メトリクス値の標準化による fault-prone モジュール判別モデルの精度向上

藏本達也[†]　柗本真佑[†]
亀井靖高[†]　門田暁人[†]　松本健一[†]

ソフトウェア開発において，信頼性の確保，およびテストの効率化のためには，fault-prone モジュール（バグを含みやすいモジュール）を予測し，それらを重点的にテストすることが求められる．ただし，予測対象のソフトウェアが，予測モデル構築に用いたソフトウェアと異なる特徴を持つ場合，必ずしも十分な予測精度が得られない．本研究では，異なる特徴を持ったソフトウェア間においても十分な予測精度を得ることを目的として，モデル構築時，および予測時に，モデルの入力となるソフトウェアメトリクスの値を標準化する方法を提案する．提案手法の効果を確かめるために，NASA IV & V Facility Metrics Data Program で公開している 9 プロジェクト分のデータセットを対象として評価実験を行った．実験の結果，提案手法を適用しない場合と比べて，AUC の値が平均で 0.07 向上した．

Performance Improvement of Fault-prone Module Prediction by Normalization of Software Metrics

Tatsuya Kuramoto,[†] Shinsuke Matsumoto,[†]
Yasutaka Kamei,[†]Akito Monden,[†] and Ken-ichi Matsumoto[†]

In software development, it is important to assign test effort to fault-prone modules, which are likely to include a bug, to ensure the reliability and not to waste test effort for fault-less modules. However, enough prediction accuracy is not always achieved when characteristics of a target project is different from that of a base project used for building a prediction model. This paper proposes a method to normalize software metrics used as predictor variables of the model, to get enough prediction accuracy for such a cross-project prediction. We evaluated the proposed method using data sets of nine projects from NASA IV & V Facility Metrics Data Program. As a result of the experiment, compared with a conventional method, the AUC value improved 0.07 on average.

1. はじめに

ソフトウェアテストおよび保守において fault-prone モジュール（欠陥を含む確率の高いモジュール）を特定することは，テストの効率化やソフトウェアの信頼性を向上するうえで重要である[4]. そのため, 複数のモジュールから計測されたメトリクス（コード行数やサイクロマティック数等）を説明変数とし，欠陥の有無を目的変数とする fault-prone モジュール判別モデルが多数提案されている[1][2][3][5][6][9][10]. 一般にこれらの fault-prone モジュール判別モデルは，過去のバージョンで計測されたモジュールデータに基づき構築され，現行バージョンのモジュールに残存する欠陥の判別に利用される[6][12].

ただし，前バージョンのデータセットが存在しない場合，例えば新規開発プロジェクトやモジュールのメトリクス計測を事前に行っていない場合などには，判別モデルを構築するためのデータセット（以降，フィットデータ）を用意できないため判別モデルを構築できない．このような場合にはオープンソースソフトウェアから計測されたデータセットや，web 上に公開されているデータセットを用いることで判別モデルの構築に利用することは可能である．

しかしながら 12 個の異種プロジェクト間での fault-prone の判別を試みた Zimmermann らの研究によると，十分な精度で判別できた組み合わせは 622 通りの組み合わせうちわずか 3.4%である. 異種プロジェクト間での判別時に高い判別精度が確保できなかった理由として，Zimmermann らはプロジェクト間の性質の違いやデータセット中のメトリクスの分布の違いなどを挙げている[11].

そこで本稿では，判別モデル構築用の前バージョンのデータセットを用意できない場合にも高い精度での判別を可能とするために，異種プロジェクト間の分布の違いを補正する標準化処理法を提案する．提案手法はデータの正規分布化を試みる対数変換と，データの分布の広がりを補正する Z-score 法の 2 つの手法からなる．この標準化処理法を判別モデル構築用のフィットデータ，及び判別対象となるテストデータの両方に適用することにより, 2 つのデータ間の分布の違いを補正する. 実験には NASA IV & V Facility の公開している 9 個のデータセット[7]を用い，計 72 通りの組み合わせ全てについて実験を行う．

以降，2 章で提案する標準化処理法について説明する．3 章で実験の方法とその手順について述べ，4 章でその実験結果について述べる．5 章で実験結果に対する考察を行い，6 章で本稿のまとめと今後の課題を述べる．

