B-01

オープンソースソフトウェアのコードレビューにおけるレビュアーの活動の分析

Analyzing Code Review Activities in An Open Source Software Project

梁　軍偉 †　　水野　修 †
JunWei Liang　Osamu Mizuno

1. はじめに

ソフトウェアの品質を保証するためには, 早期の不具合発見が必須である. コードレビューはそうした早期のバグ発見に有効な手法であると目されている [9]. コードレビューにおけるレビュアの活動は様々なソフトウェア品質に関する知見を含んでいると考えられ, その分析がソフトウェアの品質に対する新たな発見を引き出すと期待される. しかしながら, 特にオープンソースのソフトウェア開発では, コードレビュー活動は系統的に実施されていない. 多くの開発ではメーリングリスト上でのレビューが行われているのが現状である. これに対して, ツールを用いたコードレビュー手法が提案され, 一部のソフトウェア開発では実施されてきている. 本研究では, ツールベースのコードレビュー手法に対して、リポジトリマイニングを行い, コードレビューにおけるレビュアの活動がレビューにどういった影響を及ぼしているかを分析する.

2. コードレビューリポジトリ

2.1 対象プロジェクト

オープンソースソフトウェアプロジェクトにおけるコードレビュア活動の特徴，および，コードレビュープロセスの効率を左右する要因を調査するため，コードレビューツールを用いる Chromium プロジェクトを対象として分析を行った。

オープンソースソフトウェアプロジェクトにおけるコードレビューには，主にメールベースとツールベースの 2 つの方法がある．ツールベースコードレビューは以下の利点がある．

1. コードラインごとにコメントができる．
2. コードパッチの履歴が完全に残る．
3. バージョン管理システムおよびバグ管理システムとの対応関係が付けられる．
4. コードレビューリポジトリとしての豊富なデータ．

図 1 は Chromium コードリポジトリからのコミットログの例である．この例では，レビューの URL とバグ ID の情報が確認できる．

しかし，コードレビューツールを用いるオープンソースソフトウェアプロジェクトはまだ少数である．Chromium は数少ない実例の 1 つであり,「Rietveld」というコードレビューツールを用いている.「Rietveld」は Google 社内で用いるツール「Mondrian」のクローン版であり，オープンソースソフトウェアとして公開されている．どちらも Python の作者グイド・ヴァンロッサム氏によって開発されたツールである．

また,「Rietveld」を用いるコードレビューのプロセスは概ね以下のようになっている．

1. 開発者が差分ファイルを投稿し（レビュー票の作成），説明文も書いてレビュアに依頼する．
2. レビュアが差分ファイルをチェックしてメッセージやインラインコメントを書く．
3. 開発者はレビュアが指摘した問題を修正して新しい差分ファイルを投稿する．
4. 開発者とレビュアがウェブページでコメントを書いたり，返事したりして議論を行う．
5. 指摘された問題を全部確認できれば，レビュー票の状態を「Closed」に変更し，コードリポジトリにコミットする．

2.2 マイニングコードレビューリポジトリ

上で述べたように，Chromium は「Rietveld」を用いてコードレビューリポジトリを管理している.「Rietveld」はウェブベースのツールであり，差分情報やインラインコメント閲覧などの便利な機能を実装している．Chromium のコードレビューリポジトリのデータを収集するため, 我々はツールを開発してレビュー票のウェブページをクローリングした．また，クローリングしたデータをドキュメント指向データベースである「MongoDB[1]」に保存し, 必要なデータのみを抽出して実験を行った．表 1 は収集したデータの詳細である．本稿では，状態が「Closed」であるレビュー票のタイトル，説明文，内容，メッセージとコードパッチセットのデータを用いて分析を行う．

† 京都工芸繊維大学　大学院工芸科学研究科
Graduate School of Science and Technology, Kyoto Institute of Technology

