Vol.2018-SE-200 No.7
2018/12/3

ランダムフォレストを用いたソフトウェア不具合修正予測におけるインパクトスケールの有用性

野中 誠[1]　中嶋 久彰[2]　伊藤 雅子[2]　山田 弘隆[2]

概要：本稿では，筆者らの組織で独立に開発された二つの製品システムをそれぞれ構成する二つのソフトウェアを対象に，統合・評価段階において検出された不具合修正により変更が生じるファイルをランダムフォレストにより予測した結果を述べる．筆者らの組織で運用している開発環境基盤から得られたインパクトスケールを含むプロダクトメトリクスとその差分を測定し，不具合修正予測の説明変数とした．分析の結果，一方のソフトウェアではインパクトスケールとその差分を含む方がそうでない場合に比べて予測性能が上回る結果が示されたが，他方のソフトウェアではそうでない結果が示された．ただし，インパクトスケールとその差分を含む予測結果はいずれも先行研究に比べて高い予測性能であったことから，インパクトスケールの有用性を確認することができた．

Makoto Nonaka[1]　Hisaaki Nakajima[2]　Masako Ito[2]　Hirotaka Yamada[2]

1. はじめに

エンハンス開発，すなわち製品システムを構成する既存ソフトウェアを再利用して新しい製品システムを開発する場合は，ソフトウェアの構造が複雑化し，大規模化する傾向にある．そのため，ソフトウェアの構造を理解したり，ソフトウェア変更による影響範囲を把握したりするのに多くの時間と労力を要することが，エンハンス開発の課題の一つとなっている．実例として，不具合修正のための一つのモジュールへの変更が複数回に及んだり，一つの不具合に対する延べ修正日数が長期にわたる例も報告されている [1]．

この課題に対する解決策の一つとして，ソフトウェアの欠陥や不具合修正により変更が生じる可能性が高いモジュールを予測し，その結果をソフトウェア開発の過程で活用するという考え方がある．しかし，ソフトウェア欠陥予測研究の多くはオープンソースソフトウェアを対象としており，産業界でのソフトウェア開発を対象とした研究事例は十分に多いとは言えない [2]．加えて，産業界でのソフトウェア開発では，コードの変更履歴などを容易に収集することが難しい場合もある [3]．

筆者らは，産業界でのソフトウェア開発において，不具合修正により変更が生じるモジュールをインパクトスケール [3][4] などのメトリクスを用いて機械学習により予測し，その結果をソフトウェア開発の過程で活用する研究に取り組んでいる [5]．その際，実用的な精度で変更を予測できることに加えて，産業界でのソフトウェア開発において運用可能な開発環境基盤から得られるメトリクスを用いて予測できることも重要な要件としている．加えて，本取組みの結果が，組織的なソフトウェア品質改善につながる情報となることも重視している．

本稿では，筆者らの所属組織で運用中の開発環境基盤 [1][6] を通じて得られたソフトウェア開発データを用いて，不具合修正により変更が生じるファイルをランダムフォレストにより予測した結果を報告する．これは，筆者らが文献 [5] で報告した内容に，これとは独立に開発されたソフトウェアのデータを分析した結果を加えたものである．

本研究では次の二つのリサーチクエスチョンを設定する．

RQ1: インパクトスケールは不具合修正により変更が生じるファイルの予測に有用なメトリクスか？

RQ2: 産業界でのソフトウェア開発において運用可能な開発環境基盤から得られるメトリクスを用いて，不具合修正により変更が生じるファイルを実用的な予測性能で予測できるか？

これらの問いに答えるために，独立に開発された二つのソフトウェアを分析対象とし，インパクトスケールを含む

[1] 東洋大学経営学部
[2] 富士通株式会社

プロダクトメトリクスをファイル単位で測定する．そして，すべてのファイルについて，不具合修正のための変更が生じたか否かを記録し，これを目的変数とする．こうして収集したデータについて，説明変数の選択と，教師あり復元抽出によるサンプリング手法を適用した上で，ランダムフォレストによる予測とその評価を行う．

本研究の結論を先取りして述べると，RQ1 について，インパクトスケールは不具合修正により変更が生じるファイルの予測に有用なメトリクスであると言えるが，これを用いない場合でも高い予測性能が得られるケースがあることが示された．ただし，予測性能の評価指標には大きな差異がないことや，インパクトスケールはモジュール間の間接的な関係を考慮したメトリクスであり，組織的な品質改善へと結びつけられる可能性があることなどを総じて考えれば，インパクトスケールは有用なメトリクスであると筆者らは考えている．また，RQ2 について，本研究の分析結果は先行研究に示された予測性能よりも良いことから，産業界でのソフトウェア開発において運用可能な開発環境基盤から得られるメトリクスを用いて，不具合修正による変更が生じるファイルを十分な予測性能で予測できることを示した．