[†] 奈良先端科学技術大学院大学，情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

2. 提案手法

異なるプロジェクト間で fault-prone モジュールを予測する手法に関しての従来の研究は fault-prone モジュール判別モデルの構築に主眼を置き，様々なモデルが提案されてきたが，期待する予測精度が得られない理由の 1 つとしてモデル構築に使用するプロジェクト（フィットデータ）と予測対象プロジェクト（テストデータ）間でのメトリクスの分布の違いが影響していることが考えられる．本稿では，フィットデータとテストデータのメトリクスの分布の違いを補完することに着目し，以下の処理をおこないメトリクス値の標準化を図る．

2.1 対数変換

一般に，ソフトウェアのモジュールの特徴としてメトリクスは図 1 に示すイメージ図のように 0 付近に多数分布している．ここではプロジェクト間で分布の形が不揃いであるため，フィットデータとテストデータの正規分布化を意図して式(1)により対数変換をおこなう．

$$Y = \log_{10} X \tag{1}$$

式中の X はメトリクス値を指す．

対数変換によりメトリクスの分布は図 2 のように正規分布の形へと変換される．

なお，対数変換をおこなう際，メトリクスの初期値が 0 の場合は変換が不可能であるため，全てのメトリクス値に 1 を加算した後に対数変換をおこなっている．

2.2 Z-score 変換

対数変換後のデータは正規分布の形へと変換されているが，ここではフィットデータとテストデータの分布を揃えるために，式(2)により Z-score 変換をおこなう．これにより図 3 に示すように平均が 0，標準偏差が 1 の分布へと変換される．

$$Z = \frac{Y - \bar{Y}}{\nu} \tag{2}$$

式中の $\bar{Y}$，ν はそれぞれ Y の平均値，標準偏差を示す．

3. 実験

3.1 実験概要

実験では，プロジェクトごとに 2 章で述べた標準化処理を実施し，予測対象プロジェクト(テストデータ)とは異なるプロジェクト(フィットデータ)を用いて fault-prone モジュール判別モデルを構築し，fault-prone モジュールの予測実験をおこなった．また，標準化処理を適用しない場合の予測実験も併せておこない，標準化処理の有無による効果を実験的に評価した．本実験では，fault-prone モジュール判別における標準

