B-05

OSS 開発におけるパッチレビュープロセスの効率化に向けたコミッターの分類

藤田 将司†　伊原 彰紀†　大平 雅雄†　松本 健一†
Shoji Fujita　Akinori Ihara　Masao Ohira　Ken-ichi Matsumoto

1．　はじめに

近年，オープンソースソフトウェア（OSS）は，個人ユーザの利用のみならず，企業における業務での利用が進んでいる．OSS が広く普及するにつれて，OSS の社会的重要性は高まってきており，発見された不具合に対して迅速な対応が強く求められるようになってきている．

一般的な OSS プロジェクトでは，不具合の修正は次のように行われる．(1) 利用者あるいは開発者が，発見した不具合をプロジェクトが運用するメーリングリスト (ML) あるいは不具合管理システム (Bug Tracking System: BTS) に報告する．(2) 不具合報告を受けた開発者らは，不具合を修正するためのパッチ（ソースコード差分）を作成し ML や BTS へ投稿する．(3) 投稿されたパッチは，他の開発者からレビューを受け，(4) 妥当性が確認されれば正式に承認されプロダクトに反映される．

OSS プロジェクトによっては膨大な数の不具合が報告されるという現状[3]もあり，不具合修正の効率化を目的とした研究が盛んに行われている．例えば，上述の (1) に関連するものとして，開発者が不具合修正に取り掛かり易い不具合報告の書き方について調べた研究[5][7]などがある．また，(2)に関連するものとして，開発者が投稿するパッチの特徴を調べた研究[4][7]などがある．

一方，(3), (4)に関連するパッチのレビュープロセスそのものの効率化や短縮化を指向した研究はほとんど見当たらない．パッチが投稿されてもレビューされずに放置されたり[6]，レビュー後に妥当性が確認されているにも関わらず承認されずプロダクトに反映されないといった問題も報告されており，パッチのレビュープロセスを改善することは，不具合修正の迅速化に寄与するものと考えることができる．

そこで本稿では，OSS 開発におけるパッチレビュープロセスの効率化へ向けて，開発者が投稿したパッチがどの様にレビューされ最終的にプロダクトへ反映されるのかについて分析する．本分析の狙いは，パッチのレビューと承認に大きな影響を持つコミッターを選出するための要件を明らかにすることである．

コミッターとは， CVS や Subversion などの構成管理ツールで管理されているソースコードに対して，承認されたパッチを適用したり，あるいは自身のコードを直接書き込む権限（コミット権限）を与えられた一部の開発者を指す．コミッターは， プロダクトに修正を加える際の手間をできるだけ減らすために，多くの OSS プロジェクトで採用されている開発者の役割である[3].

コミッターの人数が増えれば，レビュー作業そのものを省略化や，パッチ投稿からプロダクトへの反映までの時間の短縮化を期待できる．コミッターは通常，プロジェクトへの多大な貢献が認められた開発者が他の開発者からの推薦を経て選ばれる[5]が，プロジェクトの進展に大きな影響を与え得る役割であるため，プロジェクトの状況に沿って慎重に選出される必要がある．そのため，プロジェクトに参加している多くの開発者の中からコミッターを選出することは容易ではなかった．

本稿で行う分析では，PostgreSQL プロジェクトを対象として，OSS 開発におけるパッチレビュープロセスの実態と，既存のコミッターがどの様にパッチレビュープロセスに貢献しているのかについて調査する．コミッターの貢献内容の詳細が分かれば，コミッターを選出するための要件を抽出することができると考えたためである．

本稿の構成は以下の通りである．続く 2 章では，OSS 開発におけるパッチレビュープロセスの概要と，本稿で用いる用語の定義を行う．3 章では，パッチレビュープロセスを効率化させる上で明らかにする必要のある Research Questions について述べる．4 章では，ケーススタディで行う分析方法について説明し，5 章では，PostgreSQL プロジェクトを対象として行ったケーススタディの結果をまとめる．6 章では，結果をもとに議論を行う． 7 章で関連研究の紹介と本研究との違いを延べ，最後に 8 章においてまとめと今後の課題を示す．

2．OSS 開発におけるパッチレビュープロセス

本章では，OSS のパッチレビュープロセスを紹介するとともに，本稿で用いる用語を定義する．

OSS プロダクトの修正は，既存のソースコードと修正後のソースコードとの差分，すなわちパッチを適用することで行われる．図 1 に，パッチが投稿されてからプロダクトに反映されるまでのパッチレビュープロセスを示す．図 1 中の横軸は時間経過を，縦軸にパッチレビュープロセスに登場するアクターを表す．なお本稿では，コミット権限を持つ開発者を「コミッター」，コミット権限を持たない開発者を「非コミッター」と定義する．両者を区別しない場合には，単に「開発者」と呼ぶこととする．