2. メトリクス解析による品質リスクヘッジに向けた取組み

図 1 に，筆者らの組織におけるメトリクス解析による品質リスクヘッジの取組みの全体像を示す．筆者らは製品システムを構成するソフトウェアの開発を通じて，メトリクス解析による内部品質の可視化に 2012 年より取り組んできた [6]．ここで，内部品質とはソフトウェアの保守性の度合いを意味しており，外部品質すなわちソフトウェアで実現した機能動作が要求を満たす度合いと区別している．内部品質の品質特性には様々なものが考えられるが，筆者らは特に解析性，変更性，独立性，試験性に着目してこれらの可視化に取り組んでいる [6]．

この取組みの延長として，現在は，ソフトウェアの内部品質に関わる品質リスクを回避する取組み，すなわち品質リスクヘッジに向けた取組みを進めている．具体的な内容として，メトリクス分析の結果をエンハンスのベースとなる母体ソフトウェアの評価に用いたり，ソフトウェア構造の劣化抑止に適用したりするなどしている．最近では，不具合修正により変更が生じるファイルを機械学習により予測する研究に取り組んでいる [5]．これらの取組みは開発現場への実適用を試みるなどして，現場からフィードバックを受けながら継続的に実施している．

こうした一連の取組みを支えているのが，筆者らの組織で運用している開発環境基盤である（**図 2**）．この開発環境基盤は CI (Continuous Integration) 環境に実現されており，様々なプロダクトメトリクスをファイル単位で収集

図 1 メトリクス解析による品質リスクヘッジの取組みの全体像．

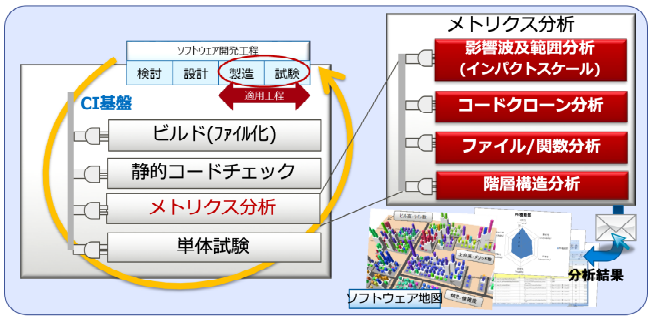

図 2 ソフトウェア開発環境基盤と構造劣化防止の取組み．

している．開発環境基盤の具体的な活用方法として，開発初期の検討工程においては母体ソフトウェアの評価に適用し，内部品質に関わる品質リスクを洗い出して開発計画に反映している．また，コーディング工程においてはメトリクスの値を監視し，ソフトウェア構造の劣化を検知した際には開発者へアラートを通知する仕組みにしている．これにより，開発者はテスト工程に入る前にソフトウェア構造の劣化を定量的に知ることができ，工程の手戻りがなく改善アクションを取りやすい仕組みとなっている．加えて，可視化の工夫としてソフトウェア地図 [7] を活用したりしている．

開発環境基盤で収集しているメトリクスの一つにインパクトスケール [3][4] がある．インパクトスケールは，ファイルの変更による影響波及量を表すメトリクスである．インパクトスケールが不具合修正にかかわる変更が生じるファイルの予測の精度向上に寄与するのであれば，設計やコーディングの段階でモジュール間の影響波及に関わる設計の改善を促す効果が期待される．

3. 分析対象

筆者らの組織で独立に開発された二つの製品システムをそれぞれ構成するソフトウェア X と Y を分析対象として，不具合修正による変更が生じるファイルの予測を行う．

3.1 プロダクトメトリクスの測定

図 3 に，分析対象ソフトウェアの開発プロセスと，本研究で用いているプロダクトメトリクスの測定時点との関係を示す．分析対象ソフトウェアは，いずれもインクリメンタル型開発プロセスに従って開発された．本研究で用い

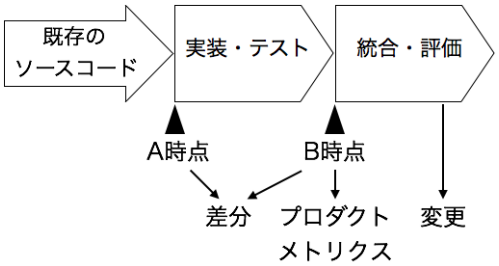

図 3 開発プロセスと測定時点との関係（出典：文献 [5]）.

