ソフトウェア開発プロジェクトをまたがるfault-proneモジュール判別の試み——18プロジェクトの実験から得た教訓——

藏本　達也[†]　　亀井　靖高[††]　　門田　暁人[†]　　松本　健一[†]

Fault-prone Module Prediction Across Software Development Projects ——Lessons Learned from 18 Projects——

Tatsuya KURAMOTO[†], Yasutaka KAMEI[††], Akito MONDEN[†], and Kenichi MATSUMOTO[†]

あらまし　ソフトウェアテスト・保守において，限られたリソースで信頼性を確保するために，faultの有無を推定するモデル（fault-proneモジュール判別モデル）が数多く提案されている．しかし，fault-proneモジュール判別モデルの構築には，同一プロジェクトの過去バージョンの開発で計測されたメトリックスと欠陥データが必要であり，開発データの計測・蓄積が行われていない企業や，新規開発プロジェクトでは導入が困難であった．そこで本論文では，他のプロジェクトのデータを利用してモデル構築・判別を行う上で有用と考えられる手法を明らかにするため，四つのリサーチクエスチョンを実験的に検証した．18個のプロジェクトデータを用いた実験を通して，(1) ランダムフォレストはプロジェクトをまたがる判別に効果を発揮する，(2) 学習データに対する前処理（正規化）の効果はない，(3) データセット間に類似性が確認できるならば，高い精度での判別が期待できる．(4) 複数のプロジェクトのデータを用いた集団学習の効果はある，といった教訓が得られた．

キーワード　プロジェクト間予測，fault-proneモジュール判別，ランダムフォレスト，実証的実験

1. まえがき

ソフトウェアテストや保守において，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，テスト工数割当やリファクタリングの優先順位の決定に役立つ[9], [18]．そのため，モジュールのメトリックス（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（fault）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おり，企業における適用事例も報告されている[14], [16], [19]．

ただし，開発データの計測・蓄積を行っていない企業や，新規開発プロジェクトでは，性能のよいfault-proneモジュール判別モデルの構築は困難である．一般に，高い精度でfault-proneモジュール判別を行うために，判別モデルの構築は，同一プロジェクトの過去のバージョンの開発で計測されたメトリックスと欠陥データを用いることが望ましいためである[28]．従来，オープンソースソフトウェアなどの公開されているデータや，他プロジェクトのデータを利用してモデル構築・判別を行うことも試みられているが，多くの場合，高い判別精度は得られない[25], [28]．また，どのような条件であれば高い精度が得られるのかについても必ずしも明らかではない．

ただし，他プロジェクトのデータを用いることに見込みがないわけではない．他プロジェクトのデータを利用する場合の利点として，複数のデータが利用可能なこと（複数のデータの中から適切なものを選んだり，複数のデータを学習データとして利用できること）が挙げられる．そのため，予測対象プロジェクトの特徴に合わせてうまく学習データを選ぶことができれば，高い精度が得られる可能性がある．また，複数の学習データや判別モデルを用いた集団学習[3]により，予

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, 8916–5 Takayama, Ikoma-shi, 630–0192 Japan
[††] 九州大学大学院システム情報科学研究院，福岡市
Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka-shi, 819–0395 Japan

測精度を高められる可能性がある．また，近年，学習データにオーバフィッティングしにくいモデリング手法としてランダムフォレスト [1] が提案されている．プロジェクトをまたがる判別においては学習データと予測対象データの性質（fault の有無に影響する要因）が異なっていることから，学習データに過度に適合するのを回避できる点でランダムフォレストは効果を発揮する可能性がある．更には，モデル構築の前処理として，データの正規化を行うことで，プロジェクト間の違いを減らし，判別精度を高められる可能性がある．

以上のことから，本論文では，18 個のプロジェクトのデータを用いた実験を通して，次の四つの質問（リサーチクエスチョン）に答えることを目的とする．

(RQ1) ランダムフォレストは，プロジェクトをまたがる判別に効果を発揮するか？

(RQ2) 学習データに対する前処理（正規化）の効果はあるか？

(RQ3) 高い判別精度が得られる学習データ（プロジェクト）の選定方法は何か？予測対象プロジェクトに応じてうまく学習データを選ぶことができれば，十分な精度が得られる可能性がある．

(RQ4) 複数のプロジェクトのデータを用いた集団学習の効果はあるか？

本論文では，NASA が公開しているデータセット（11 プロジェクト）に加えて，オープンソースソフトウェアプロジェクトから筆者らが収集したデータセット（7 プロジェクト）を用いて，RQ1〜RQ4 を実験的に検証する．

以降，**2.** では関連研究について述べ，**3.** では各リサーチクエスチョンの詳細とその検証方法を述べる．**4.** では実験について述べる．**5.** では，実験結果のまとめと考察を述べ，**6.** では本論文の制約を述べる．**7.** は全体のまとめと今後の課題である．

2. 関連研究

2.1 Fault-prone モジュール判別モデル

これまでに，モジュールのメトリックス（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用いられてきた．

(1) ロジスティック回帰分析

ロジスティック回帰分析では，判別モデル（判別関数）はロジスティック関数の形で表される．各モジュールについて p 個の説明変数の値が観測されているとき，判別モデルは次式で表される．

$$P(y|x_1,\cdots,x_p) = \frac{1}{1+e^{-(\alpha_1 x_1+\cdots+\alpha_p x_p+\beta)}}$$

ここで $y \in \{0,1\}$ は群を表す目的変数，x_i は説明変数，α_i は判別係数であり，$P(y|x_1,\ldots,x_p)$ は，説明変数の組 $x_1,\ldots,x_p$ に対して，y が 1 の値をとる条件付き確率である．本論文では，$P(y|x_1,\ldots,x_p)$ が 0.5 以上の値をとる場合，fault-prone モジュールであると判定する．

近年，fault-prone モジュールを判別する際のモデリング手法としては最も利用されているのが本手法であり，本論文においても，次に紹介するランダムフォレストの比較対象として採用する．

(2) ランダムフォレスト

ランダムフォレストは，判別モデルとして分類木（決定木）または回帰木を用いて集団学習を行う手法であり，2001 年に Breiman によって提案された [1]．モデル構築用のデータセットに対して繰り返しランダムサンプリング（復元抽出）を行い，得られたサンプル群から多数の分類木を構築する．そして，各分類木の出力の多数決（または各回帰木の出力の平均）により最終的な判別結果を得る．従来の集団学習と異なり，各モデル構築時に全ての説明変数を用いるのではなく，ランダムに選択された一部の説明変数のみを用いる点に特色がある．

