OSS開発におけるレビュアー間の合意形成の分析

林 宏徳[1,a)]　伊原 彰紀[1,b)]　松本 健一[1,c)]

概要：近年，オープンソースソフトウェアの高機能化により，不具合修正のために関連する複数機能の同時変更が必要となる場合が増加している．複数機能を変更するために各機能の担当者は意思疎通を図りながら協調作業を行う．しかしながら，先行研究において，変更されたソースコードのプロダクトへの反映に複数の開発者が関わった場合，誤った修正を発見できず手戻り（再修正）が発生する可能性が高くなることが分かった．これは，開発者にとって担当外の機能に対する理解不足や不十分なレビュー等が原因の一つと示唆される．本稿では，複数の開発者によるレビューの合意形成をはじめとする協調作業を分析し，再修正への影響を明らかにする．Openstack プロジェクトを対象にケーススタディを行った結果，レビューを行った開発者の中で一人でも合意していない場合，全員が合意した場合よりも再修正の発生する可能性が高いことが分かった．

キーワード：オープンソースソフトウェア，不具合修正プロセス，レビュー，協調作業

An Analysis of Consensus among Developers in a Review Phase of OSS Developments: A Case Study of Openstack Project

Abstract: Various functions of large-scale OSS projects make more complex bug-fixes and lead to collaboration between developers. Our previous study revealed that re-open bugs are likely to be due to the collaboration at a commit phase in bug-fix process. We consider that an insufficient consensus in collaboration may be a one of the causes. In this paper, we investigate the collaborations in a review phase to know how we can take approaches to avoid the re-opened bug-fixes. Results of our case study using Openstack project found that a consensus of developers in a review phase is important to avoid the re-opened bug-fixes.

Keywords: Open Source Software, Bug-fix Process, Review, Collaboration

1. はじめに

近年，Apache や Linux に代表されるオープンソースソフトウェア（OSS）が官公庁，教育機関だけでなく，商用ソフトウェアの一部にも利用されるようになった [10]．その結果，OSS プロジェクトは多くのユーザから不具合報告を得ることができるようになった [2][9]．

しかしながら，約 15%の不具合は正しく修正されておらず，リリース後に再度修正（再修正）が必要となり，手戻りが発生している [16]．大規模 OSS 開発における不具合修正は，再修正を必要としない場合に平均 149 日，再修正を必要とする場合に平均 371 日を要している．修正の遅延を解決するために再修正を未然に防ぐことは早急の課題といえる．

OSS の高機能化に伴い，不具合のために関連する複数機能の修正が必要となる機会が増えたため，各機能の担当者が意思疎通を図りながら協調作業を行うことが必要不可欠となっている [14][21]．その一方で，開発者が地理的に分散している OSS 開発の協調作業は困難な場合が多いことも指摘されている [1][11]．先行研究において，修正されたソースコードのコミットに複数の開発者が関わると，ソフトウェアリリース後に再修正が発生する可能性が高くなることが分かった [8][20]．これは，修正されたソースコードの検証（レビュー）において，開発者間の不十分な合意形成が再修正を引き起こす一つの原因であると示唆される．

本稿では，大規模 OSS プロジェクト（Openstack プロ

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

ジェクト）を対象に複数の開発者によるレビューの合意形成をはじめとする協調作業を分析し，再修正への影響を明らかにする．

開発者によるレビューの合意形成を分析するにあたり，本稿では，レビュー管理リポジトリを積極的に利用している大規模 OSS (Openstack) プロジェクトを対象に以下のリサーチクエスチョンに取り組む．

RQ1:開発者はレビューのために協調作業をどの程度実施しているのか？

RQ2:修正されたソースコードはレビュアー間の合意の上でプロダクトに反映されているのか？

RQ3:レビュアー間の合意形成が再修正の有無に影響しているのか？

開発者によるレビューの合意形成を分析することで，レビューの協調作業の必要性，及び，レビュアー間の合意形成が再修正に及ぼす影響の理解を促し，再修正を未然に防ぐ方法論の確立が期待できる．

