OSS 開発のコードレビュー依頼に貢献する開発者の予測

Toward Identifying Developers Contributed to Code Reviewer Requests -Case Study of Qt and OpenStack Projects-

小野 健一[*] **伊原 彰紀**[†] **坂口 英司**[‡] **平尾 俊貴**[§] **松本 健一**[¶]

あらまし 本論文では，OSS 開発のソースコードレビュープロセスを対象に，変更されたソースコード（パッチ）のレビュー依頼に対して開発者が貢献するか否かを，開発者のレビュー経験に着目した分析を行った．Qt，OpenStack プロジェクトを対象としたケーススタディを通して，開発者は過去に数多くレビューしたソースコードに関する依頼ほど，開発者はレビュー依頼に貢献することを確認した．

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の多くは，営利目的や義務的に貢献しているのではなく，趣味や自己啓発のためにボランティアで貢献していることが多い [1] [3]．従って，開発者は，依頼されたタスク（不具合修正やソースコードレビュー）に必ずしも取り組む必要はなく，依頼に貢献しない場合は，別の開発者が当該タスクに取り組むことになる．

本論文で対象とする OSS 開発におけるコードレビューは，変更されたソースコードをバージョン管理システムにコミットする前に品質を検証する作業を指す．OSS プロジェクトには，当該ソフトウェアについての理解が未熟な開発者も貢献しているため，提出されるパッチが OSS にとって有益なものとは限らない．例えば，既に修正済みの変更，欠陥が混入している変更，コーディングルールに従っていない変更なども提出される [6]．従って，コードレビューは，多面的な検証を実施するために複数のレビュアーに依頼することが多い．従来研究では，コードレビューを依頼する適任のレビュアーを推薦するためのアプローチが提案されている [7]．我々は，レビュー依頼に対する貢献の有無の調査を行い，Qt プロジェクトでは依頼されたレビュアーの約 50%，Openstack プロジェクトでは依頼されたレビュアーの約 33%が貢献していないことを確認した．OpenStack では低品質なパッチの統合を防ぐために，2 人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いる [1]．従って，レビュアーは依頼を拒否する場合があるため，依頼者は 3 人以上のレビュアーに依頼することが必要であるが，多数のレビュアーに依頼することはレビュアーにとって負担になり，各プロジェクトが必要最低限のレビュアーに依頼する手法が求められる．

本論文では，OSS プロジェクトにおけるコードレビュー依頼に対する開発者の貢献の有無を調査し，開発者が過去にレビューしたファイルやモジュールの経験に基づき，依頼に貢献するか否かを予測する手法を提案する．

