コードレビューにおける誤評価する検証者の特定に向けて

平尾 俊貴[1,a)]　伊原 彰紀[1,b)]　松本 健一[1,c)]

概要：コードレビューとは，不具合修正や機能追加などを目的として変更されたソースコード（パッチ）を第三者の開発者（検証者）が査読する作業である．コードレビューでは，ソフトウェア開発経験が少ない検証者もパッチを評価する．このような検証者が誤評価した場合，検証者間の議論を困惑させると考えられる．本論文では，誤評価する検証者を特定する指標を調査すると共に，指標の有効性の評価に向けた取り組みを述べる．

キーワード：コードレビュー，検証者，パッチ

Toward Identifying an Incorrectly Critiquing Reviewer in Code Review

Toshiki Hirao[1,a)]　Akinori Ihara[1,b)]　Kenichi Matsumoto[1,c)]

Abstract: Code review is a broadly adopted software quality practice wherein reviewers (i.e., team members) critique each other's patches (i.e., changes to a software system for quality improvement). In code review, not all reviewers have significant experience in developing the patches that have been created by the patch authors. Therefore, the less-experienced reviewers who critique the patches incorrectly may cause complicated discussions. This paper aims to clarify the criteria for identifying reviewers who are likely to evaluate the patches incorrectly, and then evaluate the effectiveness of these criteria.

Keywords: Code Review, Reviewer, Patch

1. はじめに

オープンソースソフトウェア（OSS）開発では，不具合修正や機能追加などを目的として変更されたソースコード（パッチ）を，第三者の開発者（以降、検証者）が査読して品質を評価するコードレビューが実施されている．複数の検証者が異なった視点からパッチを評価するため，OSS プロジェクトへパッチを統合する前に不具合を発見できる [4]．

OpenStack [*1] や Qt [*2] などの多くの OSS プロジェクトでは，Gerrit Code Review [*3] をはじめとするコードレビュー管理システムを導入している．コードレビュー管理システムでコードレビューを管理する形態を Modern Code Review と呼ぶ [1]．Gerrit Code Review では，複数の検証者がオンライン上で協調しながらパッチを評価する．検証者が過去にコードレビューに参加した数（レビュー経験数）は様々であるため [5]．我々の先行研究では，レビュー経験数が少ない検証者は，レビュー経験数が多い検証者に比べて，パッチを誤評価する傾向があることを確認した [2]．頻繁にパッチを誤評価する検証者がコードレビューに参加すると，検証者間の議論を困惑させると考えられる．

本論文では，OpenStack のコードレビュー履歴を用いて，誤評価する検証者を特定する指標を調査する．また，その指標を基に誤評価する検証者の予測モデルを構築して，指標の有効性を評価する．

[1] 奈良先端科学技術大学院大学
Takayama-cho 8916-5, Ikoma, Nara 630–0192, Japan
[a)] hirao.toshiki.ho7@is.naist.jp
[b)] akinori-i@is.naist.jp
[c)] matumoto@is.naist.jp
[*1] https://www.openstack.org/
[*2] https://www.qt.io/developers/
[*3] https://www.gerritcodereview.com/

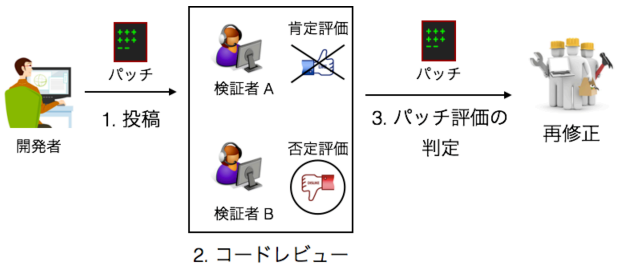

図 1 検証者のパッチ評価結果の正誤を判定する概念図

2. 実験方法

2.1 データセット

本研究では，Gerrit 上でコードレビューを管理している OpenStack を実験対象とする．McIntosh ら [3] が公開した OpenStack のコードレビュー情報（パッチ投稿時間，検証者間の議論など）を用いる．