本稿の構成は以下の通りである．続く 2 章では分析対象となる不具合の修正プロセスとレビューについて説明し，3 章では本稿におけるリサーチクエスチョンについて述べる．4 章ではケーススタディについて説明すると共に結果を示し，5 章で考察を行う．6 章で関連研究についてまとめ，最後に 7 章で本稿のまとめと今後の課題について述べる．

2. OSS 開発における不具合管理とレビュー

本章では，OSS 開発における不具合の修正プロセスと当該プロセスにおけるレビューについて説明する．

2.1 不具合修正プロセス

不具合修正プロセスは，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作業からなる．図 1 に代表的な不具合修正プロセスを示す [6][19]．

(a) 報告者が不具合の情報（バージョン，機能名，重要度等）を不具合管理システムに登録．
(b) 管理者が不具合の修正を適切な開発者（修正者）に依頼．
(c) 修正者が不具合を修正．
(d) 変更されたソースコードについて，レビュアーが正しく修正されたと判断した場合，コミッター (プロダクトに変更されたソースコードを反映する権限を持った開発者) がプロダクトに変更を反映．

もしレビュアーが誤った修正を発見できなかった場合，欠陥が混入したソースコードがプロダクトに反映されるため，再修正が必要となる．

先行研究 [21] において，複数の開発者がコミットに関わる場合，再修正が発生する可能性が高いことが分かった．

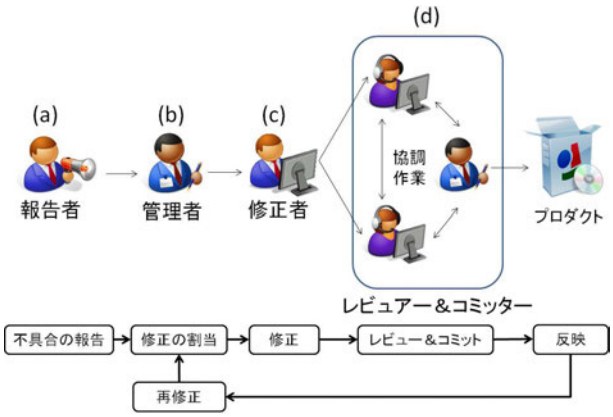

図 1 代表的な不具合修正プロセス

本稿では修正されたソースコードのレビューにおける開発者間の不十分な合意形成が再修正を引き起こす一つの原因と考え，レビューにおける開発者の協調作業について分析する．

2.2 レビュー

レビューは，開発中のプログラムのソースコードを精読する作業である．レビューの目的はプログラムの開発過程において混入されたソースコード中の欠陥を発見することである [4]．レビュアーが欠陥を発見した場合，再度ソースコードの修正が行われる．しかしながら，ソースコード中の欠陥をレビュアーが見過ごしたままプロダクトに反映した場合，リリース後に不具合報告が届き，再修正が実施されるため，完全に修正されるまでに時間がかかる．

OSS 開発においてレビューは，複数人で行われることが多く [5][13][14]，各レビュアーは修正されたソースコードを評価する．ただし，各レビュアーには異なる権限が与えられているため，レビュアーによって発言の影響力が異なる [7]．例えば，レビュアーが変更されたソースコードに対して否定的な評価を付けたとする．その後，当該レビュアーが持つ権限より強い権限を有する別のレビュアーが支持した場合，当該ソースコードはプロダクトに反映される可能性がある．もし否定的な意見が正しい場合，当該不具合は，後に再修正としてプロジェクトに報告されると考えられる．

本稿では，開発者間の不十分な合意形成が再修正を引き起こす原因と考え，レビューにおけるレビュアー間の合意形成をはじめとする協調作業を分析し，再修正との関係を分析する．

