社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

ソフトウェア品質の第三者評価を目的とした メトリクス基準値導出法の提案

浦田　大地† 　藤原　雄介† 　平山　力地† 　濱崎　一樹† 　吉田　則裕†
飯田　　元†

† 奈良先端科学技術大学院大学 情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
E-mail: †{daichi-u,yusuke-fuj,rikichi-h,kazuki-h,yoshida}@is.naist.jp, ††iida@itc.naist.jp

あらまし　ソフトウェア開発現場では，定量的指標による品質評価が行われている．この手段として，構成管理データなどから計測可能なメトリクスに対する基準値を用いた手法が知られている．一方で，基準値は経験的，慣習的に決定されるため，現場への導入時や第三者評価に用いる際の説得力に欠ける．そこで本研究では，ソースコードデータセットから計測したメトリクス値の分布に依る基準値の導出法を検討した．また，複数の OSS を対象に評価実験を行った結果，提案手法で導出したメトリクス基準値により低品質なメソッドを含むファイルを発見できた．
キーワード　メトリクス基準値，第三者評価，ソフトウェア品質

Deriving Baseline Metric Values for Third-party Assessment of Software Quality

Daichi URATA†, Yusuke FUJIHARA†, Rikichi HIRAYAMA†,
Kazuki HAMASAKI†, Norihiro YOSHIDA†, and Hajimu IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916–5, Ikoma-shi, Nara, 630–0192 Japan
E-mail: †{daichi-u,yusuke-fuj,rikichi-h,kazuki-h,yoshida}@is.naist.jp, ††iida@itc.naist.jp

Abstract　Quantitative indicators have been used for quality assessment during actual software development. During the quality assessment, metric values are computed from configuration management system, and then compared with baseline metric values. However, because the baseline metric values are often determined from the convention and experience of a team, it is difficult to convince outsiders especially in the case of third-party assessment. This paper presents an approach to deriving baseline metric values from distribution of metric values computed from source code dataset. In the case study of several OSS systems, we confirmed that derived baseline metric values are useful for finding files including low quality methods.
Key words　baseline metric values, third-party assessment, software quality

1. はじめに

ソフトウェア開発企業では，定量的指標を利用したソフトウェア品質の評価が実施されている．評価方法の 1 つとして，ソフトウェアの構成管理データから計測可能なメトリクスに対する基準値を利用した手法が知られている [1] [2]．基準値を利用した手法は計測したメトリクス値と基準値とを比較する単純な方法であるため，開発担当者自身による品質管理のみでなくソフトウェア品質の第三者（品質保証部門や発注者を含む）評価の手段としても有効である．

しかし基準値の決定方法は一般に経験的または慣例的なものであり，基準値を用いた評価手法を開発現場へ新規に導入する際や第三者に対する説明に用いる際の説得力に欠ける．また，評価の対象となるドメインへの知見が十分に蓄積されていない場合，経験に基づいて基準値を決定することは困難である．

そこで，統計的根拠に基づいた基準値の導出方法を検討することは，基準値を利用したソフトウェア品質評価に客観性を与える上で有益である．大量のソフトウェアから計測したメトリ

クスの統計情報に基づいて基準値を決定することで，基準値に一般性を与えることもできる．したがって，未経験のドメインが評価対象であっても同様の基準値を用いて評価可能となることが期待できる．

本稿では，構成管理データやソースコードデータセットから統計的根拠に基づいてメトリクス基準値を導出する手法を提案する．提案手法では，ソースコードデータセットから計測したメトリクス値の分布に基づいて基準値を導出する．一例として，カリフォルニア大学アーバイン校が提供しているソースコードデータセットである UCI Source Code Data Sets(注1)（以下 UCI Datasets と表記する）を用いて基準値を導出した．

またケーススタディとして，UCI Datasets から提案手法したがって導出した基準値を利用して評価実験を行った．評価実験の対象には，JHotDraw や jEdit など 5 つのオープンソースソフトウェア（OSS）を用いた．各対象の複数のリリースバージョンから計測したメトリクスの時系列変化を，提案手法で導出したメトリクス基準値により評価した．ケーススタディの結果，基準値に基づいて低品質なメソッドを含むファイルを発見できた．