従来の分類木は，説明変数と目的変数の間の非線形の関係を表現できるという特長があったが，その半面，判別精度は必ずしも高くなかった．ランダムフォレストは，分類木の特長を受け継ぎつつ，判別精度と汎化能力が高まっており，モデル構築用データに過度に適合（オーバフィッティング）しない点に特長がある．

(3) その他のモデル

(1)，(2) で紹介した以外にも，多数の判別手法が fault-prone モジュール判別に用いられている．例えば，線形判別分析 [22]，分類木 [8]，ニューラルネットワーク [5]，Support Vector Machine [27]，ベイズ識別器 [4]，k 近傍識別器 [10] などである．ただし，最近の研究では，ランダムフォレストが最有力であることが示されている [13] ことと，紙面の都合より，本論文では割愛する．

2.2 本論文の背景となる研究

Briand ら [2] は，開発メンバーと開発言語が共通の 2 プロジェクト間で，fault-prone モジュールの予測を試みた．予測モデルとして，ロジスティック回帰分

析，及び，ロジスティック回帰分析を改良した Multivariate Adaptive Regression Splines（MARS）を用いた結果，ロジスティック回帰分析では適合率 74%，再現率 45%となり，MARS では適合率 68%，再現率 48%となった．再現率と適合率のバランスを評価する cost-benefit 分析により，MARS モデルがロジスティック回帰分析よりも優れていることを示した．

Zimmermann ら [28] は，予測モデルとしてロジスティック回帰モデルを用い，12 個の異種プロジェクト間での fault-prone の判別を試みた結果，十分な精度で判別できた組合せは 622 通りの組合せのうち，わずか 3.4%であった．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，プロジェクト間の開発対象システムの違いやデータセット中のメトリックスの分布の違いなどを挙げている．

これらの研究から，プロジェクト間で fault-prone モジュール判別を行うためには，何らかの工夫を要することがうかがえる．

2.3 プロジェクト間予測の精度向上の先行研究

Sato ら [24] は，開発時期，開発者，開発ドメインなどが全く異なる二つのプロジェクト間での予測を試みた．二つのプロジェクトの双方において fault の有無と相関の高いメトリックス（ソースコード行数，最大ネストレベル）のみを抽出してモデル学習すると，プロジェクト間予測がうまくいくことを示している．ただし，目的変数である fault と説明変数である各メトリックスとの相関は，予測対象とするプロジェクトについては未知であるため，Sato らの成果をそのまま利用することはできない．

Turhan ら [25] は，10 個の異なるソフトウェア開発プロジェクトによるプロジェクト間予測を実施した．予測モデルとしてベイズ識別器を用い，モデルの学習データとして，自プロジェクト以外の全プロジェクトのデータを混在したものを用いた．実験の結果，プロジェクト間予測では，適合率が大幅に悪化するものの，ランダム予測よりは優れていることを示した．また，k 近傍法を用いて学習データに用いるプロジェクトの数を減らすことで，適合率が改善することを示した．

木浦ら [12] は，複数のプロジェクトのデータを用いた集団学習の効果を評価した．集団学習の方法として，Turhan らのように他プロジェクトのデータを混在させてモデル構築する方法，及び，プロジェクトごとにモデルを構築し，各モデルの出力の平均値を判別結果とする方法の二通りを行っている．実験の結果，ロジスティック回帰分析においては，いずれの方法も効果があることを示している．

これらの先行研究を受けて，本論文では，次節及び及び **3.** で述べるとおり四つのリサーチクエスチョンを設けた．

2.4 リサーチクエスチョンの設定

Zimmermann ら [28] が示したように，異種プロジェクト間で十分な精度が得られない原因は，学習データと予測対象データの性質が異なっていることに起因する．そこで，RQ1 では学習データにオーバフィッティングしにくいランダムフォレストを使うことで，精度向上を図る．また，RQ2 では，データセットに対する前処理を行い，学習データと予測対象データの分布を近づけることで精度向上を図る．

RQ3 と RQ4 は，プロジェクト間予測の精度向上の先行研究の改良または拡張である．RQ3 では，Sato らの研究 [24] の改良として，予測対象プロジェクトの目的変数の情報を用いずに学習データを選定し，類似するプロジェクト間で予測を行うことで精度向上を図る．また，RQ4 では，木浦ら [12] の実験を拡張し，より多くのプロジェクトのデータを用いて集団学習の効果を確認する．

2.5 予測精度の評価方法

Briand ら [2] の研究をはじめ，fault-prone モジュール判別の研究では適合率と再現率，及びその調和平均である F1 値がよく用いられてきた [11], [15]．しかし，これらの尺度は，モデルの性能のみならず，データセットの性質に大きな影響を受けてしまう点が問題である [7]．一般に，fault を含むモジュールが fault を含まないモジュールよりも少ないと，これらの尺度の値は低下する傾向にある．

また，ソフトウェア開発現場では，fault あり/なしの判別結果よりもむしろ，fault を含む度合い（期待値や確率）を知ることが必要である．ロジスティック回帰分析やランダムフォレストなどの判別モデルでは，fault を含む度合いを出力できるため，fault を含む可能性のより高いモジュール（例えば上位 10%）だけを重点的にテストする，といったことが可能である．このような使い方を想定した場合には，適合率，再現率，F1 値などの尺度ではモデルの性能を評価するには不十分である．

そこで，本論文では，適合率，再現率に代わるより適切な指標として，Alberg Diagram の AUC (Area Under the Curve) を用いた評価を行う．Alberg Diagram

の AUC の詳細は **4.2** で述べる．

3. リサーチクエスチョン

本論文では，他プロジェクトのデータを利用してモデル構築・判別を行う上で有用と考えられる手法を調査するため，四つのリサーチクエスチョン（RQ）を実験的に検証する．以降，各 RQ の概要と動機を整理するとともに，RQ を評価するための実験方法について述べる．

RQ1　ランダムフォレストの効果

[概要・動機] RQ1 では，プロジェクト間の fault-prone モジュール判別において，集団学習法の 1 種であるランダムフォレスト法の効果を確かめる．

RQ1 の新規性，及び，有効性は，Lessmann らの研究 [13] においてプロジェクト内予測で有効であると述べられているランダムフォレストの有効性を，プロジェクト間予測において評価している点である．

[実験方法] 全てのプロジェクトの組合せについて，一方のプロジェクトをフィットデータ（モデル学習用データ）として用い，他方をテストデータ（モデル評価用データ）に用いてランダムフォレストの性能評価を行う．比較対象として，fault-prone モジュールの予測研究に従来よく用いられているロジスティック回帰分析を採用する．

