情報処理学会論文誌 Vol.58 No.4 845–854 (Apr. 2017)

コードレビュー分析におけるデータクレンジングの影響調査

戸田 航史[1,a)] 亀井 靖高[2] 吉田 則裕[3]

受付日 2016年8月10日, 採録日 2017年1月10日

概要：本論文ではコードレビュー分析に対してデータクレンジングが与える影響を調査する．調査では，オープンソースソフトウェア開発プロジェクトである Android，Chromium，OpenStack の 3 プロジェクトを対象とした．クレンジングはレビュアとレビュー開始・終了日時の 2 つに対して実施した．レビュアへのクレンジングとして，ビルドやテストの自動化を行う bot の除去を，レビュー開始・終了日時へのクレンジングとして，実際のレビューの状況をふまえた補正を行った．3 プロジェクトから取得したデータを用いて，レビュア，レビュー開始・終了日時について，クレンジングを行ったデータと行わなかったデータをそれぞれ作成，比較した．比較の結果，bot によるレビューは OpenStack では全体の 19.4%を占めていること，レビュー開始・終了日時へのクレンジングの有無により，レビュー期間に有意な差が出ることが分かった．この結果から，データクレンジングはコードレビュー分析に影響を与えうることが分かった．さらに具体的なコードレビュー分析として，レビュー経験とレビュー期間の関係（相関）の分析を対象に，レビュア，レビュー開始・終了日時のデータクレンジングを実施したデータと実施しなかったデータを分析し，その結果を比較した．比較の結果，クレンジングを行わなければ相関係数に影響を与えることが分かった．

キーワード：コードレビュー，リポジトリマイニング，データクレンジング

Investigating the Effect of Data Cleaning Techniques for Code Review Analysis

Koji Toda[1,a)] Yasutaka Kamei[2] Norihiro Yoshida[3]

Received: August 10, 2016, Accepted: January 10, 2017

Abstract: In this paper, we investigate the effect of data cleansing techniques for code review analysis. We choose three open source software projects, Android, Chromium and OpenStack, then collect code review data from them. We perform two data cleansing techniques to the dataset. 1. remove bots from reviewers. 2. Correct review start and end time for reviewing time calculation. Then, we compare cleaning data and not cleaning data about each cleansing techniques and evaluate their effect. The results show both cleansing techniques effect to code review analysis, because 1. bots accounts for 19.4% in OpenStack review. 2. corrected reviewing time is significantly different from not corrected one. Additionally, we investigate a change of correlation coefficient of reviewers' experience and the reviewing time by performing both data cleansing techniques. The result shows cleansing to reviewers effect to the correlation.

Keywords: code review, repository mining, data cleansing

1. はじめに

ソフトウェア開発におけるレビューとは，設計文書やソースコードを人が読み，設計の誤りやコードの記述ミス，コーディングルールの違反等の問題がないかを目視により検査するプロセスのことである．レビューの実施により，欠陥の早期発見や修正が可能となり，欠陥のおよそ 60%を

[1] 福岡工業大学
Fukuoka Institute of Technology, Fukuoka 811–0295, Japan
[2] 九州大学大学院
Kyushu University, Fukuoka 819–0395, Japan
[3] 名古屋大学
Nagoya University, Nagoya, Aichi 464–8601, Japan
[a)] toda@fit.ac.jp

発見可能であることが報告されている [1], [2].

近年ではコードレビューの手法としてツールの利用を前提とする Modern Code Review（MCR）が採用されている [3]．MCR はパッチに含まれる欠陥の検出やコードの品質向上を目的として行われ，オープンソースソフトウェア（OSS）の開発においても広く用いられてている．OSS における MCR プロセスの一例を説明する．OSS では開発はパッチ（ソースコード差分）を介して行われるが，投稿されるすべてのパッチはレビュー管理システムに登録され，必ず他の開発者からのレビューを受ける．レビューではパッチの内容について議論が行われ，妥当と判定されれば，プロダクトに反映（コミット）される．OSS を対象としたレビューデータの研究やデータ自体の公開が広く行われている [4], [5], [6].