本稿の構成として，まず 2 章では本稿で扱うメトリクスやソースコードータセットについて説明する．3 章では提案手法である統計情報を利用したメトリクス基準値の導出法について述べ，4 章で実際に提案手法を用いて基準値を導出した例を示す．続いて 5 章では，4 章で示した基準値による評価実験の手順と結果を述べたうえで，提案手法および例示した基準値について議論する．6 章では関連研究に触れ，7 章で本稿の要点をまとめる．

2. 対象データおよびメトリクス

基準値の導出に利用したソースコードデータセットおよび基準値を導出したメトリクスについて以下に述べる．

2.1 対象としたソースコードデータセット

基準値の導出には UCI Datasets を利用した．UCI Datasets は，カリフォルニア大学アーバイン校から提供されているソースコードデータセットである．UCI Datasets は約 18,000 件のオープンソース Java プロジェクトを含んでいる．これらのデータはオープンソースリポジトリや開発者のリポジトリから 2010 年 4 月 22 日時点において収集されたアーカイブである．UCI Datasets に含まれるコンテンツの概要を表 1 に示す．実際には 18,000 件を超える Java プロジェクトが含まれているが，1 行以上の Java ソースコードが記述されたファイルが存在するプロジェクトは 13,000 件程度である．

ただし，基準値の導出においてはツールやメトリクスの性質を考慮して，UCI Datasets に含まれるプロジェクトのうち LOC の合計が 1,000 未満のものを除外している．結果的に，基準値導出の対象としたプロジェクトは 10,087 件である．

2.2 対象としたメトリクス

基準値の導出対象としたメトリクスは，解析ツール Understand によって計測可能なメトリクスのうち 45 種類である．Understand はアメリカの SciTools(注2) が開発した有償のソフトウェア構造解析ツールである．Understand は C, C#, Java をはじめとしたソースコードを解析してソフトウェアの構造を可視化する．ソースコード解析の機能として複雑度，ボリューム，オブジェクト指向に関するメトリクスなど約 70 種類のコードメトリクス (注3) (注4) を分析できる．本研究では，このうちプロジェクトごとに計測できる 45 種類のメトリクスについて基準値を導出し，ケーススタディを実施した．

3. 提案手法

提案手法では，構成管理データから計測したメトリクス値の分布に基づいて基準値を決定する．基準値には，箱ひげ図において頻繁に用いられている以下の値を利用する．

上限値　第 3 四分位値 + 1.5(第 3 四分位値 − 第 1 四分位値)
下限値　第 1 四分位値 − 1.5(第 3 四分位値 − 第 1 四分位値)

したがって，導出する基準値は当該メトリクスの値を正常値とみなす上限値および下限値とする．

3.1 基準値の導出手順

以下の手順で基準値を導出する（図 1 参照）．

（1） ソースコードデータセット作成

開発現場における構成管理データや OSS の公開リポジトリなどから，基準値を導出するためのソースコードデータセットを作成する．

（2） メトリクス計測

ソースコードデータセットに含まれるすべての要素から，基準値を導出したいメトリクス値を計測する．

（3） 統計処理

計測したメトリクス値の集合から第 1 四分位および第 3 四分位値を求め，後述する式にしたがって基準値（上限値および下限値）を導出する．

3.1.1 基準値の導出式

基準値導出対象のデータから計測したメトリクス値の集合を M，M の第 1 四分位値および第 3 四分位値をそれぞれ $q1$ および $q3$ としたとき，メトリクス基準値を次式で定義する．

上限値: $min(q3 + 1.5IQR, max(M))$
下限値: $max(q1 - 1.5IQR, min(M))$
$IQR = q3 - q1$

図 1　基準値導出手順の概要

(注1)：http://www.ics.uci.edu/ lopes/datasets/SDS_source-repo-18k.html

(注2)：http://www.scitools.com/
(注3)：http://www.scitools.com/features/metrics.php
(注4)：http://www.techmatrix.co.jp/i/qu/understand/app/documents/metrics.html