るプロダクトメトリクスは，次に示す二つの時点での測定結果に基づいている．一つ目は，当該プロジェクトでの機能追加を行う前の時点であり（これを A 時点と呼ぶ），既存のソースコードを測定対象としている．二つ目は，当該プロジェクトでの機能追加とテストが完了した時点であり（これを B 時点と呼ぶ），既存のソースコードおよび新規ならびに変更ソースコードを測定対象としている．説明変数として用いるプロダクトメトリクスは，B 時点で測定された値と，A 時点から B 時点までの差分をファイル単位で示した値を利用する．

B 時点以降は統合・評価段階であり，他のシステムとの統合や第三者などによる評価が行われる．この統合・評価段階で検出された不具合について，その修正のために生じたファイルの変更を測定し，不具合修正予測における目的変数としている．

不具合修正により変更が生じたファイルの測定は，ソースコード構成管理ツールのコミット履歴に基づいて行った．筆者らの組織では，ソースコードを変更して構成管理システムにソースコードを登録するときに，変更内容を記載したチケットの番号をコメントとして記載している．また，変更種別の情報として，不具合修正または機能追加のいずれかをチケットに記載している．これらのうち不具合修正のみを対象として，チケット番号を手がかりとしたスクリプト処理により不具合修正と変更ファイルを紐付けた．

なお，本研究では不具合の原因となる欠陥が混入された時点を測定していないため，A 時点以前から残存していた欠陥が原因の変更なのか，A 時点から B 時点の間の開発・テスト段階で混入した欠陥が原因の変更なのか，あるいは B 時点以降の統合・評価段階で新たに混入した欠陥が原因の変更なのかを区別できていない．したがって，本研究での不具合修正予測とは，より厳密には「開発・テスト段階の完了時点（B 時点）においてどのような状態のファイルだと，欠陥の混入時期によらず，統合・評価段階（B 時点以降）に検出された不具合の修正のために変更が生じやすいか」という問いを扱っていることになる．

3.2 分析対象ソフトウェアの統計量

表 1 に，分析対象ソフトウェアの不具合修正ファイル数とその比率を示す[*1]．X の不具合修正ファイル比率は 3.75%，Y は 11.55%であり，いずれも不具合修正ファイルの比率が小さい不均衡データである．Malhotra ら [2] によると，機械学習を適用したソフトウェア欠陥予測研究で使用されたデータセットのうち，欠陥率が 5%未満のデータセット数は約 10%であったことから，X は特に不均衡の度合いが高いといえる．

表 1 分析対象ソフトウェアの不具合修正ファイル比率.

対象	全ファイル数	修正ファイル数	比率 (%)	差分 0
X	1840	69	3.75	1192
Y	251	29	11.55	26

なお，表 1 に示した通り，A 時点から B 時点の間で差分の値が 0 であったファイル数は，X が 1192 件（64.8%），Y が 26 件（10.4%）であった．このように，X と Y では，規模だけでなく，A 時点と B 時点の間における母体ソフトウェアに対する変更比率も大きく異なり，性質の異なるプロジェクトであることが読み取れる．

B 時点で測定したプロダクトメトリクスについて，X と Y それぞれの平均値と中央値を**表 2** に示す[*2]．Y ではオブジェクト指向設計に関するメトリクスの値が測定されなかったため，それらのメトリクスの欄には NA と示している．また，実行ルート数の平均値は，X と Y のいずれも極端に大きな値の影響を受けた値となったため，ここでは非表示としている．

なお，これらのメトリクスについて A 時点と B 時点との差分を測定したものも説明変数として用いているが，その統計量は省略する．X の差分については，表 1 に示した通り差分 0 のファイル数が半数を超えていることから，すべてのメトリクスの中央値が 0 であった．

4. ランダムフォレストを用いた不具合修正予測の手順

本節では，本研究で適用したランダムフォレストによる不具合修正予測の手順を述べる．これは先行研究で示された方法に概ね基づいている．**図 4** に分析手順の概略を示す．

まず，表 2 に示したプロダクトメトリクスとその差分の中から，特徴選択により説明変数を絞り込む．特徴選択の方法として，先行研究 [8][9] に倣い，相関に基づく特徴選択手法を用いる．なお，特徴選択における探索手法として，文献 [9] に倣い Greedy Stepwise (forward) 法を用いる．その際，本研究はインパクトスケールに着目していることから，特徴選択の結果としてインパクトスケールとその差分の両方あるいはいずれか一方が説明変数として選択されていることを確認しておく必要がある．

[*1] X には外れ値と判定すべき規模の大きなファイル 1 件があったため，このファイルはあらかじめ除外している．
[*2] それぞれのメトリクスの説明については文献 [6] を参照されたい．

表 2　プロダクトメトリクスと統計量.