3. リサーチクエスチョン

レビューにおける合意と不具合の再修正について明らかにするため，本章では，3 つのリサーチクエスチョン（以下，RQ）を述べる．

RQ1: 開発者はレビューのために協調作業をどの程度実施しているのか？

OSS 開発におけるレビューには複数人のレビュアーが関わることが多い [5][13][14]．RQ1 では，Openstack プロジェクトのレビューにおける協調作業がどの程度行われているのかを確認する．(1) レビューされるファイル数，(2) レビュアーの人数，(3) レビュアーの活動量に着目し定量的に調査することで OSS 開発における協調作業の実態を明らかにする．

RQ2: 修正されたソースコードはレビュアー間の合意の上でプロダクトに反映されているのか？

OSS 開発における不具合修正では，修正されたソースコードに対して，複数人のレビュアーが評価を行う．前章で説明した通り，否定的な評価が付けられていたとしても，強い権限を持つレビュアーが支持する場合，修正されたソースコードはプロダクトに反映される可能性がある．RQ2 では，このような場合を合意形成に失敗したレビューとし，全レビュアーが支持する場合を成功したレビューとし，レビュアー間の合意形成を分析する．

RQ3: レビュアー間の合意形成が再修正の有無に影響しているのか？

先行研究 [21] の結果から，再修正はレビューにおける開発者間の不十分な合意形成が原因の一つと考えられる．RQ3 では，RQ2 において合意形成を成功/失敗したレビューが後に再修正されたか否かを分析し，合意に基づく協調作業が必要であるか否か明らかにする．

4. ケーススタディ

本章では，レビュアーの協調作業の実態を明らかにするために，Openstack プロジェクトを対象にケーススタディを行った結果を述べる．

4.1 分析対象データ

4.1.1 Openstack プロジェクト

Openstack プロジェクトは 2010 年に Rackspace Cloud と NASA によって始められた IaaS クラウドコンピューティングプロジェクトである．AMD，Intel，IBM をはじめとする 150 社以上が参画しており，ユーザ数も増加傾向にある．IaaS 型のサービスが一定のユーザ数を維持するためにはサービスの稼働率が重要であるため，不具合発生時には早急な対処が求められる．従って，Openstack プロジェクトでは不具合修正における再修正を防ぐことは重要な課題と考えられる．

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 [*1]，レビューの進捗などを管理するためにレビュー管理システム Gerrit Code Review(以下, Gerrit) [*2] を利用している．本稿では, Hamasaki らが提供するレビューデータ [7] の中から，不具合修正のために実施され，且つ，一回以上プロダクトに反映された 2,539 件のレビューデータを対象にケーススタディを行う．

表 1　レビュアーの評価範囲

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

4.1.2 Gerrit Code Review

Gerrit は Google Inc. から提供されるウェブベースのレビュー管理システムであり，新規開発，または，変更されたソースコードがプロジェクトに投稿されてからレビューを経てプロダクトに反映するまでの情報（進捗，議論）がシステムに記録される．開発者は誰でもソースコードを Gerrit にアップロードすることができ，レビュアーは投稿されたソースコードを評価する．また，開発者やレビュアーはコメント欄において議論をすることができる．

Gerrit では，レビュアーが 5 段階の評価（”+2”, ”+1”, ”0”, ”-1”, ”-2”）を付けることができ，”+2”の評価が付けられると，コミッターが変更されたソースコードをプロダクトに反映することができる．評価値は加算的に扱われず，レビュアーの評価度合いを表す．たとえ 2 名のレビュアーが”+1”の評価を付けたとしても”+2”として扱われないため，コミット可能な状態にはならない．

Openstack プロジェクトでは，レビュアーを 2 種類（コアレビュアー，一般レビュアー）に区別することができる．表 1 は，Gerrit における各レビュアーが与えることのできる評価範囲を示す．一般レビュアーは”+2”，”-2”の評価を付ける権限がなく，プロジェクトから権限が与えられている一部のレビュアー（コアレビュアー）は 5 段階全ての評価を付けることができる．レビュアーが投稿されたソースコードに対して”+2”の評価を付けた場合，たとえ”-1”等の評価が他のレビュアーによって付けられていたとしてもコミッターが承認すればプロダクトに反映される．

