情報処理学会論文誌 Vol.57 No.10 1–5 (Oct. 2016)

レビュア間の合意形成と不具合再修正に関する一考察 〜OpenStackプロジェクトを対象としたケーススタディ〜

林 宏徳[1,a)] 伊原 彰紀[1,b)] 松本 健一[1,c)]

受付日 xxxx年0月xx日, 採録日 xxxx年0月xx日

概要：本論文では，オープンソースソフトウェア（OSS）開発における不具合修正プロセス中の検証フェーズに着目し，開発者（レビュア）間の合意形成と不具合再修正の関係についてOpenStackプロジェクトを事例に分析を行った．分析の結果，プログラム変更内容に対してレビュアの合意が得られない（賛成，および，反対するレビュアがともにいる）にもかかわらずプロダクトに統合された場合，再修正が必要となる可能性が高いことが確認された．

Toward Identifying a Re-opened Bug Based on Reviewers Disagreement -Case Study of OpenStack Project-

Hayashi Hironori[1,a)] Ihara Akinori[1,b)] Matsumoto Ken-ichi[1,c)]

Received: xx xx, xxxx, Accepted: xx xx, xxxx

Abstract: This paper conducted an empirical study to identify a re-opened bug based on reviewers disagreement. Using OpenStack project dataset as case study, we found patches that reviewers did not reach consensus are likely to fix again after integrating them.

Keywords: Code Review, Consensus, Re-opened bug, Open Source Software

1. はじめに

OSSプロジェクトには，日々数十件から数百件の不具合が報告されており，その約6%〜26%は再修正を必要とする [3]．再修正とは，一度修正されたはずの不具合が将来のリリースで再び発見/修正することを指す．OSS開発者は長くても1年程度しかプロジェクトに在籍しないことが多く，OSSにおける不具合再修正は，必ずしも一度目に修正した開発者によって行われるとは限らない．不具合の修正作業を適切な開発者に割り当てるためにも，再修正の早期特定が求められる．

Shihabら [3] は，一度目の修正完了時に将来の再修正の有無を予測するためのモデルを開発しており，ソースコードの変更内容を検証する開発者（レビュア）間のコミュニケーション量，議論の内容が不具合再修正予測モデルの精度向上に強く影響していることを明らかにした．しかし，Shihabらの研究では，具体的なレビュア間の議論の内容まで調査が行われておらず，再修正を防ぐための具体的な知見が得られていない．その理由は，不具合修正における議論の内容が当時はリポジトリに記録されておらず，詳細な分析まで実施できなかったためである．しかし，昨今のOSS開発では，レビュー管理システムが使用されることが多くなり，ソースコードレビュー中の開発者の議論や合意形成のデータが取得できるようになった．

本論文では，不具合管理システム，および，レビュー管理システムの両方を長期間利用しているOpenStackプロジェクトを対象に，レビュア間の合意形成プロセスが再修正と関係しているのかを調査する．

[1] 奈良先端科学技術大学院大学
Nara Institute Science and Technology
Takayama, Ikoma-shi, Nara 630–0192, Japan
[a)] hironori-ha@is.naist.jp
[b)] akinori-i@is.naist.jp
[c)] matumoto@is.naist.jp

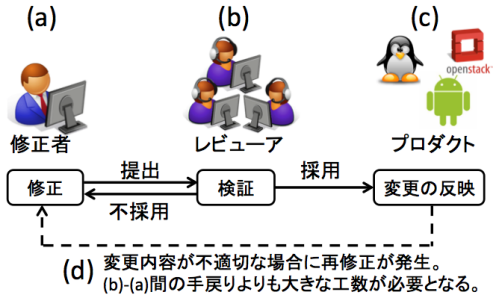


図 1 OSS 開発における不具合修正プロセス
Fig. 1 Bug fixing process in OSS project.

2. 不具合修正における検証作業

2.1 不具合修正プロセス

不具合修正は，不具合がプロジェクトに報告された後，ソースコードが修正され，修正がプロダクトに反映されるまでの一連のフェーズからなる．修正されたソースコードが検証作業を経てプロダクトに反映されるまでのプロセスを図 1 に示し，各フェーズにおける開発者の役割を説明する．

(a) 修正：修正依頼を受けた開発者（修正者）が不具合の修正を行う．
(b) 検証：修正されたソースコード（以下，パッチ）の妥当性をレビュアが確認する．パッチが不適切と判断されると，修正者に再度パッチの作成が依頼される．
(c) 修正の反映：妥当性が確認されたパッチはプロダクトに反映される．
(d) 再修正の依頼：不具合修正が不適切であった場合，再修正が依頼される．当初の修正者と異なる開発者に作業を割り当てざるを得ないこともあり，(b)-(a) 間の手戻りよりも修正に大きな工数を要する [3,6].

