Vol.2018-SE-198 No.33
2018/3/10

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

野中 誠[1]　山田 弘隆[2]　中嶋 久彰[2]　伊藤 雅子[2]

概要：本稿では，ソフトウェアの不具合修正にかかわる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するにあたって，プロダクトメトリクスに加えてインパクトスケールを用いることで予測性能が向上することを示す．筆者らの組織のソフトウェア開発プロジェクトを対象に，統合・評価段階において検出された不具合の修正のために変更されるファイルを予測するにあたって，組織で運用している開発環境基盤から得られたプロダクトメトリクスとその差分を説明変数として用いた．また，変更が生じたファイルの数はそうでないものに比べて大幅に少ないという不均衡データであったため，復元抽出によるリサンプリング手法を適用した．予測性能を評価した結果，インパクトスケールは不具合修正にかかわる変更ファイルの予測に有用なメトリクスであることが示された．また，本研究で示された予測性能は，先行研究で示された予測性能に比べて高い水準にあることが示された．

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

1. はじめに

ソフトウェアの欠陥や不具合修正[*1]にかかわる変更が生じる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ことにより，ソフトウェア品質の改善による生産性の向上が期待できる．例えば，欠陥を含む可能性が高いモジュールのテストを優先的に実施することで，テスト効率の向上を図ることができる．また，そうしたモジュールのレビューを重点的に実施することで欠陥を早期に摘出し，テスト段階での不具合検出数を減らすことができる．テスト段階での不具合検出数が少ないプロジェクトは，生産性が高く [1]，リリース後のソフトウェア品質が良い傾向にあることが示されている [2]．さらに，欠陥が混入される可能性が高いモジュールの構造を設計段階やコーディング段階で見直すことで，欠陥の混入やレビューでの欠陥見逃しを抑制する効果が期待できる．

ソフトウェア欠陥予測[*2]はこれまでに多くの研究がなされており，近年は機械学習の適用が主流となっている [3][4]．欠陥予測に利用されるメトリクスとして，特定の版に対するプロダクトメトリクスだけでなく，コードの変更履歴 [5] などのプロセスメトリクスが利用されている [6]．

こうした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プロジェクトを対象とした欠陥予測に機械学習を適用した事例は少ない [4]．加えて，産業界のソフトウェア開発ではコードの変更履歴などのプロセスメトリクスを容易に収集できないという課題がある [7]．

筆者らは，産業界のソフトウェア開発において，不具合修正にかかわる変更が生じるモジュールを予測し，その結果を活用することにより，品質の改善による生産性の向上を目指している．そのためには，実用的な精度で変更を予測できることに加えて，産業界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で予測できることも重要な要件となる．さらに，変更予測の結果が，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実施されたネットワーク製品のソフトウェア開発を対象に，組織で運用中の開発環境基盤 [8][9] から得られるデータを用いて，不具合修正にかかわる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．この開発環境基盤は CI (Continuous Integration) 環境に実現されており，様々な

[1] 東洋大学経営学部
[2] 富士通株式会社
[*1] 本稿では fault を欠陥，failure を不具合と呼んでいる．
[*2] 本稿では欠陥予測と不具合修正にかかわる変更予測を総称して「欠陥予測」と呼ぶ．

プロダクトメトリクスをファイル単位で収集している．また，それらのプロダクトメトリクスに加えて，インパクトスケール [7][10] をファイル単位で測定している．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．インパクトスケールを用いた欠陥予測を回帰分析で実施した研究はあるが [7][10]，機械学習で実施した研究は筆者らが知る限りにおいて報告されていない．

以上の議論を踏まえて，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かかわる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，実用的な予測性能で不具合修正にかかわる変更が生じるファイルを予測できるか？

これらの問いに答えるために，まず，分析対象のソフトウェアについて，インパクトスケールを含むプロダクトメトリクスをファイル単位で測定する．また，これらを異なる 2 時点で測定し，後の時点で測定した値と，前の時点からの差分を説明変数とする．そして，すべてのファイルについて，不具合を修正するために生じた変更の有無を目的変数とする．説明変数の選択と，復元抽出によるリ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う．

本研究の結論を先取りして述べると，インパクトスケールは不具合修正にかかわる変更が生じるファイルの予測に有用なメトリクスであったことと，他のプロダクトメトリクスにインパクトスケールを組み合わせることで先行研究よりも高い予測性能となったことが示された．

