6K-06

コーディング規約違反メトリクスに基づく ソフトウェア不具合予測手法の予測性能向上

山本　真[†]　　名倉　正剛[‡]　　高田　眞吾[†]
慶應義塾大学[†]　　南山大学[‡]

1. はじめに

ソフトウェア開発において，コードレビューやテストといった品質保証活動を重点的に実施すべきモジュールを早期に発見するため，ソフトウェア不具合予測の研究が盛んに行われている[1]．我々も以前，コーディング規約に対する違反に基づく不具合予測手法を提案した[2]．この手法では，ソフトウェアの変更に対するコーディング規約違反の増加量を説明変数として，教師あり学習によって不具合予測を行った．

本研究では，この手法の精度向上を目的として追実験を行い，その結果判明した問題点に対して手法を改善した．

2. 既存手法とその課題

本節では，先行研究[2]の概要と本研究で対象とする課題を述べる．

先行研究では， OSS リポジトリに存在するプロジェクトの Java ソースコードに対して，Java コーディング規約違反分析ツール Checkstyle[*1] を用いて変更前後のコーディング規約違反を抽出した．そして規約種類ごとに違反の増加量を算出した値を編集行数で正規化することで，正規化コーディング規約違反メトリクス$NormV_R$を定義した．次に Commit Guru[*2] によりコミットと不具合の対応が公開されている OSS プロジェクトを分析することにより，$NormV_R$ と不具合混入に有意な関連がある 16 個の規約を明らかにした．その上でこれらの規約群と不具合混入の間に有意な関連 (有意水準 1%)がある 14 個の既存 OSS プロジェクトに対する $NormV_R$の値を教師データとして学習することで，開発中のプロジェクトのコミットに不具合が混入しているかをランダムフォレストにより判別させた．

我々は既存手法に対して，正規化コーディング規約違反メトリクスの抽出の追実験を行い，各規約に関してその値の分布を観察した．その結果，一部のコミットにおいて，複数の規約に関する正規化コーディング規約違反メトリクスが，編集量に比べはるかに大きい値を示した．ここから次の問題が判明した．

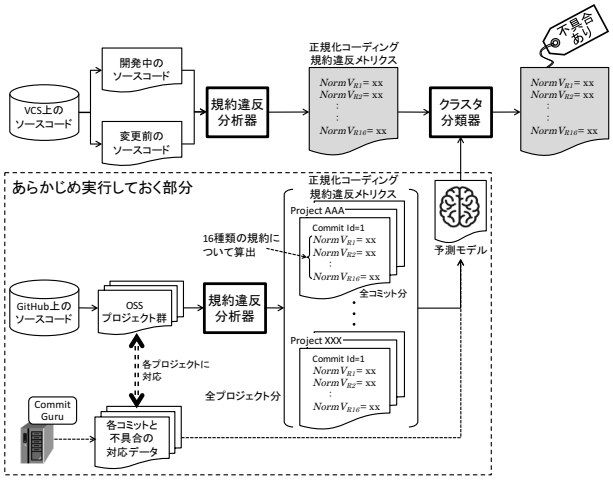

図 1 先行研究[2]での不具合予測手法の概要

問題1　先行研究[2]では，一部のコミットに対してコーディング規約違反メトリクスの抽出が適切に行われていなかった．

また，不具合予測において分類手法の選択が予測性能に影響を及ぼすという調査結果[3]を考慮すると，次の問題が存在する．

問題2　先行研究[2]では，大量のデータに対する計算コストと予測精度の両立のため，予測モデル構築にランダムフォレストを利用した．しかし，これが最適な方法であるのかが明らかではない．

3. メトリクス抽出方法の改善

問題 1 を解決するため，既存手法によって抽出されたコーディング規約違反メトリクスのデータにおいて，複数の規約で編集量に比べてはるかに大きな値を示したコミットの変更内容と，コーディング規約違反メトリクスの変化を調査した．その結果，それらのコミットは次の 2 種類に分類できた．

1 つは，ファイル内容の変更がなく，ファイル名が変更されたコミットである．ファイル名変更がファイル削除と新規作成のセットとみなされたため，変更後に検出されたコーディング規約違反はそのコミットで新規に現れたとみなされ，その増加量がコーディング規約違反メトリクス算出に用いられた．

もう 1 つは，直前のコミットに，構文の誤りによって Checkstyle の実行に失敗するファイルが含ま

Improving performance in software defect prediction based on metrics of coding standard violation
Makoto Yamamoto[†], Masataka Nagura[‡], Shingo Takada[†]
[†]Keio University, [‡]Nanzan University