[1] http://mongodb.org

```
commit log
Author:dmazzoni@chromium.org
Date:Mon Mar 14 06:56:33 2011 UTC (3 months ago)
Log Message:Re-land: Refactor Views accessibility.
BUG=74988
TEST=none
Review URL: http://codereview.chromium.org/6581010
```

図 1: コミットログの例

表 1: 収集したデータの詳細

	Count
Issues created until 2011-05-24	101,182
Issues in 'Closed'	98,585
+ assigned reviewers > 0	92,147
+ the number of patch sets = 1	58,939
+ committed	51,990
Patch sets collected	212,994

2.3 マイニングツール

図 2 は我々が開発したツールの全体構成である．クローリングサーバはクローラクライアントを管理し，クライアントからのデータを解析してバックエンドデータベースに保存する．クローリングサーバは「Rails[2]」を用いて実装している．バックエンドデータベースは MongoDB を用いる．

クローラクライアントは実際のウェブページをクローリングする．クライアントはそれぞれ独立して，クローリングサーバによって管理される．クライアントはサーバから受け取ったクローリングタスクによって，クローリングを行う．それらの工夫によって，データ収集の効率やツールの柔軟性などを高めることができた．

このツールを用いて Chromium のコードレビューリポジトリを 6 個のクライアントでクローリングした結果，10 日間で約 700 万のウェブページをクローリングできた．

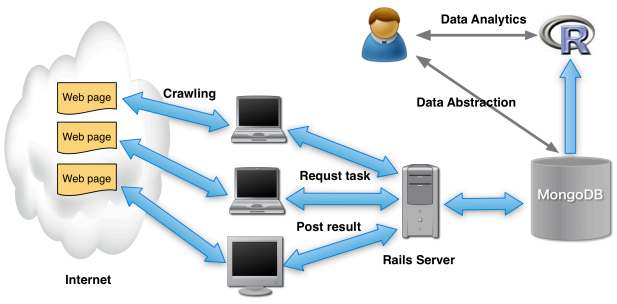

図 2: マイニングツールの構成

3. 研究設問

本稿では，コードレビュープロセスにおけるレビュアの活動について，以下の研究設問を立て，分析を行う．

設問 1 コードレビューにおけるレビュア人数はどうなっているか？

コードレビューにおけるレビュア人数についての調査は，適切なレビュア人数の推定に重要である．Lee らは，Linux プロジェクトにおけるレビュー毎にレビュアの平均人数は 2.35 であると述べている [4]．Rigby らと Asundi らは，Apache を対象とした分析でも類似の結果が得られたと報告している [1, 7]．

Chromium では，レビューで問題が発見されなかったとしても，レビュアが「LGTM (Looks Good To Me)」というようなコメントを書くべきであるとしている[3]．そのため，レビュー票に記録されたコメントの筆者 (レビュー票の筆者を除く) を数えれば，レビュアの人数が推定できると思われる．

本研究では，上記のような推定されたレビュアを「実レビュア」，レビュー票に指名されたレビュアを「割り当てレビュア」とそれぞれ表記する．また，単に「レビュア」と呼ぶ場合,「実レビュア」のことを指す．

設問 2 レビュアの開発経験とレビュー実績との関係性はあるか？

開発経験は古くからレビューの効率や効果に関わる最も重要な要因と目されている [5, 10]．Chromium に用いられるレビューツール Rietveld では，レビュー票を作成する時に，コードパッチを一緒に付けなければならない．そのため，レビュー票の作成履歴によって，レビュアの開発経験を評価することができる．

本研究では，レビュー票の作成履歴を用いてレビュアの開発経験を評価した上，分析の妥当性を確認するため，コードリポジトリから抽出したコミット履歴も用いられている．また,「レビュー実績」とはレビュアが実施したレビューの数のことを指す．

設問 3 よくレビューに割り当てられるレビュアはどのようなレビュアか？

Chromium において，誰がどのレビューを実施するべきかというような規則がないため，開発者がレビュー票を作成する際，自分で適切なレビュアを探さなければならない．例えば，コードリポジトリのコミット履歴から，同じファイルやモジュールを変更したことがある開発者を

[2]http://rubyonrails.org/

[3]http://www.chromium.org/developers/

探してレビューを割り当てるという方法が勧められている[3].

