寄与率とも符合する。なお，今回の測定サンプルの抽出数及び抽出範囲は必ずしも十分ではないため，この結果の意味するところが真に一般的であるかどうかを判定するのは難しい。この問題は，より多くのサンプルをより広範囲にわたって抽出することで解決される。

4. 結言

本論文では，メトリクスを定量的に検証する一つの方法として，主成分分析・相関分析を用いる方法を提案した。本手法は，いくつかのメトリクスが与えられたとき，(1) 対象としているソフトウェア属性のどれだけの側面を表現できるのか，(2) 冗長なメトリクスは含まれていないか，という 2 点について検証を行うことができる。本手法を従来の検証法と組み合わせることで，より的確なメトリクスの検証が可能になると考えられる。

参考文献

[1] 片山卓也，土居範久，鳥居宏次 監訳：ソフトウェア工学大事典，朝倉書店，1998.

[2] S.D. Conte, V.Y. Shenm, H.E. Dunsmore : Software engineering metrics and modles, Benjamin Cummings Publishing Inc., 1986.

[3] J.L. Elshoff : An investigation into the effects of the counting method used on software science measurements, SIGPLAN Notices, vol.13, no.2, pp.30–45, 1978.

[4] T.J. McCabe : A complexity measure, IEEE Trans. Software Eng., vol.SE-2, no.4, pp.308–320, 1976.

[5] N.E. Fenton and S.L. Pfleeger : Software Metrics : A Rigorous and Practical Approach 2nd ed., PWS Publishing Company, 1997.

[6] N. Fenton : Software measurement : a necessary scientific basis, IEEE Trans. Software Eng., vol.20, no.3, pp.199–206, 1994.

[7] E.J. Weyuker : Evaluating software complexity measures, IEEE Trans. Software Eng., vol.14, no.9, pp.1357–1365, 1988.

[8] L.C. Briand, S. Morasca, and V.R. Basili : Property-based software engineering measurement, IEEE Trans. Software Eng., vol.22, no.1, pp.68–85, 1996.

[9] V.R. Basili, L.C. Briand and W.L. Melo : A validation of object-oriented design metrics as quality indicators, IEEE Trans. Software Eng., vol.22, no.10, pp.751–761, 1996.

[10] 塩谷 實：多変量解析概論，朝倉書店，1990.

[11] 本多 正久：インフォメーション・アナリストのための多変量解析の実際，産能大学出版部，1993.

[12] 宮本一郎：よくわかる統計学概要，学術図書，1993.

[13] L.C. Briand, J.W. Daly, J. Wüst : A unified framework for cohesion measurement in object-oriented systems, Empirical Software Engineering, vol.3, pp.65–117, 1998.