表 1　UCI Source Code Data Sets

Originating Repository	Apache	Java.net	Google Code	Sourceforge	Other	Total
プロジェクト数	84	3,412	5,361	99,691	2	18,828
ファイル数	559,140	289,310	402,070	1,983,044	7,346	3,237,910

ここで，関数 $min()$ は引数の最小値を，$max()$ は引数の最大値を返す．この上限値および下限値を基準値とし，上限値より大きいあるいは下限値未満のメトリクス値を外れ値とする．

4.　導出した基準値

前述した手順にしたがって，UCI Datasets を対象に 45 種類のメトリクスについて基準値を導出した．UCI Datasets を基に作成した基準値の一覧は表 2(a) を参照されたい．各メトリクスの概要については公式サイトの解説ページ (注3) (注4) に詳しい．

また，メトリクス名に「Count」を含むものについては LOC（Understand におけるメトリクス名は CountLineCode）で割ることによって正規化した値についても基準値を作成した（表 2(b) を参照）．これは，当該メトリクスがその性質上，最も一般的なメトリクスである LOC と高い相関を示したためである．

利用する基準値（上限値，下限値のいずれかあるいは両方）は，メトリクスの性質に応じて適宜選択されるべきである．

5.　ケーススタディ

表 2(a) および表 2(b) に示した基準値にしたがって，表 3 に示した OSS プロジェクトを対象にケーススタディを行った．ケーススタディの目的は，提案手法により決定した基準値により低品質なメソッドを特定できる実例を示すことである．

5.1　ケーススタディ用のソースコードデータセット

ケーススタディ用のソースコードデータセットは，5 つの OSS プロジェクトを対象に収集した．対象とした OSS の名称とバージョン数を表 3 に示す．

5.2　ケーススタディ手順

ケーススタディは，以下の手順で行った（図 2 参照）．

（1）　メトリクス計測

対象プロジェクトの全バージョンに含まれる全てのソースコードから，基準値を導出した 45 種類のメトリクス値を計測した．

（2）　メトリクスの時系列変化を取得

各メトリクスごとに，計測値のバージョン推移を追跡することでメトリクスの時系列変化を取得した．

（3）　基準値との比較

各メトリクスの時系列変化を基準値と比較して，外れ値の出現を観察した．

（4）　低品質メソッドの特定

Understand の Treemap 化機能を利用し，外れ値の出現前後のプロジェクトからメトリクス値に変化のあったファイルを特定した．この機能は，メトリクス値をマップサイズ・マップカラーによって可視化できる．メトリクス値が高いものほどサイズは大きく，カラーが濃く表示される (注10)．特定したファイルを調査することで，低品質メソッドの存在を確認した．

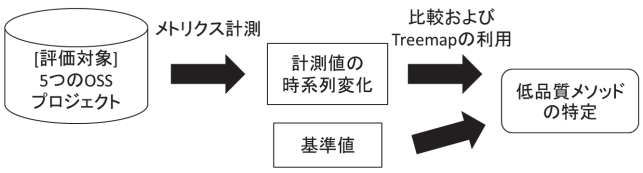

図 2　ケーススタディの手順

5.3　結果・考察

ケーススタディの結果より，メトリクスの時系列変化から外れ値の観測されたバージョンを特定することができた．外れ値が観測されたメトリクスの時系列変化を図 3 に示す．図 3 の横軸は OSS のバージョンを，縦軸はメトリクス値を表す．水平方向の直線は上下それぞれ上限値および下限値を，破線は上下それぞれ第 3 四分位値および第 1 四分位値を表している．