しかしながら，OSS にはそれぞれ独自の開発プロセスや指針が存在するため，コードレビューデータの分析を含め，OSS から取得したデータをそのまま利用することが分析結果に少なくない影響を与えることが知られている．このため多くの研究でデータクレンジングが実施されている．データクレンジングとは，データの品質改善を目的としたデータ中のエラーの特定，除去等の処理を指し [7]，ソフトウェアリポジトリマイニングにおける重要性も指摘されている [8], [9]．ただし，多くの研究ではデータクレンジングは分析目的に適するようにエラーを除去するという手段として用いられるにとどまり，実際にデータクレンジングによる分析結果の変化を定量的に調査した研究は筆者の知る限り存在しない．

そこで本研究では Android[*1]，Chromium[*2]，OpenStack[*3]という 3 つの OSS 開発プロジェクトにおけるコードレビューデータを分析対象として，データクレンジングの効果を定量的に評価する．評価のためのアプローチとして，レビュアとレビュー開始・終了日時（すなわちその差分であるレビュー期間）に対してそれぞれデータクレンジングを実施し，その影響を調査する．この 2 つを対象として選んだ理由としては，レビュアはコードレビューに直接関係するためであり，レビュー期間はその長期化が重大な問題となりうる（有用な機能の提供や重大な不具合の修正の遅れ）ためである．これを 2 つのリサーチクエスチョン（RQ）としてまとめ，以下に示す．

RQ1：bot の除外はレビュアの分析に影響を与えるか

現状のレビュー管理システムのリポジトリ上に記録されているデータには人間によるレビューと bot によるレビューが混在している．人間によるレビューには人間の判断が必要であるのに対し，bot によるレビューはルーチンワーク，すなわちビルドやテスト結果の記述といった人間の判断が必要ない（介入しない）ものである．しかしながら既存のコードレビュー研究の多くは人間によるレビューを分析の対象としており，bot は除外（クレンジング）の対象である．そこで本 RQ ではレビュアに対するクレンジングの実行，すなわち bot の除去がレビュアの分析に与える影響を調査する．

RQ2：レビュー開始・終了日時へのクレンジングはレビュー期間に影響を与えるか

レビュー期間とは，バージョン管理システムに記録されているパッチ作成日時およびコミット日時の差分，もしくはレビュー管理システムに記録されているレビューページ開始日時および最終更新日時の差分を意味する．しかし，このようにして算出されるレビュー期間は実際のレビュー開始日時とレビュー終了日時の差分（レビュー期間）を表さない場合があり，レビューデータ，特にレビューに要する時間に着目した分析において障害となる．そこで本 RQ ではレビュー期間に対するクレンジングがレビュー期間の分析に与える影響を調査する．

以降，2 章では RQ への回答のために実施した調査の概要について説明し，3 章，4 章で RQ への解答を示す．5 章では 2 つの RQ をふまえた議論を，6 章でまとめと今後の課題を述べる．

2. 調査概要

2.1 調査目的

本章ではコードレビューの分析におけるデータクレンジングの影響を調査する．調査では実際のプロジェクトにおいてレビュアへのデータクレンジングとレビュー開始・終了日時へのデータクレンジングがコードレビュー分析に与える影響について分析する．また本研究では分析対象となるレビューは人によるものに限定し，bot によるものは対象外とする．この点については 6 章で詳述する．

具体的には，オープンソースソフトウェア開発プロジェクトの各種開発管理システムから入手したデータに対し，クレンジングを行ったデータセット（クレンジングデータ）と行わなかったデータセット（非クレンジングデータ）を作成し，レビュア，レビュー期間がデータクレンジングの有無によってどのように変化するか調査する．

分析対象として 3 種類のプロジェクト，Android, Chromium, OpenStack を選択した．以降，3 プロジェクトの概要について述べ，引き続き分析の元となるデータセットの取得手順を説明する．続いてクレンジングデータセットと非クレンジングデータセットの作成方法について説明し，最後に分析手順を述べる．

2.2 調査対象プロジェクト概要

本研究で分析対象とする 3 プロジェクトの概要を述べる．Android プロジェクトは Android OS を，Chromium

[*1] https://source.android.com/index.html
[*2] https://www.chromium.org/
[*3] https://www.openstack.org/

表 1 バージョン管理，レビュー管理から得るメトリクス
Table 1 Metrics collected from version control and review management system.

メトリクス	取得元	データ型
パッチ作成日時	バージョン管理システム	日付型
パッチコミット日時	バージョン管理システム	日付型
レビューページ作成日時	レビュー管理システム	日付型
レビューページ最終更新日時	レビュー管理システム	日付型
パッチ投稿者名	バージョン管理システム	文字列型
レビュア名	レビュー管理システム	文字列型

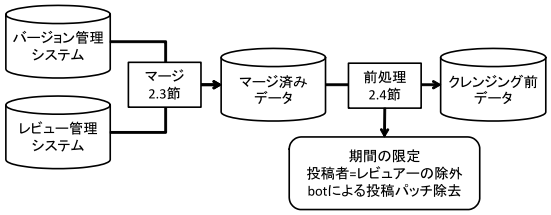

図 1 データ取得手順
Fig. 1 A procedure of collecting data and preprocessing.

