LO-003

低コストな Java プログラムの生産量計測

Production Metrics of a Java Program at a lower cost

五味 弘† **小川 浩司**† **鈴木 剛**†
Hiroshi Gomi Hiroshi Ogawa Tuyoshi Suzuki

１．はじめに

Java プログラムに対する生産性の評価を行なうためにはそのプログラムの生産量を計測する必要がある．またその計測コストが小さいことが求められる．

この生産量の計測とその生産に要したコストにより，生産性に関するプログラム開発者の評価，また Java の各種フレームワークやツールの生産性評価が可能となる．

本稿では，まず実務で運用するときの計測時の制限について述べ，従来の計測手法とその制限に対する評価を示す．次に低コストな計測手法の概要を紹介する．さらに計測結果の考察を行ない，最後に計測手法の妥当性の評価を行なう．

２．計測における制限

Java プログラムの生産量を実務で計測するときに，以下の制限が必要になる．

1. プログラムが一部不足しているときでも計測可能
2. 実行環境がないときでも計測可能

これらからプログラムの静的解析による計測が必要になる．逆に作成中の Java プログラムの一部で計測が可能になり，第三者が随時，または複数の計測対象を一括で計測することが可能になり，効率的に計測の運用ができる．

3. 計測コストが小さいこと

計測コストを低く抑えるためには，上記の 1 と 2 による第三者の一括計測と，さらに機械的に計測できることが求められる．

３．従来の生産量の計測手法

以下に従来の計測手法[2-9][12-13]を注目する入力により，分類する．

(1) ソースコードの一次量
例．コード行数，クラス数，メソッド数，オブジェクト指向ソフトウェアメトリクス[1]
これは意味のないプログラムも平等に計測するため，望ましくない．

(2) ソースコードの二次量
例．マケイブ数[10][12]，ソフトウェアサイエンス量
一次量から何らかの評価関数を用いて計測する方法であり，いくつか報告されている[8][10][12][13]．ここでは評価関数の妥当性とその計測コストが高いことが問題になる．またオブジェクト指向メトリクスを使用していないものは採用できないが この二次量による生産量測定で妥当性があり，低コストな計測手法を提案する．

(3) 機能量 --- 例．ファンクションポイント，フィーチャーポイント，ユースケースポイント
これらは機能に注目した計測指標であるため，ソースコードの情報を利用していないので情報量が少なく望ましくない．この指標を本計測指標の妥当性の検証として使用することが可能である．

(4) マイナスの生産量 --- 例．バグ件数
バグ件数などは生産を阻害する指標であり，マイナスの生産量である．多くが静的解析では計測できないため，計測は高コストであり，本計測指標には含めない．但し，この値は別に収集・管理するために生産量の評価関数の妥当性の検証に使用する．

†（社）沖ソフトウェア株式会社, Oki Software Co.,Ltd.

４．生産量の低コストな計測手法

以下の生産量指標により，Java プログラムの一部から機械的に生産量を計測することにより，低コストで運用する．

４．１ 生産量指標

Java のソースコードの一部を入力として，生産量を計測する．これは，３章の(2)の二次量で計測する手法の一つであり，(3)や(4)から妥当性の検証を行なう．

提案する計測指標はステートメント数を基準とし，これにメソッドサイズやメソッド数，クラス数に注目して，ステートメント数当たりの機能（これを LOC パフォーマンスと呼ぶ）や障害件数との関連を経験的に主観評価して得た評価関数を採用する．以下にこの評価関数を示す．

生産量 = ステートメント数 × プログラム率 **(式 1)**

プログラム率 = ((1.0 - dev1 + 0.5) ** 係数 1) **(式 2)**
× ((dev2 +0.5) ** 係数 2)
× ((1.0 - dev3 + 0.5) ** 係数 3)