RQ2　学習データ正規化の効果

[概要・動機] プロジェクト間で判別を行う場合，システム種別や規模等の違いによってフィットデータとテストデータ間のメトリックス分布に違いが生じやすいと考えられる．一般に，判別モデルは，フィットデータとテストデータ間におけるメトリックスの分布が，ある程度同じ傾向であることを前提としている．そのため，メトリックス分布の違いはモデルの性能低下を引き起こす可能性が高い．そこで，判別モデルを構築する前段階の正規化処理として，対数変換や Z-score 変換処理を施すことによってプロジェクト間での説明変数の分布を近づけることで，判別精度が向上する可能性がある．

[実験方法] フィットデータとテストデータ間の各説明変数の分布を近づけるために各データセットの各説明変数に対して，前処理として正規化（RQ2-1. 対数変換，RQ2-2. Z-score 変換，RQ2-3. 対数変換後に Z-score 変換処理）を行ってから，モデル構築・判別を行う．判別モデルにはランダムフォレストを用いる．

RQ3　学習データ選択の効果

[概要・動機] 他プロジェクトのデータが複数ある場合，予測対象プロジェクトの特徴に合わせて適切な学習データを選ぶことができれば，高い精度が得られる可能性がある．

そこで，RQ3 では，高い判別精度が得られる学習データ（プロジェクト）の選定方法を明らかにする．本論文では，三つの選定方法について評価する．

一つ目（RQ3-1）は，自身をうまく予測できるデータセットをフィットデータとして用いる方法である．プロジェクト内の予測精度が高いデータセットは，プロジェクト間の予測に用いても高い精度が得られるという仮説に基づいている．

二つ目（RQ3-2）は,「説明変数間の関係」が類似するプロジェクトをフィットデータとして用いる方法である．ただし，予備実験の結果，全ての説明変数を対象とした場合には効果がなかったため，フィットデータにおいて fault の有無に大きく寄与する変数だけに着目して，説明変数間の関係を数値化し，類似度を算出する．

三つ目（RQ3-3）は,「プロジェクトの特徴を表すメトリックスが似ているならば，プロジェクト間予測はうまくいく」という仮説に基づく．本論文では，プロジェクトの特徴を表すメトリックスとして，計測が容易なメトリックスであるソフトウェア規模（ソースコード総行数）とモジュール粒度（1 ファイル当りの行数）に着目する．

[実験方法]（RQ3-1）テストデータ内で 2-fold 交差検証を 10 回試行し，その際の判別精度の平均値とそのフィットデータを用いて他のプロジェクトを判別した際の判別精度の平均値との相関を調べる．RQ3 でも同様に，判別モデルにはランダムフォレストを用いる．

（RQ3-2）「説明変数間の関係の類似度」を定義するにあたって，各フィットデータについて下記手順 1 から 4 によって類似度を求める．(1) 説明変数の中から fault の判別に寄与する変数のみを選ぶ．具体的には，フィットデータの各説明変数と目的変数にあたる fault の有無との相関を調べ，相関の強い説明変数上位三つを選択する．(2) 選ばれた三つの説明変数間の相関係数（これらを R1, R2, R3 とする）を求める．(3) フィットデータで選択された説明変数と同じものをテストデータでも選択し，それら説明変数間の相関係数（これらを Q1，Q2，Q3 とする）を求める．(4) 手順 2, 3 で求められた二つのベクトル (R1, R2, R3) と (Q1,

表 1　NASA データセットの概要
Table 1　Summary of NASA data.

データセット	モジュール数	コード行数 [KSLOC]	モジュール粒度 [KSLOC/module]	バグモジュール率 [bug module/module]
CM1	505	16,903	33.47	0.095
JM1	10,878	457,177	42.03	0.193
KC1	2,107	42,963	20.39	0.125
MC1	4,625	66,583	14.40	0.015
MC2	161	6,134	38.10	0.323
MW1	403	8,341	20.70	0.067
PC1	1,059	25,922	24.48	0.072
PC2	4,505	26,863	5.96	0.005
PC3	1,511	36,473	24.14	0.106
PC4	1,347	30,055	22.31	0.132
PC5	15,414	161,695	10.49	0.033

Q2, Q3) のユークリッド距離を，プロジェクト間の類似度の尺度として用いる．

（RQ3-3）ソフトウェア規模 s とモジュール粒度 m の組 (s, m) をプロジェクトの特徴ベクトルとする．特徴ベクトル間のユークリッド距離が最も近いデータセットを選定し，モデル構築・判別を行う．

RQ4　**複数データセットによる集団学習の効果**

[概要・動機] 複数プロジェクトのデータが利用できる場合，集団学習により予測精度を高められる可能性がある．本論文では，各データセットから構築された判別モデルを用いて集団学習する方式（RQ4-1）と，先にデータを結合してから一つの判別モデルを作る方式（RQ4-2）の両方について評価する．

[実験方法] （RQ4-1）では，判別対象プロジェクト以外の全てのデータセットでそれぞれ判別モデルをランダムフォレストによって構築し，それらの出力の平均値を判別結果として用いる．

（RQ4-2）では，モデルの学習データとして，自プロジェクト以外の全てのプロジェクトのデータを結合してから一つの判別モデルを構築する．

RQ1～4 **の位置付け**

一般に，fault-prone モジュール判別においては，(a) フィットデータの選定，(b) データに対する前処理，(c) 適切なモデルの採用，(d) モデルの学習，といった手続きが考えられる．RQ1～4 では，各手続きにおける精度向上策の効果を確認していると位置づけられる．

4. 実　　験

本実験の目的は，RQ1～4 を実験的に検証し，プロジェクト間での fault-prone モジュールの判別精度の向上に有効な手法を明らかにすることである．本実験結果の一般性を向上させるため，NASA が公開しているデータセット（11 プロジェクト）に加えて，オープンソースソフトウェアプロジェクトから筆者らが収集したデータセット（7 プロジェクト）を用いて同様の結果が見出されるかの確認を行う．

4.1　データセット

実験には NASA IV & V Facility Metrics Data Program [21] で公開されているデータセットを用いた．本論文では，公開されているもののうちから C 言語で記述された 11 個のデータセットを実験対象とした．実験で用いたデータセットの概要を表 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目的変数はモジュールのバグの有無とし，説明変数はモジュールのプロダクトメトリックスとした．

また，Open Source Software（以降，OSS と表記）の Eclipse の三つのサブプロジェクト（Platform, JDT, PDE）や Mylyn, Chrome, Firefox, Netbeans それぞれから筆者らがバグの有無，及び，メトリックスを計測し，計七つのデータセットを作成した．データセットの概要については表 2 に示す．

紙面の都合上，使用したメトリックスの詳細は割愛するが，NASA データでは，11 プロジェクトで共通に計測されている 20 個の説明変数（ソースコード行数，条件分岐数，サイクロマティック数，Halsted のソフトウェアサイエンス尺度など）を用いた．また，OSS データでは，オブジェクト指向メトリックスを中心とする 21 種類のプロダクトメトリックスを Understand [26] を用いて計測した．fault の計測には，SZZ アルゴリズム [23] を用いた．なお，NASA データと OSS データの両方に共通するメトリックスがほとんどないため，本論文では，NASA データによる OSS データの予測

表 2　OSS プロジェクトから収集したデータセットの概要
Table 2　Summary of collected data from OSS projects.