2.2 RQ1：誤評価した検証者が参加したパッチはどのくらい存在するのか？

検証者の評価結果の正誤を判定する方法を説明する．図 1 は，コードレビューにおいて二人の開発者の評価結果が異なった例である．検証者 A は投稿されたパッチに対して肯定的な評価をしており，検証者 B は否定的な評価をしている．しかし，投稿されたパッチは未完成なパッチであったため，プロジェクトから再修正を要求されている．検証者 A は投稿されたパッチが未完成であったにも関わらず肯定的に評価したため，図 1 では検証者 A は誤った評価をしたと判定する．この正誤判定の定義に基づいて，実験対象パッチ群でどのくらいパッチを誤って評価する検証者が存在するのか調査する．我々が調査した結果，OpenStack に 2011 年から 2014 年までに投稿されたパッチの中で，31%は誤評価した検証者が参加していたことを確認した．

2.3 RQ2：どのようなパッチは誤評価されやすいのか？

検証者は主に二つの方法で開発者にフィードバックを与える．一つ目は，ソースコード中の指摘箇所にコメントを直接書き込む方法（以降，直接的指摘）である．二つ目は，コードレビュー掲示板上で複数の検証者が議論する方法（以降，間接的指摘）である．直接的指摘と間接的指摘に着目して，以下の 3 つの手順に従って RQ2 の分析を進める．

(1) 検証者が誤評価したパッチを，直接的指摘もしくは間接的指摘に分類する．
(2) 手順 1 で分類されたパッチに対して，どのような内容のパッチが存在するのかを調査するために，パッチ情報（コミットメッセージ，検証者間の議論など）を人手で読む．
(3) 手順 2 で得られた調査結果を基に，誤評価しやすい検証者を特徴付ける指標（バグ修正に関する内容のパッチ検証経験数，リファクタリングに関する内容のパッチ検証経験数など）の候補を見つける．

2.4 誤評価する検証者を特徴付ける指標の有効性評価

誤評価する検証者を特定するために有効な指標を明らかにするために，誤評価する検証者の予測モデルを構築する．この予測モデルでは，パッチを評価する検証者に対して，誤評価するか否かをコードレビュー開始前の時点で予測する．この予測結果を基に，直接的指摘及び間接的指摘を受けたパッチに対して，どの指標が有効であるか明らかにする．さらに，直接的指摘及び間接的指摘を受けたパッチの内容別に有効な指標を明らかにして，投稿されたパッチの内容から誤評価する検証者を特定する手法の提案に取り組む．

3. おわりに

コードレビューでは，全ての検証者が正確にパッチを評価できるとは限らない．本論文では，OpenStack を実験対象として，誤評価する検証者を特徴付ける指標の分析方法を説明した．また，RQ2 で得られる指標を基に，誤評価する検証者予測モデルの構築して，指標の有効性の評価に向けた取り組みを述べた．

本論文では OpenStack を実験対象としているため，他の OSS プロジェクトにも同様の分析結果及び予測結果が得られるとは限らない．今後は，他の OSS プロジェクトに対しても調査を進め，他プロジェクトで用いることができる指標の提案を目指す．

参考文献

[1] M. E. Fagan, “Design and code inspections to reduce errors in program development,” *IBM Systems Journal*, vol. 15, no. 3, pp. 182–211, 1976.
[2] T. Hirao, A. Ihara, Y. Ueda, P. Phannachitta, and K. Matsumoto, “The impact of a low level of agreement among reviewers in a code review process,” in *The 12th International Conference on Open Source Systems*, 2016, pp. 97–110.
[3] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, “An empirical study of the impact of modern code review practices on software quality,” *Empirical Software Engineering*, vol. 21, no. 5, pp. 2146–2189, 2016.
[4] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, “Investigating code review practices in defective files: An empirical study of the qt system,” in *Proceedings of the 12th Working Conference on Mining Software Repositories*, 2015, pp. 168–179.
[5] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K. ichi Matsumoto, “Who should review my code? a file location-based code-reviewer recommendation approach for modern code review,” in *Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering*, 2015, pp. 141–150.