但し，１メソッド当たりの平均ステートメント数と１クラス当たりの平均メソッド数，プログラム当たりのクラス数は標準正規分布に従うとして，各々偏差値を求め，その値を dev1, dev2, dev3 とする．注意：平均ステートメント数など理論的に上限がなく対称でない分布に対して，簡単化のために正規分布の仮定をしている．これにより大きく逸脱しているものを一定の範囲内に抑える評価関数になっている．

この三者には相関関係が存在し，また LOC パフォーマンスに与える影響度も異なるので，各係数で調整する．この係数調整では後述する生産量計測の妥当性の検証で行なう主観評価に基づいて調整する．

一方，ステートメント数や 1 メソッド当たりの平均ステートメント数など，部分的な Java プログラムの静的解析だけで計測できるため，その計測コストは小さい．例えば，[11]の JMA によって，これらの計測が可能である．

これにより，生産量は経験的主観に合致した値になり，かつ低コストで計測が可能になる．

４．２ 生産量指標の根拠

計測コストを小さくするために Java プログラムにおける標準的な一次量を基に機械的に計測できる計測指標を定義した．今回は標準的な一次量としてメソッドサイズやメソ

ッド数，クラス数に注目した．これらの値とプログラムのLOC パフォーマンスや品質との相関を既存のプログラムから主観的に推量した．この結果，以下の知見を得た．

(1) 大きいメソッドが存在するプログラムは LOC パフォーマンスが悪い．これはメソッド抽出が不完全であり，メソッドの共通化が弱いためである．また１行当たりのバグ件数が多いことが見られた．

(2) 1 メソッド・クラスは悪い．クラスが適当な大きさであっても、1 メソッド・クラスは(1)と同様な傾向性が認められた．

後は同様に「クラスが大きいものは悪い」やメソッドの大きさが適当でも「 1 クラス」のときは悪いなども得られた．これらの経験的知見を基に生産量の計測指標の評価関数(式 1)，(式 2)を作成した．

５．計測結果とその評価

以下に 4.1 の計測指標で計測した結果の一部を以下の表１に示す．ここで示す計測対象のシステム A から D は JDK や Tomcat などのオープンソースのものである．

表１．Java プログラムの生産量の計測結果(一部)

評価項目	A	B	C	D	平均	標準偏差
(a) ステートメント数	440	242	3271	36887		
(b) 平均ステートメント数/メソッド	12.2	6.5	8.1	6	7.8	3.5
(c) 平均ステートメント数/メソッドの偏差値 1-dev1	0.37	0.54	0.49	0.55		
(d) 平均メソッド数/クラス	4.5	2.2	3.3	11.3	6.9	3.4
(e) 平均メソッド数/クラスの偏差値 dev2	0.43	0.36	0.40	0.63		
(f) 平均ステートメント数/クラス	54.9	14.3	26.73	67.8	40.9	21.4
(g) 上記の偏差値 1 - dev3	0.44	0.63	.057	0.37		
(h) 主観評価	0.90	0.95	1.00	1.15		
備考	Swing	Swing	Swing			
ProgramRate (係数 1 , 1, 0)	0.81	0.89	0.89	1.19		
ProgramRate (係数 2 , 2, 0)	0.66	0.8	0.79	1.41		
ProgramRate（係数 1, 1, 1）	0.76	1.01	0.95	1.04		
ProgramRate（係数 3/4, 6/4, 3/4）	0.77	0.9	0.88	1.13		
ProgramRate（係数, 0, 0, 1）	0.87	1.04	0.99	1.05		

但し，(h) の主観評価は， 0.6 から 1.4 までの 0.2 きざみの 5 段階評価(数値が大きい方が高い評価)を行なった平均値である．ここも正規分布を仮定する．また，係数を変更した 5 種類のプログラム率(ProgramRate) を計算した．平均値や標準偏差は表中の 4 システムだけでなく，計測したプログラムで特異なものを除いたすべての計測結果の平均値と標準偏差である．

例えば，表１からは A のシステムではステートメント数は 440 行であるが，プログラム率は各係数を平等にしたときはプログラム率が 0.76 であるので，その生産量は 440×0.76 = 334 行と生産量を計測した．