データセット	バージョン	モジュール数	コード行数 [KSLOC]	モジュール粒度 [KSLOC/module]	バグモジュール率 [bug module/module]
Platform	3.2	6,776	1,347,235	198.82	0.155
PDE	3.2	452	55,749	123.34	0.179
JDT	3.2	4,025	994,067	246.97	0.097
Mylyn	3.0	1,502	241,081	160.51	0.413
Netbeans	5.0	9,332	1,999,316	214.24	0.146
Firefox	2.0	6,468	3,446,619	532.87	0.071
Chrome	4.0	4,185	1,807,778	431.97	0.011

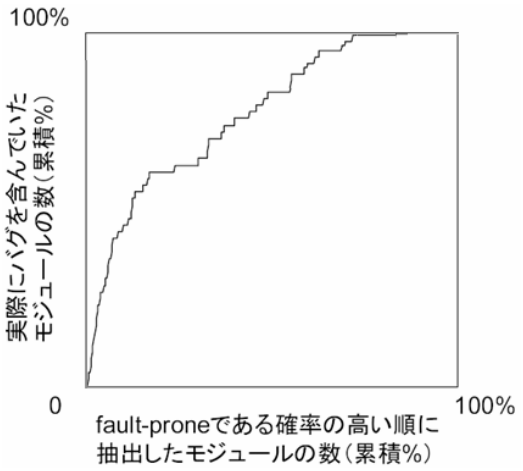

図 1　Alberg Diagram の例
Fig. 1　Example of Alberg Diagram.

（あるいはその逆）は行わない．

4.2 評価指標

本論文では fault-prone モジュール判別モデルの評価指標として Alberg Diagram [22] の AUC (Area Under the Curve) を用いる．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 1）．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 1 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．

この評価指標の第一の利点は，AUC の値が fault あり/なしの判別境界と独立なため，データセットの偏りに影響を受けにくいことである．また第二の利点は，AUC が高いモデルほど，上位モジュールにより多くバグを含むことになるため，開発現場での利用に即していることである（以降，本論文では，Alberg Diagram の AUC のことを，単に AUC と記す）．

表 3　ランダムフォレストによるモデル構築での判別結果（NASA）
Table 3　Result of random forest (NASA).

		テストデータ										
		CM1	JM1	KC1	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
フィットデータ	CM1	—	0.65	0.73	0.85	0.62	0.69	0.66	0.81	0.70	0.61	0.89
	JM1	0.73	—	0.71	0.85	0.62	0.80	0.69	0.81	0.73	0.70	0.90
	KC1	0.74	0.63	—	0.86	0.57	0.77	0.64	0.84	0.71	0.64	0.91
	MC1	0.64	0.63	0.66	—	0.60	0.75	0.62	0.78	0.69	0.67	0.84
	MC2	0.71	0.59	0.69	0.79	—	0.73	0.58	0.85	0.56	0.50	0.89
	MW1	0.71	0.66	0.70	0.88	0.64	—	0.62	0.81	0.71	0.62	0.85
	PC1	0.73	0.63	0.70	0.88	0.58	0.69	—	0.87	0.76	0.75	0.86
	PC2	0.77	0.65	0.74	0.88	0.63	0.82	0.68	—	0.73	0.63	0.89
	PC3	0.76	0.64	0.73	0.89	0.60	0.77	0.82	0.84	—	0.72	0.88
	PC4	0.58	0.60	0.69	0.92	0.53	0.62	0.73	0.86	0.72	—	0.88
	PC5	0.68	0.59	0.71	0.88	0.61	0.70	0.65	0.83	0.68	0.65	—

表 4　ロジスティック回帰によるモデル構築での判別結果（NASA）
Table 4　Result of logistic regression (NASA).

		テストデータ										
		CM1	JM1	KC1	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
フィットデータ	CM1	—	0.53	0.74	0.75	0.53	0.74	0.66	0.51	0.67	0.58	0.77
	JM1	0.73	—	0.73	0.89	0.61	0.80	0.70	0.88	0.75	0.73	0.91
	KC1	0.79	0.55	—	0.61	0.59	0.78	0.56	0.20	0.61	0.44	0.73
	MC1	0.76	0.57	0.73	—	0.53	0.79	0.79	0.84	0.71	0.74	0.84
	MC2	0.76	0.59	0.70	0.77	—	0.69	0.62	0.75	0.63	0.55	0.87
	MW1	0.75	0.57	0.63	0.77	0.49	—	0.69	0.78	0.74	0.58	0.64
	PC1	0.73	0.52	0.49	0.84	0.52	0.80	—	0.78	0.78	0.75	0.66
	PC2	0.26	0.35	0.25	0.19	0.38	0.24	0.34	—	0.31	0.42	0.07
	PC3	0.76	0.53	0.58	0.90	0.57	0.84	0.75	0.86	—	0.70	0.74
	PC4	0.50	0.53	0.68	0.76	0.41	0.42	0.60	0.85	0.63	—	0.71
	PC5	0.74	0.60	0.73	0.88	0.63	0.77	0.70	0.88	0.73	0.70	—

4.3 実験結果

(RQ1) ランダムフォレストの効果

ランダムフォレストでモデル構築した結果を表 3，ロジスティック回帰分析でモデル構築した結果を表 4 にそれぞれ示す．評価値は AUC で，小数第 3 位を四捨五入して小数第 2 位で表記している．また，箱ひげ図で示したものを図 2 に示す．

ランダムフォレストでモデル構築して判別した場

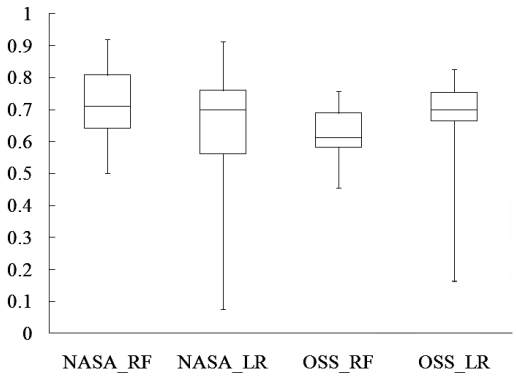

図 2　RQ1：各モデル (RF/LR) の性能比較
Fig. 2　RQ1: Result of random forest v.s. logistic regression analysis.