[*]Ken-ichi Ono, 奈良先端科学技術大学院大学 情報科学研究科, ono.kenichi.ob4@is.naist.jp

[†]Akinori Ihara, 奈良先端科学技術大学院大学 情報科学研究科, akinori-i@is.naist.jp

[‡]Hideshi Sakaguchi, 奈良先端科学技術大学院大学 情報科学研究科, sakaguchi.hideshi.rv4@is.naist.jp

[§]Toshiki Hirao, 奈良先端科学技術大学院大学 情報科学研究科, hirao.toshiki.ho7@is.naist.jp

[¶]Ken-ichi Matsumoto, 奈良先端科学技術大学院大学 情報科学研究科, matsumoto@is.naist.jp

[1]http://docs.openstack.org/infra/manual/developers.html

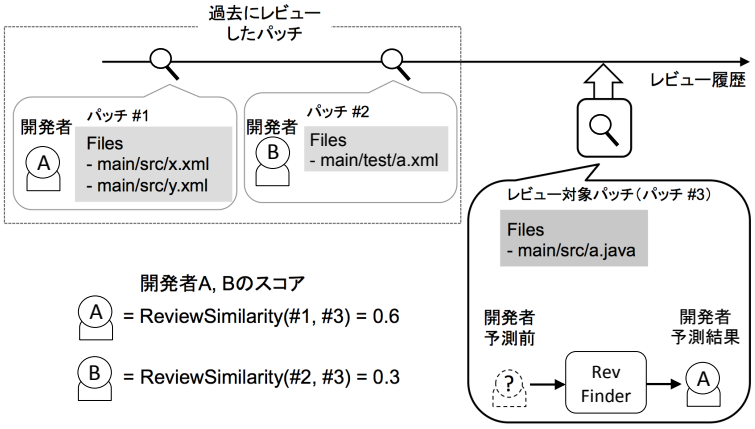

図 1　RevFinder を用いたレビュー依頼する開発者の推薦手法

2　コードレビュー

昨今の OSS 開発では，開発者間でコードレビューの内容を共有するために，Gerrit，Phabricator，ReviewBoard などの Web でレビュー内容を閲覧することができるレビュー管理システムを利用している．レビュー管理システムは，過去に取り組まれたコードレビューを一元管理することによって容易に追跡可能にし，重複したパッチの作成を防ぐなどの効果がある [4] [5]．このようなレビュー管理システムを使ったレビュープロセスを “Modern Code Review” と呼ばれている [2].

2.1　コードレビュープロセス

Modern Code Review におけるコードレビュープロセスは以下の手順を踏む.

(1) 開発者は変更・新規開発したソースコードをレビュー管理システムに投稿し，投稿者，または，コミッターが投稿されたソースコードのレビューを別の開発者（レビュアー）に依頼する.
(2) レビュアーは，投稿されたソースコードをバージョン管理システムに統合することに賛成するか否かを意思を表明する，または，パッチ作成者にコメントを残す.
(3) コミッターはレビュアーの評価を参考に，ソースコード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.

本論文では，レビュープロセスにおけるレビュー依頼に対する開発者の貢献の有無に着目する．1 章でも述べたように，OSS 開発はボランティアとして参加することが多く，レビュアーはレビュー依頼に対して必ずしも対応する義務はない．その実態を確認するため，Qt プロジェクト，OpenStack プロジェクトを対象にレビューが依頼される人数と，依頼に貢献する開発者数を調査した．両プロジェクト共に，2 人の開発者にレビュー依頼していることが多く，たとえ 4 人に依頼しても貢献する開発者は 1 人のみであることが多く，レビュー依頼に貢献する開発者の特定が必要であることを確認した.

2.2　関連研究におけるレビュー依頼手法

Thongtanunam ら [7] らは，レビュアーを推薦する RevFinder を開発している．RevFinder は，レビュー対象パッチのソースコードと，過去にレビューしたパッチのソースコードのリンク（ファイルパス）との類似度 (ReviewSimilarity) を計算し，パッチの知識を有するレビュアーを推薦する手法である．図 1 は，レビュー対象パッ