[*1]https://checkstyle.sourceforge.io/
[*2]http://commit.guru/

れるコミットである．Checkstyle の実行に失敗したファイルのコーディング規約違反が全て 0 として扱われ，抽出対象コミットにおける規約違反メトリクスが正しく計算されなかった．

そこで，本研究では，前者については削除されたファイルと新規作成されたファイルで内容が完全に一致するようなペアが存在した場合をファイル名の変更とみなし，変更時にはメトリクスが変化していないものとすることで対応した．後者については，変更前のファイルの分析において Checkstyle の実行が失敗した場合，実行が成功するファイルに到達するまで再帰的に直前コミット時点での該当ファイルを利用して Checkstyle を実行し，直近で Checkstyle の実行が成功するファイルを変更前のファイルとみなすことで対応した．この結果，先行研究[2]で分析対象としたプロジェクトに対し，先行研究とは異なる$NormV_R$の値を得た．得られた値を利用することで，不具合発生に有意に関連するコーディング規約を特定した．この規約群は先行研究で特定したコーディング規約とは一部異なる規約を含む 16 規約であった．

先行研究と同様にこれらの規約群と不具合混入の間に有意な関連がある 10 プロジェクトを利用し，不具合予測性能を交差検証により検証した．この際に，先行研究と同様にランダムフォレストにより予測モデルを構築した．比較のため先行研究で利用した 16 規約を利用し先行研究で学習に利用した 14 プロジェクトに対しても同様の交差検証を行うことで，改善前の状況での予測性能を評価した．この結果を，表 1 に示す．AUC が 66.2% であったのが 71.2% となり，メトリクス抽出方法の改善により予測性能を向上できた．

表 1 予測性能の比較 (単位: %)

	精度	適合率	再現率	F1 値	AUC
改善前	78.8	62.2	40.9	48.9	66.2
改善後	83.1	63.8	50.5	55.3	71.2

4. 分類手法の選択

問題 2 を解決するため，教師あり学習による分類手法のうち，次の 8 種類の分類手法を利用して予測モデルを構築した上で不具合予測を実施し比較した．

- Gradient Boosting Tree (GBT)
- XGBoost
- LightGBM
- Adaboost
- Random Forest (RF)
- Support Vector Machine (SVM)
- Naïve Bayes (NB)
- k-Nearest Neighbor (kNN)

3 節で改善後の交差検証で対象にした 10 プロジェクトに対する$NormV_R$の値を利用し，各分類手法に対して不具合予測性能を交差検証で検証した．この結果を 表 2 に示す[*3]．k-Nearest Neighbor を用いた場合の性能が一番高くなり， AUC が 73.1% になった．この結果より，コーディング規約違反の傾向を利用して不具合予測を実施する際の予測モデルとして，k-Nearest Neighbor が適していることが分かった．

k-Nearest Neighbor では 16 規約に対するコーディング規約違反メトリクスデータを 16 次元のベクトルとみなし，不具合予測を行う対象のコミットデータとベクトル間距離が近い学習データのみを利用して予測を行う．コーディング規約違反の出現傾向が似たコミットのデータを学習に利用したために予測性能が高くなったと考えられる．

表 2 分類手法間の予測性能の比較 (AUC 順，単位: %)

	精度	適合率	再現率	F1 値	AUC
kNN	81.9	59.0	57.9	57.4	73.1
AdaBoost	83.2	64.7	49.7	55.9	70.9
RF	83.3	65.5	49.3	55.0	70.9
XGBoost	83.1	64.6	49.4	54.7	70.8
NB	82.3	61.5	50.4	54.1	70.6
GBT	83.2	65.1	48.7	54.4	70.6
LightGBM	83.2	65.3	48.0	54.2	70.4
SVM	78.8	0.0	0.0	0.0	50.0

5. 結論

コーディング規約違反メトリクスに基づくソフトウェア不具合予測手法について，精度向上のための改善点を実験的に明らかにした．そこで判明した改善点に対して改善を行った．既存手法との予測性能の比較の結果，AUC の値が 66.2% から 73.1% となった．これにより，不具合予測の性能向上を確認した．

謝辞 本研究の成果の一部は，科研費基盤研究(C)17K00110，2019 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

参考文献

[1] 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.106-117, (2012).
[2] 名倉 正剛，田口 健介，高田 眞吾,“コーディング規約違反メトリクスに基づく ソフトウェア変更に対する不具合予測手法の提案,” ソフトウェアエンジニアリングシンポジウム 2019 論文集, pp.190-198, (2019).
[3] B. Ghotra, S. McIntosh, A. E. Hassan, “Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models,” Proc. of the 37th Int'l Conf. on Software Engineering, pp.789-800 (2015).

[*3] なおここではハイパーパラメータのチューニングも行ったため，ランダムフォレストで予測した場合も表 1 とは値が異なる．