OSS 開発におけるソースコード静的解析手法を用いたパッチ検証手法の提案

安藤聡志[†1] 平尾俊貴[†2] 伊原彰紀[†2]
松本健一[†2] 関浩之[†1]

OSS 開発では, 欠陥の修正やパフォーマンスの向上を目的に変更されたファイル（パッチ）に対して一人以上の開発者（レビューア）が品質検証を行う．本研究は，品質の高いパッチの投稿を促すために，投稿前にソースコード静的解析に基づきパッチが採択されるか否かを判定する手法を提案する．本手法の開発に向けて，本論文では Qt プロジェクトに投稿されたパッチ 450 件を対象にパッチに採択，再投稿，不採択されるソースコードの変更行数を調査した結果，変更行数が多くなるほど再投稿される割合が高くなることがわかった．

Toward Identifying Accepted Patches Based on Static Code Analysis

Satoshi Andou,[†1] Toshiki Hirao,[†2] Akinori Ihara,[†2]
Ken-ichi Matsumoto[†2] and Hiroyuki Seki [†1]

In open source software (OSS) projects, more than one developer (reviewer) verifies a source code (patch) that was changed to fix a bug or improve the performance. In order to lead high-quality patch submission, this study proposes an approach to identify a patch that reviewers agree using static code analysis method. From our case study using Qt project dataset, larger size of the changed codes is likely to be requested to resubmit the patch.

1. はじめに

ソフトウェア開発プロセスにおいて，新しく開発，変更されたソースコード（パッチ）を製品に統合する前に，欠陥の有無，可読性などを一人以上の開発者（レビューア）が検証する．ソフトウェア開発企業を始めとするクローズドなソフトウェア開発は，検証作業としてテストケース，テストコードを用意することも多いが，オープンソースソフトウェア (OSS) 開発では，テストコードを十分に管理されていないことが多く，レビューアによる検証がソフトウェアの品質を確保する上で重要な作業である．

これまで，OSS 開発におけるコードレビューはメーリングリスト，不具合追跡システムなどにパッチを投稿し，レビューアが検証していた．しかし，これらのツールは投稿されたパッチ，検証プロセスを管理することを目的としていないため，開発者はパッチの修正箇所，検証ステータス，レビューア間の意見の合意などを追跡することは難しく，ソフトウェア工学研究でコードレビューの研究をすることは容易ではなかった．昨今，パッチ検証のためのツールとして，Gerrit や Review Board をはじめとするレビュー管理システムが開発され，現在これらのツールを利用している OSS プロジェクトも存在する．

レビュー追跡システムでは，投稿されたパッチに対して，レビューア間で合意形成した結果，最終的にパッチの採択，再投稿，不採択を決定する．しかし，OSS プロジェクトには日々数十件のパッチが投稿され，必ずしも高品質なパッチが投稿されているわけではない．低品質なパッチの投稿は，検証作業に時間を浪費し，また再投稿が必要となる場合は，開発者とレビューアの双方にとって不利益なものであり，投稿前にパッチの品質を検証し，高品質なパッチを投稿することが必要である．Weißgerber ら[1] は，大規模 OSS プロジェクト FLAC と OpenAFS を対象に採択されるパッチの特徴を調査し，パッチの行数（変更行数）が少ないほど採択される可能性が高いことを示している．しかし，ソフトウェア工学において一般的に使用されるメトリクス（複雑度，クラス数など）の調査は未だ行われていない．

†1 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†2 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

本研究は，品質の高いパッチの投稿を促すために，投稿前にソースコード静的解析に基づきパッチが採択されるか否かを判定する手法を提案する．本手法の開発に向けて，本論文では Qt プロジェクトに投稿されたパッチ 450 件を対象に，従来研究と同様にパッチに採択，不採択されるパッチサイズ，さらに，従来研究では調査されていなかった再投稿されたパッチサイズについても調査する．