プロジェクトは，Web ブラウザである Chromium と OS である Chromium OS を，OpenStack はクラウド基盤構築ソフトウェアの開発を目的とする OSS 開発プロジェクトである．バージョン管理システムとしては，Android および OpenStack は git を，Chromium は Subversion と git を利用（併用．リポジトリは同期されている）し，レビュー管理システムとしては Android，OpenStack は Gerrit を，Chromium は Rietveld を採用している．

2.3 調査対象データの取得とマージ

レビュア，レビュー期間の分析のためには，バージョンおよびレビュー管理システム上に記録されている各パッチについてレビュア名，パッチ投稿日時，レビュー完了日時等のメトリクスが必要となる．これらのメトリクスはバージョン管理システムとレビュー管理システムに別個に記録されているため，以下の手順でパッチごとにバージョン管理システムとレビュー管理システムの紐付けを行い，メトリクスを取得する．

(1) バージョン管理システムに記録されている全パッチのデータを抽出し，パッチの ID（ハッシュ値），変更行数等のメトリクスと対応するレビューの ID[*4]を抽出する．
(2) レビュー管理システム（gerrit，Rietveld）の履歴から，各レビューの開始時間，最終更新時間等のメトリクスとレビュー ID，もしくはレビュー対象パッチの git のハッシュ値を抽出する．
(3) 全パッチについてバージョン管理，レビュー管理システムそれぞれから得られたメトリクスを，レビュー ID もしくは git のハッシュ値を用いて紐付けする．

今回は，バージョン管理システムのデータは各プロジェクト git リポジトリ直接取得し，レビューデータは文献 [6], [10] のデータを採用し，紐付けを行った．

作成したリポジトリから**表 1** のメトリクスを抽出し，以降の分析に用いる．

2.4 調査データ前処理手順

2.3 節で得られたデータに対し 3 つの前処理を施す．ここまでの処理手順を**図 1** に示す．前処理完了後のデータ（図中クレンジング前データ）に対し，データクレンジングを行い，分析を実施する．

(1) 分析対象期間の限定
本研究で対象とするプロジェクトは，たとえば Android の最古のコミットが 2008/10/21 であるのに対し，OpenStack は 2011/01/04 であり，現在までの開発期間が大きく異なる．分析対象期間を統一するため，本研究ではレビューデータと紐付けできた最新のコミットから起算して過去 3 年分を分析対象とした．
(2) パッチ投稿者自身によるレビューコメントの除外
レビュアからつけられたコメントに対し，パッチの投稿者（製作者）がコメントを返すことがある．しかし，データ上では投稿者からのコメントもレビューコメントとして扱われるため，そのままではパッチ投稿者もレビュアとしてカウントされてしまう．MCR においてコードレビューはパッチに含まれる欠陥の検出やコードの品質向上を目的として第三者により実施されるもので，パッチ投稿者からのコメントはこの定義には従わない．よって，各パッチレビューにおいてパッチを投稿した開発者は，レビューコメントを返してい

[*4] たとえば Chromium プロジェクトでは，レビューの結果コミットと判定されたパッチは，レビュー管理システムにより自動的に挿入されるレビュー ID とともに，バージョン管理システムからコミットされる．

表 2 分析対象データ詳細
Table 2 Summary of studied three projects.

プロジェクト	期間	コミット数	レビュア数
Android	2011/12/31 – 2014/12/05	23,606	1,021
Chromium	2009/10/31 – 2012/10/31	87,219	2,110
OpenStack	2011/12/02 – 2014/12/01	90,778	2,860

てもレビュアとしては扱わないものとした．

(3) bot による投稿パッチの除外
レビューデータ上にはしばしば bot からの投稿が存在している[*5,*6]．これらのパッチは自動生成されるものであり，MCR ではレビューとして人の生成物を第三者がレビューすることを想定しており，自動生成されたパッチはこの定義に従わないため除外した．bot は後述の 3 章と同様の手順で特定した．

ここまでの処理が完了後のプロジェクトごとの分析対象期間，コミットされたパッチ数，レビュア数を**表 2** に示す．

3. RQ1. bot の除去はレビュアの分析に影響を与えるか

本章および次章では，本研究で注目する 2 つのデータクレンジングについて，それが必要となる理由（動機）と対策（方法）を述べ，引き続いて調査から得られた結果とリサーチクエスチョン（RQ）への回答を述べる．説明の都合上，後述する分析で用いた 3 種類の OSS プロジェクト，Android，Chromium，OpenStack とそこで動作しているバージョン管理システム git，およびレビュー管理システム Gerrit，Rietveld を例に用いている．