2. 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．

2.1 ソフトウェア欠陥予測の手法

ソフトウェア欠陥予測の研究には，欠陥を予測対象とするもの（例えば [3][4]）と，不具合の修正にかかわる変更を予測対象とするもの（例えば [11]）がある．欠陥を予測対象とする場合は，不具合の原因箇所である欠陥に着目できるという利点があるが，その箇所または欠陥を含むモジュールを何らかの方法で特定する必要がある．一方，変更を対象とする場合は，コミット履歴から変更されたファイルのデータを容易に測定できるという利点があるが，欠陥を含むファイルの変更にかかわって変更された他のファイルも対象となることに留意する必要がある．予測研究という観点からは，両者にはデータを準備する過程に違いがあるものの，得られたデータに適用する解析手法については共通である．

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 [12] のほか，分類木 [13][14]，ナイーブベイズ [15][16]，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 [17] も行われている．

Catal ら [3] 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 [4] 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．

2.2 欠陥予測に用いる説明変数とその選択手法

Malhotra ら [4] によると，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 [18] 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 [6] 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 [5] 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，使用性の高い予測モデルを得るために，特徴選択の手法が使用される．Malhotra ら [4] 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 [19] 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．

2.3 インパクトスケール

2.2 節で示した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 [7][10] 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 [7][10] 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．

2.4 不均衡データを対象としたサンプリング手法

一般に，欠陥を含むモジュール数は，欠陥を含まないモジュール数に比べて少ない．このように分類によって件数に大きな差があるデータは不均衡データと呼ばれている．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 [20]．サンプリングの手法には様々なものがあるが，Malhotra ら [11] は，復元抽出によるリサンプリング手法が SMOTE (synthetic minority over sampling technique)[21] 手法に比べて予測精度が高いことを示している．

2.5 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 [4]．纐纈ら [16] 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 [24] 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 [25] 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．

3. 分析対象

筆者らの所属組織で開発された二つのネットワーク製品を構成する 47.6 万行のソフトウェアを分析対象として，不具合修正にかかわる変更が生じたファイルの予測をランダムフォレストにより行う．これら 2 製品のソースコードの一部は共通設計となっており，同じソースコードを共有している．そのため，これら 2 製品を分けて扱わずに一つのソフトウェアとして扱う．

図 1 に，分析対象ソフトウェアの開発プロセスと測定時点との関係を示す．このソフトウェアはインクリメンタル型開発プロセスに従って開発されており，既存のソースコードをベースとして 2016 年 9 月（これを A 時点と呼ぶ）に機能追加を開始した．その後，開発チームによる機能の実装とテストが行われ，2016 年 11 月（これを B 時点と呼ぶ）にこれらのプロセスが完了した．さらにその後，統合・評価段階へと移行し，他のシステムとの統合と第三者などによる評価が行われた．この段階で検出された不具合の修正のために生じたファイルの変更を，2017 年 3 月までの範囲で測定した．

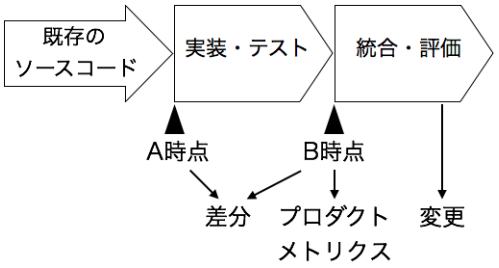

図 1 開発プロセスと測定時点の関係．

不具合修正にかかわる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．筆者らの所属組織では，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，チケットの種別として，不具合修正または機能追加のいずれかを記載している．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，本研究の変更予測は「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．

表 1 に，分析対象ソフトウェアの変更ファイル数を示す．なお，外れ値と判定すべき規模の大きなファイル 1 件は表 1 から除外している．全ファイル数 1840 のうち，欠陥修正にかかわる変更ファイル数は 69 であり，変更ファイル数の比率は 3.75%であった．Malhotra ら [4] 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%である．したがって，分析対象のソフトウェアは，先行研究で使用されたデータセットに比べて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変更ファイル数.