2. 調 査 方 法

レビュー管理システムでは，パッチをプロジェクトへ統合する権限を持つコントリビューター（コミッター）が投稿されたパッチをバージョン管理システムに統合するか否かを決定する．コミッターはレビューアの意見を参考にパッチに対して，採択（プロダクトに統合），不採択（検証作業の終了），再投稿（パッチ再変更を要求）を決定する．

本論文では，同程度の変更行数のパッチをグルーピングすることにより，変更行数による採択，不採択，再投稿の割合を明確に比較する．グルーピングには，変更行数に基づいて K-means 手法を用いることによってパッチを 4 つのグループに分類する．さらに各グループにおいて採択/再投稿/不採択されたパッチの割合を計測し，変更行数による採択/再投稿/不採択されるパッチ数を比較する．

3. ケーススタディ

本論文は，採択されやすいパッチの特徴を理解するために，Qt プロジェクトに投稿されたパッチ 450 件（2011 年 5 月 17 日-2011 年 6 月 10 日）を対象にケーススタディを行った．Qt は UI フレームワークであり，ソフトウェア開発企業 Digia の一部門によって開発されているが，Qt プロジェクトは OSS 開発のように不特定多数の開発者が貢献している．Qt プロジェクトではレビュー管理システム Gerrit を用いてレビューデータを管理しており，採択されたパッチ 277 件，再投稿されたパッチ 155 件，不採択されたパッチ 18 件が含まれる．

図 1，図 2 はそれぞれパッチの追加行数，削除行数（X 軸）による採択，再投稿，不採択されたパッチ数の割合（Y 軸）を示す．調査の結果，追加行数，削除行数ともに，変更行数が増加するにつれ採択されるパッチの割合は減少し，再投稿と判定されるパッチの割合は増加する．特に，最も変更行数が少ないパッチのグループ（平均追加行数：30.6 行，平均削除行数：24.0 行）と 2 番目に変更行数が小さいパッチのグループ（平均追加行数：1131.7 行，平均削除行数：1901.7 行）とで，採択，再投稿されるパッチの割合は逆転し，従来研究の知見であるパッチの変更行数が小さいほど採択される確率が高いことを確認した．

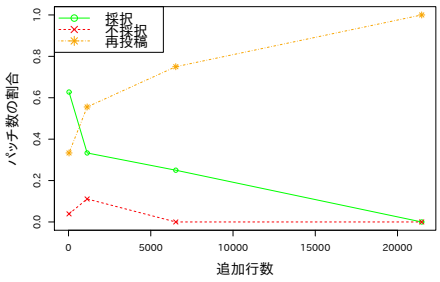

図 1 追加行数とパッチ評価結果の関係

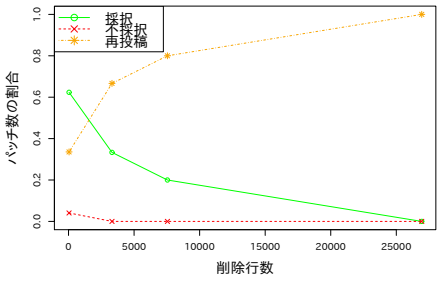

図 2 削除行数とパッチ評価結果の関係

4. まとめと今後の課題

本論文では Qt プロジェクトに投稿されたパッチ 450 件を対象にパッチが採択，再投稿，不採択されるパッチの変更行数を調査した結果，変更行数が多くなるほど再投稿される割合が高くなることがわかった．今後は複雑度をはじめとする，その他のメトリクスについても調査する．さらに，本論文で対象としたデータセットには，パッチの採択/再投稿/不採択の判断はもちろん，再投稿されたパッチがどのように更新されたのかを追跡することができ，今後はパッチの採択/不採択の判定のみならず，再投稿されたパッチがレビューアの再修正要求を満たしているか否かを評価する手法を提案する．

謝辞 本研究の一部は，頭脳循環を加速する戦略的国際研究ネットワーク推進プログラムによる助成を受けた．

参 考 文 献

1) Peter Weißgerber, Daniel Neu, Stephan Diehl, “Small Patches Get In!,” *Proceedings of the International Working Conference on Mining Software Repositories (MSR’08)*, pp.67-76, 2008.