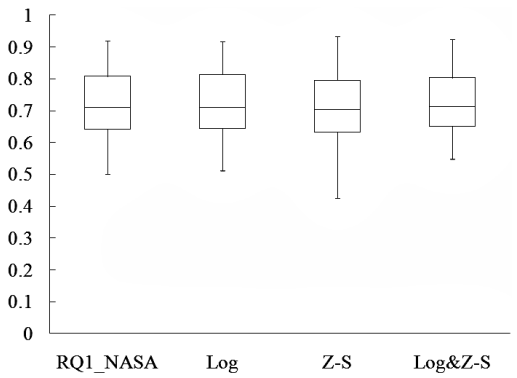

図 3　RQ2：正規化処理の効果 (NASA)
Fig. 3　RQ2: Effect of normalization of metrics (NASA).

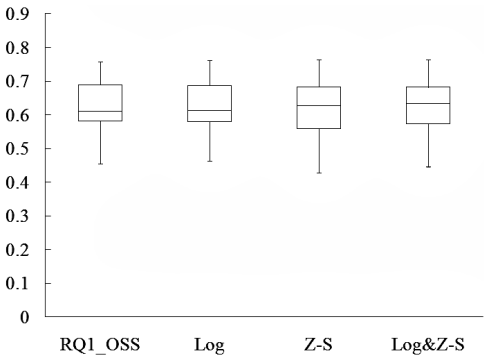

図 4　RQ2：正規化処理の効果 (OSS)
Fig. 4　RQ2: Effect of normalization of metrics (OSS).

合，AUC の値にして最小値 0.50，最大値 0.92，平均値 0.72 が得られた．一方，ロジスティック回帰分析でモデル構築して判別した場合，最小値 0.07，最大値 0.91，平均値 0.65 が得られた．これより，ランダムフォレストでのモデル構築は安定して fault-prone モジュールを判別できている一方で，ロジスティック回帰分析でのモデル構築は，データセットの組合せによっては全く判別に至らない結果を含んでいることが分かる．特に，プロジェクト PC2 をフィットデータにした場合に，ランダムに判別するよりも極端に悪い判別結果が得られている．OSS 開発プロジェクトから計測したデータセットの実験結果ついては，NASA データセットの場合と同様に，ランダムフォレストを用いることで安定して fault-prone モジュールを判別できた（紙面の都合上，表は割愛する）．ランダムフォレストの場合，AUC の値で最小値 0.46，最大値 0.76，平均値 0.62 が得られた．一方，ロジスティック回帰分析の場合，最小値 0.16，最大値 0.82，平均値 0.67 が得られた．最大値，平均値においてロジスティック回帰分析でのモデル構築の方が優位である．しかしながら，最小値に注目すると，ロジスティック回帰分析では 0.16 と極端に悪い値となっており，開発現場で用いることは危険であり，ランダムフォレストの方が実用的であると思われる．

(RQ2) 学習データ正規化の効果

NASA データにおけるランダムフォレストの実験結果の箱ひげ図を図 3 に示す（RQ1 においてランダムフォレストの有用性が確認されたので，以降，紙面の都合上，ランダムフォレストの結果のみ示す）．各正規化処理においてどの程度の判別精度の向上が見られたのかを比較するために，RQ1 で行った表 3 の結果（RQ1_NASA）も付記している．図 3 から分かるように，全ての実験条件において若干の差異は認められるが，顕著な変化は確認できなかった．同様に，OSS のデータセットにおいて顕著な変化は認められなかった（図 4）．このことから，プロダクトメトリックスを正規化処理することによる効果は期待できないことが分かった．

(RQ3) 学習データ選択の効果

[RQ3-1 自身をうまく予測できるデータセット]

図 5 に NASA データセットにおける交差検証の結果とプロジェクト間予測の結果の関係を示す．X 軸は各プロジェクトの交差検証の結果を，Y 軸は各プロジェクトをフィットデータとして用いて他プロジェクトを予測した結果を平均値で示している．図 6 は OSS

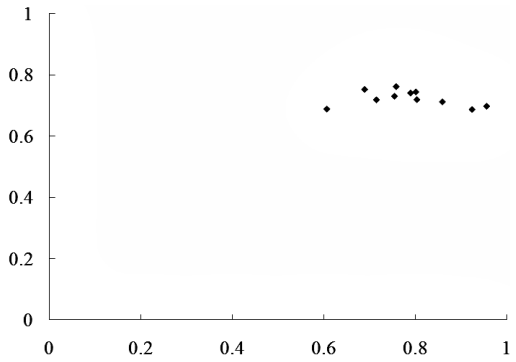

図 5　RQ3-1：交差検証による予測精度とプロジェクト間予測による予測精度の関係 (NASA)

Fig. 5　Relationship between cross-validation prediction and cross-project prediction (NASA).

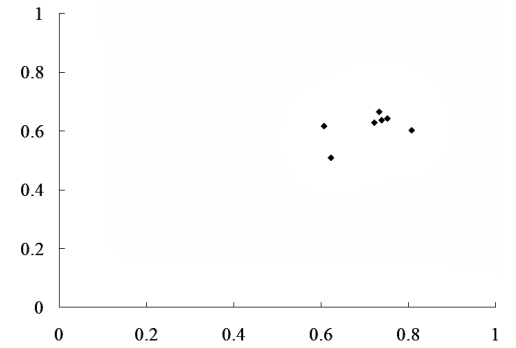

図 6　RQ3-1：交差検証による予測精度とプロジェクト間予測による予測精度の関係 (OSS)

Fig. 6　Relationship between cross-validation prediction and cross-project prediction (OSS).

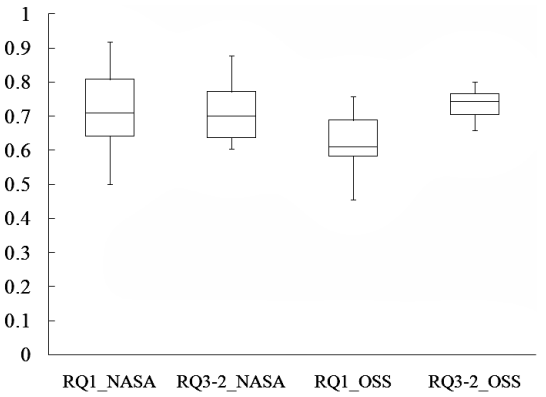

図 7　RQ3-2：説明変数間の関係の類似性に基づく予測結果

Fig. 7　RQ3-2: Prediction result based on similarity of relation between explanatory variables.