	X		Y	
メトリクス	mean	med.	mean	med.
コード行数（LOC）	252.6	138	881.2	295
循環複雑度（CC）	11.7	5	15.7	8
修正循環複雑度（MCC）	11.0	5	14.8	6
本質的複雑度（EC）	2.6	1	6.7	3
実行ルート数（Route）	–	6	–	21
非構造化度（Knots）	13.5	2	38.5	3
ネスト数（MaxNest）	2.3	2	3.1	3
関数の入力数（Input）	13.4	10	30.2	17
関数の出力数（Output）	15.6	12	20.0	10
クラス凝集度（LCOM）	52.3	62	NA	NA
クラス結合度（CBO）	6.5	5	NA	NA
派生クラス数（NOC）	1.2	0	NA	NA
継承ツリー深度（DIT）	0.9	0	NA	NA
インパクトスケール（IS）	20.9	7.6	82.6	37.7
クローン率（CVR）	.21	.00	.12	.05

mean：平均値　med.：中央値

表 3　予測性能の評価指標.

評価指標	定義
正確度（Accuracy）	$\frac{TP+TN}{TP+FP+FN+TN} \times 100$
精度（Precision）	$\frac{TP}{TP+FN} \times 100$
再現率（Recall）	$\frac{TN}{FP+TN} \times 100$
F 値（F-measure）	$\frac{2 \times Recall \times Precision}{Recall+Precision}$
G-mean	$\sqrt{\frac{TP}{TP+FP} \times \frac{TN}{TN+FN}}$
AUC (Area Under ROC Curve)	横軸に FP 率，縦軸に再現率をとった ROC 曲線の下側面積

注）
TP：変更のあったファイルが「変更あり」と正しく予測された件数
TN：変更のなかったファイルが「変更なし」と正しく予測された件数
FP：変更のなかったファイルが「変更あり」と誤って予測された件数
FN：変更のあったファイルが「変更なし」と誤って予測された件数

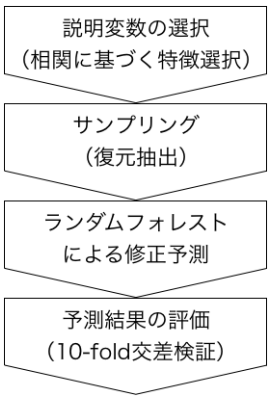

図 4　分析手順.

不均衡データの予測には，復元抽出によるサンプリング手法を適用することによって予測精度が高まることが先行研究により示されている．本研究でも，Malhotra ら [8] の知見に基づいて復元抽出によるサンプリング手法を適用する．同手法を適用するにあたっては，教師ありか否かにより結果が異なるが，本研究では教師ありによるサンプリング手法を用いる．

予測性能の評価にあたっては，10-fold 交差検証を用いる．そして，その結果に対して**表 3** に示した評価指標を用いて予測性能を評価する．ソフトウェア欠陥予測の評価指標として正確度（Accuracy），精度（Precision），再現率（Recall），F 値（F-measure）などがよく利用されるが，これらの評価指標は不均衡データを使用するときには不向きであることが指摘されている [10][11]．不均衡データを使用するときの評価指標として G-mean[12] や，ROC (Receiver Operating Characteristic) 曲線の下側面積を表す AUC (Area Under ROC Curve)[12][13] などが示されている．ただし，これまでの多くのソフトウェア欠陥予測研究は正確度や精度などを利用していることから [2]，比較のしやすさを考慮して，本研究でもこれらを評価指標として示す．

5. 分析結果

X と Y の二つのソフトウェア開発データに対して，機械学習ツール WEKA[14] を使用して不具合修正ファイルの予測を行った．以下にその結果を示す．なお，WEKA のオプションは変更せず，すべてデフォルト値のままとした．

5.1 RQ1. インパクトスケールは不具合修正により変更が生じるファイルの予測に有用なメトリクスか？

表 2 に示したすべてのメトリクスを対象に，相関に基づく特徴選択を行った結果を**表 4** に示す．○印を付けたものが特徴選択によって選択されたメトリクスである．なお，NA を付けたものは 3.2 項で述べた理由のため，これらを除外した上で相関に基づく特徴選択を行った．また，比較のために，インパクトスケールおよびその差分を除いた場合に同様の方法で特徴選択を行った結果も示している．

表 4 に示した通り，インパクトスケールは X と Y のいずれにおいても説明変数として選択された．X においては，インパクトスケールの差分も説明変数として選択された．また，非構造化度（Knots）も X と Y のいずれにおいても説明変数として選択された．しかし，ソースコード行数（LOC）などの先行研究でよく使用されているメトリクスは説明変数として選択されなかった．

この結果から，少なくとも特徴選択の段階において，インパクトスケールは本研究の分析対象において有用なメトリクスである可能性が高いといえる．インパクトスケールの有用性については，次項の予測精度も考慮して評価する．

表 4　特徴選択の結果.