設問 4 1 つのレビューでどのぐらいの欠陥が発見されているか？

レビューにおける欠陥の発見数はレビューの効果を評価するために重要なメトリックスである [2,3,6]．欠陥の発見数を測定できれば，発見数の多いレビューと発見数の少ないレビューを比較することで，それらの相違点あるいは特徴を判別することが期待される．

設問 5 レビュアの人数はレビューにどんな影響を与えるか？

レビュアの最適な人数に関する研究では，レビュアが 2 人の場合とレビュアが 3 人以上の場合とでは，ほぼ同じ効果があると報告されている [5, 10]．それに対して，レビュアが 1 人の場合はペアプログラミングと類似した状況になるため，より高い効果が得られることも期待される [8].

4. 分析

本節では，各研究設問に対して，Chromium のコードレビューリポジトリから抽出したデータを用いて分析した結果を説明する．

4.1 分析に用いるメトリクス

本研究では，コードレビューにおけるレビュアの活動を分析するため，8 つのメトリクスを定義する．

まず，レビュアに関して，以下のメトリクスを定義する．

- レビュアごとに割り当てられたレビューの数 (n^r_{ai})
- レビュアごとに実施したレビューの数 (n^r_{di})
- レビュアごとに作成したレビュー票の数 (n^r_{ci})

次に，レビュー票に関して，以下のメトリクスを定義する．

- レビュー票ごとの割り当てレビュアの数 (n^i_{ar})
- レビュー票ごとの実レビュアの数 (n^i_{rr})
- レビュー票ごとのコードパッチセットの数 (n^i_{ps})
- レビュー票ごとのメセッジの数 (n^i_{ms})
- レビュー票ごとのインラインコメントの数 (n^i_{ic})

レビュアとレビュー票にかかわるメトリクスの統計情報を，表 2，3 と表 4，5 にそれぞれ示す．

表 2: レビュアにかかわるメトリクスの要約統計量

	Min.	Median	Mean	Max.
n^r_{ai}	0.00	8.00	127.00	3261.00
n^r_{di}	1.00	6.00	93.11	2199.00
n^r_{ci}	0.00	8.00	84.97	1851.00

表 3: レビュアにかかわるメトリクスの順位相関行列

	n^r_{ai}	n^r_{di}	n^r_{ci}
n^r_{ai}	1		
n^r_{di}	0.94	1	
n^r_{ci}	0.89	0.88	1

4.2 コードレビューにおけるレビュア人数はどうなっているか？(設問 1)

収集したデータより，Chromium プロジェクトにおける 1 回以上レビューを実施したレビュア（実レビュア）の人数は 1,147 であった．図 3 にレビュー票の数とレビュー票ごとの実レビュアの人数 n^i_{rr} の関係を示す．図に示すように，98,585 件のレビュー票のうち実レビュアが 1 人しかいないレビュー票が最も多くあり，60,432 件（61.30%）がある．それに対して，実レビュアが 6 人以上参加するレビュー票が 149 件（0.15%）しかない．

また，表 4 によって Chromium におけるレビュー票ごとに実レビュア人数 n^i_{rr} の中央値が 1 であり，平均値は 1.08 である．それに対して，レビュー票ごとに割り当てレビュア人数 n^i_{ar} の中央値も 1 であるが，平均値は 1.41 である．この結果は，一部の割り当て請求に対して応答がなかったことを示している．

4.3 レビュアの開発経験とレビュー実績との関係性はあるか？(設問 2)

図 4 は，レビュアごとに実施したレビューの数 n^r_{di} と作成したレビュー票の数 n^r_{ci} との関係を示す．この図から，レビュアの開発経験が多いほど，実施したレビューの数が多いという傾向が見られる．さらに，スピアマンの順位相関係数によって，0.88 という強い相関を示している（表 3）．この結果は，レビュアは開発経験が多いほど，レビュー実績も多いということを示している．