全ファイル数	変更ファイル数	比率 (%)
1840	69	3.75

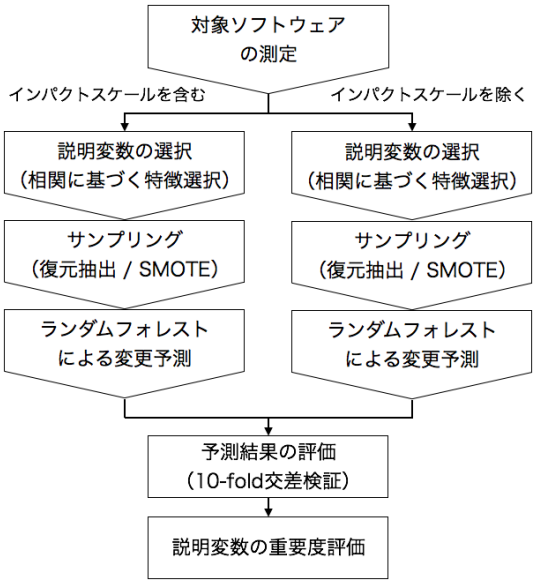

図 2　分析手順の概略.

4. 準備

ここでは準備として，対象ソフトウェアの測定，基本統計量，および分析手順について説明する．**図 2** に分析手順の概略を示す．

4.1 対象ソフトウェアの測定

分析対象ソフトウェアについて，筆者らの組織の開発環境基盤を用いて測定したプロダクトメトリクスの一覧と，その中央値と四分位範囲を**表 2** に示す．なお，それぞれのプロダクトメトリクスの説明については文献 [8] を参照されたい．これらのプロダクトメトリクスについて，図 1 に示した B 時点で測定したものと，A 時点と B 時点との差分を測定したものを説明変数として用いる．

なお，差分の統計量については，すべてのメトリクスにおいて第 1 四分位数と中央値はいずれも 0 であった．第 3 四分位数については，コード行数の差分が 1，インパクトスケールの差分が 0.35 であり，それ以外のメトリクスはいずれも 0 であった．

4.2 説明変数の選択

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 [11][25] 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 [25] 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．

表 2　プロダクトメトリクスと統計量.

メトリクス	Q_1	中央値	Q_3
コード行数（LOC）	56	138	290
サイクロマチック複雑度（CC）	2	5	11
縮退サイクロマチック複雑度（MCC）	2	5	10
本質的複雑度（EC）	1	1	1
実行ルート数（Route）	2	6	39
非構造化度（Knots）	0	2	10
ネスト数（MaxNest）	1	2	3
関数の入力数（Input）	4	10	16
関数の出力数（Output）	5	12	22
クラス凝集度（LCOM）	0	62	90
クラス結合度（CBO）	2	5	9
派生クラス数（NOC）	0	0	1
継承ツリー深度（DIT）	0	0	1
インパクトスケール（IS）	3.0	7.6	23.0
クローン率（CVR）	.00	.00	.36

注）Q_1 は第 1 四分位数，Q_3 は第 3 四分位数を表す.

また，比較のために，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インパクトスケールとその差分を除いた上で，同様の方法で特徴選択を行う．

4.3 サンプリングと予測性能の評価

2.4 節で述べたように，不均衡データである変更予測にはサンプリング手法の適用が望ましい．Malhotra ら [11] の知見に基づいて相関に基づく特徴選択手法を適用するが，比較のために SMOTE 手法も適用する．

予測性能の評価にあたっては，10-fold 交差検証を用いる．これは，元のデータを 10 個のブロックに分割し，9 ブロックの学習用データから予測モデルを構築し，残る 1 ブロックの評価用データを予測するという手順を 10 回繰り返すことで，すべてのデータを 1 回ずつ予測する手法である．この手法は多くの欠陥予測研究で利用される手法である．

10-fold 交差検証の結果に対して，**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）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 [15][26]．不均衡データを使用するときの評価指標として G-mean[27] や，ROC (Receiver Operating Characteristic) 曲線の下側面積を表す AUC (Area Under ROC Curve)[27][28] などが示されている．ただし，これまでのソフトウェア欠陥予測研究は正確度や精度なども多用している [4] ことから，比較のしやすさを考慮して，本研

表 3　評価指標の定義.