	IS 含む		IS 除く	
メトリクス	X	Y	X	Y
LOC	–	○	○	○
CC	–	–	–	–
MCC	–	–	–	–
EC	–	–	–	–
Route	–	–	–	–
Knots	○	○	–	○
MaxNest	–	–	–	○
Input	○	–	○	○
Output	–	○	–	○
LCOM	–	NA	–	NA
CBO	–	NA	–	NA
NOC	–	NA	–	NA
DIT	–	NA	–	NA
IS	○	○	NA	NA
Cvr	–	–	–	–
Δ LOC	–	–	–	–
Δ CC	–	–	–	–
Δ MCC	–	–	–	–
Δ EC	○	–	○	–
Δ Route	–	○	–	○
Δ Knots	○	–	○	–
Δ MaxNest	–	–	–	–
Δ Input	–	–	–	–
Δ Output	○	–	○	–
Δ LCOM	–	NA	–	NA
Δ CBO	–	NA	–	NA
Δ NOC	–	NA	–	NA
Δ DIT	–	NA	–	NA
Δ IS	○	-	NA	NA
Δ Cvr	○	–	○	–

注）Δ で始まるメトリクスは差分を表す.

5.2 RQ2. 産業界でのソフトウェア開発において運用可能な開発環境基盤から得られるメトリクスを用いて，不具合修正により変更が生じるファイルを実用的な予測性能で予測できるか？

前項で述べた手順により選択されたメトリクスを用いて，教師あり復元抽出によるサンプリング手法を適用した上でランダムフォレストによる変更予測を行い，10-fold 交差検証により予測性能を評価した．その結果を**表 5** に示す．比較のために，インパクトスケールおよびその差分を除いた場合に同様の分析手順を実施したときの予測性能の評価結果も示す．インパクトスケールとその差分を含んだときの X と Y の不具合修正予測における混同行列を**表 6** と**表 7** にそれぞれ示す.

インパクトスケールとその差分を含むメトリクスを用いた不具合修正予測は，これらを含まない場合の予測に比べて，X については予測性能が概ね上回るという結果が示された．しかし，Y については，多くの評価指標についてインパクトスケールを含まない場合の方が予測性能が高いという，RQ1 で想定していた内容とは異なる結果が示された.

表 5　不具合修正予測の結果.

	X		Y	
指標	IS 含む	IS 除く	IS 含む	IS 除く
正確度	**.982**	.980	.920	**.928**
精度	**.875**	.808	.714	**.824**
再現率	.609	.609	**.517**	.483
F 値	**.718**	.694	.600	**.609**
G-mean	**.928**	.892	.819	**.878**
AUC	**.951**	.908	.877	**.893**

表 6　予測結果の混同行列（X：教師あり復元抽出・IS 含む）.

		予測		
		変更なし	変更あり	合計
実測	変更なし	1765	6	1771
	変更あり	27	42	69

表 7　予測結果の混同行列（Y：教師あり復元抽出・IS 含む）.

		予測		
		変更なし	変更あり	合計
実測	変更なし	216	6	222
	変更あり	14	15	29

ただし，インパクトスケールとその差分を含む予測結果は，X と Y のいずれにおいても先行研究に比べて高い予測性能であることから，インパクトスケールの有用性が否定されるものではないと筆者らは考えている．Malhotra ら [2] によると，ランダムフォレストを用いた先行研究 7 件における正確度の平均値は .7563，標準偏差は .1566 であり，ランダムフォレストを用いた先行研究 35 件における AUC の平均値は .83，標準偏差は .09 であった．これらの先行研究の分布に対して，本研究における「IS 含む」の予測性能の z 得点は，X の正確度が 1.44，AUC が 1.34 であり，Y の正確度が 1.05，AUC が 0.52 である.

また，表 5 の「IS 含む」と「IS 除く」の比較で数値に差があるが，その差は小さいものも多く，サンプルによって評価結果が変動する可能性が否めない．このようなことからも，評価指標の大小で単純に結論づけるのではなく，先行研究に比べて優れた予測性能であることを考慮して評価すべきと考える．さらに，インパクトスケールはモジュール間の間接的な関係を考慮したメトリクスであり，組織的な品質改善へと結びつけられる可能性があることなどを総合的に判断すれば，RQ1 について，インパクトスケールは不具合修正予測において有用なメトリクスであると筆者らは考えている．そして，RQ2 について，産業界でのソフトウェア開発において運用可能な開発環境基盤から得られるメトリクスを用いて，不具合修正による変更が生じるファイルを十分な予測性能で予測できたと筆者らは考えている.