図 2 は Gerrit のレビュー結果画面を示す．図 2 下部の表はレビュー結果であり，David Kranz が”+1”，Christopher MacGown が”-1”そして Mark McLoughlin が”+2”(レビュー結果画面では，”+2”はチェックで表示される) の評価を付けている．レビュアーの一人が”+2”をつけたため，コミッター (i.e., Mark McLoughlin) がプロダクトに変更を反映する（Approved にチェックで表示される）．

*1 https://bugs.launchpad.net/
*2 https://code.google.com/p/gerrit/
*4 Change ID: I7aa4b539393df15f3b2c950cf7aeca4691ed3d73
https://review.openstack.org/#/c/6921/

図 2 Gerrit のレビュー結果画面[*3]

表 2 各レビューの基本統計量

	最小値	中央値	平均値	最大値
ファイル数	0	1	2.93	164
レビュアー数	1	4	4.41	18
コメント数	0	4	6.37	62

4.2 分析

本節では各 RQ に答えるための分析方法，及び，分析結果について述べる．

RQ1: 開発者はレビューのために協調作業をどの程度実施しているのか？

分析方法

Gerrit から，レビュー対象となるファイル数，レビュー毎のレビュアー数，レビュアーのコメント数を取得する．

本稿の分析対象ファイルは.py ファイル，および，.java ファイルのみとする．また，コメント数の計測は，レビュアーが投稿したコメントを取得し，Gerrit Trigger Jenkins Plugin によって自動投稿されるコメントは除外する．

分析結果

表 2 に各レビューのファイル数，レビュアー数，コメント数の基本統計量を示す．Openstack プロジェクトでは，不具合修正後，1~ 2 ファイルがレビューされ，中には 164 ファイルのソースコードがレビューされることもある．また，各レビューに対してレビュアーが約 4 人参加し，約 4~6 件のコメントが投稿されている．多くのレビューでは複数人のレビュアーが協調作業を行っていることが分かる．

先行研究 [21] と同様に，レビューされるファイル数に応じて，協調作業が実施されるか（複数のレビュアーが参加しているか）否かを確認するために，図 3 にレビューされるファイル数とレビュアー数の関係を箱ひげ図で示す．横軸は各レビューで対象となったファイル数，縦軸はレビュアー数を示す．レビュー対象となるファイル数に関係なく，レビュアー数は約 4 人であった．レビュー対象となるファイル数に依存せず，たとえレビュー対象のファイル数が少なくても一定のレビュアーが協調作業を行っていることが明らかになった．

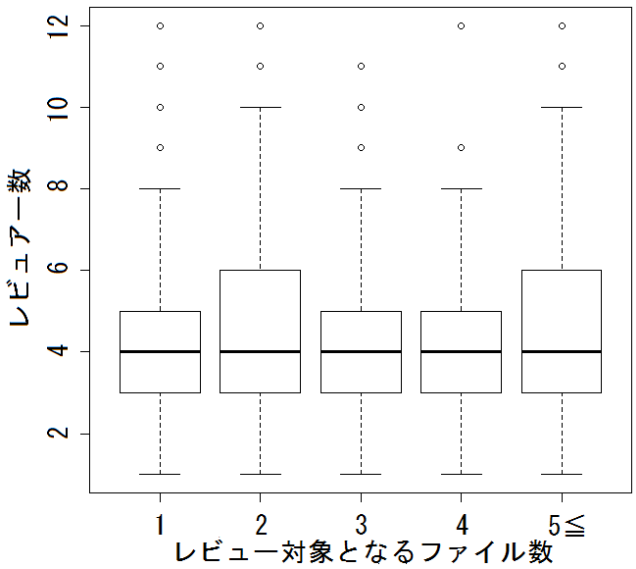

図 3 レビュー毎のファイル数と開発者数

RQ1 の回答: レビュー対象ファイル数に関係なく，約 4 人のレビュアーが協調作業を行っている．

RQ2: 修正されたソースコードはレビュアー間の合意の上でプロダクトに反映されているのか？

分析方法

Gerrit の各レビュー結果から各レビュアーがつけた評価を集計する．レビュアーの中で一人でも”-1”もしくは”-2”の評価を付けたままプロダクトに反映されていれば，合意形成を失敗したレビューとして扱い，それ以外を合意形成に成功したレビューと判断する．合意形成に成功/失敗したレビュー数を比較する．