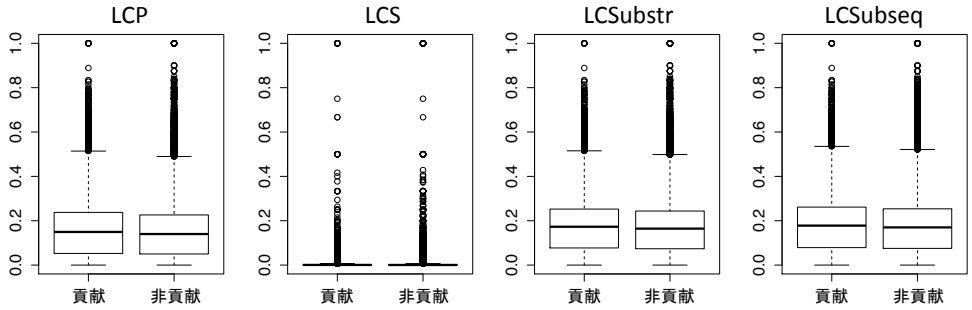

図 2　貢献するパッチと貢献しないパッチの ReviewSimilarity の分布（Qt）

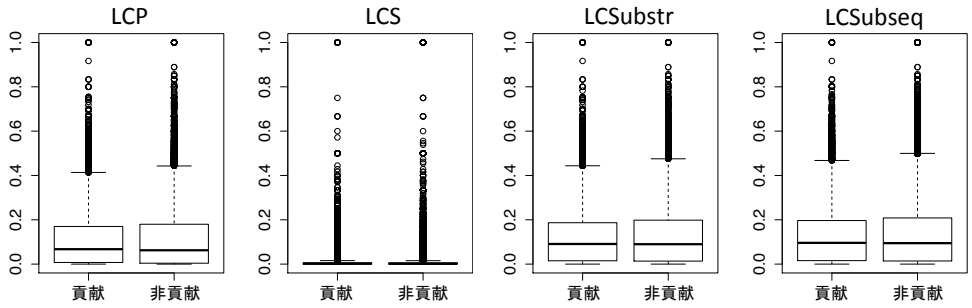

図 3　貢献するパッチと貢献しないパッチの ReviewSimilarity の分布（OpenStack）

チに関するレビュアー A とレビュアー B の経験を RevFinder によって比較する例を示す．レビュアー A は過去にパッチセット#1 を検証している．パッチセット#1 とパッチセット#3 の類似度が，ReviewSimilarity(#1, #3) = 0.6 とする．一方，レビュアー B はパッチセット#2 をレビューしている．パッチセット#2 とパッチセット#3 の類似度は，ReviewSimilarity(#2, #3) = 0.3 とする．類似度が高いほど，レビュー対象のソースコードを検証した経験が多いことを意味するため，RevFinder はレビュアー A を推薦する．このように RevFinder は全開発者に対して 4 種類の類似度計算方法を提案し，類似度が上位（依頼されたパッチのレビュー経験を最も多く持つ）のレビュアーを推薦する．本論文でも同様に 4 種類の類似度を使い，分析を行うが，類似度の計算方法は紙面の都合上割愛する．

3　パッチのレビュー経験を有する開発者の貢献の有無

3.1　データセットと分析概要

本論文では，従来研究で提案された手法によって推薦された開発者がレビュー依頼に貢献しているのか否かを，Qt プロジェクト，OpenStack プロジェクトを対象に確認する．両プロジェクトは，投稿されたパッチ，レビューの依頼，そして，パッチに対する議論，評価をレビュー管理システム Gerrit を利用して一元管理している．本論文では，McIntosh ら [5] が公開したデータセットを利用する．

本章では，レビュー依頼に貢献した開発者（貢献者）と依頼に貢献しなかった開発者（非貢献者）の ReviewSimilarity を比較し，マンホイットニーの U 検定を用いて，統計的な有意差があるか否かを確認する．

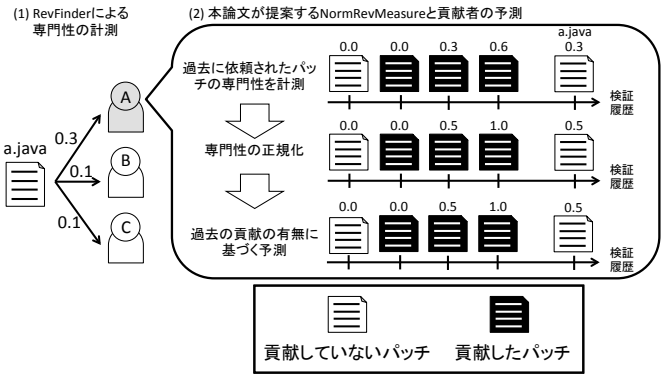

図 4　NormRevMesure の算出方法と予測の概要

3.2　分析結果

図 2，図 3 は，貢献者と非貢献者の ReviewSimilarity の分布を箱ひげ図で示している．また，箱ひげ図の上には Thongtanunam らが提案している ReviewSimilarity を求める 4 つの類似度計算手法を用いている．両プロジェクトでは，ReviewSimilarity に統計的優位差（マンホイットニーの U 検定を用いて有意水準 5%）を確認したが，これはサンプルサイズが大きいため有意差が出た可能性が高く，ReviewSimilarity が高いパッチのレビュー依頼に開発者は必ずしも貢献していないことが示唆される．

4　開発者のレビュー経験に基づく貢献の有無

従来手法では，レビュー対象のソースコードに対して適任のレビュアーを推薦することを目的としていた．しかし，3 の結果から，レビュー対象のソースコードに対して適切と判断されたレビュアーが必ずしも依頼に貢献するとは限らないことを明らかにした．本章では，レビュアーにとって過去によくレビューしたファイルやモジュールに関するパッチの依頼ほど貢献するか否かを調査する．

4.1　開発者のレビュー経験に基づく類似度計算手法

レビュアーのパッチへの興味の度合いが高いか否かを判断するために，従来研究と同様に RevFinder で計測する 4 種類の ReviewSimilarity を用いて，レビュアーが過去に依頼されたパッチから，レビュー対象のパッチの類似度を求める．しかし，依頼された開発者によって，ReviewSimilarity の値域が異なると考えられる．従って，本論文では，レビュアーにとってのパッチへの興味の度合い理解するために，過去に依頼されたパッチの ReviewSimilarity の分布に基づき正規化したスコア NormRevMesure を提案する．図 4 に正規化を行う際の算出方法の例を示す．