図 3 の外れ値が観測されたバージョンに対して TreeMap を

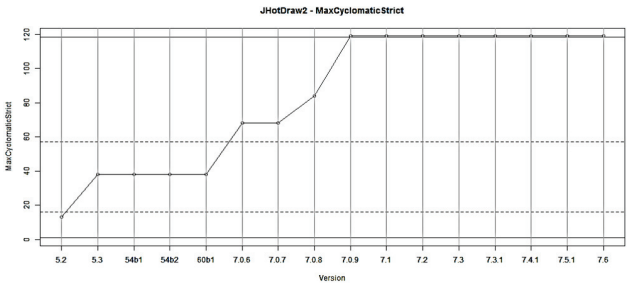

図 3　観測された外れ値（JHotDraw の MaxCyclomaticStrict メトリック）

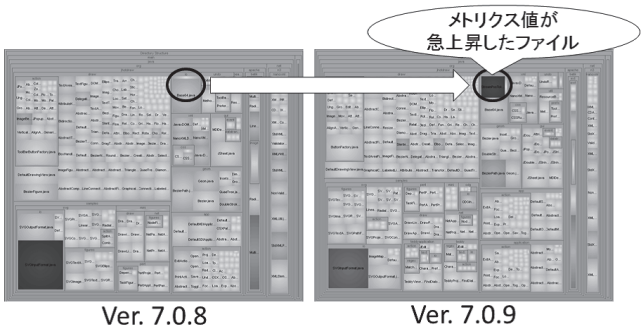

図 4　Understand の TreeMap 化機能を用いたメトリクス値の分析（JHotDraw の ver.7.0.8 と ver.7.0.9 との比較）

(注5)：http://ant.apache.org/
(注6)：http://carol.ow2.org/
(注7)：http://www.dnsjava.org/
(注8)：http://www.jedit.org/
(注9)：http://www.jhotdraw.org/

(注10)：http://www.techmatrix.co.jp/quality/understand/function/metrics.html

表 2　導出したメトリクス基準値一覧

メトリクス名	基準値		メトリクス名	基準値	
	上限	下限		上限	下限
AvgCyclomatic	3.5	1	CountLineCode	33259	1000
AvgCyclomaticModified	3.5	1	CountLineCodeDecl	11373.5	100
AvgCyclomaticStrict	3.5	1	CountLineCodeExe	15193.75	156
AvgEssential	1	1	CountLineComment	16543.5	0
AvgLine	358	8	CountSemicolon	18089.75	252
AvgLineBlank	46.5	0	CountStmt	23731.5	294
AvgLineCode	215	6	CountStmtDecl	10518.5	108
AvgLineComment	118.5	0	CountStmtExe	13369	134
CountDeclClass	525.5	1	Cyclomatic	6178	32
CountDeclClassMethod	234.5	0	CyclomaticModified	5938.5	32
CountDeclClassVariable	416	0	CyclomaticStrict	6570	32
CountDeclFile	370.75	1	Essential	3812.5	18
CountDeclFunction	3162.75	18	MaxCyclomatic	101.5	1
CountDeclInstanceMethod	2526	0	MaxCyclomaticModified	85.5	1
CountDeclInstanceVariable	972	0	MaxCyclomaticStrict	118.5	1
CountDeclMethod	2768	18	MaxInheritanceTree	11.5	1
CountDeclMethodAll	31361	31	MaxNesting	14	0
CountDeclMethodDefault	62.5	0	RatioCommentToCode	1.225	0
CountDeclMethodPrivate	252.5	0	SumCyclomatic	6685.5	32
CountDeclMethodProtected	149	0	SumCyclomaticModified	6443.25	32
CountDeclMethodPublic	2243.5	4	SumCyclomaticStrict	7109	32
CountLine	57788.75	1157	SumEssential	4168	18
CountLineBlank	7820.75	0	-	-	-