分析結果

図 4 は，対象レビュー 2,539 件においてレビュアーが付けた各評価の総数を表す度数分布を示す．レビュアーは”+2”もしくは”0”の評価を多く付けている一方，否定的な評価 (i.e., ”-1”, ”-2”) を付けているケースは 235 件 (”-1”が 183 件 (約 5%)，”-2”が 52 件 (約 3%)) であった．

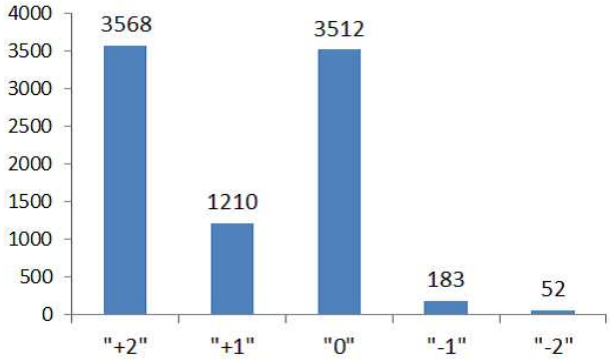

図 4 レビュアーに付けられたレビュー評価

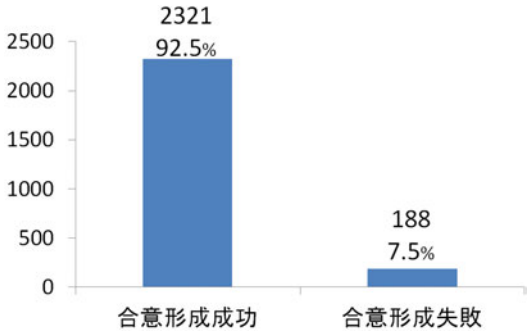

図 5 合意形成の成功/失敗したレビュー件数

表 3 レビューの合意形成と不具合の再修正

合意形成	レビュー件数	再修正の発生件数	再修正の発生率
成功	2,321	367	15.81%
失敗	188	47	25.00%

図 5 は，合意形成が成功/失敗したレビューの件数を示す．全てのレビューが合意形成に成功しているわけではなく，188 件（約 8%）のレビューは合意形成に失敗していることが分かる．

RQ2 の回答: 188 件（約 8%）のレビューは合意形成に失敗しているにもかかわらずプロダクトに反映されている．

RQ3: レビュアー間の合意形成が再修正の有無に影響しているのか？

分析方法

レビューに関連付けられた不具合管理システムの情報から，修正が完了した（変更内容がプロダクトに反映された）回数を計測し，2 回以上修正が完了した不具合を再修正が発生した不具合とする．RQ2 で分類されたレビューについて，再修正の有無を調査することで，レビューにおける合意と再修正の関係を明らかにする．

分析結果

表 3 は，レビューにおける合意形成が成功した場合と失敗した場合の再修正の発生率（=再修正の発生件数/レビュー件数）を示す．合意形成が失敗したレビューは合意形成に成功したレビューより約 10%多く再修正が発生しており，カイ二乗検定によって統計的有意差（有意水準 1%）が認められた．従って，レビューにおける合意形成は，再修正に影響していることが明らかになった．

RQ3 の回答: レビュアー間の合意形成は，再修正の有無に影響する．

5. 考察

5.1 レビューにおける合意形成と不具合の再修正

ケーススタディから，合意形成を失敗した場合，再修正が発生する可能性が高いことが分かった．本節では合意形成の事例を示す．図 6 は再修正が発生したレビューにおける合意形成の議論（事例 1[*5]），図 8 は再修正が発生しなかったレビューにおける合意形成の議論（事例 2 [*6]）を示す．