3.1 動機

近年の OSS では Continuous Integration（CI）の導入がさかんであり，ビルドやテストの自動化が行われている．今回分析対象としたプロジェクトにおいても各種自動化が行われており，bot として稼働している．しかしながら，たとえば自動テスト用の bot は，レビューページが作成（パッチが投稿）された時点やパッチがアップデートされた時点で自動テストを実行し，その結果をレビューコメントとして残す．この場合，データ上では bot はレビュアとして取り扱われ，分析において障害となるため，除外する必要がある．また外部プロジェクトのライブラリ等の変更を反映する目的で，一部のパッチの投稿者やコミッタが bot の場合もあり，同様に除外する必要がある．

3.2 方法

今回の分析において除外の対象となる bot の特定方法は各プロジェクトで異なる．共通の処理として，各プロジェクト中のパッチ投稿者およびレビュアからメールアドレスに bot，jenkins，gerrit，no-reply，ci，build のいずれかを含むものを抽出し，さらに目視で bot を特定した．これは「ci」や「bot」は人名の一部に含まれる場合があるためである．Android，OpenStack ではさらに別の方法でも bot を特定した．

(1) Android
文献 [11] で指摘されている Deckard Autoverifier（deckard@android.com）を除外した．

(2) OpenStack
OpenStack では稼働中のすべての bot は専用の Web ページ[*7]にまとめられている．本論文では各ページ内 Contact Information にあげられているすべてのメールアドレスについて，bot かどうかを目視で確認し，判定した．具体的には，そのメールアドレスを用いたパッチが投稿されていないか，もしくは投稿されているがマージ（コミット）されていないか（bot のテストとしてパッチ投稿がなされる場合があるが，コミットはされない），bot の開発者の開発行動に使われていないかについて目視で確認した．さらに残りのレビュアの中から目視で発見した trivial-rebase@review.openstack.org を bot と特定した．

上記手順の結果，Android からは 3 名，Chromium からは 2 名，OpenStack からは 121 名をレビュア中の bot として除去対象とした．

3.3 結果

各プロジェクトのクレンジング前データから，全レビュアについてレビューに参加したパッチの数（レビュー回数）を測定した．レビュー回数上位 10 名について，bot か否かおよびレビュー回数とともに**表 3** に示す．

bot に対する除去（クレンジング）対象となるレビュアの数は，Android 3 名，Chromium 2 名，OpenStack 121 名であり，表 2 のレビュア数と比較すると，その割合は非常に小さい（最も高い OpenStack でも 5%以下）．しかしながら表 3 から，OpenStack では上位 10 名中 3 名が bot であり，残りのプロジェクトでも 2 位が bot である．次に，各プロジェクトにおける bot によるレビュー回数が全レビュー回数に占める割合（bot レビュー割合）を**表 4** に

*5 https://review.openstack.org/#/c/82230/
*6 https://review.openstack.org/#/c/86238/
*7 ThirdPartySystems
https://wiki.openstack.org/wiki/ThirdPartySystems

表 3 レビュー回数上位中の bot
Table 3 Reviewer ranking of reviewing frequency.

	Android		Openstack		Chromium	
順位	bot	回数	bot	回数	bot	回数
1	N	3,233	N	6,986	**Y**	**9,289**
2	**Y**	**2,179**	**Y**	**6,021**	N	5,793
3	N	2,084	N	4,140	N	5,188
4	N	1,857	N	3,959	N	5,084
5	N	1,597	N	3,814	**Y**	**4,455**
6	N	1,322	N	2,811	N	4,020
7	N	1,238	N	2,643	**Y**	**3,970**
8	N	1,167	N	2,560	N	3,929
9	N	1,145	N	2,541	N	3,844
10	N	1,092	N	2,277	N	3,535

表 4 プロジェクト別 bot レビュー割合
Table 4 Reviewing ratio by bots.

	全体	上位 10 名
Android	6.0%	12.9%
Chromium	3.6%	15.9%
OpenStack	19.4%	36.1%

示す．運用されている bot の数が多い OpenStack においては全体で 19.4%，表 3 で示した上位 10 名に限定するとすべてのプロジェクトにおいてレビューの回数の 1 割以上が bot によるものとなる，すなわち，この結果は bot の数がレビュアに対してごく少数であっても，レビュー回数や全体のレビューに占める割合では無視できないほど大きくなりうる，ということを示しており，bot の除去はレビュアの分析に影響を与えうる，と考えられる．

> ***bot* によるレビューが全体の *20%*程度を占める場合があり，*bot* の除去はレビュアの分析に影響を与えることが分かった．**

4. RQ2. レビュー開始・終了日時へのクレンジングはレビュー期間に影響を与えるか

4.1 動機

コードレビューに関わるメトリクスのうち，日時の関連するものとしては，バージョン管理システムからパッチの製作日時とコミット日時が，レビュー管理システムからはレビューページの作成日時とレビューページの最終更新日時があげられる．たとえばレビュー期間というメトリクスを利用したい場合，単純な方法としてはバージョン管理システム，レビュー管理システムのどちらか一方を採用し，その差分をとる，というものが考えられる．しかしながら，実際のリポジトリデータから取得したデータを利用する場合には，その方法で取得した日時が信用できない事例が発生する．具体例として以降 4 種類をあげ，説明する．