2.2 レビュア間の合意形成

検証フェーズでは，レビュアがパッチを精読し，その妥当性についての議論を行う [1]．レビュアが，不完全な変更内容を見落とすと，リリース後に再度不具合報告が届き，再修正を要することがある．近年，OSS 開発では，パッチの検証をするために Gerrit Code Review(以下，Gerrit) [*1]や Reviewboard [*2]等のレビュー管理システムの利用が増加している．レビュー管理システムでは，投稿されたパッチを検証するためにレビュー票が作成され，レビュー票には，新たなパッチがプロダクトに反映されるまでの進捗や，修正者とレビュアの議論が記録される．

通常，投稿されたパッチは，レビュア間の意見が一致した上でプロダクトに反映が決定されるが，レビュー中に議論が分かれ，パッチに対する否定的な意見が考慮されずに，プロダクトに反映される事がある．OpenStack プロジェクトのレビュー票 2,539 件を調査した結果，188 件 (約 8%) のパッチは否定的な評価がついたまま（つまり，合意が形成されないまま），プロダクトに反映されていた．パッチに対する否定的な意見が正しい場合，当該不具合は再修正を要する可能性がある．本論文では，検証フェーズにおけるレビュア間の協調作業の実態，および，合意形成と再修正との関係について分析する．

表 1 レビュアの評価範囲
Table 1 Reviewer's voting.

	評価範囲
一般レビュア	-1, 0, +1
コアレビュア	-2, -1, 0, +1, +2

3. 開発者の協調作業と再修正の分析

3.1 分析対象

OpenStack プロジェクトは，不具合を管理するために不具合管理システム Launchpad.net [*3]，レビュー作業を管理するためにレビュー管理システム Gerrit [*4]を利用している．本論文では，Hamasaki らが公開している Openstack プロジェクトのレビュー記録 [2] から，Launchpad.net 上の不具合票と関連付けられており，且つ，プロダクトに反映されたことのある 2,539 件のレビュー票を分析する．

レビュー管理システム Gerrit では，投稿されたパッチに対して，レビュアが表 1 の通り 5 段階の評価を投稿する．

当該プロジェクトでは，コアレビュアと一般レビュアの 2 種類のレビュアが存在し，コアレビュアは 5 段階全ての評価を付ける権限を持つが，一般レビュアには “+2”，“-2” の評価を付ける権限がない．たとえ一般レビュアが否定的な評価（“-1” 等）を付けていたとしても，コアレビュアが “+2” の評価をつけた場合，パッチはプロダクトに反映可能である．ただし，レビュアによる評価は加算的に扱われないため，2 名のレビュアが “+1” の評価をつけたとしても “+2” として扱われない．したがって，パッチをプロダクトに反映するためには，コアレビュアによる評価が必須である．

本論文では，不具合修正のために作成されたパッチが，レビュー管理システムで議論され，プロダクトに反映された後，当該パッチの再修正が決定したか否かを Launchpad に記録された修正プロセスから判定する．

3.2 Research Questions

本論文では，検証フェーズにおけるレビュア間の合意が形成される過程を分析し，レビュア間の協調作業と再修正との関係を調査するため次の 2 つのリサーチクエスチョン

*1 https://code.google.com/p/gerrit/
*2 http://www.reviewboard.org/
*3 Launchpad.net: `https://bugs.launchpad.net/`
*4 Gerrit: `https://code.google.com/p/gerrit/`

表 2 レビューの合意形成と不具合の再修正
Table 2 Re-opened bug fixed after reaching consensus or not among reviewers.

合意形成	レビュー票の件数	再修正の発生件数	再修正の発生率
あり	2,321	367	15.8%
なし	188	47	25.0%

（RQ1, RQ2）に取り組む．

RQ1: 合意が形成されない場合に再修正が発生するのか？

プロダクトに反映されたパッチの中には，一般レビュアの否定的な評価が無視され，レビュアが合意に至らないままプロダクトに反映されるものがある．本 RQ では，一般レビュアからの否定的な意見にかかわらず，コアレビュアによって問題ないと判断されれば再修正が発生しないのかを明らかにする．

分析方法: 検証対象のパッチに対して，一般レビュアもしくはコアレビュアの否定的な評価（-1 または-2）がつけられたままプロダクトに反映された場合を「合意形成なし」，最終的に全てのレビュアが肯定的な評価（+1 または+2）をつけてプロダクトに反映された場合を「合意形成あり」と定義する．プロダクトに反映されたのち，再び修正が必要となったか否かを不具合管理システムの修正履歴から調査することによって，レビュア間の合意と再修正の関係を明らかにする．