事例 1 は，Peng がパッチ（変更されたソースコード）を投稿し，それに対して Willian がパッチの書き方について否定的な意見を述べるとともに評価”-1”を付けている．Peng の返答に対して Willian はコアレビュアーの意見に従うと述べているが，”-1”を取り消すことなかった．また，このレビューには他に 3 人のレビュアーが肯定的な評価を付けている（図 7）．この後，変更されたソースコードはプロダクトに反映されたが，後に再修正が発生している．合意形成に失敗し，再修正が発生した典型的な例である．

事例 2 は，Alex がパッチを投稿し，それに対して Johannes（コアレビュアー）が”+2”の評価を付けているため，プロダクトへの反映が可能となった．しかしながら，Vish（コアレビュアー）が次のコメントでデシリアライズに関する安全性に問題があるとし，”-2”の評価を付けたため，Alex は Vish の意見を基に新たにパッチを投稿した．Vish は新たなパッチで問題が解決されたと判断し，”+2”に評価を付け変えた．その後，新たな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．このレビューにおける最新パッチに対する評価は図 9 の通りである．

事例 1 で，レビュアー間の合意形成が失敗した時，事例 2 のように再度変更内容を検討していれば再修正が発生しなかったかもしれない．レビュアー間で合意形成できていない場合，コアレビュアーはもちろん，パッチ投稿者もレビュアーと再度議論することが必要であると示唆される．

5.2 結果の妥当性

本稿では，再修正が発生する原因として，開発者間の合意形成をはじめとする協調作業に着目した．実際に開発者の不十分な合意形成が原因で再修正を引き起こした事例は存在していた．しかしながら，再修正はその他にも原因がある可能性がある．先行研究において，再修正は単に変更量に依存していないことが明らかにしたが [8]，その他の要因も考えられる．開発スキルを正確に判断することは容易ではないが，今後の研究で開発者の経験（プロジェクトでの活動期間など）を開発スキルとするなどが考えられる．今後，その他の要因についても検討する．

[*5] Change ID: I5a0c975ab7998627a213ac4c69c037e9e2d95bfa https://review.openstack.org/#/c/5220/
[*6] Change ID: Ife3b64b19fe8abbc730184d4ee7d9fcabfd29db3 https://review.openstack.org/#/c/5749/

Peng Young (Mar 12, 2012)
Uploaded patch set 5.

Willian Molinari (Mar 13, 2012)
Patch Set 5: I would prefer that you didn't submit this. This code is deprecated on the new versions of sqlalchemy:[URL] The new way to implement it:[URL]

Peng Young (Mar 14, 2012)
Patch Set 5.
@Willian Molinari, please refer to comments in:[URL]

Willian Molinari (Mar 14, 2012)
Patch Set 5:
Let's see what the quantum core reviews would say about it. I don't know why not to implement on the non-deprecated way, but if they prefer to do the same as melange (accept and change it on Folsom) I'll remove my -1.*** (中略) ***

Salvatore Orland (Mar 14, 2012)
Patch Set 5: Looks good to me (core reviewer)

図 6 事例 1：合意形成が行われなかった議論例

Reviewer	*Verified*	*Code-Review*	*Approved*
Peng Yong			
Jenkins	+1		
Willian Molinari		-1	
dan wendlandt		+1	
Brad Hall		+1	
Edgar Magana		✓	
Salvatore Orlando		✓	✓

図 7 事例 1: レビューの評価

本稿では Openstack プロジェクトを対象にケーススタディを行ったが，クラウドサービスという歴史の浅いプロジェクトの性質上，多くのデータセットを確保することが容易ではなかった．近年，レビュー管理システムを採用し，公開している OSS プロジェクトは増加傾向にあるが，十分なデータセットを公開しているプロジェクトは少ない．

Peter ら [14] は，OSS や商用ソフトウェアを含めた 13 のプロジェクトにおけるレビューを対象に，レビュー期間，レビュアーの人数等の観点からプロジェクトを比較している．本稿の RQ1 の分析結果と，Peter らが行ったレビューの調査結果を比較すると，Openstack プロジェクトのレビュアー数，コメント数は同程度であるため，他のプロジェクトでも同じような協調作業が行われていると考えられる．従って，他のプロジェクトを対象にケーススタディを行っても，同様の結果が得られると考える．

Alex Meade (Mar 27, 2012)
Uploaded patch set 3.