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）	$\frac{TN}{FP+TN} \times 100$
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」と予測された件数
TN：変更のあったファイルが「変更なし」と予測された件数
FP：変更のなかったファイルが「変更あり」と予測された件数
FN：変更のなかったファイルが「変更なし」と予測された件数

究でもこれらを評価指標として用いる．

4.4　説明変数の重要度評価

1 節で述べたとおり，本研究では予測性能のみを追求するのではなく，変更予測によって組織的なソフトウェア品質改善につながる情報が得られることも重視している．前項の予測性能の評価方法ではこうした情報を得にくいため，サンプリング後のデータを母集団とした上で，決定木分析とランダムフォレストにより説明変数の重要度評価を行う．その結果を踏まえて，組織的なソフトウェア品質改善につなげるための実務への示唆を考察する．

ただし，この母集団はサンプリングのランダムさに依存するため，つねに一貫した結果が得られるわけではないことに留意する必要がある．

5.　結果

得られたデータを用いて，機械学習ツール WEKA[29] を使用して変更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かかわる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 に示したインパクトスケールとその差分を含む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．また，インパクトスケールとその差分を除外した上で，同様に特徴選択を行った．それらの結果を**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，インパクトスケールとその差分を除外した場合にはこれらのメトリクスが選択されることはないので，NA（該当なし）と示している．

インパクトスケールとその差分を含むすべてのメトリクスを対象として特徴選択を行った結果，インパクトスケールとその差分のいずれも選択された．しかし，LOC やオブジェクト指向メトリクス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．

表 4　特徴選択の結果.

メトリクス	IS 含む	IS 除く
LOC	–	○
Knots	○	–
IS	○	NA
Input	○	○
Δ Knots	○	○
Δ EC	○	○
Δ IS	○	NA
Δ CVR	○	○
Δ Output	○	○

注）Δ で始まるメトリクスは差分を表す.

表 5　変更予測の結果.

	復元抽出		SMOTE	
指標	IS 含む	IS 除く	IS 含む	IS 除く
正確度	**.982**	.980	.941	.939
精度	**.875**	.808	.703	.633
再現率	**.609**	**.609**	.326	.362
G-mean	**.928**	.892	.817	.776
AUC	**.951**	.908	.919	.887

表 6　変更予測結果の混同行列（復元抽出・IS 含む）.

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

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．表 4 に示したメトリクスを用いた変更予測の予測性能が優れていれば，インパクトスケールの有用性を主張することができる．

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，実用的な予測性能で不具合修正にかかわる変更が生じるファイルを予測できるか？

表 5 に，表 4 に示したメトリクスを対象に，復元抽出によるリサンプリング手法と，SMOTE 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結果を示す．また，わかりやすさのために，インパクトスケールとその差分を用いた変更予測における混同行列を**表 6** に示す．

SMOTE 手法を適用する場合に比べて，復元抽出によるリサンプリング手法を適用した場合の方が予測性能が高いことがわかる．また，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変更予測は，これらを含まない場合の変更予測に比べて，再現率を除くすべての評価指標において予測性能が高いことがわかる．Malhotra ら [4] 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「復元抽出・IS 含む」の予測性能の z 得点は，正確度が 1.44，AUC が 1.34 であることから，本研究の結果は先行研究に比べて高い予測性能であるといえる．

これらの結果から，RQ2 の答えとして，復元抽出によるリサンプリング手法を適用した場合に高い予測性能で変更を予測できたといえる．また，予測性能が高いことから，RQ1 の答えとしてインパクトスケールが変更予測に有用なメトリクスであるといえる．

5.3 サンプリング後のデータを母集団としたときの説明変数の重要度評価

RQ1 の答えを異なる観点から探り，かつ組織的なソフトウェア品質改善につながる情報を得るために，サンプリング後のデータを母集団とした上で，決定木分析とランダムフォレストにより説明変数の重要度評価を行う．

まず，表 4 に示したインパクトスケールとその差分を含むメトリクスを対象に，復元抽出によるリサンプリング手法により「変更あり」を増やすオーバーサンプリングを行った．そして，オーバーサンプリング後のデータを母集団とした決定木分析を行った．その結果を**図 3** に示す．図中の終端ノードの TRUE は「変更あり」，FALSE は「変更なし」を表している．なお，決定木のサイズが大きくなったため，同じメトリクスが分岐ノードに 2 度出現しない決定木となるように剪定している．