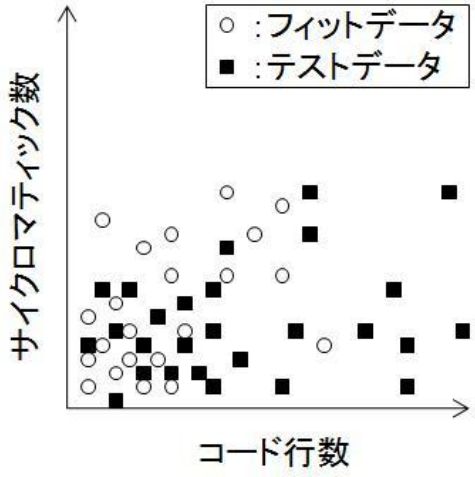

図 1　コード行数とサイクロマティック数の関係

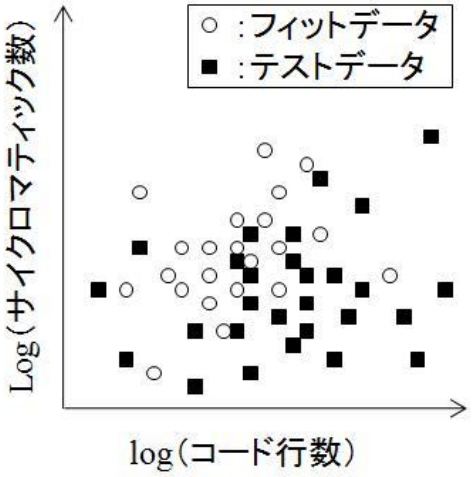

図 2　対数変換後の分布

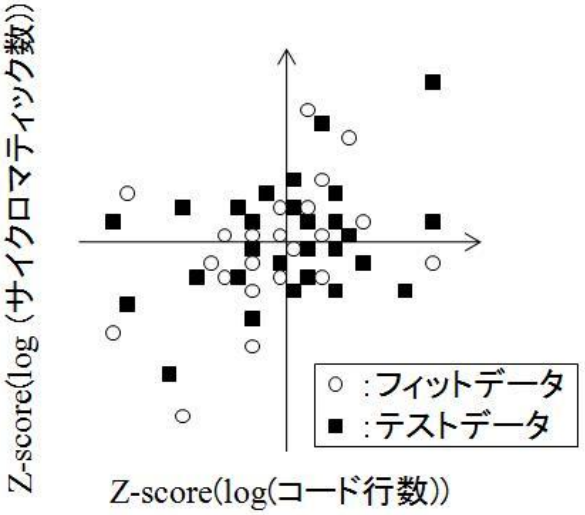

図 3　Z-score 変換後の分布

表 1　実験に使用したデータセット

プロジェクト名	モジュール数	バグ含有率[%]	コード行数[KSLOC]
CM1	505	9.5	20
JM1	10,878	19.3	18
KC1	2,107	15.4	25
MW1	403	7.7	8
PC1	1,107	6.9	40
PC2	5,589	0.4	26
PC3	1,563	10.2	40
PC4	1,458	12.2	36
PC5	17,186	3.0	164

的なモデルであるといわれているロジスティック回帰分析を用いた[1].

3.2 データセット

実験には NASA IV & V Facility Metrics Data Program で公開しているデータセットを用いた．NASA IV & V Facility Metrics Data Program とは，NASA の独立検証機構である IV & V Facility が過去のプロジェクトにおける成果物の一部をリポジトリに蓄積し，一般に公開している活動である．公開されているデータセットは複数のプロジェクトで開発されたモジュールに関するメトリクスを記したものである．本稿では，9 個のデータセットを実験対象とした．各データセットの概略を表 1 に示す．表中のバグ含有率とはモジュール総数に対してのバグを含んだモジュール数の割合を指す．また，fault-prone モジュール判別モデルを構築する際の従属変数はモジュールのバグの有無とし，独立変数はモジュールのメトリクスとした．実験に用いたメトリクスの詳細を表 2 に示す.

3.3 評価指標

fault-prone モジュール判別モデルの予測精度の評価指標としては, Alberg Diagram[8] の AUC(Area Under the Curve)を用いた．Alberg Diagram とはバグを含んでいる可能性の高い順(rank-order)にモジュールを抽出した際に，実際にバグが含まれるモジュールをどれだけ抽出できたかその割合をグラフ化したものである(図 4 参照)．グラフの横軸は rank-order に抽出されたモジュールの割合を表し，縦軸は判別モデルによって抽出された，実際にバグを含んでいたモジュールの割合を表す．AUC とは曲線下面積のことであり，予測精度が高いほど左上に凸形状になり曲線下面積が大きくなる．値域は[0,1]をとり，図 4 に示す例の場合 AUC の値は 0.8 程度となり，予測精度としては高いことを示す．また，無作為に抽出した場合，その値はおおよそ 0.5 となる．

表 2　実験に使用したメトリクス

メトリクスの名前	内容
LOC BLANK	空白行の行数
BRANCH COUNT	分岐の数
LOC CODE AND COMMENT	コメント入りコード行数
LOC COMMENT	コメント行数
CYCLOMATIC COMPLEXITY	サイクロマティック複雑度
DESIGN COMPLEXITY	McCabe の design complexity
ESSENTIAL COMPLEXITY	McCabe の essential complexity
LOC EXECUTABLE	実行可能コード行数
HALSTEAD CONTENT	Halstead の content
HALSTEAD DIFFICULTY	Halstead の difficulty
HALSTEAD EFFORT	Halstead の programming effort
HALSEAD ERROR EST	Halstead の error estimate
HALSTEAD LENGTH	Halstead の length
HALSTEAD LEVEL	Halstead の level
HALSTEAD PROGTIME	Halstead の programming time
HALSTEAD VOLUME	Halstead の volume
NUM OPERANDS	オペランド数
NUM OPERATORS	オペレータ数
NUM UNIQUE OPERANDS	オペランドの種類数
NUM UNIQUE OPERATORS	オペレータの種類数
LOC TOTAL	総行数

3.4 実験手順

実験については以下に示す 3 つの手順で構成される.

手順 1. 標準化処理

用意した 9 個のデータセットに対し，標準化処理を適用する.