6. 考察

6.1 本研究の意義

本研究の意義として，文献 [5] に引き続き，産業界でのソフトウェア開発プロジェクトを対象としたランダムフォレストによる不具合修正予測の事例を示したことが挙げられる．そして，先行研究で議論されてきた知見を踏まえて各種の手法を適用し，先行研究に比べて高い予測性能を示したことは，ソフトウェア欠陥予測研究の促進に貢献したといえる．

本研究が実務に与える示唆として，インパクトスケールで示される影響波及量の大きいファイルは不具合修正にかかわる変更が生じやすいことを実証している点が挙げられる．そして，このようなファイルは，設計段階で影響波及の度合いが小さくなるように構造を見直すことが望ましいことを示唆している点が挙げられる．

影響波及の度合いが小さくなるようにソフトウェアを設計することの重要性は一般に知られているものの，実際にはエンハンス開発の過程でその度合いが少しずつ大きくなってしまい，保守作業を難しくする要因になっていることがしばしばある．不具合修正予測におけるインパクトスケールの有用性を示したことにより，そのようなコードを早い段階から見直す行動を後押しする根拠情報を提供したことに本研究の実務的意義があると考える．

6.2 妥当性への脅威

妥当性への脅威として，表 5 に示した予測結果について，インパクトスケールを含む場合と含まない場合を比べたときの大小関係が X と Y で一貫しておらず，インパクトスケールの有用性の根拠に欠ける面があることは否めない．また，X の評価指標においてもインパクトスケールを含む場合と含まない場合の差が小さいため，統計的に有意な差があるとはいえない可能性が考えられる．今後，10-fond 交差検証を複数回繰り返して実施するなどして，サンプリングのばらつきを考慮した上で性能評価を行う必要がある．

7. 関連研究

ここでは，本研究にかかわりのあるいくつかの研究を紹介し，欠陥予測研究の動向を概観する．なお，本節の記述は文献 [5] で示した内容に基づいている．

7.1 ソフトウェア欠陥予測モデルの構築手法

ソフトウェアの欠陥予測モデルの構築には，ロジスティック回帰分析 [15] のほか，分類木 [16][17]，ナイーブベイズ [10][18]，ランダムフォレストなどの機械学習の手法がよく用いられる．最近では深層学習を用いた研究 [19] も行われている．

Catal ら [20] によると，2005 年から 2008 年までに出版された欠陥予測研究の 66%が機械学習の手法を用いている．また，Malhotra ら [2] によると，ランダムフォレストは他の機械学習の手法に比べて精度が高い傾向にある．

7.2 欠陥予測に用いる説明変数とその選択手法

Malhotra ら [2] によると，欠陥予測研究においては LOC (Lines of Code) やサイクロマチック複雑度などの手続き型のプロダクトメトリクスが欠陥予測研究の説明変数として利用されることが多い．また，オブジェクト指向型のプロダクトメトリクスである CK (Chidamber and Kemerer) メトリクス [21] の中では，RFC (response for a class) や CBO (coupling between objects) が多く利用されている．さらに，畑ら [22] によると，特定の版に対するプロダクトメトリクスだけでなく，コード変更履歴 [23] などのプロセスメトリクスが多くの欠陥予測研究で利用されている．

説明変数の次元数を減らし，予測モデルの使用性を高めるために特徴選択の手法が使用される．Malhotra ら [2] によると，機械学習を用いたソフトウェア欠陥予測研究の約半数が特徴選択を使用しており，相関に基づく特徴選択手法 (Correlation based Feature Selection: CFS) がもっともよく使用されている．Khoshgoftaar ら [24] は，特徴選択によりすべての説明変数のうちの 15〜30%程度を用いた場合でも，すべての説明変数を用いた場合と同程度の予測精度が得られたことを示している．

7.3 インパクトスケール

多くのプロダクトメトリクスは，基本的に，測定対象モジュール内の情報のみを定量化しているか，または測定対象モジュールと他モジュールとの直接的な関連を定量化している．一方，小林ら [3][4] が提案したインパクトスケールは，直接的な関連に加えて間接的な関連を持つモジュールの影響も考慮しており，変更による影響波及量を定量化している点に特徴がある．

小林ら [3][4] は，インパクトスケールを用いることで，他のプロダクトメトリクスのみを説明変数とした場合に比べて欠陥の予測性能が高まったことを示している．ただし，これらの研究では回帰分析を用いているため，機械学習の手法を適用したときにインパクトスケールが有用であるかどうかは示されていない．