Johannes Erdfelt (Mar 27, 2012)
Patch Set 3: Looks good to me (core reviewer)

Vish Ishaya (Mar 27, 2012)
Patch Set 3: Do not submit
I think the deserialization is still unsafe. I think we should probably wrap all base exceptions in a NovaException and force a check that the incoming exception module is an allowed class before constructing it. Do a quick test to verify that it is insecure as is.

*** (中略) ***

Alex Meade (Mar 29, 2012)
Uploaded patch set 4.

Vish Ishaya (Apr 3, 2012)
Patch Set 4: Looks good to me (core reviewer)
Ok I'm satisfied. Thanks alex.

図 8 事例 2：合意形成が行われた議論例

Reviewer	*Verified*	*Code-Review*	*Approved*
Alex Meade			
Sandy Walsh		+1	
Johannes Erdfelt		✓	
Chris Behrens		✓	✓
Jenkins	+1		
SmokeStack			
Vish Ishaya			

図 9 事例 2: レビューの評価

6. 関連研究

6.1 不具合の再修正に関する研究

ソフトウェア開発において，プロジェクトは保守作業に膨大なコストをかけているため，ソフトウェア工学の分野では不具合修正に関する研究が盛んに行われている．例えば，ソフトウェアの不具合の混入箇所の特定に関する研究 [18]，修正担当者の決定方法に関する研究 [2]，再修正不具合の予測技術に関する研究 [16] などが挙げられる．特に不具合修正プロセスのにおける再修正は，修正コストの増大や修正期間の延期につながるため大きな課題となっている．Shihab らは，不具合修正プロセスを 4 つの観点（作業習慣，不具合の特徴，不具合修正，作業者）から調査し，どの変数が再修正と深い関係を持つかを明らかにした [16]．その結果，不具合管理システムにおける開発者間のコミュニケーション情報が再修正に影響していることを分析から

明らかにした．また，調査した変数を用いて構築した再修正を予測するモデルは，84%~ 90%の予測精度を実現した．しかしながら，再修正の発生を防ぐために開発者が具体的に取るべき行動については明らかにしていない．従って，本稿では再修正を防ぐための開発者の行動指針を示した．

6.2 レビューにおける協調作業に関する研究

近年，レビューに関する研究も数多く取り組まれている [3][12][15]．Yang らの研究では [17]，レビューにおける協調作業に関する研究を行っている．Android プロジェクトのレビューデータを Gerrit から取得し，レビュアーのソーシャルネットワーク分析を行うことでレビュアー間の関係を定量的に明らかにした．Yang らの研究は，本研究と同様にレビューにおける協調作業ついて分析を行っているが，不具合の再修正という観点については分析されておらず，本研究と目的が異なる．

7. おわりに

本稿では，OSS の不具合修正において，レビューの協調作業と再修正の関係について Openstack プロジェクトを対象に分析した．本稿で得られた知見を以下に示す．

- レビュー対象となるファイル数に関係なく，複数人の開発者によってレビューが実施される．
- レビューは必ずしも全員の合意の下で行われているわけではない．Openstack プロジェクトでは約 3%の否定的なレビュアーの意見が無視されており，合意形成が失敗したレビューが約 8%存在する．
- 合意形成が成功した不具合修正に比べ，合意形成が失敗した場合は，後に再修正が発生する可能性が高い．再修正を防ぐために，合意形成が失敗した場合は，再度変更内容を検討すべきである．

今回のケーススタディでは Openstack プロジェクトのみを対象としたが，今後は，より結果の妥当性を向上させるために，異なるプロジェクトを対象に同様の分析を行う予定である．また，本稿で，レビューにおける協調作業について定量的に調査を行い，議論内容について事例を示した．今後は議論内容の観点から合意形成の種類を分類し，再修正を事前に防ぐ方法論について言及したい．

謝辞 本研究の一部は，文部科学省科学研究補助費（挑戦的萌芽: 25540026，若手 B: 課題番号 25730045，基盤 B: 課題番号 23300009）による助成を受けた．

参考文献

[1] R. Abreu and R. Premraj. How developer communication frequency relates to bug introducing changes. In *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution and software evolution workshops(IWPSE'09)*, pages 153–158, 2009.