手順 2. fault-prone モジュール判別モデルの構築

標準化処理を適用したデータセットのうちの 1 つをフィットデータとして，fault-prone モジュール判別モデルを構築する．相互に相関が強いメトリクスは，予測精度に大きく影響するため変数増減法により相関の強いメトリクスを排除した.

手順 3. fault-prone モジュールの予測

構築した判別モデルを用いてフィットデータに使用したデータセット以外の 8 個のデータセットをそれぞれ予測し，予測結果から AUC を算出した．本稿において

は 9 個のデータセットを用いて予測実験をおこなうため，全 72 件の予測結果が得られる．

4. 実験結果

標準化処理の工程を適用しなかった場合の実験結果を表 3 に，本稿で提案している標準化処理を適用した場合の実験結果を表 4 に示す．表 4 中の白抜きは標準化処理を適用せずに予測した結果と比較して評価指標 AUC の値が低下しているケースを示し，灰色地は AUC の値に変化がなかったケース，白地は AUC の値に改善がみられたケースを示す．

提案手法を適用することにより，72 件の内，60 件の予測結果が向上し，AUC の平均値は 0.67 から 0.74 に 0.07 向上した．また，一部のデータセットにおいては標準化処理を適用することで，大幅に予測精度が向上したものがあった．例えば，KC1 から PC2 を予測した場合，AUC の値で 0.21 から 0.83 へと予測精度が向上し，PC1 から PC5 を予測した場合，0.65 から 0.90 へと予測精度が向上した．

また，予測結果をテストデータごとに箱ひげ図で表したものを図 5 に，フィットデータごとに箱ひげ図に表したものを図 6 に示す．例えば，図 5 の“CM1”は，CM1 を予測した結果の箱ひげ図であり，図 6 の“CM1”は，フィットデータに CM1 を用いて予測した結果の箱ひげ図である．図中の白地の箱ひげ図は標準化処理を適用しなかった予測実験によるもので，灰色地の箱ひげ図は標準化処理を適用した予測実験によるものである．

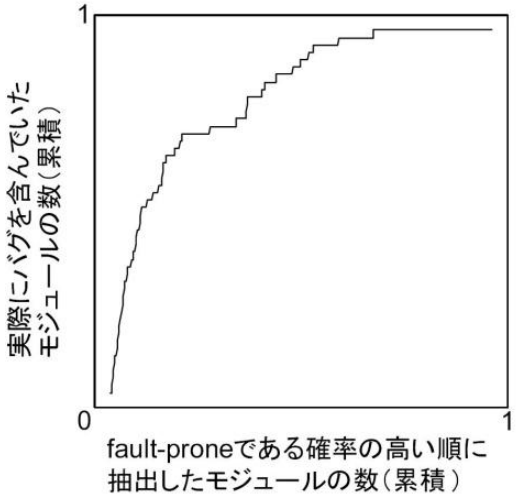

図 4　Alberg Diagram の AUC の例

表 3　予測結果（標準化処理なし）[AUC の値]

フィットデータ ＼ テストデータ	CM1	JM1	KC1	MW1	PC1	PC2	PC3	PC4	PC5
CM1		0.57	0.71	0.72	0.61	0.67	0.66	0.52	0.63
JM1	0.77		0.76	0.79	0.71	0.87	0.78	0.72	0.88
KC1	0.75	0.58		0.75	0.49	0.21	0.57	0.35	0.73
MW1	0.79	0.63	0.69		0.70	0.78	0.75	0.54	0.73
PC1	0.76	0.55	0.55	0.71		0.66	0.74	0.72	0.65
PC2	0.74	0.56	0.75	0.56	0.63		0.64	0.72	0.75
PC3	0.79	0.57	0.76	0.70	0.69	0.88		0.77	0.80
PC4	0.46	0.45	0.54	0.30	0.58	0.86	0.54		0.55
PC5	0.77	0.64	0.78	0.68	0.69	0.87	0.71	0.69	

AUC 平均値：0.67

表 4　予測結果（標準化処理あり）[AUC の値]

