Vol.2014-SE-184 No.6
Vol.2014-EMB-33 No.6
2014/5/20

パッチレビュープロセスによるパッチ作成者の継続性の違い

大坂 陽[1,a)]　伊原 彰紀[2]　亀井 靖高[1]　松本 健一[2]　鵜林 尚靖[1]

概要：OSS 開発では，開発者が作成したコード（パッチ）に対して，コア開発者が検証するパッチレビューの一連の作業（パッチレビュープロセス）を繰り返して，品質の高いパッチだけがプロダクトに反映される．しかしながら，全てのパッチが検証されるとは限らず，検証されずに長期間対応されない場合，開発者のプロジェクトに対するモチベーションが低下することが考えられる．本研究では，データセットとして Git プロジェクトを対象として，パッチレビュープロセスと開発者の継続性との関係を分析した結果，1) パッチレビュープロセスによりプロジェクトに取り込まれた割合は約 26%である，2) 全開発者の約 30%が継続的に開発を行い，この開発者が全パッチの約 80%を作成している，3) 初回パッチがレビューされるか否かよりも，初回パッチが取り込まれるか否かのほうが，開発者の継続性に影響を与える可能性が高い，ということがわかった．

Osaka Ataru[1,a)]　Akinori Ihara[2]　Kamei Yasutaka[1]　Kenichi Matsumoto[2]　Ubayashi Naoyasu[1]

1. はじめに

コスト削減や開発スピードの向上を目的に商用ソフトウェアの開発に高品質なオープンソースソフトウェア（OSS）を利用する企業が増加している．また，情報処理推進機構の調査 [1] によると，商用ソフトウェア開発事業の約 50%は OSS 関連事業であることが報告されていることからも，OSS に対する注目の高さがうかがえる．その一方で，緊急時の技術的サポートが得にくい，セキュリティに対する OSS コミュニティの対応に不安があるという点で OSS 利用に不安を抱える企業も少なくない．

OSS 開発者はプロジェクトへの参加・離脱が自由であるため，全てのモジュールが継続して同じ開発者に保守されるとは限らない．保守担当のモジュールに対する開発経験が過去にない場合，開発者はモジュールの理解に時間がかかったり，誤った理解に基づく修正を行うことで欠陥を混入させてしまったりする可能性が高い．従って，同一開発者のプロジェクト対する継続的な貢献は，OSS の品質維持に求められる要因の 1 つである．

本研究では，プロジェクトに継続的に貢献する開発者の特徴を明らかにすることを目的とし，パッチレビュープロセスと開発者の継続性との関係を分析する．OSS 開発では，開発者が作成したパッチ（不具合修正や機能追加のために作成したソースコード差分）[2] に対して，コア開発者によるパッチレビューに関連する一連の作業（パッチレビュープロセス）を繰り返す．そして，パッチレビュープロセスを通して得られる品質の高いパッチだけを開発プロジェクトの成果物（ソフトウェア）に反映する．しかしながら，全てのパッチがレビューされるとは限らず，レビューされずに長期間対応されない場合，開発者のプロジェクトに対するモチベーションが低下することが考えられる．

本稿では，パッチレビュープロセスが開発者の継続性へ与える影響を調べるために，次の 3 つのリサーチクエスチョンに答える．

RQ1: パッチレビュープロセスによりプロジェクトに取り込まれたパッチはどの程度あるのか． パッチ開発者は自身が作成したパッチがプロジェクトに反映されることで，そのプロジェクトへの開発のモチベーションを維持し，継続的に開発に取り組むと考えられる．RQ1 では，どの程度のパッチがプロジェクトに取り込まれたかを調べるために，パッチとプロジェクトのコミットログとのリンク付けを行った．

RQ2: 継続的にパッチを投稿している開発者はどの程度いるのか． OSS 開発では，開発者のプロジェクトへの参

[1] 九州大学
Kyushu Uniersity
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] osaka@posl.ait.kyushu-u..ac.jp