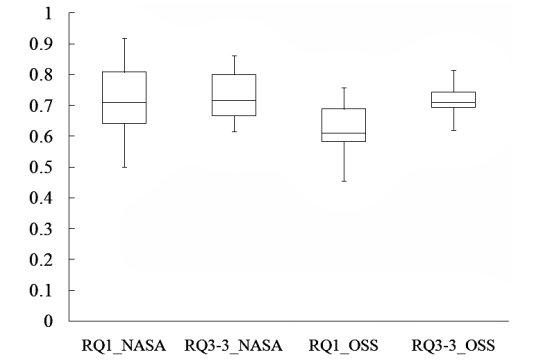

図 8　RQ3-3：プロジェクトメトリックスの類似性に基づく予測結果

Fig. 8　Prediction result based on similarity between project metrics.

データセットの場合の結果である．

図 5，図 6 共に X 軸の値と Y 軸の値の相関は弱い．X 軸の値が大きいプロジェクトは「自身をうまく予測できるデータセット」であるが，そのようなデータセットをフィットデータとしても高い予測精度が見込めるわけではないことを示している．

[RQ3-2 説明変数間の関係の類似性]

NASA データセット及び OSS データセットの実験結果の箱ひげ図を図 7 に示す．図 7 から見て取れるように，どちらのデータセットにおいても下限値が向上しているのがはっきりと確認できる．特に，OSS 開発のデータセットにおいては，RQ1 の結果と比較すると最大値，中央値が共に大きく向上しており，「RQ1 の判別結果の平均値と，説明変数間の関係の類似性に基づく判別結果の平均値に差はない.」という帰無仮説に対して Welch の t 検定を用いた結果，p 値は 0.00007 となり，有意水準 1%で有意な差が認められた．そのため，説明変数間の関係の類似性による学習データの選択は精度向上が期待できると考える．なお，Welch の t 検定を用いた理由は，比較対象である判別結果の分布について等分散性を考慮しなくても用いることができるためである．

[RQ3-3 プロジェクトメトリックスの類似性]

NASA データセット及び OSS データセットの実験結果の箱ひげ図を図 8 に示す．

図 8 から確認されるように，NASA 及び OSS のデータセット共に RQ3-2 の結果と同様となった．OSS 開発のデータセットにおいては，「RQ1 の判別結果の平均値と，プロジェクトメトリックスの類似性に基づく判別結果の平均値に差はない.」という帰無仮説に

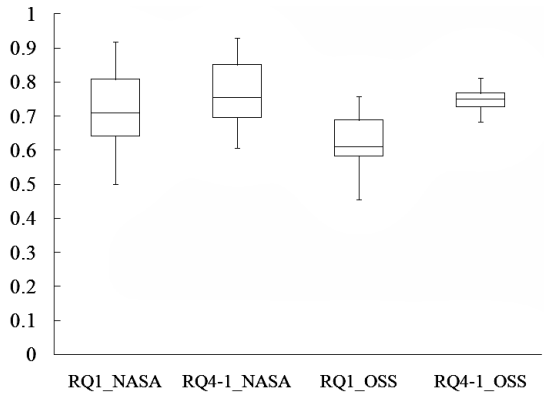

図 9　RQ4-1：複数のデータセットによる判別結果の統合
Fig. 9　RQ4-1: Prediction by combined outputs of models each built from different dataset.

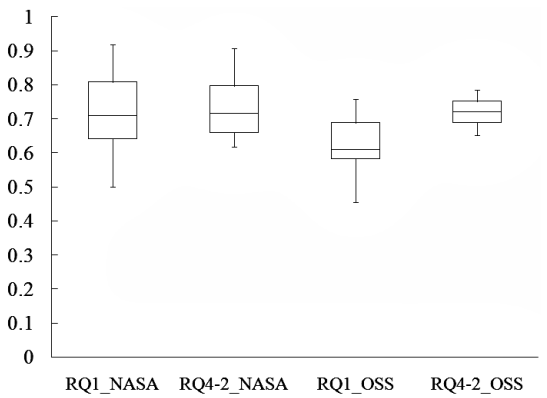

図 10　RQ4-2：結合データセットに基づく判別
Fig. 10　RQ4-2: Prediction by a model built from a combined dataset.

対して Welch の t 検定を用いた結果，p 値は 0.00313 であり，有意水準 1%で統計的に有意な差であると認められた．そのため，プロジェクトメトリックスの類似性による学習データの選択は精度向上に期待できると考える．

(RQ4) 複数データセットによる集団学習の効果

[RQ4-1 各プロジェクトからの判別の統合]

NASA 及び OSS のデータセットの実験結果を図 9 に示す．同図より，NASA データセットにおいては RQ1 の結果と比較して最小値が向上していることが確認できた．また，OSS データセットにおいては RQ1 の結果と比較して全体的に判別精度が向上しており，「RQ1 の判別結果の平均値と，各フィットデータからの判別結果の平均値に差はない.」という帰無仮説に対して Welch の t 検定を用いた結果，p 値は 0.00001 であり，有意水準 1%で統計的に有意な差であると認められた．そのため，各プロジェクトからの判別の統合は精度向上に期待できると考える．

[RQ4-2 統合されたデータセットからの判別]

NASA 及び OSS のデータセットの実験結果を図 10 に示す．

同図より，NASA データセットにおいては RQ1 の結果と比較して最小値が向上していることが確認できた．また，OSS データセットにおいては RQ1 の結果と比較して全体的に判別精度が向上していた.「RQ1 の判別結果の平均値と，判別対象プロジェクト以外のデータセットを統合して一つのプロジェクトとしてモデル構築を行い判別した結果の平均値に差はない.」という仮説のもと，Welch の t 検定を行った結果，p 値は 0.00042 となり，有意水準 1%で有意な差が認められた．そのため，統合されたデータセットからの判別は精度向上に期待できると考える．

5.　実験結果のまとめと考察

本章では，**4.** の実験結果をまとめるとともに，結果に対する考察を述べる．実験を通して分かったことは，以下のとおりである．

・RQ1：ロジスティック回帰分析における AUC の最小値は NASA データで 0.07，OSS データで 0.16 であり，ランダム予測（AUC = 0.5）と比べても著しく悪くなる場合があることが分かった．このことから，プロジェクト間予測にロジスティック回帰分析を使うことは極めて危険であるといえる．

・RQ1：一方，ランダムフォレストは，AUC の最小値が NASA データで 0.5，OSS データで 0.46 であった．このことから，ランダムフォレストは最悪ケースでもランダム予測（AUC = 0.5）と同程度の精度となることから，ロジスティック回帰分析より実用的である．