① 開発者によって ML や BTS に投稿されたパッチは，1 人又は複数の開発者から修正内容が適切かどうかをチェックする（妥当性確認）ためのレビューを受ける．
② 適切であると判断された場合は，コミッターによってパッチを適用したソースファイルが，CVS や Subversion などの構成管理ツールへコミットされ，プロダクトに反映される．
③ 不十分あるいは適切ではないと判断された場合は，コミッターからパッチに対するフィードバックを返す．開発者が再度パッチを編集（修

† 奈良先端科学技術大学院大学　情報科学研究科
Graduate School of Information Science,
Nara Institute of Science and Technology

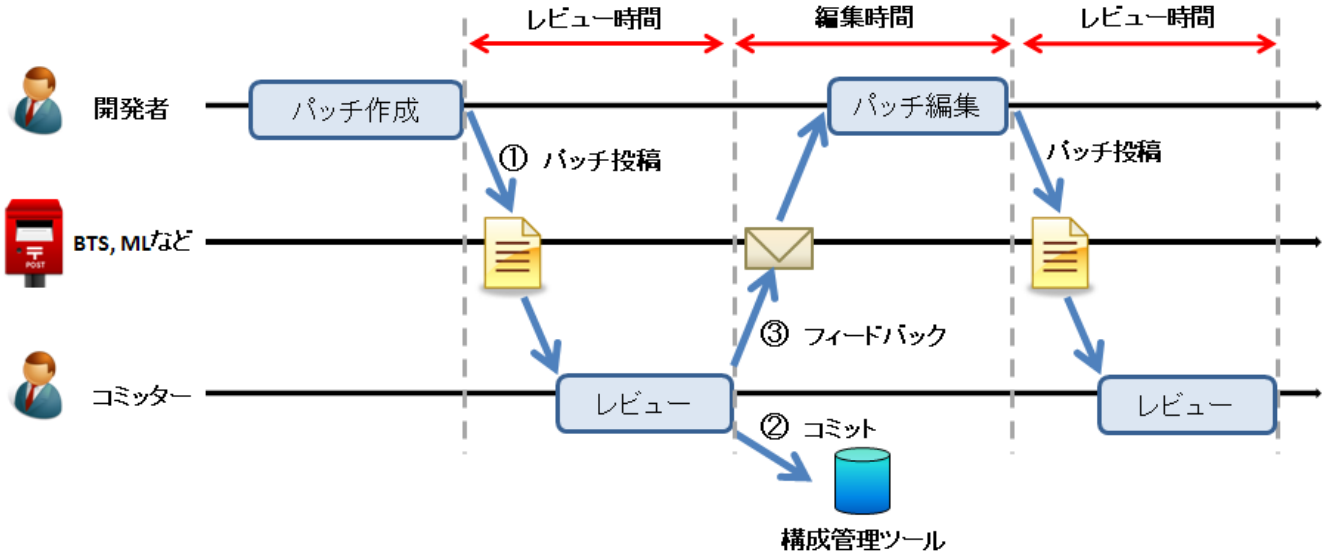


図 1　OSS 開発におけるパッチレビュープロセス

正）して投稿しレビューに合格すれば，②と同様，プロダクトに反映される．

図 1 では，レビュープロセスにおける時間の経過を「レビュー時間」と「パッチ編集時間」の 2 つに分類している．それぞれ，以下のように定義する．

レビュー時間：開発者によってパッチが投稿されてから，パッチがレビューされ妥当性が確認されるまでの時間．パッチが不十分あるいは適切ではないと判断される場合は，フィードバックが返されるまでの時間．

パッチ編集時間：投稿されたパッチが他の開発者によってフィードバックを受けてから，再び投稿されるまでの時間．

3．　Research Questions

本稿では，レビュープロセスにおける，編集時間とレビュー時間について，以下のように Research Questions を立て，分析を行う．

RQ1：パッチ編集時間・レビュー時間にはどのような傾向があるか

RQ2：パッチ編集回数・レビュー回数にはどのような傾向があるか

RQ3：コミッターのパッチ編集とレビューにはどのような関係があるか

コミッターは非コミッターの中から積極的にプロジェクトに参加している開発者をコミッターに推薦する．その時，コミッターは非コミッターのプロジェクトに対する貢献や非コミッターが作成するパッチの品質などを参考にしてコミッター候補者に推薦する開発者を決定する．貢献はパッチ編集回数や，パッチ編集時間などが挙げられる．また，パッチの品質としては，編集したパッチに含まれる不具合数などが挙げられる．しかし，コミッターに推薦するための基準は明確にされてない．さらに，コミッターは非コミッターにコミット権限付与後，その非コミッターに開発者から依頼されるパッチのレビュー作業に貢献することを期待してコミット権限を与えるが，コミット権限を与えた後，その開発者がレビュー作業に積極的に貢献するどうかを予測することは困難である．