(a) Understand より得られるメトリクスの基準値一覧

メトリクス名	基準値		メトリクス名	基準値	
	上限	下限		上限	下限
CountDeclClass	3.79E-02	2.83E-04	CountLineBlank	4.31E-01	4.67E-02
CountDeclClassMethod	2.05E-02	0	CountLineCode	1	1
CountDeclClassVariable	3.30E-02	0	CountLineCodeDecl	5.49E-01	1.50E-01
CountDeclFile	2.94E-02	4.14E-05	CountLineCodeExe	6.67E-01	2.22E-01
CountDeclFunction	1.73E-01	1.99E-02	CountLineComment	1.24E+00	0
CountDeclInstanceMethod	1.52E-01	8.23E-03	CountSemicolon	7.13E-01	3.79E-01
CountDeclInstanceVariable	7.23E-02	0	CountStmt	9.00E-01	5.49E-01
CountDeclMethod	1.61E-01	1.49E-02	CountStmtDecl	5.14E-01	1.42E-01
CountDeclMethodAll	2.89E+00	2.31E-02	CountStmtExe	5.90E-01	1.90E-01
CountDeclMethodDefault	6.42E-03	0	Cyclomatic	2.67E-01	9.69E-02
CountDeclMethodPrivate	2.29E-02	0	CyclomaticModified	2.56E-01	9.77E-02
CountDeclMethodProtected	1.51E-02	0	CyclomaticStrict	2.86E-01	9.83E-02
CountDeclMethodPublic	1.42E-01	1.17E-03	Essential	1.93E-01	3.63E-02
CountLine	2.57E+00	1.01E+00	-	-	-

(b) LOC（CountLineCode）により正規化した基準値の一覧

表 3　ケーススタディ用のソースコードデータセット

名称	apacheAnt(注5)	CAROL(注6)	Dnsjava(注7)	jEdit(注8)	JHotDraw(注9)
バージョン数	24	12	62	73	16

用いた分析を行った結果，該当バージョンにはそれ以前のバージョンには存在しない複雑なメソッドを含むファイル（StreamPostTokenizer.java）が含まれていることを確認した（図 4 参照）．なお図 4 において，マップの大きさおよび濃淡は，それぞれ LOC および外れ値が観測された MaxCyclomaticStrict メトリックのバージョン内における値の高低比をそれぞれ表している．

本ケーススタディにおいて，OSS の各バージョンにおけるメトリクス値と基準値データとを比較することで，外れ値が観測されたバージョンを特定することができた．また，TreeMap を合わせて利用することで，外れ値を観測した原因となったファイルおよびメソッドを特定することができた．

本ケーススタディでは，分析者は対象としたオープンソースソフトウェアに関する知識を特に持っていなかったが，メトリクス値の時系列データや各メトリクスの基準値データ，TreeMap のみ用いて品質に問題が発生したバージョン及びファイルを容易に分析することができた．表計算ソフトやソフトウェア開発企業において広く使用されている Understand の機能のみを用いているため，高度な専門知識を持たない第三者であっても同様の品質評価を行う事ができると考えられる．また．今回用いたメトリクスであれば，企業のソフトウェア開発が対象であっても，同様の方法で分析できると考えられる．

ただしより実用性の高い解析を行うためには，基準値を導出するメトリクスの種類を増加させる必要があると考えられる．現状では静的解析ソフトウェア Understand が計測可能なメトリクスから基準値を導出しているが，今後は企業で行うソフトウェア開発において頻繁に使用されるテストカバレッジ等のメトリクス基準値を導出する必要があると考えられる．企業が行うソフトウェア開発に適用するためには，これに加えて各種メトリクスの自動収集・整理のためのツールセットの整備・公開が今後の課題となる．

5.4 妥当性への脅威

5.4.1 基準値導出方法の妥当性

本稿では，ソースコードデータセットからメトリクス値を計測し，計測値集合の四分位値から基準値を導出する手法を提案した．ケーススタディでは，提案手法で導出した基準値を利用して実際に低品質メソッドを特定できた．

しかし，メソッドが低品質であると決定づけた根拠は著者らのレビューによるものであり，客観性に欠ける可能性がある．加えて，提案手法で導出した基準値を利用した評価を行った際に，外れ値が検出されたソフトウェアが低品質であるという統計的根拠はない．提案手法で導出した基準値の客観的かつ統計的な妥当性を議論するためには，バグ管理システムの情報を利用して，外れ値の観察されたバージョン（あるいはファイル）とそれ以外のものでバグ報告件数などに統計的有意差が存在することを確認すべきであると考える．