4.2　分析結果

図 5 はそれぞれ Qt プロジェクト，OpenStack プロジェクトにおける，レビュアーが貢献したパッチと貢献していないパッチの NormRevMesure の分布を表す．各プロジェクトでは Thongtanunam らが提案した 4 種類の ReviewSimilarity をそれぞれ用いて NormRevMesure を計測した結果を示す．プロットは 1 人のレビュアーを表し，横軸はレビュアーがレビュー依頼に貢献した全てのパッチの NormRevMeasure（中央値），縦軸はレビュアーがレビューに貢献していないパッチの NormRevMeasure（中央値）である．散布図には，全プロットを対象とした線形近似直線を示す． 近似線の傾きは全ての類似度計算手法において 1 未満であり，レビュアーは過去にレビューしたファイルやモジュールに関するパッチの依頼ほど優先的に貢献している

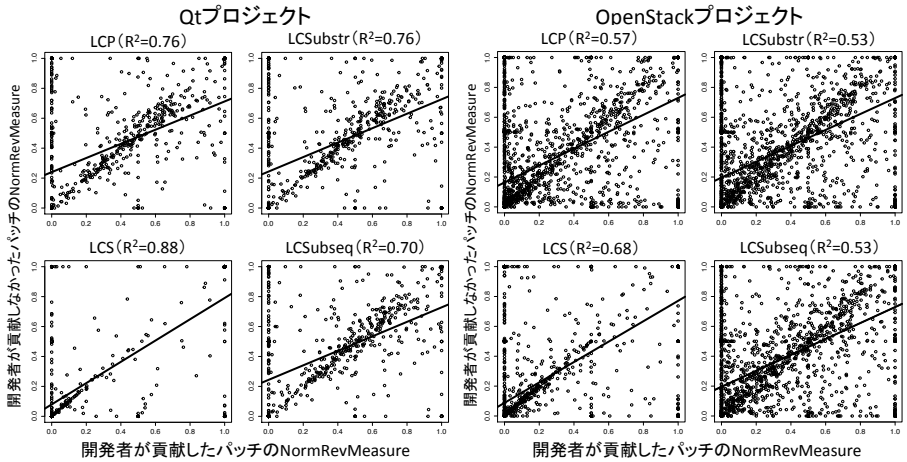

図 5　開発者が貢献した/貢献しなかったパッチの NormRevMesure の分布

ことが考えられる．しかし，傾きが 1 の付近に位置するレビュアー，また，過去にレビューしたファイルやモジュールに関するパッチの依頼にもかかわらず貢献していないレビュアー（散布図の左上）も存在していることを確認した．

5　コードレビュー依頼に貢献する開発者の予測

本論文で，レビュー対象ファイルをレビューした経験を有するレビュアー（ReviewSimilarity が高いスコアを持つレビュアー）であったとしても，依頼に貢献するとは限らないことを明らかにした．その一方で，レビュアーは，過去にレビューした経験の多いパッチ（NormRevMesure の値が高いパッチ）ほど貢献することを明らかにした．以上より，本論文では，NormRevMesure の値が高いレビュアーほど貢献するという方針で予測モデルを作成する．予測モデルを作成するにあたって Qt プロジェクトでは 2011 年 5 月～2011 年 12 月までのデータを用いてモデルを作り，2012 年 1 月～2013 年 11 月までのデータの予測を，OpenStack プロジェクトでは 2011 年 7 月～2011 年 12 月までのデータを用いてモデルを作り，2012 年 1 月～2014 年 5 月までの予測をそれぞれ行った．また評価指標として適合率，再現率，F1 値を用いてモデルの評価を行った．その結果を，表 1 に示す．

この予測結果を使い，依頼人数，予測した貢献者数，実際の貢献者数の関係について調査した．本調査では，各プロジェクトにおいて最も予測精度（F1 値）が高い手法（Qt プロジェクトにおいては LCS 手法，OpenStack プロジェクトでは LCP 手法）を用いて貢献者数を予測する．表 2 は Qt プロジェクトと OpenStack プロジェクトの依頼人数，予測した貢献者数（中央値，平均値），実際の貢献者数（中央値，平均値）を示す．