7.4 不均衡データを対象としたサンプリング手法

不均衡データである欠陥予測に機械学習の手法を適用するときに，少数派の分類のデータを人工的に増加させるオーバーサンプリングなどのサンプリング手法を用いることで予測精度が向上することが示されている [25]．サンプリングの手法には様々なものがあるが，Malhotra ら [8] は，復元抽出によるリサンプリング手法が SMOTE (synthetic minority over sampling technique)[26] 手法に比べて予測

精度が高いことを示している．

7.5 機械学習による欠陥予測の産業界での適用

これまでに示されている欠陥予測研究は，データへのアクセスのしやすさと，予測手順の再現性担保のために，オープンソースソフトウェアなどの公開データを分析対象としたものが多い．一方で，産業界のソフトウェア開発プロジェクトのデータに機械学習の手法を適用した欠陥予測研究は非常に少ない [2]．纐纈ら [18] はベイジアンネットワークなどの手法を適用して予測精度を比較評価しているが，ランダムフォレストを適用していない．Weyuker ら [27] はランダムフォレストを適用して欠陥予測を行っているが，不均衡データに対するサンプリングを適用していない．Koroglu ら [9] はサンプリングと特徴選択を行った上で複数の手法を評価し，ランダムフォレストの予測精度が高かったことを示しているが，インパクトスケールのような複数モジュールにまたがるプロダクトメトリクスを利用していない．

これらの先行研究に対する本研究の新規性は，産業界のソフトウェア開発プロジェクトのデータにランダムフォレストを適用するにあたり，インパクトスケールを含むプロダクトメトリクスを説明変数とし，サンプリングと特徴選択を行った上で予測結果を評価している点にあると考える．

8. 結論

本稿では，筆者らの組織で独立に開発された二つの製品システムをそれぞれ構成するソフトウェア X と Y を対象に，統合・評価段階において検出された不具合修正により変更が生じるファイルをランダムフォレストを用いて予測した．分析にあたって，組織で運用している開発環境基盤から得られたインパクトスケールを含むプロダクトメトリクスと，それらの差分をファイル単位で測定し，不具合修正予測の説明変数とした．また，それぞれのファイルについて，不具合修正による変更の有無を目的変数とした．インパクトスケールを用いた場合と用いなかった場合のそれぞれについて，相関に基づく特徴選択および復元抽出によるサンプリング手法を適用した上でランダムフォレストによる変更予測を行い，10-fold 交差検証を行った．その結果，インパクトスケールとその差分を含むメトリクスを用いた不具合修正予測は，これらを含まない場合の予測に比べて，X については予測性能が概ね上回るという結果が示されたが，Y についてはそうでない結果が示された．ただし，インパクトスケールとその差分を含む予測結果は，X と Y のいずれにおいても先行研究に比べて高い予測性能であることから，インパクトスケールの有用性は確認することができた．インパクトスケールがモジュール間の間接的な関係を示したメトリクスであり，組織的な品質改善へと結びつけられる可能性があることなどを総じて考えると，インパクトスケールは有用なメトリクスであると考える．

今後の課題として，変更回数などのように変更の影響度を考慮した予測や，不具合の原因である欠陥に着目した予測を視野に入れながら，産業界における複数のソフトウェア開発プロジェクトを対象に変更または欠陥予測を適用する必要がある．

参考文献

[1] 山田弘隆，伊藤雅子，山瀬清美，中嶋久彰，長尾徳富，小林克己， 縣博之，森田純恵，菊池慎司， 野中誠：ソースコードメトリクスと品質リスクとの関係分析とそれに基づくリスクヘッジ手法に向けて，ソフトウェア品質シンポジウム 2017 講演論文集 (CD-ROM)，pp. 1–8 (2017).

[2] Malhotra, R.: A systematic review of machine learning techniques for software fault prediction, *Applied Soft Computing*, Vol. 27, pp. 504–518 (2015).

[3] 小林健一，松尾昭彦，井上克郎，早瀬康裕， 上村学，吉野利明：大規模ソフトウェア保守のための影響波及量尺度インパクトスケール，情報処理学会論文誌，Vol. 54, No. 2, pp. 870–882 (2013).

[4] Kobayashi, K., Matsuo, A., Inoue, K., Hayase, Y., Kamimura, M. and Yoshino, T.: ImpactScale: Quantifying change impact to predict faults in large software systems, *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, IEEE, pp. 43–52 (2011).

[5] 野中誠，山田弘隆，中嶋久彰，伊藤雅子：インパクトスケールを用いた不具合修正にかかわるソフトウェア変更の予測，研究報告ソフトウェア工学 (SE)， Vol. 2018, No. 33, pp. 1–8 (2018).