この分析からは 1,147 人の実レビュアのうち，390 人（34.00%）の実レビュアが，レビュー票も作成したこと

表 4: レビュー票にかかわるメトリクスの要約統計量

	Min.	Median	Mean	Max.
n^i_{ar}	0	1	1.408	25
n^i_{rr}	0	1	1.083	19
n^i_{ps}	1	1	2.102	51
n^i_{ms}	0	2	3.245	127
n^i_{ic}	0	0	2.984	551

表 5: レビュー票にかかわるメトリクスの順位相関行列

	n^i_{ar}	n^i_{rr}	n^i_{ps}	n^i_{ms}	n^i_{ic}
n^i_{ar}	1				
n^i_{rr}	0.70	1			
n^i_{ps}	0.31	0.41	1		
n^i_{ms}	0.61	0.82	0.60	1	
n^i_{ic}	0.33	0.42	0.62	0.63	1

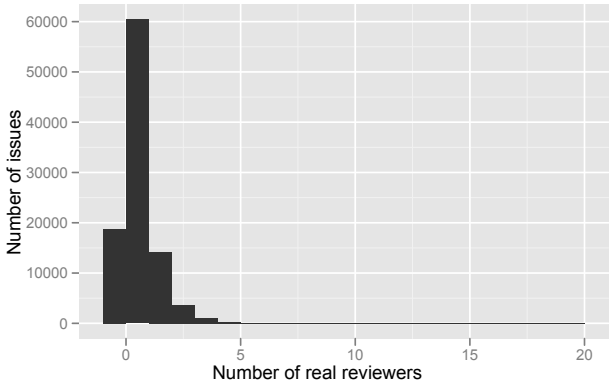

図 3: レビュアとレビュー票のヒストグラム

がなく，コミットしたこともない，つまり開発履歴が存在しないということが確認された．また，390 人のうち 382 人（97.95%) が実施したレビューの数が 10 件未満である．さらに，条件を加えると，390 人のうち 309 人（79.23%）が実施したレビューの数が 10 件未満であり，レビューに割り当てられたことがほとんどないということが確認された．我々は，これらの 309 人が Chromium プロジェクトにおいて，どのような貢献者であるかという疑問に対して，更なる分析を行った．

まず，多くのオープンソースプロジェクトと同じように Chromium においても，開発者を大きく分けると，コミット権を持つコミッターと非コミッターという 2 つのグループになっている．また，非コミッターは能力を見せたうえ，複数のコミッターから推薦をもらえば，コミッターになり得る．コミッターになる際,「example@chromium.org」という形のアカウントが配布される．このことから，アカウントの「@」より後ろの部分によって，開発者のプロジェクトにおける役割が推測できると考えられる．そこで，我々は，Chromium における貢献者を「google.com」,「chromium.org」,「gmail.com + 他」という 3 つのグループに分けて分析を行った．それぞれ，Google 職員，コア貢献者，普通の貢献者と見なす．