図 3 から，インパクトスケールが 5.941 以上[*3]（ノード 1）で，インパクトスケールの差分が少しでも大きくなった場合（ノード 3）に，不具合修正にかかわる変更が生じる確率が 0.75 程度（ノード 9）であることがわかる．一方で，インパクトスケールが 5.941 未満（ノード 1）の場合は変更が生じにくいことがわかる（ノード 2）．また，インパクトスケールが 5.941 以上（ノード 1）で，その差分がほとんど増えていない場合（ノード 3）には，非構造化度が 4.5 以上[*4]（ノード 6）か否か，さらには非構造化度が 4.5 以上の場合に関数の入力数が 8.5 以上[*5]（ノード 4）か否かで変更が生じる確率が異なることがわかる．

同様に，すべてのデータを母集団として，ランダムフォレストにより変更予測モデルを構築し，予測に寄与した変数の重要度を分析する．**表 7** に，変更予測に用いたメトリクスと，その重要度を示す指標である mean decrease Gini (MDG) を示す．インパクトスケールとその差分の MDG

[*3] インパクトスケールの中央値は 7.6 である（表 2）．
[*4] 非構造化度の中央値は 2 である（表 2）．
[*5] 入力の数の中央値は 10 である（表 2）．

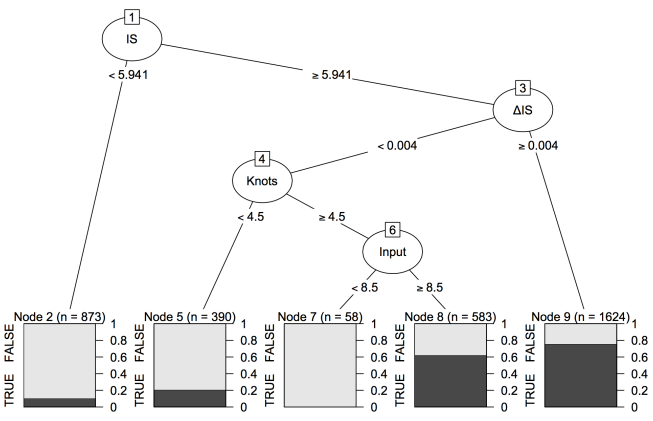

図 3 復元抽出によるリサンプリング後を母集団としたときの決定木．

表 7 変数の重要度．

メトリクス	mean decrease Gini
IS	398.4
ΔIS	329.0
Input	272.7
Knots	257.3
ΔCVR	136.3
ΔOutput	122.8
ΔKnots	61.8
ΔEC	17.6

値が大きいことから，これらのインパクトスケールが変更予測に大きく貢献していることがわかる．一方で，非構造化度の差分と本質的複雑度の差分は，他のメトリクスに比べて変更予測にほとんど寄与していないことがわかる．

これらの結果も，RQ1 の答えとしてインパクトスケールが変更予測に有用なメトリクスであることを支持しているといえる．

6. 考察

6.1 研究の意義

本研究の意義として，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変更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．

また，インパクトスケールの有用性を示したことも，プロダクトメトリクス研究における本研究の貢献といえる．2.3 節で述べたとおり，インパクトスケールは，モジュール間の直接的な関連に加えて，間接的な関連を持つモジュールも考慮した上で影響波及量を測定しているという特徴がある．不具合修正にかかわる変更予測においてインパクトスケールが有用であるということを示したことで，影響波及の度合いを低くした良い構造へと設計を改善することが期待できる．

6.2 実務への示唆

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設計することの重要性は，ソフトウェア工学における基本的事項ではあるものの，実際には影響波及の度合いが大きいプログラムとしてコーディングされてしまうことがしばしばある．そのようなコードを早い段階から見直す必要性は認識されているものの，多くの技術者が関与する大規模なソフトウェア開発において実際に行動に移すことは容易ではない．表 7 に示したように，変更予測におけるインパクトスケールの重要度の大きさ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意義があると考える．