・RQ1：ランダムフォレストはオーバフィッティングしにくい特徴をもつため，ロジスティック回帰分析よりも判別精度が高くなったと考える．

・RQ2：説明変数に対して対数変換や Z-score 変換といった正規化処理を施したとしても，プロジェクト間の判別精度が大きく改善することは望めない．

・RQ2：ランダムフォレストは非線形モデルである分類木を用いており，データの正規性を仮定していないことから，効果がなかったと考える．なお，紙面の

都合上掲載していないが，ロジスティック回帰分析に対しては，正規化の効果が見られた（ただし，ランダムフォレストの精度には及ばなかった）．

・RQ3-1：プロジェクト内の判別精度と，他プロジェクトを予測した場合の判別精度の間には関係が見られなかった．これは，たとえ自身をうまく予測できるデータセットを用いたとしても，予測対象のプロジェクトの特徴が異なれば，うまく予測できないためであると考える．このことから，あるプロジェクトにおいて判別がうまくいっているからといって，その判別モデルを他プロジェクトに使う理由にはならない．

・RQ3-2：説明変数間の関係が似ている他プロジェクトがあれば，そのデータセットを用いることは有望である．

・RQ3-3：同様に，プロジェクトの特徴（規模，モジュール粒度）の類似性が確認できるならば，そのデータセットを用いることは有望である．

・RQ4-1：複数のデータセットそれぞれで判別モデルを構築し，判別した結果の平均値を用いることで判別精度の向上が望める．

・RQ4-2：同様に，複数のデータセットを統合して一つのフィットデータとして fault-prone モジュールを判別することで判別精度の向上が望める．

・RQ4-1 及び RQ4-2：複数のデータをフィットデータとして用いたため，一つデータセットを用いて判別するよりもオーバフィッティングを避ける効果が大きかったと考える．

以上の結果・考察より，他プロジェクトのデータを用いて fault-prone モジュール判別モデルを構築する際には，ランダムフォレストが有望であり，メトリックスに対する正規化処理は行う必要がないことが分かった．更に，類似プロジェクトの選択若しくは集団学習を行うことが効果的であることが分かった．そこで，以降では類似プロジェクトの選択と集団学習を組み合わせた方法についても検討する．

ここでは，RQ3-2（または RQ3-3）の方法により類似プロジェクトを三つ選択してそれぞれ判別モデルを構築し，RQ4-1 の方法により，三つのモデルの出力の平均値を判別結果として用いることとした．RQ3-2 と RQ4-1 を組み合わせた結果を図 11 に，RQ3-3 と RQ4-1 を組み合わせた結果を図 12 に示す．いずれの場合にも，RQ3-2，RQ3-3，RQ4-1 単独で実施した場合とほとんど変わらない結果が得られた．このことから，現時点では，組合せによる精度の向上は見込めないといえる．ただし，組合せによる方法は有望であると考える．それは，類似したプロジェクトの選択（RQ3-2 または RQ3-3）によって予測に寄与しないプロジェクトが除去され，RQ4-1 の集団学習において精度向上に役立つプロジェクトのみが利用されるからである．そのため，より多くの他プロジェクトのデータが利用できる場合には，効果を発揮する可能性があり，その評価は今後の課題である．

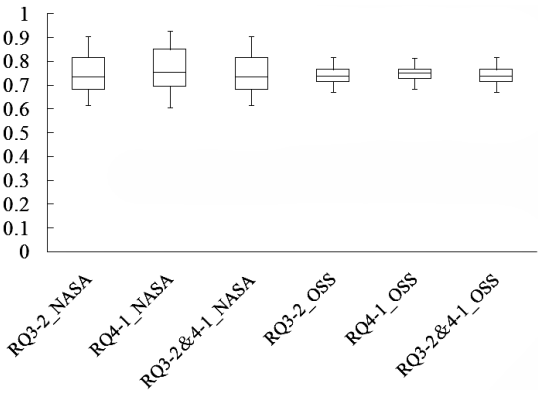

図 11　RQ3-2 と RQ4-2 の組合せによる結果
Fig. 11　Result of Combination of RQ3-2 & RQ4-2.

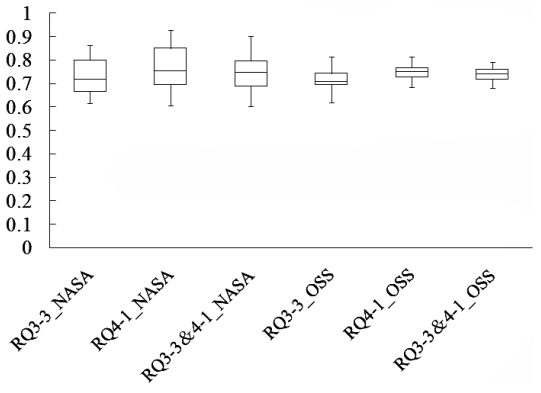

図 12　RQ3-3 と RQ4-2 の組合せによる結果
Fig. 12　Result of Combination of RQ3-3 & RQ4-2.

6. 本論文の妥当性

本章では，本論文の妥当性について議論する．

[内的妥当性] 本実験における OSS プロジェクトの fault の計測には SZZ アルゴリズム [23] を用いた．SZZ アルゴリズムは，多くの fault-prone モジュール判別の研究 [15], [17] で使われているアルゴリズムであるが，CVS のログコメント中に記録されている fault しか計

測できないという制約がある．また，他の制約として，ユークリッド距離を（RQ3-2）（RQ3-3）での類似度として用いていることが挙げられる．今後の課題として，コサイン距離などの他の類似度を用いることも検討する．

[外的妥当性] 本研究の結果は，NASA が公開しているデータセットと OSS から計測したデータセットの 2 群からなる計 18 プロジェクトの実験から得られたものであるものの，結果の信頼性を向上させるためには他のデータセットを用いて実験を追加する必要がある．また，本実験では，判別モデルにおいて一般的に利用されているメトリックス（NASA データセット：主に Cyclomatic 複雑度や Halstead 複雑度，OSS データセット：主に CK メトリックス）を用いているので，（特殊なメトリックスを用いるよりは）一般性のある結果であると考えるが，一般性をより向上させるためには今回用いた以外のメトリックスによる実験を追加することが望ましい．

7. むすび

本論文では，プロジェクトをまたがる fault-prone モジュールの判別精度向上に有用な手法を明らかにするために，NASA が公開しているデータセットと OSS から計測したデータセットの 2 群からなる計 18 のデータセットを用いて，四つのリサーチクエスチョンを実験的に検証した．

実験の結果，(1) ランダムフォレストはプロジェクトをまたがる判別に効果を発揮する，(2) 学習データに対するして前処理（正規化）の効果はない，(3) データセット間に類似性が確認できるならば，高い精度での判別が期待できる．(4) 複数のプロジェクトのデータを用いた集団学習の効果はある，といった教訓が得られた．

今後の課題として，類似プロジェクトの選択と複数データセットを用いた集団学習の組合せ方法について，更なる検討・評価を進めることが挙げられる．

謝辞 本研究の一部は，文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究費補助金（基盤研究 (C)：課題番号 22500028）に基づいて行われた．

文　献

[1] L. Breiman, “Random forests,” Mach. Learn., vol.45, pp.5–23, 2001.