フィットデータ ＼ テストデータ	CM1	JM1	KC1	MW1	PC1	PC2	PC3	PC4	PC5
CM1		0.60	0.72	0.80	0.70	0.85	0.72	0.69	0.87
JM1	0.75		0.78	0.80	0.75	0.88	0.79	0.78	0.94
KC1	0.77	0.60		0.74	0.63	0.83	0.63	0.60	0.92
MW1	0.80	0.63	0.77		0.75	0.87	0.79	0.72	0.94
PC1	0.75	0.60	0.71	0.77		0.85	0.80	0.80	0.90
PC2	0.78	0.62	0.78	0.76	0.68		0.75	0.71	0.93
PC3	0.77	0.59	0.73	0.78	0.76	0.79		0.79	0.89
PC4	0.58	0.47	0.47	0.36	0.66	0.54	0.72		0.70
PC5	0.77	0.65	0.79	0.78	0.70	0.87	0.75	0.70	

AUC 平均値：0.74

提案手法を適用した場合の箱の両端は，図 5・CM1 と図 6・PC3 を除き，適用しない場合と比べて大きいことが見てとれ，提案手法を適用することで予測精度がより向上しているのが確認できた．また，全てのデータセットにおいて偏りなく予測

精度の向上が図れているのが確認でき，提案手法を適用した場合の最小値は，図 5・KC1 を除き，適用しない場合と比べて大きかった．このことから，提案手法は予測を大きく外す可能性を低減していることが示唆された．

5. 考察

実験において予測精度が悪化したケースが 9 件確認された．この原因は，外れ値の影響によるものだと考えられる．ここでいう外れ値とはデータセット PC1, PC2, PC3，PC4 および PC5 において確認できたコード行数が 0 のモジュールを指しているが，それぞれのデータセットを構成するモジュール総数の数%～20%程度を占めている（表 5 参照）．一般に，コード行数が 0 のモジュールは考えられないが，NASA IV & V Facility の公開しているデータセットのコード行数は，単位が KSLOC であることから 500 行未満のものは全て 0 で記載されていると推察できる．それらの外れ値が，本来異なる特徴を有するプロジェクトを補正するためにおこなう標準化処理の阻害要因になっていると考えられる．しかしながら，外れ値を含むデータセットに係わる予測実験の全ての予測精度が悪化しているわけではなく， 組み合わせによっては大幅に予測精度が向上しているものも確認できるため，今後さらなる分析が必要である．

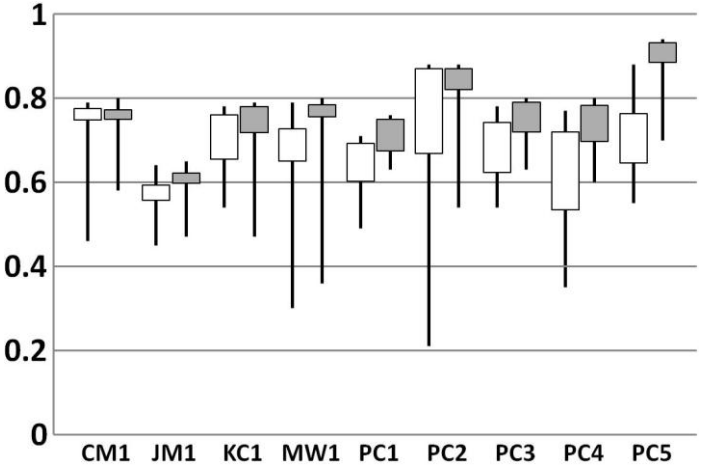

図 5　テストデータごとの予測結果分布[AUC の値]

表 5　プロジェクトと外れ値の関係

プロジェクト名	CM1	JM1	KC1	MW1	PC1	PC2	PC3	PC4	PC5
構成モジュール数	505	10,878	2,107	403	1,107	5,589	1,563	1,458	17,186
コード行数 0 の数	0	0	0	0	48	1,084	52	111	1,772
コード行数 0 含有率	0.0%	0.0%	0.0%	0.0%	4.3%	19.4%	3.3%	7.6%	10.3%

6. おわりに

本稿では，異なるプロジェクト間でのメトリクスの分布傾向を揃えるためにメトリクスの標準化処理を提案した. NASA IV & V Facility が公開している 9 プロジェクト分のデータセットを用いて，提案手法の効果を実験的に評価した．実験から得られた知見は以下のとおりである．