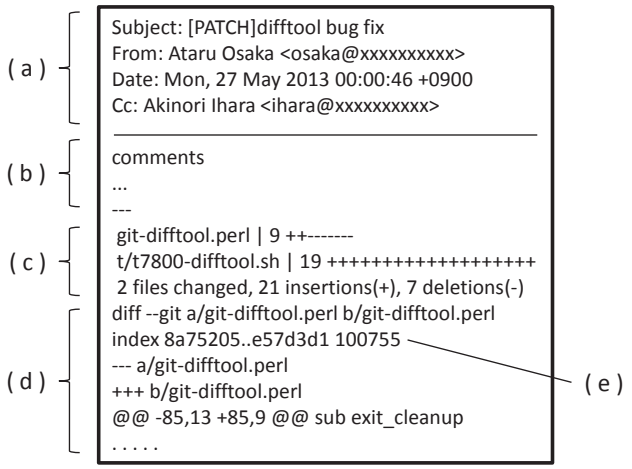

図 1 Git プロジェクトのポリシーに基づいたパッチの投稿例

加・離脱が自由であるため，継続的に開発を行う開発者が少なく，OSS の品質維持において問題となっている．RQ2 では，パッチ開発に継続的に貢献している開発者はどの程度いるのか調べるために，パッチの投稿回数と開発者の関係を調べた．

RQ3: レビューの有無は開発者の継続性に影響を与えるのか． 開発者がメーリングリストにパッチを投稿しても見逃されて，レビューアからレビューを受けられないことがある [3]．この様なパッチはプロジェクトに取り込まれにくく，また，取り込まれない事実はパッチ開発者のモチベーションを下げ，開発者の継続性を低下させると考えられる．RQ3 では，レビューの有無による開発者の継続性への影響を調べた．

以降，第 2 章では，OSS 開発におけるパッチについて説明する．第 3 章では，本研究で扱った 3 つの RQ によるケーススタディを説明する．第 4 章では関連研究として，パッチレビュープロセスと OSS 開発者の活動期間について述べる．最後に第 5 章においてまとめて今後の展望について述べる．

2. OSS 開発

2.1 パッチ

OSS 開発では，Git や CVS などの版管理システムによってソースコードが管理され，バグ修正や機能拡張のためのソースコードの変更前後の差分コードであるパッチによって，ソースコードの修正が行われる [4]．パッチには，図 1 に示すように，変更したファイルとその変更箇所の差分やどのコミットから変更を加えたか，などの情報が付加されている．

(a) メーリングリストに投稿されたメールの情報で，件名，投稿者（パッチ開発者），日付などがある．

(b) メールの本文で，コミット[*1]のコメントとしてコミットログに書き込まれる．

(c) パッチで変更されたファイルの統計情報で，どのファイルがどの程度変更されたかがわかる．

(d) パッチで変更された各ファイルの詳細な変更箇所を示す．

(e) そのファイルが以前に変更されたときのリモートリポジトリのコミットハッシュ，変更されたあとのローカルリポジトリのコミットハッシュ，及び，ファイルのパーミッションの情報が記されている．

本研究で分析対象とする Git プロジェクトでは，パッチの送信方法についてドキュメントが用意されている．Git プロジェクトでは，図 1 に示すように，パッチを最小限の機能ごとに分割し，メールに添付せずメール本文に載せる．