本稿では，OSS 開発者のコミッターと非コミッターの貢献を把握するためにパッチの編集時間と編集回数についてそれぞれ RQ1，RQ2 で調査する．また，コミッターが非コミッターに比べてパッチのレビューに貢献しているかを把握するために，OSS 開発者のレビュー時間とレビュー回数を調査し，コミッターと非コミッターの貢献の違いを分析する．

RQ1，RQ2 の結果を踏まえて，RQ3 でコミッターのパッチ編集とレビューの関係を調査する．

4．　分析方法

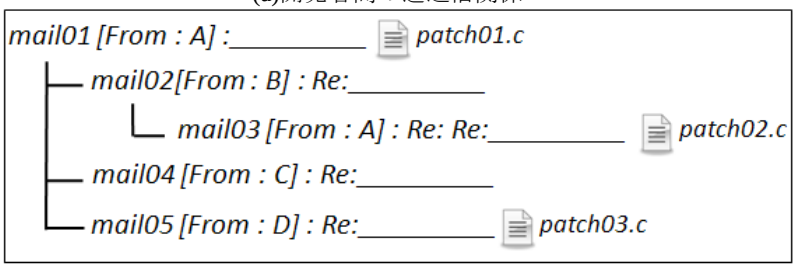
本稿では，レビュープロセスの効率化のために，OSS 開発でのパッチ編集時間とレビュー時間，パッチ編集回数とレビュー回数を調査し，それぞれの関係について分析を行う．本章では，分析を行う際の対象データの取得と，分析方法について述べる．

4.1　分析対象データの取得

本稿では，OSS 開発で利用されている BTS もしくは ML アーカイブなどから開発者がパッチ編集を行った回数と各パッチ編集に要する時間，レビューを行った回数と各レビューに要する時間を取得する．図 2 は BTS もしくは ML での送返信の関係から，返信にかかった時間がレビュープロセスのどの時間に対応しているかを示す概略図である．図 2 (a)は開発者間の送返信関係を示し，メッセージの横にあるファイルはパッチファイルとする．図 2 (b)は各メッセージの送信情報を示す．図 3 はメッセージの送返信情報と貢献の種類を示す．本節では図 2 を用いて，パッチ編集とレビューの取得方法を示す．

開発者 A が mail01 でパッチを投稿し，それに対して開発者 B が mail02 で返信する場合，mail02 はパッチに対するフィードバックとみなし，レビューを行ったとする．

(a)開発者間の送返信関係



(b)メッセージの送信日時とパッチの有無

ID	送信日時	送信者	パッチ
mail01	2010/4/20 9:55	A	あり
mail02	2010/4/20 13:18	B	なし
mail03	2010/4/20 15:53	A	あり
mail04	2010/4/20 21:40	C	なし
mail05	2010/4/21 2:38	D	あり

(c)メッセージの送返信情報と貢献の種類

送信情報			返信情報			経過時間（時間）	対象
ID	送信日時	送信者	ID	返信日時	返信者		
mail01	2010/4/20 9:55	A	mail02	2010/4/20 13:18	B	3.38	レビュー
mail02	2010/4/20 13:18	B	mail03	2010/4/20 15:53	A	2.58	パッチ編集
mail01	2010/4/20 9:55	A	mail04	2010/4/20 21:40	C	11.75	レビュー
mail01	2010/4/20 9:55	A	mail05	2010/4/21 2:38	D	16.72	パッチ編集

図 2　メッセージの送返信関係と時間の対応の概略

その時，レビュー時間は mail01 の送信時間と mail02 の送信時間の差とする．一方，mail05 のようにパッチファイルを添付したメッセージを返信した場合，新たなパッチ投稿とみなし，パッチ編集を行ったとする．その時，パッチ編集時間は mail01 の送信時間と mail05 の送信時間の差とする．
ただし，返信の行われていないメッセージ，送信・返信共にパッチが添付されていないメッセージは分析対象外とする．
　また，コミッターは構成管理ツールに 1 回以上コミットを行った開発者とする．ただし，構成管理ツールに登録されている開発者名と ML もしくは BTS に登録されている開発者名の一致が確認できた開発者のみをコミッターと判断する．