[6] 三神郷子，中嶋久彰：メトリクスを用いてネットワークソフトウェアの内部品質を可視化する技術，電子情報通信学会論文誌 B， Vol. 100, No. 5, pp. 356–364 (2017).

[7] Kobayashi, K., Kamimura, M., Yano, K., Kato, K. and Matsuo, A.: SArF map: Visualizing software architecture from feature and layer viewpoints, *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, IEEE, pp. 43–52 (2013).

[8] Malhotra, R. and Khanna, M.: An empirical study for software change prediction using imbalanced data, *Empirical Software Engineering*, Vol. 22, No. 6, pp. 2806–2851 (2017).

[9] Koroglu, Y., Sen, A., Kutluay, D., Bayraktar, A., Tosun, Y., Cinar, M. and Kaya, H.: Defect prediction on a legacy industrial software: A case study on software with few defects, *Conducting Empirical Studies in Industry (CESI), 2016 IEEE/ACM 4th International Workshop on*, IEEE, pp. 14–20 (2016).

[10] Menzies, T., Greenwald, J. and Frank, A.: Data mining static code attributes to learn defect predictors, *IEEE transactions on software engineering*, Vol. 33, No. 1, pp. 2–13 (2007).

[11] Gao, K., Khoshgoftaar, T. M. and Napolitano, A.: Combining Feature Subset Selection and Data Sampling for Coping with Highly Imbalanced Software Data., *the 23rd International Conference on Software Engineering and Knowledge Engineering*, pp. 439–444 (2015).

[12] He, H. and Garcia, E. A.: Learning from imbalanced data, *IEEE Transactions on knowledge and data engineering*, Vol. 21, No. 9, pp. 1263–1284 (2009).

[13] Lessmann, S., Baesens, B., Mues, C. and Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings,

IEEE Transactions on Software Engineering, Vol. 34, No. 4, pp. 485–496 (2008).

[14] Witten, I. H., Frank, E., Hall, M. A. and Pal, C. J.: *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann (2016).

[15] Schneidewind, N. F.: Investigation of logistic regression as a discriminant of software quality, *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, IEEE, pp. 328–337 (2001).

[16] Khoshgoftaar, T. M. and Seliya, N.: Software quality classification modeling using the SPRINT decision tree algorithm, *International Journal on Artificial Intelligence Tools*, Vol. 12, No. 03, pp. 207–225 (2003).

[17] 纐纈伸子，川村真弥，野村准一， 野中誠：プロセスおよびプロダクトメトリクスを用いた Fault-Prone クラス予測の適用事例，研究報告組込みシステム (EMB)， Vol. 2010, No. 6, pp. 1–8 (2010).

[18] 纐纈伸子，川村真弥，野村准一， 野中誠：プロセスメトリクス利用による Fault-Prone クラス予測精度の向上，ソフトウェア品質シンポジウム 2010 講演論文集 (CD-ROM)，Vol. 2010, pp. 1–8 (2010).

[19] 近藤将成， 森啓太， 水野修， 崔銀恵：深層学習による不具合混入コミットの予測と評価，ソフトウェアエンジニアリングシンポジウム 2017 論文集， Vol. 2017, pp. 35–45 (2017).

[20] Catal, C. and Diri, B.: A systematic review of software fault prediction studies, *Expert systems with applications*, Vol. 36, No. 4, pp. 7346–7354 (2009).

[21] Chidamber, S. R. and Kemerer, C. F.: A metrics suite for object oriented design, *IEEE Transactions on software engineering*, Vol. 20, No. 6, pp. 476–493 (1994).

[22] 畑秀明， 水野修， 菊野亨：不具合予測に関するメトリクスについての研究論文の系統的レビュー，コンピュータソフトウェア， Vol. 29, No. 1, pp. 1_106–1_117 (2012).

[23] Moser, R., Pedrycz, W. and Succi, G.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, *Proceedings of the 30th international conference on Software engineering*, ACM, pp. 181–190 (2008).

[24] Khoshgoftaar, T. M. and Gao, K.: Feature selection with imbalanced data for software defect prediction, *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, IEEE, pp. 235–240 (2009).

[25] Kamei, Y., Monden, A., Matsumoto, S., Kakimoto, T. and Matsumoto, K.: The effects of over and under sampling on fault-prone module detection, *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, IEEE, pp. 196–204 (2007).

[26] Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P.: SMOTE: synthetic minority oversampling technique, *Journal of artificial intelligence research*, Vol. 16, pp. 321–357 (2002).

[27] Weyuker, E. J., Ostrand, T. J. and Bell, R. M.: Comparing the effectiveness of several modeling methods for fault prediction, *Empirical Software Engineering*, Vol. 15, No. 3, pp. 277–295 (2010).