また，基準値の導出に四分位値ではなく信頼区間を用いている研究も存在する [3]．基準値を導出するうえで，いずれの統計値を利用した場合がより正確に低品質なソフトウェアを検出できるかを議論することも今後の課題である．

5.4.2 ソースコードデータセット選択の妥当性

本稿では，OSS のソースコードデータセットである UCI Datasets からメトリクス基準値を導出した．ただし，より一般的な基準値を導出するためには，商用ソフトウェアの構成管理データも新たに加える必要があると考える．

また，UCI Datasets に含まれるプロジェクトは OSS であるという点を除き，ライセンスや開発者の性質，ソフトウェアの種類などに一貫性がない．すなわち，導出される基準値をより一般化するという観点では，直感的に妥当なソースコードデータセットであるといえる．しかし，特に開発者コミュニティの文化やソフトウェアの使用目的などの要因によって，ソフトウェア品質の評価基準は大きく異なることが予想される．したがって，コミュニティの文化や使用目的が明確なソフトウェアの品質評価に用いる基準値を導出する場合は，ソースコードデータセットに含む情報に一貫性を持たせる方が低品質なソフトウェアのみをより正確に特定できる可能性がある．

6. 関 連 研 究

本稿で提案した基準値の導出手法は，統計的な値として第 1 四分位値および第 3 四分位値を利用するものである．また本稿では，提案手法を用いて OSS のソースコードデータセットである UCI Datasets から 45 種類のメトリクス基準値を導出し，OSS プロジェクトを対象に評価実験を行った．

四分位値を利用した評価基準を設けている研究として，鷲崎らによる品質評価の枠組みの提案がある [1]．この枠組みは，ソースコードを対象としてソフトウェアの内部品質を評価するためのものである．また，評価の粒度としてモジュールからシステム全体までの規模を網羅しているため，開発者から管理者まで幅広く品質評価に役立てられる．本稿の提案手法とは，評価基準に四分位値を利用しており，開発者を除く第三者評価を想定している点で共通している．しかし，鷲崎らの評価枠組みはソフトウェア開発企業のリソースを利用している点で，OSS を利用した本稿と異なる．

OSS が対象の基準値を用いたソフトウェア品質評価の試みとしては，Meyers らの研究がある [3]．Meyers らは，OSS を対象にスライスベースの凝集度メトリクスや結合度メトリクスが品質評価において有用であるかを検討した．この研究は，OSS を対象として基準値による評価手法を議論している点で本稿と共通しているが，Meyers らは基準値の導出に信頼区間を利用している点で本稿の提案手法と異なる．

7. お わ り に

本稿では，構成管理データやソースコードデータセットから統計的根拠に基づいてメトリクス基準値を導出する手法を提案した．また，ケーススタディとして提案手法により導出した基準値を利用した評価実験を行い，低品質なメソッドを含むファイルを特定する手順を示した．

今後の課題としては，テストカバレッジ等のメトリクス基準値を導出して企業のリソースを対象にケーススタディを実施することや，提案手法で導出した基準値の統計的な妥当性を検証

することが考えられる．

謝辞

本研究は独立行政法人 情報処理推進機構 2012 年度ソフトウェア工学分野の先導的研究支援事業 （ソフトウェア品質の第三者評価のための基盤技術 ーソフトウェアプロジェクトトモグラフィの開発ー）の助成を得た．

文　　献

[1] 鷲崎弘宜，波木理恵子，福岡呂之，原田陽子，渡辺博之，“プログラムソースコードのための実用的な品質評価枠組み,” 情報処理学会論文誌，vol.48，no.8，pp.2637–2650，aug 2007.

[2] 独立行政法人情報処理推進機構 ソフトウェア・エンジニアリング・センター，“ソフトウェア開発データ白書 2009,” 2009.

[3] T.M. Meyers and D. Binkley, “An empirical study of slice-based cohesion and coupling metrics,” ACM Trans. Softw. Eng. Methodol., vol.17, no.1, pp.2:1–2:27, Dec. 2007.