[*1] 版管理システムにおける変更のまとまり

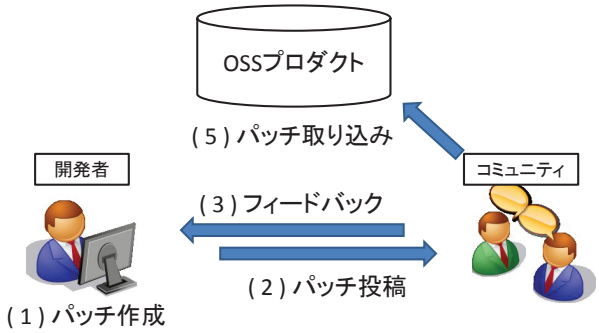

図 2 OSS 開発におけるパッチが取り込まれるまでの流れ

2.2 パッチがコードリポジトリに取り込まれるまでの流れ

図 2 は，OSS 開発におけるパッチ投稿から，プロダクト（成果物）に取り込まれるまでの流れを説明する．

(1) OSS 開発者は不具合の報告を受けると，当該不具合を修正するためにパッチを作成する．
(2) 作成したパッチを開発コミュニティに知らせるためメーリングリストに投稿する．
(3) 他の開発者のレビューを受け，フィードバックを得る．
(4) 投稿されたパッチが適切でないと判断された場合，(1) から (3) を繰り返す．
(5) 投稿されたパッチが適切であると判断された場合，そのパッチはコミッター[*2]によってプロダクトに取り込まれる．

OSS 開発では，パッチ投稿にメーリングリストが使用されることが多い [2]．それぞれの OSS コミュニティでメールの書き方のドキュメントが用意されている．パッチがそのドキュメントに準拠して作成されており，かつ，パッチの内容が適切であると判断された場合に，その OSS 開発のプロダクト（版管理システム）に取り込まれる．例えば，

[*2] OSS プロジェクトからバージョン管理システムにソースコードをアップロードする権限を与えられた開発者．大規模なプロジェクトに参加するコミッターはプロジェクトに参加する開発者の 5%にも満たない

表 1 データセットの統計

プロジェクト	コミット数	開発者数	コミットの期間	メール数	スレッド数	送信者数	メーリングリストの期間
Git	34,781	1,198	2005.4.7-2013.9.20	29,303	3,465	1,123	2012.9.9-2013.9.19

本研究で使用したメーリングリストでは，パッチはファイルを添付するのではなく本文に書くことや，パッチに対する説明文の書き方などが示されていた．

3. ケーススタディ

3.1 概要

本研究では，プロジェクトに継続的に貢献する開発者の特徴を明らかにすることを目的とする．そのために，Git プロジェクトをケーススタディのデータセットとして扱い，3 つのリサーチクエスチョンに取り組んだ．まず，本ケーススタディで用いるデータセットについて述べ，次に各 RQ の動機，アプローチ，および，結果について述べる．

3.2 対象プロジェクトとデータセット

本研究では，活発に開発が進められているプロジェクトである Git プロジェクトをケーススタディの対象とした．Git は，OSS で開発されている分散型版管理システムであり，2005 年から Linux の創始者の Linus らによって開発が行われている．

本研究では，メーリングリストに投稿されたパッチがプロジェクトに取り込まれたことを判定するために，メーリングリストだけでなく，Git プロジェクトのソースコードリポジトリが必要である．そのため，GitHub から git リポジトリをクローンし[*3]，メーリングリストのデータを Web から wget コマンドで取得した[*4]．今回使用した Git のプロジェクトは，データを取得した日時から一年前までのメールしか残されておらず，より多くのデータを扱うことは今後の課題である．

本ケーススタディで用いた Git プロジェクトのデータセットの統計を表 1 に示す．表中のスレッドは，あるパッチ投稿に対する一連のメールのやりとり（パッチレビュープロセス）を指す．コミット数は約 8 年間で 34,781 コミットで，メーリングリストの期間だけに焦点を絞ると，5,104 コミットであった．メール数はメーリングリストに投稿されたパッチに対するレビューのメールも数に入っている．