OpenStack プロジェクトの場合，2 人以上のレビュアーを確保するためには 2 人から 3 人のレビュアーにレビュー依頼する必要がある．その一方，Qt プロジェクトは一定人数以上のレビューが必要という制約がないため，実際の貢献者数が OpenStack プロジェクトよりも少なくなったと考えられる．また Qt プロジェクトも 2 人以上の人数に依頼した場合，予測人数は実際の貢献者数よりも多く予想してしまっている．我々が提案したモデルは Qt プロジェクトの場合，依頼人数が 1 人まで OpenStack プロジェクトにおいては 4 人までの人数予測には用いることができると考えられるが，依頼人数が多くなるにつれて精度が悪くなることも分かった．その理由として，実際のレビュープロセスでは，一定以上の依頼に対する貢献があり，明確な結論が

表 1　依頼人数による評価値

	Qt プロジェクト				OpenStack プロジェクト			
	LCP	LCS	LCSubstr	LCSubseq	LCP	LCS	LCSubstr	LCSubseq
適合率	0.51	0.44	0.45	0.45	0.61	0.60	0.61	0.60
再現率	0.43	0.78	0.69	0.69	0.82	0.76	0.83	0.82
F1 値	0.47	0.56	0.54	0.54	0.70	0.67	0.70	0.70

表 2　依頼人数に対する予測貢献者数と貢献者数

	Qt (LCS 手法)				OpenStack (LCP 手法)			
依頼人数	予測貢献者数（中央値）	貢献者数（中央値）	予測貢献者数（平均値）	貢献者数（平均値）	予測貢献者数（中央値）	貢献者数（中央値）	予測貢献者数（平均値）	貢献者数（平均値）
1	1	1	0.82	0.87	1	1	0.77	0.95
2	2	1	1.58	1.03	2	2	1.54	1.63
3	3	1	2.30	1.26	3	2	2.32	2.10
4	3	1	2.89	1.47	3	3	3.14	2.55
5	4	2	3.55	1.69	4	3	3.94	2.90

でている場合，それ以上貢献しないことも考えられる．

6　おわりに

本論文では，2 つの大規模 OSS (Qt，OpenStack) プロジェクトを対象に，レビュー対象ファイルをレビューした経験を有するレビュアーであったとしても，依頼に貢献するとは限らないことを明らかにした．その一方で，レビュアーは過去にレビューしたファイルやモジュールに関するパッチの依頼ほど貢献することを明らかにした．本論文では，レビュアーが過去に貢献した全てのレビュー依頼に基づき予測を行ったが，レビュアーは時間経過とともにレビュー依頼に貢献するパッチの傾向が変わっていくことが考えられる．例えば，開発者がレビュー経験のあるパッチのみ貢献していたレビュアーが経験のないパッチにも貢献するようになる可能性もある．今後は，レビュアーがレビュー依頼に対して貢献するパッチの変化についても調査する．

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] Androutsellis-Theotokis, S. and Spinellis, D.: Maria Kechagia, and Georgios Gousios. Open source software: A survey from 10,000 feet, *Foundations and Trends in Technology, Information and Operations Management*, pp. 187–347 (2011).

[2] Bacchelli, A. and Bird, C.: Expectations, Outcomes, and Challenges of Modern Code Review, *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*, pp. 712–721 (2013).

[3] Ghosh, R. A., Glott, R., Krieger, B. and Robles, G.: FLOSS: Survey of developers, *Free/Libre and Open Source Software: Survey and Study, FLOSS, Final Report. International Institute of Infonomics, Berlecom Research GmbH* (2002).

[4] Kollanus, S. and Koskinen, J.: Survey of Software Inspection Research, *The Open Source Engineering*, Vol. 3, No. 1, pp. 15–34 (2009).

[5] McIntosh, S., Kamei, Y., Adams, B. and Hassan, A. E.: The Impact of Code Review Coverage and Code Review Participation on Software Quality: A Case Study of the Qt, VTK, and ITK Projects, *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*, pp. 192–201 (2014).

[6] Tao, Y., Han, D. and Kim, S.: Writing Acceptable Patches: An Empirical Study of Open Source Project Patches, *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME'14)*, pp. 271–280 (2014).

[7] Thongtanunam, P., Tantithamthavorn, C., Kula, R. G., 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 International Conference on Software Analysis, Evolution, and Reengineering (SANER'15)*, pp. 141–150 (2015).