(1) パッチ作成日時の更新

近年の OSS プロジェクトでははは共通してバージョン管理システムとレビュー管理システムの連携されており，パッチを投稿すると自動的にレビューページが作成される．ただしレビュー中にレビュアの指摘に従ってパッチを変更した結果，git のパッチも作成日時が更新される（おそらくはパッチを作り直しており，git のリポジトリ上でも新規にパッチを作成した扱いになっている）事例があり，このような事例では git 上に記録されたパッチ作成日時を信頼できない[*8]．

(2) 一括コミット

バージョン管理システムの日時データを利用する場合にのみ起こりうる事例である．特に大規模な OSS プロジェクトのバージョン管理システムにおいては，しばしば大量のコミットがほぼ同一日時に同一人物（コミッタ）によって行われる事例がある．これは特定のコミッタがある時点でコミット可能なパッチを一括してコミットしていると推定できる．一例として，Android において同一人物，ほぼ同時刻に 4 件がコミットされる事例が存在する[*9,*10,*11,*12]．このような事例では，そのコミット日時をレビューの終了日時として採用するのは適切ではない．

(3) レビューページの追加コメント

レビュー管理システムの日時データを利用する場合にのみ起こりうる事例である．レビューが完了しパッチがコミットされた後に，当該パッチについてリバート，リベースやバグを含んでいた等の旨の投稿等がなされることがあり，その場合には当然ながらレビューページの最終更新日も更新されるため，実質的なレビュー完了日との間に乖離が生じるため，信頼できない[*13]．

(4) プロジェクト固有の問題

今回の分析対象プロジェクトでは Chromium だけが該当する事例であるが，git から正しいパッチ作成日時を得られない場合がある．Chromium では git から得られるすべてのパッチのデータにおいて作成日時とコミット日時が一致している（レビューデータから，コミット日時が作成日時に代入されていると推測される）．原因としては Chromium の git リポジトリがメインのバージョン管理システムである Subversion からの同期で作成されているためであると推測される．このような場合には git のパッチ作成日時を信頼することはできない．

4.2 方法

4.1 節で述べた状況をふまえ，実質的なレビュー開始日時

*8 https://android-review.googlesource.com/#/c/73002/
*9 https://android-review.googlesource.com/#/c/16213/
*10 https://android-review.googlesource.com/#/c/18363/
*11 https://android-review.googlesource.com/#/c/20047/
*12 https://android-review.googlesource.com/#/c/20048/
*13 https://android-review.googlesource.com/#/c/20471/

表 5 プロジェクト，クレンジング別レビュー期間（単位：時間）
Table 5 Changes of reviewing time by data cleansing (hour).

プロジェクト名	クレンジング	中央値	平均値
Android	あり	0.49	369.85
	なし	14.00	257.27
Chromium	あり	11.66	90.39
	なし	142.15	2434.94
OpenStack	あり	3.109	175.43
	なし	54.31	54.54

（実レビュー開始日時）については git から得られるパッチ製作日時とレビューページ作成日時のうち早い方を採用する．同様に実質的なレビュー終了日時（実レビュー終了日時）についてはコミット日時とレビューページ最終更新日時の組合せにおいて早い方を採用する．よって，この 2 者間の差分がデータクレンジング後のレビュー期間となる．

4.3 結果

クレンジング後のレビュー期間の比較対象となるレビュー期間は，上述の Chromium 固有の問題からレビューページ作成日時からレビューページ最終更新日時の差分とする．各プロジェクトについて，各パッチのレビュー期間の平均値，中央値をデータクレンジングの有無別で**表 5** に示す．表のとおり，クレンジングの有無によって，レビュー期間に大きな差が現れることが分かった．またクレンジングの有無の 2 群に対して wilcoxon の順位和検定を実施した結果，すべてのプロジェクトについて有意差が見られた．

レビュー開始・終了日時へのクレンジングの有無によりレビュー期間に有意な差が現れる場合があり，レビュー開始・終了日時へのクレンジングはレビュー期間に影響を与えることが分かった．

5. 議論

本章ではまず RQ1，RQ2 に対する分析を通して知見と分析結果のまとめについて述べる．続いて分析を通して得られた知見の活用について述べ，最後に本研究で実施したデータクレンジングにかかる労力とそれをふまえたデータクレンジングの実施の可否判断について述べる．