3.3 本研究で取り組む RQ

3.3.1 RQ1: パッチレビュープロセスによりプロジェクトに取り込まれたパッチはどの程度あるのか

動機．パッチ開発者は自身が作成したパッチがプロジェクトに反映されることで，そのプロジェクトへの開発のモチベーションを維持し，継続的に開発に取り組むと考えられる．RQ1 では，パッチレビュープロセスによりプロジェクトに取り組まれたパッチはどの程度あるのか調べるために，Git リポジトリのコミットログとメーリングリストに投稿されたパッチとのリンク付けを行った．

表 2 パッチが git リポジトリに取り込まれた数と割合（%）

	メール	スレッド
取り込まれた数	2,194	925
取り込まれた割合（%）	7.49	26.70

表 3 パッチ投稿回数による開発者数とスレッド数

	パッチ投稿回数		
	合計	1 回	2 回以上
開発者数	951	647	304
割合（%）	100	68.03	31.97
スレッド数	3,465	647	2,818
割合（%）	100	18.67	81.33
取り込まれたスレッド数	925	95	830
割合（%）	100	10.27	89.73

アプローチ．Git リポジトリでは，コミットを一意に識別するためにコミットハッシュがある．メール本文内のパッチには，開発者のローカルリポジトリにおけるコミットハッシュ（図 1 の (e)）は記述されている．しかしながら，OSS プロジェクトに取り込まれた際にはパッチの情報（誰がコミットを行ったなど）が更新され，コミットハッシュの値も変わってしまうため，メールに投稿されたパッチが git リポジトリに取り込まれたか否かを判断するために用いることができない．

本ケーススタディでは，パッチ投稿の日付と送信者の名前と，コミットログの Author の日付と名前が一致した場合，パッチがプロジェクトに取り込まれたと判断した．

送信者の名前は重複の恐れがあるため一意に定まるメールアドレスを使用したかったが，今回使用したデータセットでは，メールアドレスの悪用を防ぐためにドメイン名が消されていたため，使用できなかった．

結果と考察．表 2 にパッチが git リポジトリに取り込まれた数と割合を示す．メールの数がスレッドの数よりも多い理由は，一つのスレッドのやり取りの中で複数のパッチが生成され，プロジェクトに取り込まれた場合（つまり，パッチレビュープロセス）があるためである．パッチレビュープロセスを経て最終的に git リポジトリに取り込まれたか否かの結果を，スレッド単位の結果として示す．

各個別のメールに着目すると，git リポジトリに取り込まれた割合は 7.49%であった．結果が 10%未満と極端に小さい割合になった理由としては，レビューなどのフィードバックを受けて改訂を求められるメール（パッチ）は git リ

[*3] git clone https://github.com/git/git
[*4] wget http://www.spinics.net/lists/git/threads.html

表 4　レビュー無による開発者数とスレッド数

	投稿回数	
	1 回	2 回以上
開発者数	647	304
初回パッチがレビューされる割合（%）	527 81.45	263 86.51
初回パッチが取り込まれる割合（%）	95 14.68	72 23.68

ポジトリに取り込まれることがないためであると考える．

その一方で，スレッド単位の結果に着目すると，投稿されたパッチの約 26%が取り込まれていることがわかった．また，個別のメールの結果よりも，取り込まれる割合が大きいことから，パッチはレビュープロセスを経てプロジェクトに組み込まれていることがわかった．

3.3.2 RQ2: 継続的にパッチを投稿している開発者はどの程度いるのか

動機．OSS 開発では，開発者のプロジェクトへの参加・離脱が自由であるため，継続的に開発を行う開発者が少なく，OSS の品質維持において問題となっている．そこで，継続的にパッチを投稿している開発者が何%いるのかを調べた．

アプローチ．パッチ開発に継続的に貢献している開発者はどの程度いるのか調べるために，開発者をパッチの投稿回数が 1 回か 2 回以上かで分類し，開発者数，スレッド数，プロジェクトに取り込まれたスレッド数の違いを調べた．

結果と考察．表 3 にパッチ投稿回数と開発者数の関係を示す．表 1 の送信者数と表 3 の開発者数が違うのは，送信者数にはレビューだけを行うレビューアの数が含まれているためである．