[2] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, pages 361–370, 2006.

[3] A. Bacchelli and C. Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*, pages 712–721, 2013.

[4] 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.

[5] D. B. Bisant and J. R. Lyle. A two-person inspection method to improve programming productivity. *IEEE Transactions on Software Engineering*, 15(10):1294–1304, 1989.

[6] W. Cathrin, P. Rahul, Z. Thomas, and Z. Andreas. How long will it take to fix this bug? In *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07)*, pages 1–8, 2007.

[7] 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 International Workshop on Mining Software Repositories (MSR'13)*, pages 49–52, 2013.

[8] H. Hayashi, A. Ihara, A. Monden, and K. Matsumoto. Why collaboration is needed in oss project?: A case study of eclipse project. In *Proceedings of the 5th International Workshop on Social Software Engineering (SSE'13)*, pages 17–20, 2013.

[9] P. Hooimeijer and W. Weimer. Modeling bug report quality. In *Proceedings of the 22nd International Conference on Automated Software Engineering (ASE'07)*, pages 34–43, 2007.

[10] IPA(独立行政法人 情報処理推進機構). **第 *3* 回オープンソースソフトウエア活用ビジネス実態調査** *(2009* **年度調査***)*. 2009.

[11] S. Matsumoto, Y. Kamei, M. Ohira, and K. Matsumoto. A comparison study on the coordination between developers and users in foss communities. In *Proceedings of the Socio-Technical Congruence (STC'08)*, pages 1–9, 2008.

[12] M. Mukadam, C. Bird, and P. C. Rigby. Gerrit source code review data from android. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*, pages 45–48, 2013.

[13] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly and Associates, 1999.

[14] P. C. Rigby and C. Bird. Convergent contemporary software peer review practices. In *Proceedings of the 9th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC/FSE'13)*, pages 202–212, 2013.

[15] P. C. Rigby and M.-A. Storey. Understanding broadcast based peer review on open source software projects. In *Proceedings of the 33th International Conference on Software Engineering (ICSE'11)*, pages 541–550, 2011.

[16] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K. Matsumoto. Studying re-opened bugs in open source software. *Journal of Empirical Software Engineering*, 18(5):1005–1042, 2012.

[17] X. Yang, R. G. Kula, C. C. A. Erica, N. Yoshida,

K. Hamasaki, K. Fujiwara, and H. Iida. Understanding oss peer review roles in peer review social network (PeRSoN). In *Proceedings of the 19th Asia-Pacific Software Conference (APSEC'12)*, pages 709–712, 2012.

[18] J. Zhou, H. Zhang, and D. Lo. Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports. In *Proceedings of the International Conference on Software Engineering (ICSE'12)*, pages 14–24, 2012.

[19] 伊原彰紀, 大平雅雄, and 松本健一. OSS 開発における不具合修正プロセスの現状と課題：不具合修正時間の短縮化へ向けた分析. **情報社会学会誌**, 6(2):1–12, 11 2011.

[20] 林宏徳, 伊原彰紀, 門田暁人, and 松本健一. OSS 開発におけるコミッターによる協調作業の一考察. In **研究報告グループウェアとネットワークサービス**（*GN*）, pages 1–4, 2013.

[21] 林宏徳, 伊原彰紀, 門田暁人, and 松本健一. OSS 開発における一般開発者の協調作業と不具合の再修正に関する一考察. In **マルチメディア，分散，協調とモバイルシンポジウム論文集** *(DICOMO'13)*, pages 1704–1709, 2013.