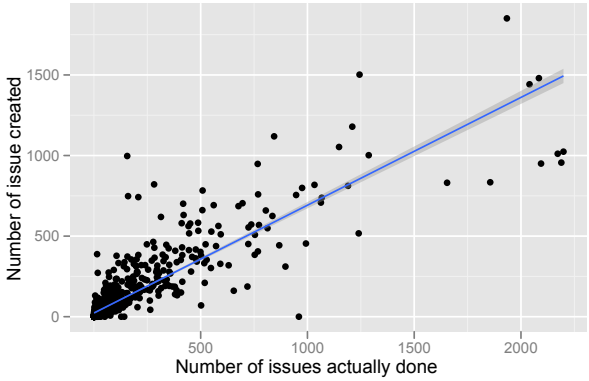

図 4: レビュアごとに作成したレビュー票と実施したレビューの関係

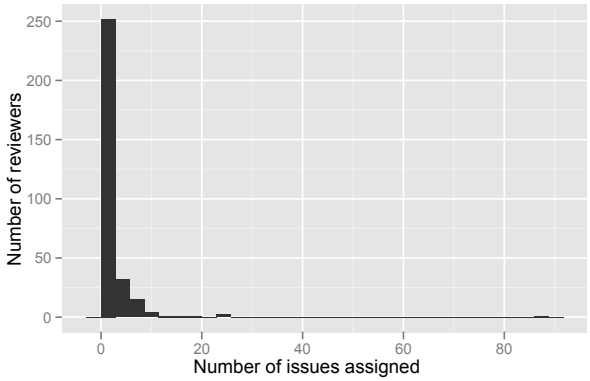

図 5: 309 人のレビュアにおいて，割り当てられたレビュー票とレビュアのヒストグラム

次に，疑問「これらの 309 人のレビュアはどのような貢献者であるか」について考察する．表 6 は上記の分け方によって，貢献者のグループ分けを示す.「309 人のレビュア」の列に注目すると，309 人のレビュアのほとんどは普通の貢献者であることが確認できる．

また，309 人のレビュアの割り当てられたレビューの数に着目して調査した結果を図 5 に示す．この図から，ほとんどのレビュアが実際は 1 回しかレビューに割り当てられなかったということが確認できる．また，割り当てられたレビューを 5 件以上持つレビュアが 5 人しかいない．87 件を持つレビュアが 1 人おり，このレビュアについて更に調査した結果，開発履歴が見つからず，レビューを 5 回しか実施していないが，Chromium プロジェクトのオーナー（マネージャー）であることが分かった．このことから，Chromium プロジェクトにおける 12 人のオーナーについても調査を行った．その結果，約半分のオーナーはコードリポジトリへコードコミットした回数が 9 回以下であることが確認できた．別のアカウントを利用している可能性も高いが，一部のオーナーが普通の貢献者であるとも考えられる．

結論として，以下のことが確認できた．

1. 309 人のレビュアがほとんどはまれにプロジェク

表 6: 貢献者グループ

	コミッター	レビュア	309 人のレビュア
@google.com	308	522	70
@chromium.org	274	242	25
@gmail.com and others	13+9	324+59	178+6
合計	604	1,147	309

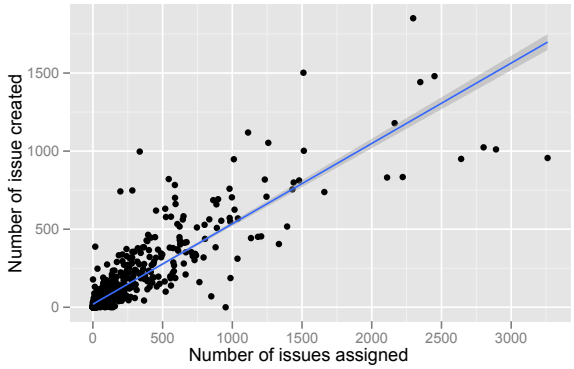

図 6: レビュアごとの作成したレビュー票と割り当てられたレビューの関係

表 7: n^r_{ai} と n^r_{ci} の関係

		n^r_{ai}	
		$= 0$	> 0
n^r_{ci}	$= 0$	78 (6.80%)	336 (29.29%)
	> 0	9 (0.79%)	724 (63.12%)

トに貢献する普通の貢献者である．

2. 相手はプロジェクトオーナー（マネージャー）だとしても，開発履歴をチェックせずにコードレビューを割り当てるべきではない．

4.4 よくレビューに割り当てられるレビュアはどんなレビュアか？(設問 3)

表 3 より，レビュアが割り当てられたレビューの数 n^r_{ai} と作成したレビュー票の数 n^r_{ci} とのスピアマンの順位相関係数は 0.89 となっている．この関係は図 6 にも見られる．