[2] L.C. Briand, W.L. Melo, and J. Wust, “Assessing the applicability of fault-proneness models across object-oriented software projects,” IEEE Trans. Softw. Eng., vol.28, no.7, pp.706–720, 2002.

[3] T.G. Dietterich, “Machine-learning research: four current directions,” AI magazine, vol.18, pp.97–136, 1997.

[4] J.P. Ganesh and B.D. Joanne, “Empirical analysis of software fault content and fault proneness using Bayesian methods,” IEEE Trans. Software Engineering, vol.33, no.10, pp.675–686, 2007.

[5] A.R. Gray and S.G. MacDonell, “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.

[6] T. Gyimothy, R. Ferenc, and I. Siket, “Empirical validation of object-oriented metrics on open source software for fault prediction,” IEEE Trans. Softw. Eng., vol.31, no.10, pp.897–910, 2005.

[7] 亀井靖高，まつ本真佑，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,” 情処学論，vol.48, no.8, pp.2651–2662, Aug. 2007.

[8] T.M. Khoshgoftaar and E.B. Allen, “Modeling software quality with classification trees,” Recent Advances in Reliability and Quality Engineering, Hoang Pham Editor, World Scientific, Singapore, pp.247–270, 1999.

[9] T.M. Khoshgoftaar and E.B. Allen, “Predicting software modules that will need rework,” Proc. 3rd Workshop on Empirical Studies of Software Maintenance (WESS ’98), pp.49–50, 1998.

[10] T.M. Khoshgoftaar, S. Naeem, and S. Nandini, “An empirical study of predicting software faults with case-based reasoning,” Software Quality Control, vol.14, no.2, pp.85–111, 2006.

[11] S. Kim, E.J. Whitehead, Jr., and Y. Zhang, “Classifying software changes: Clean or buggy?” IEEE Trans. Softw. Eng., vol.34, no.2, pp.181–196, 2008.

[12] 木浦幹雄，まつ本真佑，亀井靖高，門田暁人，松本健一，“異なるプロジェクト間における Fault-Prone モジュール判別の精度評価,” ソフトウェア工学の基礎 XIV，日本ソフトウェア科学会 FOSE2007, pp.131–136, 2007

[13] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, “Benchmarking classification models for software defect prediction: A proposed framework and novel findings,” IEEE Trans. Softw. Eng., vol.34, no.4, pp.485–496, 2008.

[14] P.L. Li, J. Herbsleb, M. Shaw, and B. Robinson, “Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc.,” Proc. 28th Int’l Conf. Software Engineering (ICSE’06), pp.413–423, 2006.

[15] O. Mizuno and T. Kikuno, “Prediction of fault-prone

software modules using a generic text discriminator," IEICE Trans. Inf. & Syst., vol.E91-D, no.4, pp.888–896, April 2008.
[16] A. Mockus and D.M. Weiss, "Predicting risk of software changes," Bell Labs Tech. J., vol.5, no.2, pp.169–180, 2000.
[17] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," Proc. Int'l Conf. Software Engineering (ICSE'08), pp.181–190, 2008.
[18] J. Munson, and T. Khoshgoftaar, "The detection of fault-prone programs," IEEE Trans. Softw. Eng., vol.18, no.5, pp.423–433, 1992.
[19] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," Proc. 28th Int'l Conf. Software Engineering (ICSE'06), pp.452–461, 2006.
[20] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," Proc. Int'l Conf. Software Engineering (ICSE'05), pp.284–292, 2005.
[21] NASA IV & V Facility Metrics Data Program, http://mdp.ivv.nasa.gov/
[22] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," IEEE Trans. Softw. Eng., vol.22, no.12, pp.886–894, 1996.
[23] J. Sliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?," Proc. Int'l Conf. Mining Software Repositories (MSR'05), pp.1–5, 2005.
[24] S. Sato, A. Monden, and K. Matsumoto, "Evaluating the applicability of reliability prediction models between different software," Proc. Int'l Workshop on Principles of Software Evolution (IWPSE'02), pp.97–102, 2002.
[25] B. Turhan, T. Menzies, A. Bener, and J. Distefano, "On the relative value of cross-company and within-company data for defect prediction," Empirical Software Engineering, vol.14, no.5, pp.540–578, 2009.
[26] Understand, Scientific Toolworks, Inc. http://www.scitools.com/
[27] F. Xing, P. Guo, and M.R. Lyu, "A novel method for early software quality prediction based on support vector machines," Proc. Int'l Symposium on Software Reliability Engineering (ISSRE'05), pp.213–222, 2005.
[28] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction," Proc. 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.

（平成 23 年 6 月 5 日受付，10 月 3 日再受付）

藏本　達也

平 13 防衛大学校・工卒．同年航空自衛隊に任官．平 19 同大学校理工学研究科前期課程了．現在，奈良先端科学技術大学院大学情報科学研究科博士後期課程在籍．航空自衛隊所属の社会人学生．

亀井　靖高　（正員）

平 17 関西大・総合情報卒．平 21 奈良先端科学技術大学院大学情報科学研究科博士後期課程了．同年日本学術振興会特別研究員 (PD)．平 22 カナダ Queen's 大学博士研究員．平 23 九州大学大学院システム情報科学研究院助教．博士（工学）．ソフトウェアメトリクス，マイニングソフトウェアリポジトリの研究に従事．情報処理学会，IEEE 各会員．

門田　暁人　（正員）

平 6 名大・工・電気卒．平 10 奈良先端科学技術大学院大学情報科学研究科博士後期課程了．同年同大学助手．平 16 同大学助教授．平 19 同大学准教授．平 15〜16 Auckland 大学客員研究員．博士（工学）．ソフトウェアメトリクス，ソフトウェアプロテクション，ヒューマンファクタの研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

松本　健一　（正員）

昭 60 阪大・基礎工・情報工学卒．平元同大大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平 5 奈良先端科学技術大学院大学助教授．平 13 同大学教授．工博．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集/利用支援の研究に従事．情報処理学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．