4.2　分析方法

4.2.1　RQ1：パッチ編集時間とレビュー時間

　コミッターと非コミッターのパッチ編集時間，レビュー時間をそれぞれ分析するために，各開発者のパッチ編集時間，レビュー時間の中央値をそれぞれ求める．次に，コミッターと非コミッターのパッチ編集時間の差，レビュー時間の差がそれぞれ統計的に有意かどうかを検定する．検定にはウィルコクソンの順位和検定を用いる．ただし，パッチ編集時間，レビュー時間共にパッチ付の返信メッセージ 5 件未満の貢献数が少ない開発者は分析対象外とする．

4.2.2　RQ2：パッチ編集回数とレビュー回数

　コミッターと非コミッターのパッチ編集回数，レビュー回数をそれぞれ分析するために，各開発者のパッチ編集回数，レビュー回数の中央値をそれぞれ求める．次に，コミッターと非コミッターのパッチ編集回数，レビュー回数の差がそれぞれ統計的に有意かどうかを検定する．検定にはウィルコクソンの順位和検定を用いる．

4.2.3　RQ3：コミッターのパッチ編集とレビューの関係

　パッチ編集回数，レビュー回数がそれぞれ 5 件以上の開発者のパッチ編集時間とレビュー時間の中央値，全開発者のパッチ編集回数，レビュー回数の中央値をそれぞれ基準とし，パッチ編集とレビューそれぞれの値でコミッターを分類し，コミット権限の与えられる開発者の傾向を分析する．

5．ケーススタディ

　本章では，パッチのレビューと承認に大きな影響を持つコミッターを選出するための要件の貢献としてパッチ編集回数や，パッチ編集時間を分析することが有効かを調べるための適用事例の一つとして PostgreSQL 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．

5.1　分析対象データ

　PostgreSQL は，オープンソースで開発されているオブジェクト関係データベース管理システムとして幅広く普及している．PostgreSQL プロジェクトでは，分析対象期間（2001 年 1 月～2008 年 12 月）に 2 つのメーリングリスト pgsql-hackers と pgsql-patches を用いてパッチの投稿，レビュー後のフィードバックを行っており，本研究では，この 2 つのメーリングリストを分析対象とする．
　コミッターと非コミッターは，メーリングリストに登場する開発者の中で，分析対象期間に構成管理ツールにコミットを行った開発者をコミッターとして区別した．本稿では，14 人の開発者がコミッターであることを確認した．分析期間中のコミッターと非コミッターがパッチ編集を行った人数とレビューを行った人数はそれぞれ，146 人と 214 人である．