また，図 3 に示した決定木から読み取れる具体的な示唆として，インパクトスケールが小さくないファイルに対して，開発やテストの過程を通じてその値が大きくなったファイルは注意すべきということである．インパクトスケールは，当該ファイルに変更がなされていなくても，影響波及関係にあるファイルが変更されると値が変更するという性質がある．したがって，当該ファイルの変更の有無によらず，インパクトスケールの差分の変化に着目することの必要性を示唆しているといえる．

6.3 妥当性への脅威

内的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差が指標によっては小さいため，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．特に，復元抽出によるリサンプリングを行ったときの正確度と再現率について，インパクトスケールを含む　場合と含まない場合での差は極めて小さく，有意な差があるとは言いがたい．ただし，精度や AUC においては十分な差が示されていることから，総合的にはインパクトスケールが有用であるといえよう．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．

外的妥当性への脅威として，本研究で示した変更予測の結果は一つのソフトウェアのみを対象としているため，インパクトスケールが変更予測に有用であるということが他のソフトウェアに対しても当てはまるとは限らない可能性が考えられる．ただし，本研究では 1840 ファイルという多くのファイルを含んだソフトウェアを対象としたことから，サンプルとしては妥当であったと考える．今後，同様の手法を他のソフトウェアに適用するなどして，インパクトスケールが不具合修正にかかわる変更予測に有用であることを検証する必要がある．

また，本研究では変更予測の結果が組織的なソフトウェア品質改善につながることを重視しているが，変更ファイルの予測性能が高さがそのような目的に役立つことを，根拠をもって示すことができていない．特に，3 節で述べたように，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，プロダクトメトリクスを測定した時点よりも前に混入した欠陥にかかわる変更であるのか，測定した時点よりも後に混入した欠陥にかかわる変更であるのかを区別できていない．とはいえ，インパクトスケールが重要な説明変数であるという結果は，ソフトウェアの設計改善に役立つ情報と考えられるので，外的妥当性を持ちうる結論である．今後，予測結果を踏まえたプロセス改善を行うなどして，その効果を定量的に示していく必要がある．

7. 結論

本稿では，筆者らの組織で開発されたネットワーク製品のソフトウェアを対象に，不具合修正にかかわる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結果を述べた．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プロダクトメトリクスとインパクトスケール，ならびにそれらの差分をファイル単位で測定し，変更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，不具合修正にかかわ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リ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．さらに，サンプリング後のデータを母集団としたときの説明変数の重要度評価を行った．その結果，次を得ることができた．

(1) インパクトスケールとその差分は不具合修正にかかわる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ことが示された．これは，特徴選択で選択されたことに加えて，説明変数の重要度評価において高い値が示されたこと，さらには変更予測の予測性能の高さから示される．
(2) 変更ファイルの予測性能は，先行研究に比べて高い水準であったことが示された．これは，先行研究の分布に対する本研究の予測性能の z 得点が，正確度 1.44，AUC1.34 であったことから示される．

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] 丸山志保，柳田礼子：レビュー重視と品質・生産性の関係分析－品質と生産性の向上は両立するかー，ソフトウェア品質シンポジウム 2017 講演論文集 (CD-ROM)，pp. 1–7 (2017).

[2] 独立行政法人情報処理推進機構 技術本部ソフトウェア高信頼化センター 高信頼性定量化部会信頼性メトリクス WG：横断的アプローチによるソフトウェア開発データの分析～高信頼性定量化部会信頼性メトリクス WG 検討報告書～，https://www.ipa.go.jp/sec/reports/20150416_1.html (2015).

[3] 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).

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

[5] 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).

[6] 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).

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] 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).

[12] 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).

[13] 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).

[14] 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).

[15] 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).

[16] 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 2017 講演論文集 (CD-ROM)，Vol. 2010, pp. 1–8 (2010).

[17] 近藤将成， 森啓太， 水野修， 崔銀惠：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).

[18] 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).

[19] 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).

[20] 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).

[21] 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).

[22] Wang, S. and Yao, X.: Using class imbalance learning for software defect prediction, *IEEE Transactions on Reliability*, Vol. 62, No. 2, pp. 434–443 (2013).

[23] Wang, S., Chen, H. and Yao, X.: Negative correlation learning for classification ensembles, *Neural Networks (IJCNN), The 2010 International Joint Conference on*, IEEE, pp. 1–8 (2010).

[24] 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).

[25] 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).

[26] 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).

[27] 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).

[28] 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).

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