また，表 7 より，全ての実レビュア中，レビューに割り当てられたことがなく，レビュー票を作成したこともないレビュアが 9 人（0.79%）しかいない．それに対して，レビューに割り当てられたことがあり，レビュー票を作成したこともあるレビュアが 724 人（63.12%）がいる．以上のことから，レビュアは開発経験が多いほど，レビューに割り当てられる可能性が高いという傾向がある．

しかし，節での知見と同様に，1,147 人のうち 336 人（29.29%）の実レビュアがレビュー票の作成履歴がないが，レビューに割り当てられたことがあることも分かった．

4.5 1 つのレビューでどのぐらいの欠陥が発見されているか？(設問 4)

Rietveld のようなツールを用いるコードレビュープロセスにおいて，レビュアが欠陥を発見した場合，ふつうはウェブページで，コードの欠陥があるところにインラインコメントを書くべきである．そして，欠陥を発見しなかったとしても，レビュアが「LGTM (Look Good To Me)」というようなコメントを書くべきである．そのため，欠陥発見の履歴がほとんどインラインコメントに記録されていると考えられる．また，レビュー票を作成した開発者がインラインコメントを受け次第に，コードパッチセットを更新するため，レビューにおける欠陥の発見を評価するにはコードパッチセットの数もメトリクスとして利用できると考えられる．

本研究で収集したデータから，98,586 件のレビュー票のうち，76,250 件（77.34%）にはインラインコメントがなく，58,939 件（59.79%）にはコードパッチセットの更新がないということが確認できた．そのため，レビュー票ごとのインラインコメントの数 n^i_{ci} とコードパッチセットの数 n^i_{ps} の中央値はそれぞれ 0 と 1（最初のパッチセット）である（表 4）．これらのことはレビューでほとんど欠陥が発見されてないことを意味する．なぜそんなに多くの「完璧な」レビューがあるのかという疑問を解明するために更なる分析を行った．

まず，レビュー票の詳細に書かれているキーワード「Committed」を用い，レビュー票をコードリポジトリのコミット履歴とマッピングする．その結果，そのような 58,939 件のレビュー票のうち，32,773 件（55.60%）のコミットされたことが確認できた．

次に，32,773 件のうち，9802 件（29.91%）のレビューが実レビュアがおらず，レビューが実施されていないことが確認できた．また，32,733 件のうち，20,405 件（62.26%）が 1 人のレビュアによって実施された．それに対して，24611 件（75.10%）が割り当てレビュアが 1 人しかいないことを確認した．

なお，コードレビューにコード変更量が少ない（1 行）場合，欠陥の発生率が低いと思われるため，我々はレビュー票に含むコードの変更量に注目して，いくつかの段階に分けて調査を行っている．表 8 が上記の 20,405

表 8: レビュー票におけるパッチセットのコード変更量

	Total 20,405 reviews
Changed lines(CL) = 1	3,384
$1 < CL < 10$	6,815
$10 \leq CL$	10,200
$20 \leq CL$	7,286
$100 \leq CL$	2,630

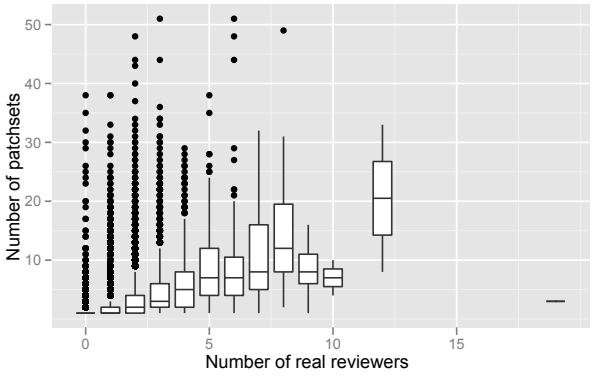

図 7: レビュアに対するコードパッチセットの関係

件のレビューに対して調査した結果である．表 8 より，コード変更量が 10 行未満のレビュー票を欠陥なしと仮定しても，コード変更量が 10 行以上で，欠陥が発見されてないレビュー票は 10200 件（49.99%）もある．