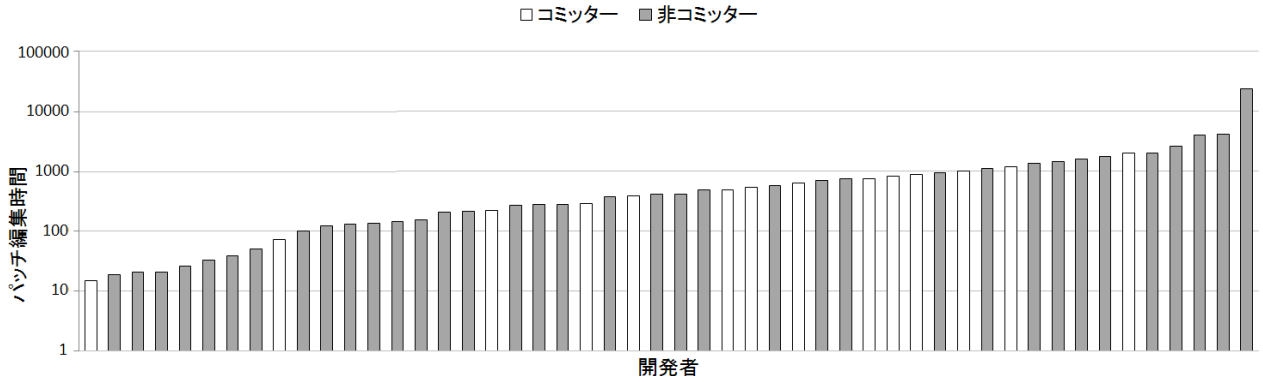


図 3　開発者のパッチ編集時間

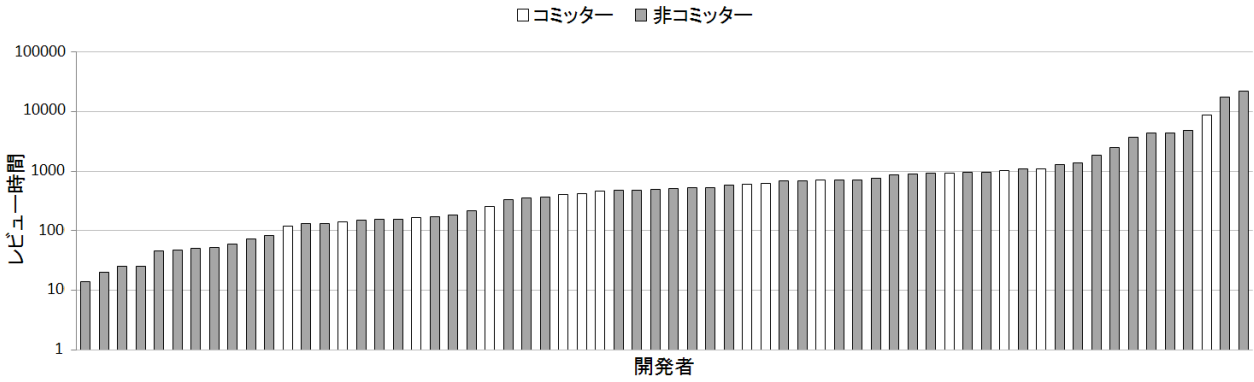


図 4　開発者のレビュー時間

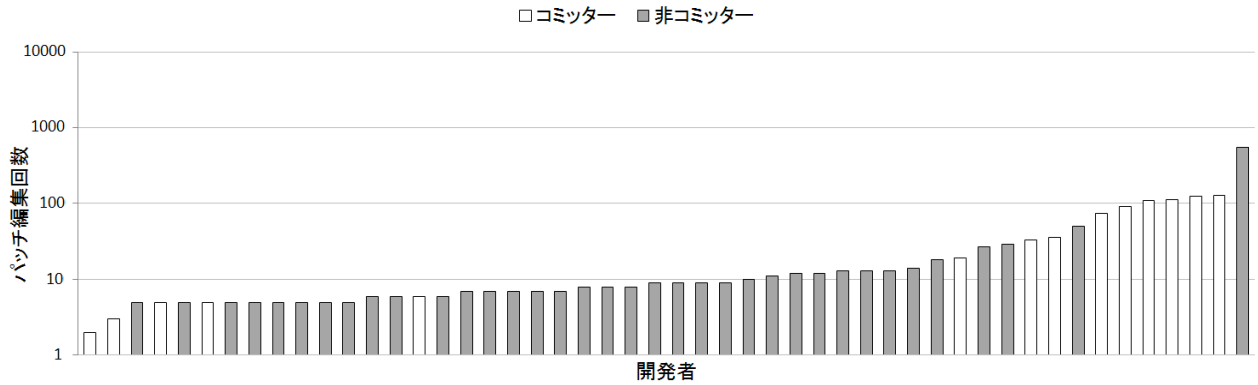


図 5　開発者のパッチ編集回数

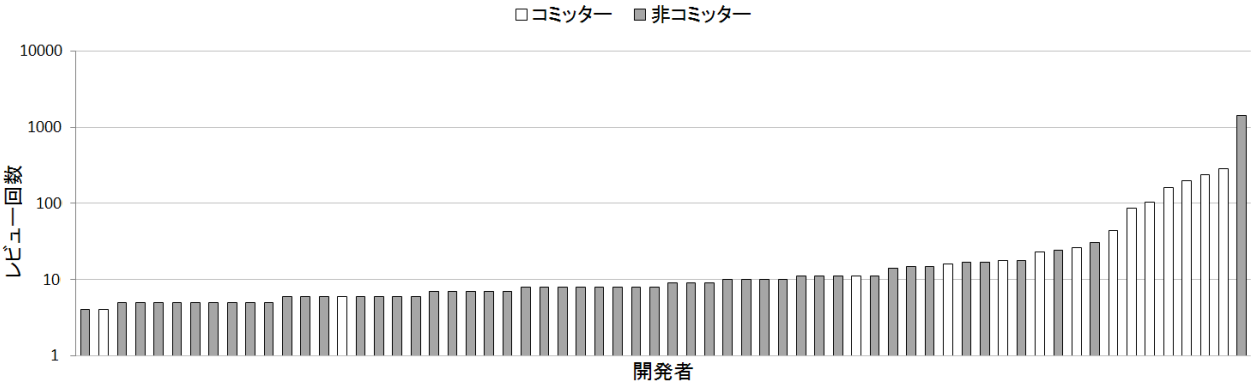


図 6　開発者のレビュー回数

5.2　分析結果

5.2.1　RQ1：パッチ編集時間とレビュー時間

コミッターと非コミッターのパッチ編集時間を図 3 に示す．横軸はパッチ編集時間の長い開発者を昇順に並べており，パッチ編集時間は各開発者の中央値を対数値に変換している．図 3 にはパッチ編集回数が，5 回以上の開発者 50 人（コミッター14 人，非コミッター36 人）を提示している．また，表 1 に全開発者，非コミッター，コミッターのパッチ編集時間の中央値，平均，標準偏差を示す．パッチ編集時間の中央値は非コミッターに比べ，コミッターの方が早いことが分かる．しかし，コミッターと非コミッターのパッチ編集時間の差をウィルコクソンの順位和検定を行った結果，p 値は 0.48 (> 0.05) であるためコミッターと非コミッターでパッチ編集時間に有意差は見られなかった．

次に，コミッターと非コミッターのレビュー時間を図 4 に示す．横軸はレビュー時間の長い開発者を昇順に並べており，レビュー時間は各開発者の中央値を対数値に変換している．図 4 にはレビュー回数が，5 回以上の開発者 64 人（コミッター14 人，非コミッター50 人）を提示している．また，表 2 に全開発者，非コミッター，コミッターのレビュー時間の中央値，平均，標準偏差を示す．レビュー時間の中央値はコミッターと非コミッターに大きな差は見られない．コミッターと非コミッターのレビュー時間の差をウィルコクソンの順位和検定を行った結果，p 値は 0.65 (> 0.05) であるためコミッターと非コミッターでレビュー時間に有意差は見られなかった．

5.2.2　RQ2：パッチ編集回数とレビュー回数