5.1 2 つのクレンジングがデータ分析に与える影響

本研究では 2 つの RQ を設定し，データクレンジングがレビュア，レビュー期間に与える影響を調査した．しかしながら，2 つの RQ は分析に用いるメトリクスへのデータクレンジングの影響を調査するにとどまっており，実際にそれらのメトリクスを用いた分析にデータクレンジングがどのような影響を与えるのかは明らかではない．そこで本章では，レビュア，レビュー期間の 2 つのメトリクスを用いる分析として，筆者らが以前に実施した，レビュー経験とレビュー期間の関係の分析 [12] を採用し，データクレンジングが与える影響について調査する．

表 6 Android におけるレビュー回数の上位割合と対応するレビュア数
Table 6 Freaquency of review and a number of reviewers in Android project.

percentile	投稿回数	投稿者数
1%	962	10
5%	**112**	**50**
10%	43	101
20%	12	202

データは 2 章で作成したクレンジング前データをもとに作成する．クレンジング前データに対し，RQ1，RQ2 で説明した方法を適用したデータをクレンジングデータ，適用しなかったデータを非クレンジングデータとして分析を行い，結果を比較する．

本論文ではレビュー経験とレビュー期間の関係として，2 者間の相関係数を採用する．ただし，現時点ではレビュー経験というメトリクスは存在しないため，これを付加する必要がある．そこで，データ中に記録されているすべてのパッチを日時でソートし，レビュアの経験回数，すなわち各パッチがそのレビュアにとって何回目のレビューか，という情報を付加し，分析に用いる．これに加え，ごく少数のパッチのレビューにしか参加していないレビュアも存在するため，以下の手順でフィルタリングと相関係数の算出を行う．まず，レビュアを分析対象期間中のレビュー回数によって順位付けする．次にレビュー回数が上位 n%（n = 1, 5, 10, 20）に該当するレビュア，およびその各区間に該当するレビュア（たとえばレビュー回数が上位 1%未満 5%以上に該当するレビュア）をそれぞれ抽出し，分析対象とする．ここで割合を採用したのは分析対象のレビュア数がプロジェクトごとに異なるためである．さらに各区間においてもレビュアごとにレビュー回数は異なる．相関係数の算出には欠損が含まれていてはならないため，その中で最もレビュー回数が少ないレビュアに合わせる形で実施し，欠損のないデータを作成，相関係数を算出する．具体例として，Andorid のクレンジングデータにおけるレビュー回数上位の割合とそれに対応するレビュア数，レビュー回数を表した**表 6** を用いて説明する．

まず表 6 中のたとえば太字で示した 5%の行は，上位 5%に相当するレビュアは 50 人おり，その中で最小のレビュー回数，すなわちレビュー回数が 50 番目のレビュアは 112 回レビューしたことを示す．ここで無欠損のデータを作成するため 50 番目以上のレビュアは 112 回以上のレビューを行っているが，分析では 113 回目以降のレビューについては無視し．50 人のレビュアについて，1 回目から 112 回目までのレビュー期間を対象とする（すなわち

表 7 レビュー回数の上位割合と対応するレビュア時間の相関係数
Table 7 Correlation coefficient of reviewers' experience and reviewing time.

	クレンジング対象	−0.01	−0.05	−0.1	−0.2
Android	なし	−0.097	−0.391	−0.501	−0.126
	両方	0.052	−0.043	0.001	−0.018
	bot	−0.105	−0.407	−0.486	−0.259
	レビュー期間	0.045	−0.053	−0.035	−0.028
Chromium	なし	−0.644	−0.848	−0.862	−0.950
	両方	0.410	0.660	0.660	0.654
	bot	−0.643	−0.848	−0.860	−0.945
	レビュー期間	0.466	0.673	0.683	0.643
Openstack	なし	0.811	0.881	0.877	0.858
	両方	0.440	0.609	0.615	0.731
	bot	0.805	0.893	0.905	0.875
	レビュー期間	0.481	0.575	0.617	0.677

	クレンジング対象	0.01-0.05	0.01-0.10	0.01-0.20	0.05-0.10	0.05-0.20	0.10-0.20
Android	なし	−0.347	−0.420	−0.189	−0.087	−0.189	−0.476
	両方	0.061	0.143	0.070	0.163	0.056	−0.175
	bot	−0.333	−0.416	−0.238	−0.035	−0.077	−0.343
	レビュー期間	0.017	0.128	−0.014	0.268	−0.084	−0.252
Chromium	なし	−0.906	−0.855	−0.955	−0.945	−0.986	−0.980
	両方	0.654	0.661	0.658	0.516	0.631	0.529
	bot	−0.902	−0.853	−0.949	−0.939	−0.984	−0.979
	レビュー期間	0.667	0.672	0.657	0.538	0.628	0.525
Openstack	なし	0.885	0.880	0.878	0.869	0.862	0.875
	両方	0.628	0.664	0.764	0.649	0.761	0.703
	bot	0.897	0.916	0.883	0.899	0.871	0.837
	レビュー期間	0.600	0.633	0.694	0.670	0.713	0.724