また，この計測手法による計測した結果から，フレームワーク別のプログラムから以下の知見が得られた．

(a) 慣れに関係せず Struts を利用プログラム --- 1.2

(b) 慣れた人がいる JSP/Servlet を直接 --- 1.1

(c) 慣れた人がいない JSP/Servlet --- 1.0

Struts の場合は，慣れによる影響が少なく，JSP/Servlet のときは慣れがやや影響することが分かる．

６．計測手法の妥当性とその評価

計測対象のプログラムを本計測指標の策定と無関係なメンバによる主観評価を行なった．

これを以下の表２と表１の(h)の欄に示す．但し表 2 のプログラム率は 1, 1, 1 のものである．

表２．主観評価とプログラム率

主観評価(h)	1.05	1.10	1.20	1.05	1.00	0.70
プログラム率	1.22	1.09	1.12	1.10	0.97	0.61
備考	Struts	Struts	独自	Struts	JSP	Swing

表 2 や表 1 から，主観評価に合致した計測結果になった．

また係数別では，主観に最も近いのは，係数 1= 3/4，係数 2 = 6/4，係数 3= 3/4 のものであった．

低コストな計測手法であるが，その結果も主観評価などと比較的高い相関関係があり，妥当性がある．

７．おわりに

Java プログラムの生産量を低コストで計測する手法について述べた．これを用いて各種のプログラムの計測を行ない，Java プログラムのフレームワークの特性や各開発者における差異を発見した．また，本計測指標の妥当性の検証をプログラムの主観評価との相関で実施し，合致することを得た．今後は FPA などの機能量計測との相関関係や，またバグ件数ととの相関関係も調査する予定である．さらに今後も多くの生産量の計測を行ない，生産性の評価を行なう予定である．

参考文献

1. Mark Rolenz, Jef Kid: "Object-Oriented Software Metrix", Prentice Hall, ISBN013179292X(1994).
2. Hirohisa AMAN et al: "A graph-based class structural complexity metric and its evaluation", IEICE Trans. on Infomation and System, vol.E85-D, no.4, pp.674-684(2002).
3. 石井康雄: "ソフトウェア生産性におけるプロジェクト管理の諸問題", 経営情報科学, vol.2, No.1, pp.11-22.
4. Robert B. Grady, Deborah L. Caswell: "ソフトウェア・メトリクス 生産性、品質の計測方法からその全社的展開まで", 日経 BP 社, ISBN4-8222-7116-1(1990).
5. Capers Jones, "ソフトウェア開発の定量化手法―生産性と品質の向上をめざして", 構造計画研究所, ISBN4320026373(1993).
6. Steve McConnell: "コードコンプリート 完全なプログラミングを目指して", 日経 BP 社, ISBN4756102107(1994).
7. C.Jones: "ソフトウェア開発の定量化手法 第 2 版", 構造計画研究所, ISBN432009722X(1998).
8. 阿萬, 山田, 野田: "属性を介したメソッド間結合に基づくクラス凝集度メトリクスの提案", ソフトウェア工学の基礎<8>, 近代科学社, ISBN4764902982(2001).
9. エドワード・ヨードン: “ソフトウェア・メトリクス”, ソフトウェア管理の落とし穴：(ISBN4-8101-8547-8), プレンティスホール, 第 7 章, pp.185-208(1993).
10. EclipseMetrics: http://www.teaminabox.co.uk/downloads/metrics/index.html
11. オージス総研:"Java Metrice Tool: JMA", http://www.ogis-ri.co.jp/otc/hiroba/others/JavaMetricsTool/JMA.html
12. マケイブのサイクロマティック数, http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html
13. 鷲崎弘宜: "ソフトウェア測定法に関する考察", http://www.fuka.info.waseda.ac.jp/~washi/other/metrics.html
14. 五味: “Java プログラムの生産性評価”, Java Technology Conference 2004, I-16(2004).