コミッターと非コミッターのパッチ編集回数を図 5 に示す．横軸はパッチ編集回数の多い開発者を昇順に並べており，パッチ編集回数は各開発者の中央値を対数値に変換している．図 5 にはパッチ編集回数が，5 回以上の開発者を提示している．表 3 に全開発者，非コミッター，コミッターのパッチ編集回数の中央値，平均，標準偏差を示す．パッチ編集回数の中央値は非コミッターよりも多いことが分かる．コミッターと非コミッターのパッチ編集回数の差をウィルコクソンの順位和検定を行った結果，p 値は 0.00 (< 0.05) であるためコミッターと非コミッターでパッチ編集回数に有意差が見られた．

次に，コミッターと非コミッターのレビュー回数を図 5 に示す．横軸はレビュー回数の多い開発者を昇順に並べており，レビュー回数は各開発者の中央値を対数値に変換している．図 5 にはレビュー回数が，5 回以上の開発者を提示している．表 3 に全開発者，非コミッター，コミッターのレビュー回数の中央値，平均，標準偏差を示す．レビュー回数の中央値は非コミッターよりも多いことが分かる．コミッターと非コミッターのレビュー回数の差をウィルコクソンの順位和検定を行った結果，p 値は 0.00 (< 0.05) であるためコミッターと非コミッターでレビュー回数にも有意差が見られた．

5.2.3　RQ3：コミッターのパッチ編集とレビューの関係

コミッターのパッチ編集時間とレビュー時間の関係を図 7 に示す．横軸がパッチ編集時間，縦軸がレビュー時間を示し，図中にパッチ編集回数が 5 件以上の開発者のパッチ編集時間の中央値 411 分（対数表記；2.61），レビュー時間の中央値 508 分（対数表記；2.71）の線を示す．

表 1　パッチ編集時間の統計表(カッコ内は対数表記)

パッチ編集時間	全開発者	非コミッター	コミッター
人数	50	36	14
中央値	411.00(3.02)	324.50(3.15)	583.00(2.91)
平均	1188.88(3.10)	1394.47(3.21)	660.21(2.92)
標準偏差	3317.03(0.37)	3877.61(0.41)	493.46(0.18)

表 2　レビュー時間の統計表(カッコ内は対数表記)

レビュー時間	全開発者	非コミッター	コミッター
人数	64	50	14
中央値	507.50(2.94)	507.50(2.96)	533.00(2.82)
平均	1479.07(3.06)	1579.25(3.11)	1121.29(2.93)
標準偏差	3572.63(0.43)	3875.23(0.45)	2132.97(0.37)

表 3　パッチ編集回数の統計表(カッコ内は対数表記)