・　標準化処理を適用することにより，予測件数 72 件の内，60 件に予測精

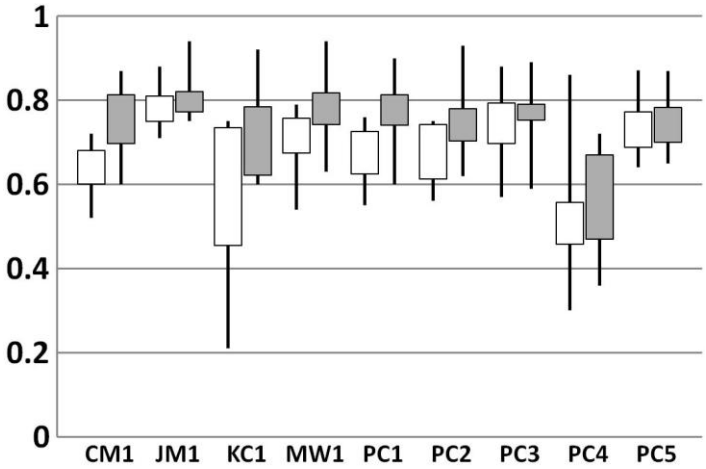

図 6　フィットデータごとの予測結果分布[AUC の値]

度の改善がみられた．AUC の値が，平均で 0.07 向上した．
- 一部のデータセットにおいては，標準化処理の適用により大幅に予測精度の向上がみられた．
- 標準化処理を適用しても AUC の値が 0.5 を下回る予測結果が 3 件確認した．

今後の課題としては，予測精度が悪くなった予測ケースの原因の解明，ならびに AUC の値が 0.5 を下回っている予測ケースの分析が挙げられる．

また，本稿ではロジスティック回帰分析により判別モデルを構築したが，他の判別モデルに標準化処理を適用してどの程度の効果が認められるかを分析する．

謝辞　本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，日本学術振興会特別研究員奨励費（課題番号 21・8995，20・9220）による助成を受けた．

参考文献

1) Basili, V.R., Briand, L.C. and Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators, *IEEE Trans. Software Engineering*, Vol.22, No.10, pp.751–761 (1996).
2) Gray, A. R. and MacDonell, S. G.: Software Metrics Data Analysis—Exploring the Relative Performance of Some Commonly Used Modeling Techniques, *Empirical Software Engineering*,Vol.4, No.4, pp.297–316 (1999).
3) Gyimothy, T., Ferenc, R. and Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction, *IEEE Trans. Software Engineering*, Vol.31, No.10, pp.897–910 (2005).
4) Li, P.L., Herbsleb, J., Shaw, M. and Robinson, B.: Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc., *Proc. 28th Int'l Conf. on Software Engineering* (*ICSE'06*), pp.413–422 (2006).
5) Munson, J. C. and Khoshgoftaar, T. M.: The Detection of Fault-prone Programs, *IEEE Trans. Software Engineering*, Vol.18, No.5, pp.423-433 (1992).
6) Nagappan, N., Ball, T., and Zeller, A.: Mining metrics to predict component failures, *Proc. Int'l Conf. on Software Engineering* (*ICSE'06*), pp.452-461 (2006).
7) NASA/WVU IV&V Facility. Metrics Data Program. http://mdp.ivv.nasa.gov/.
8) Ohlsson N. and Alberg H.: Predicting Fault-Prone Software Modules in Telephone Switches. *IEEE Trans. on Software Engineering*, Vol.22, No.12, pp. 886–894 (1996).
9) Pighin, M. and Zamolo, R.: A Predictive Metric Based on Discriminant Statistical Analysis, *Proc. 19th Int'l Conf. on Software Engineering* (*ICSE'97*), pp. 262- 270 (1997).
10) Song, Q., Shepperd, M.,Cartwright, M. and Mair, C.: Software Defect Association Mining and Defect Correction Effort Prediction, *IEEE Trans. Software Engineering*, Vol.32, No.2, pp.69-82 (2006).
11) Zimmermann T., Nagappan N., Gall H., Giger E. and Murphy B.: Cross-project Defect Prediction. *The 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (*ESEC/FSE'09*), pp.91-100 (2009).
12) 亀井靖高，柗本真佑，柿元 健，門田暁人，松本健一: Fault-proneモジュール判別におけるサンプリング法適用の効果，情報処理学会論文誌, Vol.48, No.8, pp.2651-2662 (2007).