分析結果: 表 2 は，レビュー管理システムに投稿されたパッチに対する，「合意形成あり」と「合意形成なし」のレビュー件数，再修正の発生件数と発生率（=再修正の発生件数/レビュー票の件数）を示す．「合意形成なし」のレビューは「合意形成あり」のレビュー より再修正の発生率が約 10%高く，$\chi 2$ 検定によって統計的有意差（有意水準 5%）を確認できたことから，レビューにおける合意形成の有無は，再修正の発生に関係があると示唆される．

RQ2: 不具合再修正を防ぐためにレビュアができることは？

投稿されたパッチが妥当でない場合，レビュアは，修正者に判断理由を説明し，パッチの修正を求める．しかしながら，一般開発者の不適切なパッチへの意見に誰も回答しない場合がある事が 2.2 節にて分かった．レビュアが合意を形成する議論を調査し，不具合を見逃さないための指針を立てる．

分析方法: 本論文の著者らがレビュー票中に記載されているコメント欄をレビュアや修正者の議論を観察し，「合意形成あり」「合意形成なし」それぞれの特徴を抽出する．また，「合意形成なし」のレビュー票については，188 件全てのコメント欄を読解し，そのレビューの特徴を層別することで，より具体的な問題特定を試みる．

図 2 合意が得られた議論（左: 事例 1）と合意が得られなかった議論（右: 事例 2）
Fig. 2 Discussion sample to reach consensus (left), Discussion sample not to reach consensus (right).

分析結果: 合意に基づきパッチがプロダクトに反映されたレビューの議論の例[*5]を図 2（左），合意が得られずにパッチが反映されたレビューの議論の例[*6]を図 2（右）に示す．

事例 1 は，先ず Dev8-A のパッチ投稿に対して Dev8-C が “+2” の評価を付け，プロダクトへの反映が可能となった．しかしながら，Dev8-G がパッチに問題があるとし，“-2” の評価を付けたため，Dev8-A はパッチを修正し，再投稿した．Dev8-G は問題が解決されたことを確認し，評価を “+2” に変更した．その後，パッチをプロダクトに反映した結果，再修正は発生しなかった．

一方で事例 2 は，Dev10-A のパッチ投稿に対して Dev10-C がパッチの書き方の問題を指摘し，評価 “-1” を付けた．これに対し，Dev10-A は過去の開発例を挙げ，パッチを修正しない旨の発言をしている．Dev10-A はコアレビュアの意見に従うと述べ，Dev10-C は評価を変えなかった．その後にレビューを行ったコアレビュア Dev10-G は “+2” の評価を付けるのみで Dev10-C の意見には言及しなかった．最終的に Dev10-C は “-1” の評価を取り消すことのないまま，パッチはプロダクトに反映された．事例 2 のレビュー（図 2 右）では他に 3 人のレビュアが肯定的な評価を付けているが，後に再修正が発生している．

合意形成が行われなかった全レビュー票 188 件に対し，レビュアのコメントを読解した結果，合意形成に向けた議論が行われないレビューが散見された．具体的には，パッチに対してポジティブな評価（+1 または +2）とネガティ

*5 https://review.openstack.org/#/c/4643/
*6 https://review.openstack.org/#/c/5220/

表 3 レビュアコメントと再修正発生率
Table 3 Re-opened bug fixed after reviewers disagreement.

レビュアコメント	レビュー票の件数	再修正の発生件数	再修正の発生率
あり	166	33	19.9%
なし	22	14	63.6%

ブな評価（-1 または -2）の両方が与えられているにも関わらず，その評価理由や，他のレビュアに対する意見などは一切述べられず，パッチについて議論が行われた様子がないままコアレビュアがパッチをプロダクトに反映しているものなどを指す．合意形成が行われず，レビュア間で議論されたレビュー票を「レビュアコメントあり」，議論されなかったレビュー票を「レビュアコメントなし」として層別し，再修正の発生件数との関係を表 3 の通りまとめた．

「合意形成なし」のレビュー 188 件中の「レビュアコメントなし」は 22 件（全体の約 15%）であるが，「レビュアコメントあり」の場合に比べて再修正発生率が 3 倍以上高く，χ^2 検定によって統計的有意差（有意水準 5%）が確認された．再修正の発生を防ぐためには開発者間の合意形成が必要であることは前節にて述べた通りであるが，合意形成がなかったとしてもレビュアコメントを通じて意見を交わす必要があると示唆される．