50（人）× 112（回）の行列データとして取り扱う）．

作成したデータに対し，各回のレビュー期間の中央値と回数の相関係数を算出し，分析の対象とした．

プロジェクト別の 2 種類のクレンジング手法を各々実施した場合，両方とも実施した場合，いっさい実施しなかった場合のレビュー回数とレビュー期間の相関係数を**表 7** に示す．ここで，分析方法としてレビュー回数ごとにレビュアを抽出し，その回数に該当するレビュー期間中央値との相関係数を用いた．

まずクレンジング手法を適用しなかった場合と両方を適用した場合の結果について述べた後，その結果を比較し，さらに 2 つのクレンジング手法がそれぞれデータ分析に与える影響についての結論を述べる．

(1) クレンジング手法を適用しなかった場合

表 7 より Andorid ではレビュアのレビュー回数とレビュー期間について一部で負の相関が見られた．Chromium ではすべての区間で相関係数 −0.4 以下，上位 1%以外では −0.8 以上の非常に強い負の相関が見られた．OpenStack においてはすべての区間で相関係数 0.8 以上と非常に強い負の相関が見られた．また Chromium，OpenStack では全区間で相関は有意であった．

(2) 両方のクレンジング手法を適用した場合

Andorid ではレビュアのレビュー回数とレビュー期間に相関はほとんど見られなかった．Chromium, OpenStack においてはすべての区間で相関係数 0.4，上位 1%（表中 −1%）以外では 0.5 以上の正の相関が見られた．また Chromium，OpenStack では全区間で相関は有意であった．

(3) bot の除去だけを適用した場合

表 7 から，クレンジング手法を適用しなかった場合，および両方を適用した場合と比較すると，bot の除去の影響はそれほど大きくないことが分かる．ただし Android の上位 20%や同上位 10%から上位 20%の区間等，一部においては bot が相関係数に大きな影響を与えることが分かる．

(4) レビュー開始・終了日時へのクレンジングだけを適用した場合

表 7 から，クレンジング手法を適用しなかった場合，および両方を適用した場合と比較すると，レビュー開始・終了日時へのクレンジングの影響は bot の除去に比べると相関係数への影響が非常に大きいことが分かる．ただし，Openstack の上位 20%や同上位 1%から 20%の区間のように，bot の除去と組み合わせた場合に相関係数が大きく変化する場合も見られる．よって，bot の除去とレビュー開始・終了日時へのクレンジングのどちらか一方ではなく，両方を実施すべきと考えられる．

5.2 データ分析結果のまとめ

まず，データクレンジングがレビュー期間とレビュー回数の相関係数に与える影響をまとめる．

(1) bot の除去は相関係数にそれほど大きな影響を与えない．ただし一部の条件下では影響を与えることがある．
(2) レビューの開始日時，終了日時へのクレンジングは相関係数に大きな影響を与える．
(3) bot の除去とレビューの開始日時，終了日時へのクレンジングを組み合わせることで相関係数に影響が現れる場合がある．

5.3 想定する活用方法

本研究を通して得られた，データクレンジングが分析に与える影響についての知見は，リポジトリマイニングをソフトウェア開発に役立てたい研究者にとって，リポジトリマイニングからより高い精度の知見を得るという点で有用と考えられる．たとえば，リポジトリマイニングの研究者が，レビュー時間削減についての分析を通して得られた知見を生かし，レビュー時間が短くなるようにレビュアを割り当てる方法を検討する際に，より正確な分析結果に基づき検討を行うことができると考えられる．

5.4 データクレンジングにかかる労力

5.1 節から，データクレンジングが分析に影響を与えることが示された．ただし分析内容が変わればデータクレンジングが影響をほとんど，もしくはまったく与えないこともありえ，その場合データクレンジングを行う必要がなくなる．本節ではデータクレンジングにかかる労力と影響の関係について議論する．

本研究におけるデータクレンジングは 4 つの手順からなる．

(1) 対象となるプロジェクトから提供されているドキュメントを通して，分析対象のプロセスとそこに含まれるメトリクスのデータ形式を把握する．
(2) リポジトリデータ上のメトリクスの値からクレンジングの可否を判断する．
(3) (2) でクレンジングが必要と判断した場合，クレンジング用スクリプトを作成する．
(4) (3) で作成したスクリプトを計算機で実行する．