継続的にパッチを投稿している開発者は約 30%(= 304/951) で，その約 30%の開発者が 80%以上 (= 2,818/3,465) のパッチを作成し，その多くがプロジェクトに取り込まれている (約 90%= 830/925)．この知見は，Apache と Mozilla の開発者の貢献度合いを調査した Mockus ら [5] の，2 割の開発者が 8 割のソースコードの修正を行っているという調査結果と一致している．

3.3.3 RQ3: レビューの有無は開発者の継続性に影響を与えるのか

動機．開発者がメーリングリストにパッチを投稿しても見逃されて，レビューアからレビューを受けられないことがある [3]．このようなパッチはプロジェクトに取り込まれにくく，また，パッチ開発者のモチベーションを下げ，開発者の継続性を低下させると考えられる．そこで，レビューを受けていないパッチについて調べた．

アプローチ．レビューの有無による開発者の継続性への影響を調べるために，1 回のみパッチを投稿した開発者と，2 回以上パッチを投稿した開発者の間で，初回投稿時のパッチがどの程度の割合でレビューを受けたかを調べた．

結果と考察．表 4 に初回にレビューを受けていない開発者の数を示す．表 4 に示す通り，1 回だけでパッチ投稿を終えた開発者では，82%の初回パッチがレビューを受けている一方で，2 回以上継続してパッチを投稿し続けた開発者では，87%のパッチがレビューを受けていた．両者の間に大きい関係がないことから，初回投稿時に対するレビューの有無は継続性に大きい影響を与えるわけではないことがわかった．

一方で，初回パッチが取り込まれるか否かは，1 回だけでパッチ投稿を終えた開発者と継続してパッチ投稿を続けた開発者では，15%と 24%であり，レビューの有無よりは影響を与えている可能性が高いことがわかった．

4. 関連研究

4.1 パッチレビュープロセスの分析

OSS 開発におけるパッチの投稿，組織内での開発者の役割についての研究がこれまで盛んに行われている [2][6]．Weißgerber らは，開発者に承認されやすいパッチを提示するために，FLAC と OpenAFS のプロジェクトでパッチが承認される割合，承認されやすいパッチの行数，そして，パッチが承認されるまでの時間について調査している [2]．調査の結果，行数の小さなパッチはアクセプトされやすいが，アクセプトされるまでの時間は行数に影響されないことが明らかとなった．Weißgerber らはパッチが投稿されてからパッチが承認されるまでの時間を分析している．継続的にプロジェクトに貢献する開発者を選定するためには，パッチが投稿されてから承認されるまでの開発者の活動をより詳細に分析する必要がある．

また，Jensen らは Mozilla，Apache，NetBeans を対象に，OSS プロジェクト内における複雑な組織の構造と，開発者の昇格プロセスを明らかにした [6]．例えば，Apache プロジェクトについて，開発者がパッチ投稿から，コミッターへの推薦，PMC メンバーの投票，承認を経てコミッターへ昇格するまでのプロセスと開発プロセスを併せてモデル化している．本論文が対象とするパッチレビュープロセスは Jensen らが提示する開発プロセスの一部を詳細に調査したものであり，本研究を用いることで開発者の特徴別に昇格プロセスをモデル化することが期待される．

4.2 OSS 開発者の活動期間の分析

OSS プロジェクトでは，世界中の不特定多数のボランティア開発者が専門知識を共有し，高品質なソフトウェアの開発を目指している．しかしながら，ボランティア開発者が自由に開発したソースコードをソフトウェアに反映することは，ソフトウェアの品質低下につながる恐れがある．ソースコードの品質を判断できるコミッターをはじめとする経験豊富な開発者が OSS プロジェクトには必要である [7]．