4. 考察

4.1 検証フェーズの議論

本節では RQ2 の分析結果で示した 2 つの議論事例について再修正の観点から考察する．

図 2 事例 1 の議論では，Dev8-G の指摘に対して Dev8-A がパッチの修正を行ったことで合意が形成された．一方で，図 2 事例 2 の議論では，Dev10-C の否定的な意見に対する反応はなく，Dev10-G は “+2” の評価を付け，パッチをプロダクトに反映したが，Dev10-C が投稿したコメントに対して言及しておらず，肯定的な評価をつけた意図を汲み取る事は出来ない．このように，新たなパッチに対する指摘内容が反映されているコメントは，開発者間の意思疎通を促し，再修正の予防に繋がると示唆される．

また図 2 事例 2 では，再修正の発生によって，当該バグを修正する為に 5 日間（1 回目のレビューが終了してから 2 回目のレビューが終了するまで）経過している．修正の遅延は，新たな欠陥をソースコードに混入させる可能性があるため，意思疎通を伴う正確なレビューが求められる．

5. おわりに

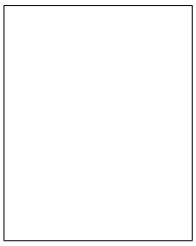
OSS 開発では，開発者間の協調作業による迅速な不具合修正が求められる一方で，不具合修正プロセスにおける検証フェーズでの不適切なコードの見逃しによって膨大な手戻りの工数が発生している．本論文では，OSS 開発における再修正コストの削減に向けて，レビュー管理システムにおけるレビュア間の合意形成と不具合再修正の関係を調査した．その結果，合意形成が得られないままプロダクトに反映されたパッチは後に再修正を必要とする可能性が高い事が分かった．また，合意が形成される過程において，修正者とレビュア間で議論を重ね，レビューにおける合意を得ることが再修正の発生を低減させることがわかった．

従来まで，不具合修正に関するパッチのコードレビューは不具合管理システムで議論されることが多かった [4,5]．不具合管理システムはコードレビューに特化したシステムではなく，議論はメーリングリストや IRC ですることが促され，過去のコードレビューの振り返りを困難にさせていた．昨今，Gerrit のようなレビュー管理システムが利用されるようになり，本論文で示したパッチに対するレビュア間の合意形成に関する調査を実施することができた．本論文では，再修正が発生する原因として，レビュア間の合意形成に着目したが，今後は，レビュアの知識や経験，パッチの難しさから，レビュアの意見の信憑性を検討する．

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．

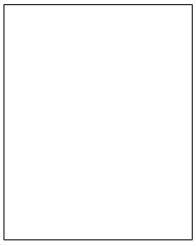
参考文献

[1] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. In *IEEE Transactions on Software Engineering*, volume 22, pages 751–761, 1996.
[2] K. Hamasaki, R. G. Kula, N. Yoshida, A. E. C. Cruz, K. Fujiwara, and H. Iida. Who does what during a code review? datasets of oss peer review repositories. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR'13)*, MSR '13, pages 49–52, 2013.
[3] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, O. Ohira, Masao, B. Adams, A. E. Hassan, and K.-i. Matsumoto. Studying re-opened bugs in open source software. In *Empirical Software Engineering*, pages 1–38, 2013.
[4] Y. Tao, D. Han, and S. Kim. Writing acceptable patches: An empirical study of open source project patches. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME'14)*, pages 271–280, 2014.
[5] P. Weißgerber, D. Neu, and S. Diehl. Small patches get in! In *Proceedings of the 5th International Working Conference on Mining Software Repositories (MSR'08)*, pages 67–76, 2008.
[6] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy. Characterizing and predicting which bugs get reopened. In *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, pages 1074–1083, 2012.



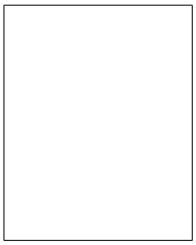
林 宏徳

2012 年岐阜工業高等専門学校電子システム工学専攻修了．同年奈良先端科学技術大学院大学入学．2014 年同大学修士課程修了．同年トヨタ自動車株式会社入社．修士（工学）．ソフトウェア工学の研究に従事．



伊原 彰紀 （正会員）

2007 年龍谷大学理工学部卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2012 年同大学博士課程修了．同年同大学情報科学研究科・助教．博士（工学）．ソフトウェア工学，特にオープンソースソフトウェア開発・利用支援の研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE 各会員．



松本 健一 （正会員）

1985 年大阪大学基礎工学部情報工学科卒業．1989 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．1993 年奈良先端科学技術大学院大学助教授．2001 年同大学教授．工学博士．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集／利用支援の研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．