オープンソース開発におけるパッチ検証者数の予測

小野健一[†1] 伊原彰紀[†1]
平尾俊貴[†1] 松本健一[†1]

オープンソースソフトウェア (OSS) プロジェクトには，多岐にわたる専門，生活習慣を持つ開発者が貢献している．しかし，OSS 開発者の多くはボランティアで参加し，依頼されたタスクに対して必ずしも取り組む義務はない．本論文は，OSS 開発を対象に，複数の開発者に依頼されるソースコード（パッチ）の検証作業の取り組みを調査した．その結果，必ずしもすべての開発者が取り組んでいない課題を確認した．依頼した開発者数に対し，実際に貢献する開発者数の予測手法を提案する．

Toward predicting the number of contributors for code review in open source development

Ken-ichi Ono,[†1] Akinori Ihara,[†1] Toshiki Hirao[†1]
and Ken-ichi matsumoto [†1]

Many developers with different expertise and different life style in all over the world contribute to open source software (OSS) project. Most developers could contribute their interest tasks, because they join the project as a volunteer. Hence, they can reject tasks that were assigned from the other developers. This study focuses on code review request. We analyze how many code review requests reviewers accepted. From a case study using Qt project dataset, we found the requested code reviews were not always accepted. This study proposes an approach to predict how many reviewers accept the code review request.

1. はじめに

オープンソースソフトウェア (OSS) プロジェクトに参加する開発者は義務的に貢献するのではなく，ボランティアとして貢献する場合が多いため，開発者は趣味や自己啓発のために取り組むことも多い．しかし，ボランティアとして参加する開発者は，依頼されたタスクに取り組むことを拒否することも可能であり，オープンなコミュニティ特有の課題が存在する．

OSS プロジェクトにおける依頼は，不具合修正依頼，質問回答依頼などが挙げられる．不具合修正依頼は，修正箇所についての高い専門知識が求められ，これまでの多くの研究が本課題に取り組んでいる．また質問回答依頼は，StackOverflow などの Q&A サービスを対象に質問方法などの研究が取り組まれている．本論文では，新しく開発，または，変更されたソースコード（パッチ）の検証依頼を対象とする．

パッチの検証は，パッチに欠陥が混入していないか，要求を満たしているか，ソースコードの可読性などを確認する工程である．しかし，OSS 開発におけるパッチの検証依頼は，不具合修正依頼と同様に，開発者は必ずしも引き受けられるとは限らない．また，プロジェクトによっては一定人数以上による検証を受けない限り，パッチをバージョン管理システムに統合しないルールを設けており，検証する開発者が少ない場合は再割り当てが必要となり，開発効率の低下を招く．

本論文では Qt プロジェクトを対象に，検証依頼を引き受ける開発者の調査を行い，検証依頼を引き受ける開発者数を予測するための提案手法を紹介する．

2. 予備調査

本章では，本研究の動機を明確にするために，パッチ検証依頼された開発者が依頼を引き受ける割合を調査する．本論文では，ケーススタディとして Qt プロジェクト [⋆1] に投稿された 42078 件のパッチにおいて，検証依頼された 939 名の各開発者を対象とする．依頼された開発者のうち実際に依頼を引き受けた開発者の

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

⋆1 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．

表 1 パッチ検証者数の予測の例

依頼された開発者	依頼を引き受けると予測される開発者	実際に依頼を引き受けた開発者数
Alice	○	
Bob	×	
Chirs	○	2 人
Dan	×	
Emily	○	

割合を調査したところ，検証を引き受けた平均割合は 67%と少なく，最も依頼数の多い開発者の上位 10 名でさえ平均割合は 80%であった．検証を行わなかった理由についてはさらに調査が必要であるが，依頼を引き受けない開発者が数多く存在していることを確認することができた．

3. パッチ検証者数の予測に向けた提案

パッチ検証は，一般的に複数の開発者に依頼される．本研究では，依頼された開発者のうち依頼を引き受ける開発者数を予測する．表 1 は，本研究で提案するパッチ検証者数の予測，及び，評価結果の例である．本例は，依頼された開発者は，OSS プロジェクトに投稿されたあるパッチの検証を依頼する候補者を示し，依頼を引き受ける場合は○，引き受けない場合は×を示している．パッチの検証を依頼する候補者の選択は，従来研究[1] で開発されたモデルを用いる．本論文では，依頼された開発者がパッチの検証に貢献するか否かを予測し，実際に貢献する開発者数を算出する．本論文は，各開発者が検証するか否かを予測することはもちろんであるが，プロジェクトがパッチ検証に貢献する開発者数を確保するために，依頼すべき開発者数の予測を目的とする．

本章では，パッチ検証者数の予測に向けた提案として，依頼した開発者の専門性，パッチ検証に対する開発者のオープン性，開発者のタスク量に基づく手法を紹介する．

3.1 開発者の専門性に基づく予測

パッチ検証依頼は，不具合修正依頼と同様，検証対象となるモジュールの専門知識を要する．従来研究[1] では，検証対象となるモジュールを検証した経験を有する開発者を推薦するモデルを提案している．従来研究が提案する手法は必ずしも検証した経験を有する開発者が検証依頼を引き受けているとは限らず，予測精度が低下していると考えられる．OSS プロジェクトでは，ボランティアとして自ら開発に参加する開発形態から，自身が関与した興味のあるモジュールに対して積極的に活動すると考えられる．本手法では，過去に検証対象となるモジュールに対しては，高い確率で依頼を引き受けると考え，そこから貢献者数を予測を試みる．

3.2 パッチ検証に対する開発者のオープン性を用いた予測

パッチ検証もボランティアとしての作業であり，2 章で示した通り開発者は必ずしも依頼を引き受けるとは限らない．一方で，多くの依頼を受けている開発者は引き受ける確率が高く，開発者のオープン性が関係していると考えられる．しかし，依頼された回数が少ない開発者に対しては，開発者のオープン性を十分に理解することは難しく，本手法は効果的でない可能性がある．

3.3 開発者のタスク量に基づく予測

不具合修正，パッチ検証の依頼が増えるにつれ，開発者が期間内に取り組むことのできるタスク量を超えてしまい，検証依頼を引き受けることができないことも予想される．しかし，同プロジェクトからの依頼以外のタスク（開発者の私生活のスケジュール等）を把握することは現実的に不可能であるため，検証依頼直前の開発者の貢献数などを計測することにより，開発者のタスク量に基づく予測を実現できると考えられる．

4. おわりに

本論文では，OSS 開発を対象に，複数の開発者に依頼されるパッチ検証作業の取り組みを調査することで，必ずしもすべての開発者が取り組んでいないことを確認した．従って，検証依頼したとしても開発者が引き受けるとは限らず，依頼した開発者のうち検証に取り組む開発者数を把握することは重要である．本論文では，パッチ検証者数の予測に向けて，依頼した開発者の専門性，パッチ検証に対する開発者のオープン性，開発者のタスク量に基づく手法を提案した．今後は，提案した手法の実装を行い，その予測精度について調査を行う．

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) Thongtanunam.P, Tantithamthavorn.C, Kula.G.R, Yoshida.N, Iida.H and Matsumoto.K: Who Should Review My Code? A File Location-Based Code-Reviewer Recommendation Approach for Modern Code Review, *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER'15)*, pp.141–150 (2015).