Bird ら [8] は，PostgreSQL プロジェクトにおける開発

者の活動期間の分析を通して，約 1 年間の活動実績がある一般開発者はコミッターに昇格する開発者として適切であると結論付けている．開発者の活動期間が 1 年よりも短い場合，コミッターとしてふさわしいかどうかを判断するのは難しく，また，1 年より長い場合，開発者はモチベーションを失ってしまう危険性が高いことを挙げている．また，Zhou ら [9] は，開発者が長期的にプロジェクトに貢献する開発者を特定するためには，何ヵ月の活動を見る必要があるかを分析している．その結果，大規模プロジェクトでは 12 ヵ月間，小規模プロジェクトでは数ヶ月間，貢献数が増加し続けていれば，開発者は長期的にプロジェクトに貢献する傾向を示している．Bird ら，Zhou らの研究から，数ヶ月でプロジェクトを離脱する開発者が多いことが示されているが，具体的な活動内容やプロジェクトを離脱した具体的な理由については議論されていない．本論文では，開発者がプロジェクトに継続的に貢献する理由を明らかにする．

5. おわりに

本稿では，プロジェクトに継続的に貢献する開発者の特徴を明らかにすることを目的とし，パッチレビュープロセスと開発者の継続性との関係について分析を行った．Git プロジェクトを対象に行ったケーススタディの結果，下記の知見が得られた．

- パッチはレビュープロセスを経てプロジェクトに取り込まれ，その取り込まれる割合は約 26%である．
- パッチを継続的に投稿している開発者は，全開発者のうち約 30%であり，この 30%の開発者が全パッチのうちの 80%を作成している．
- 初回パッチがレビューされるか否かよりも，初回パッチが取り込まれるか否かのほうが，開発者の継続性に影響を与える可能性が高い．

今後の課題としては，パッチレビュープロセスの解析のための適切なデータセットの選択が挙げられる．本ケーススタディで使用したデータセットはメーリングリストのデータが約 1 年分しか残っておらず，より正確性を求めるために，全期間のメーリングリストのデータが取得できることが望ましい．また，プロジェクトが異なると，メーリングリストや版管理システムの仕様も異なってくるため，そのプロジェクトにあった解析方法をとる必要がある．

謝辞 本研究の一部は，日本学術振興会 科学研究費補助金（若手 B：課題番号 25730045，若手 A：課題番号 24680003，挑戦的萌芽：課題番号 25540026）による助成を受けた．

参考文献

[1] 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).
[2] Weißgerber, P., Neu, D. and Diehl, S.: Small patches get in!, *Proceedings of the International Working Conference on Mining Software Repositories (MSR'08)*, pp. 67–76 (2008).
[3] Nurolahzade, M., Nasehi, S. M., Khandkar, S. H. and Rawal, S.: The Role of Patch Review in Software Evolution: An Analysis of the Mozilla Firefox, *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution and Software Evolution Workshops*, pp. 9–18 (online), DOI: 10.1145/1595808.1595813 (2009).
[4] Bird, C., Gourley, A. and Devanbu, P.: Detecting Patch Submission and Acceptance in OSS Projects, *Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR'07)*, pp. 26–30 (2007).
[5] Mockus, A., Fielding, R. T. and Herbsleb, J. D.: Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 11, No. 3, pp. 309–346 (2002).
[6] Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proceedings of the International Conference on Software Engineering (ICSE '07)*, pp. 364–374 (2007).
[7] Fogel, K.: *Producing Open Source Software: How To Run Sucessful Free Software Project*, O'Reilly Media, Sebastopol, CA (2005).
[8] Christian Bird, Alex Gourley, P. D. A. S. and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proceedings of the International Workshop on Mining Software Repositories (MSR' 07)*, p. 6 (2007).
[9] Zhou, M. and Mockus, A.: Developer Fluency: Achieving True Mastery in Software Projects, *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, New York, NY, USA, ACM, pp. 137–146 (online), DOI: 10.1145/1882291.1882313 (2010).