パッチ編集回数	全開発者	非コミッター	コミッター
人数	146	132	14
中央値	2.00(0.30)	1.00(0.00)	34.50(1.97)
平均	12.62(0.43)	8.24(0.31)	53.93(1.83)
標準偏差	50.24(0.60)	48.33(0.46)	49.18(0.29)

表 4　レビュー回数の統計表(カッコ内は対数表記)

レビュー回数	全開発者	非コミッター	コミッター
人数	214	200	14
中央値	2.00(0.30)	1.00(0.00)	35.00(1.45)
平均	15.55(0.35)	10.56(0.28)	86.93(1.56)
標準偏差	100.93(0.50)	99.65(0.39)	91.79(0.65)

それぞれの中央値を基にコミッターの分類を行った結果，パッチ編集時間，レビュー時間共に中央値より長い時間を要するコミッターは 5 人，パッチ編集時間が長く，レビュー時間が短いコミッターは 4 人，パッチ編集時間が短くレビュー時間が長いコミッターは 3 人，パッチ編集時間，レビュー時間共に短いコミッターは 2 人であった．つまり，コミッターはパッチ編集時間が長い傾向にあることが分かった．

コミッターのパッチ編集回数とレビュー回数の関係を図 8 に示す．横軸がパッチ編集回数，縦軸がレビュー回数を示し，図中に全開発者のパッチ編集回数の中央値 2 回（対数表記；0.30），レビュー回数の中央値 2 回（対数表記；0.30）の線を示す．図 7，図 8 共に点は 1 人のコミッターを示す．

基準線を基にコミッターの分類を行った結果，コミッター14 人全員がパッチ編集回数・レビュー回数ともに基準値より多いタイプのコミッターであることがわかった．

６．議論

RQ2 において，パッチ編集回数・レビュー回数ともにコミッターが非コミッターより多くなり，RQ3 においても，コミッターは開発者全体の中でパッチの編集回数，レビュー回数共に多いグループに属することが分かった．分析結果からコミッターは，昇格前の活動から昇格後通して，多くのパッチを編集・投稿し，その貢献が認められコミッターへ昇格したと考えられる．レビュー回数に関しては，非コミッターもフィードバックを返すことは可能であるが，パッチを適切であると判断し，構成管理ツールへコミットすることはコミッターしか行うことが