以上のことから，これらのレビュー票について欠陥を見逃した可能性が高いと推定できる．また，4.6 節の分析結果「1 人のレビュアに比べ，複数のレビュアが参加するレビューのほうが，パッチセットの更新数が多い」によって，レビューに複数のレビュアを割り当てれば，より多くの欠陥が発見できると期待される．

4.6 レビュアの人数はレビューの欠陥発見にどんな影響を与えるか？(設問 5)

表 5 に示すように，レビュー票ごとのメッセージ数は実レビュアの人数との相関が 0.82 であり，コードパッチセット数との相関が 0.60 である．どちらも正の相関があるが，レビュアの人数 n^i_{rr} とコードパッチセットの数 n^i_{ps} との相関が 0.41 しかないため，関係性が明らかになっていない．

一方，4.5 節で示したように，98,585 件のレビューにはレビュアが 1 人しかないレビューが 60,432 件（61.3%）もある．これは，ちょうどペアプログラミングのような状態になっていると考えることができ，Chromium におけるコードレビューでは，レビュアが 1 人の場合に最も効果が高くなっていると推察することができる．

しかし，n^i_{ps} と n^i_{rr} の箱ひげ図（図 7）には，n^i_{rr} の増加につれて，n^i_{ps} の平均値も増えていく傾向が見られる．そこで，我々はレビューを n^i_{rr} によって，レビュー票をレビュアが 1 人と 1 人以上との 2 つの群に分けて統計検定を行った．それぞれの群において，n^i_{ps} の平均値は 1.86 と 3.70 である．2 つの群における n^i_{ps} の平均値の差に対して Wilcoxon の順位和検定を行った結果，$p < 0.001$ であったため，平均値には統計的に有意な差が見られた．

したがって，Chromium におけるコードレビューでは，1 人のレビュアに比べ，複数のレビュアが参加するレビューのほうが，パッチセットの更新数が多かったことが分かる．

5. まとめ

本研究では，コードレビューにおけるレビュアの活動は様々なソフトウェア品質に関する知見を含んでいると考え, その分析を行うために，ツールベースのコードレビューを実施したリポジトリからのデータを取得した．そして，5 つの研究設問に対して分析を実施することにより，レビュアがレビューに与える影響の分析を行った．

参考文献

[1] J. Asundi and R. Jayant. Patch review processes in open source software development communities: A comparative case study. In *HICSS: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 10, 2007.

[2] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[3] P. M. Johnson. Reengineering inspection. *ACM Communications*, 41(2):49–52, 1998.

[4] G. Lee and R. Cole. From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization Science*, 14(6):633–649, 2003.

[5] A. A. Porter, H. P. Siy, A. Mockus, and L. G. Votta. Understanding the sources of variation in software inspections. *ACM Transactions Software Engineering Methodology*, 7(1):41–79, 1998.

[6] A. A. Porter, H. P. Siy, C. A. Toman, and L. G. Votta. An experiment to assess the cost-benefits of code inspections in large scale software development. *IEEE Trans. on Software Engineering*, 23:329–346, 1997.

[7] P. Rigby and D. German. A preliminary examination of code review processes in open source projects. Technical report, Technical Report DCS-305-IR, University of Victoria, 2006.

[8] P. C. Rigby, D. M. German, and M.-A. Storey. Open source software peer review practices: a case study of the apache server. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 541–550, New York, NY, USA, 2008. ACM.

[9] P. C. Rigby and M.-A. Storey. Understanding broadcast based peer review on open source software projects. In *Proc. of 33rd International Conference on Software Engineering*, pages 74–83, 2011.

[10] C. Sauer, D. Jeffery, L. Land, and P. Yetton. The effectiveness of software development technical reviews: A behaviorally motivated program of research. *Software Engineering, IEEE Transactions on*, 26(1):1–14, 2000.