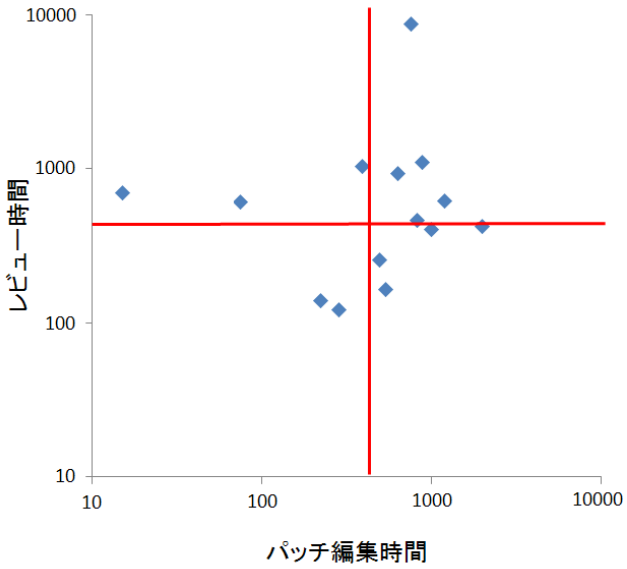


図 7　パッチ編集時間とレビュー時間による
コミッターのタイプの分類

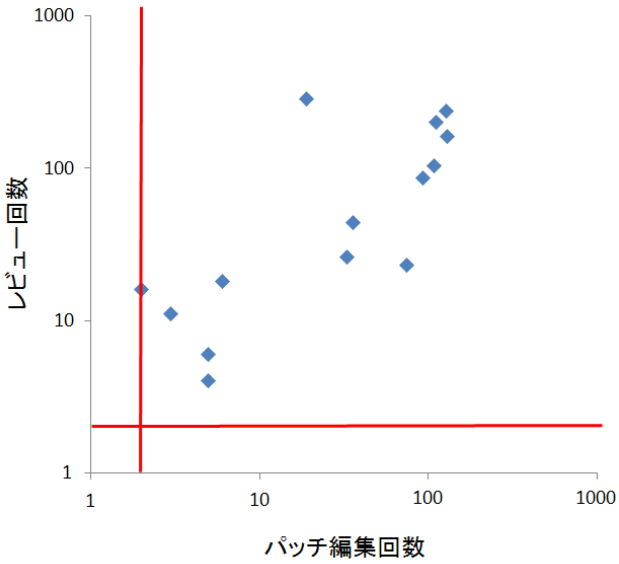


図 8　パッチ編集回数とレビュー回数による
コミッターのタイプの分類

できないため，必然的にコミッターのレビュー回数の方が多くなったと考えられる．

しかし，RQ3 のパッチ編集時間とレビュー時間の関係の結果から，コミッターは必ずしもパッチ編集やレビューに要する時間が早いわけではなく，PostgreSQL のコミッターの多くは開発者の中でパッチ編集に長い時間を要していることが分かった．この結果から，パッチ編集に長い時間をかけて，品質の高い多くのパッチを投稿することでコミッターに推薦されたことが考えられる．レビュープロセスの短縮化を目的としたとき，パッチ編集時間や編集回数だけでレビュー時間の短い開発者を決定することはできないことが分かった．

また，本稿では PostgreSQL プロジェクトに対してケーススタディを行ったが，この結果は全ての OSS プロジェクトに対して一般性を保証するものではない．PostgreSQL はコミッターの人数が特に少なく，プロジェクト管理者がコミッターの活動を把握することが容易であるが，多くのパッチが投稿されたとき，レビュー時間が長期化することが考えられる．また，PostgreSQL はパッチの投稿に ML を利用している OSS プロジェクトであるが，不具合管理システムに添付する形でパッチの投稿が行われている OSS プロジェクトの分析は行っていない．パッチの投稿ルールの違いによる，パッチレビュープロセスの違いも結果に影響を及ぼす可能性がある．

7．関連研究

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[5][7]．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[7]．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．コミッターへの候補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[5]．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．本研究の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．

8．おわりに

本稿では，OSS 開発におけるパッチレビュープロセスの効率化へ向けたコミッター選出支援のために，パッチ編集時間とレビュー時間を定義し，それぞれにかかる時間と回数の実態を調査した．ケーススタディとして PostgreSQL プロジェクトを対象に分析を行った結果，以下の知見を得ることができた．

- コミッターと非コミッター間でパッチ編集時間とレビュー時間に有意な差は見られなかった
- コミッターと非コミッター間でパッチ編集回数とレビュー回数に有意な差が見られた
- コミッターはパッチ編集回数，レビュー回数共に多いが，パッチ編集やレビューに要する時間が必ずしも早いわけではない

今後は，プロジェクト管理者が適切なコミッターを選出するために，パッチの編集回数や編集時間だけでなく，パッチの品質（不具合の混入率が低いパッチ，可読性の高いパッチなど）も着目する必要があると考えられる．さらに．多くの OSS プロジェクトでは，公式ウェブサイト上にコミッターになるためにはどうすればよいかを記

載されており，そのうち多くの場合は，継続的にパッチを投稿してプロジェクトに貢献することも１つの条件となっている．本研究では，時系列の要素を考慮せずに開発者の活動を調査したが，対象データを時系列に分析することで，継続的にパッチを投稿する開発者の活動も把握ことができると考えられる．さらに，OSS 管理者がコミッター候補者を選ぶための具体的な基準値を提示し，非コミッターがコミッターに昇格後，そのコミッターがレビュープロセスの効率化に貢献できるかを評価したいと考えている．

謝辞

本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は文部科学省科学研究補助費（若手 B：課題番号 22700033）による助成を受けた．

参考文献

[1] Bettenburg N., Just S., Schroter A., Weiss C., Premraj R., Zimmermann T.: What makes a good bug report, Proceedings of the 16th ACM SIGSOFT International symposium on foundations of software engineering (SIGSOFT'08/FSE-16), pp.308-318, 2008

[2] Fogel, K.: Producing Open Source Software: How to Run a Successful Free Software Project, O'Reilly, 2005

[3] Hailpern B., Santhanam P.: Software debugging, testing, and verification, IBM Systems fournal, pp.4-12, 2002.

[4] Hooimeijer P., Weimer W.: Modeling bug report quality, Proceedings of the 22nd IEEE/ACM International conference on automated software engineering (ASE2007), pp.34-43, 2007

[5] Jensen C., Scacchi W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, Proceedings of the 29th International conference on software engineering (ICSE2007), pp.364-374, 2007

[6] Wang, Y., Guo, D. and Shi, H.: Measuring the evolution of open source software systems with their communities, SIGSOFT Softw. Eng. Notes, Vol.32, No.6, p.7, 2007.

[7] Weißgerber P., Neu D., Diehl S.: Small patches get in!, Proceedings of the 5th International workshop on mining software repositories (MSR2008), pp.67-76, 2008.