具体例として本実験における OpenStack での bot のクレンジングを実施する際の手順を以下に述べる．

(1) プロジェクトから提供されるパッチ投稿からコミットまでのプロセスを示したドキュメント（Openstack の場合，Developer's guide[*14]）を通してレビューとは人間の判断が必要なもの，テストとは人間の判断が必要ではないものを指すこと，およびテストが bot によって実施されていることを理解（データ形式の把握）する．
(2) レビュー管理システム上にレビュアとして bot が含まれていることを確認し，クレンジングが必要と判定する．それにともない，bot 一覧が記載された Web ページから bot のリストを取得する．
(3) R 言語で bot リストに含まれるレビューを分析対象から除外するスクリプトを実装する．
(4) 実装したスクリプトを計算機上で実行する．

(1) は，プロジェクトからドキュメントが提供されており，一般的に容易に理解可能と考えられる．(2) は，メトリクスのデータ形式とリポジトリデータ上の値から，クレンジングの必要性に容易に判断が可能である．(3) については，本研究で使用した各スクリプトはいずれも R 言語で 100 行程度であり，R 言語を習熟している第 1 著者はいずれも 3 時間以内で開発できた．(4) については，本研究のスクリプトは，いずれの対象に対しても実行に要する時間は 3 時間程度であった．

(1)〜(4) 全体を通して，1 プロジェクトあたり 1 人日程度の工数を確保可能あれば，第 1 著者には実施可能であり，(1), (2) については，一般のコードレビューに関する知識は必要であるが，特定のオープンソースプロジェクトに関する知識は必要ない．(3) については，R 言語に対する一定のスキルがあれば十分と考える．これらをふまえ，R 言語に対する一定のスキルを有する研究者であり，かつ 1 人日程度の工数を確保できるのであれば，本研究のクレンジングを適用できると考えられる．

クレンジングを実施すべきかの判断については，たとえば本研究では bot の除去というクレンジングを実行したが，bot が動作する頻度がきわめて小さい場合は，bot に関するクレンジングが不要となる場合が考えられる．期間に関するクレンジングについては，git と gerrit/rietveld から取得できるメトリクスに原因があるため，これらシステムを使用する限りは必要と考えられる．

6. 関連研究

6.1 データクレンジング

本研究で主題としたデータクレンジングについて，その重要性が既存研究において指摘されている．たとえば，Zimmermann ら [8] はバージョン管理システムのデータを分析するにあたって行うべき前処理について 4 つの例をあげ，また前処理の必要性について議論している．Mockus [9] は分析においてデータクレンジングが全体の 95%を占めていると述べ，その重要性を指摘している．ただしこれらの研究はデータクレンジングの必要性，重要性については議論されているものの，その具体的な分析への影響については検証していない．

また本研究に類似した既存研究として，Issue Report に

[*14] http://docs.openstack.org/infra/manual/developers.html

おけるミスラベリングを扱った論文が存在する [13]．ただし当該論文はミスラベリングというバグの有無に関するラベル付けを誤ったという現象が欠陥予測に与える影響を調査しており，データクレンジングのようなプロセスを提案した本研究とは異なる．また，ミスラベリングが出力のみを対象としているのに対し，データクレンジングがプロセスとそれにともなう出力を対象としているという点でも異なる．

6.2 コードレビュー

本研究ではコードレビュー分析を対象にデータクレンジングを実施した．コードレビューについての論文は数多く存在し，たとえば Gousious ら [14] は GitHub におけるコードレビューの対象である Pull Request について総合的な分析を実施しており，その中で各 Pull Request のコミットの可否およびコミットまでの時間と各種メトリクス（変更行数，コメント数，プロジェクト全体のコード行数用）との関係を調べている．Thongtanunam ら [15] は 4 種類の OSS プロジェクトを対象に，コードレビューにおいてレビュアの割当てがレビュー期間に与える影響について調査している．しかしながら既存研究では，前処理に相当するデータクレンジングを実施はしているものの，その効果については検証しておらず，本研究とは目的が異なる．

またコードレビューを対象とした既存研究の多くは分析対象となるレビューを人間によるものに限定しており，bot によるものは対象としていない [4], [12], [15], [16], [17]．これは現状ではコードレビューに関する研究が，その多くにおいて目的を人によるレビューの効率化としているためである．本研究でもそれに従い，レビュアの対象を人に限定している．ただし近年 CI（Continuous Integration）の研究が進んでおり [18], [19]，今後の研究動向によっては bot の活動を分析の対象とすることもありうる．

7. おわりに

本論文ではコードレビュー分析においてデータクレンジングが与える影響について調査した．具体的には，1) bot の除去はレビュアの分析に影響を与えるか，2) レビュー開始・終了日時へのクレンジングはレビュー期間に影響を与えるか，という 2 点に対し調査を実施した．そのアプローチとして，Android，Chromium，OpenStack の 3 つの OSS プロジェクトからコードレビューに関するデータを取得した．取得したデータに対してクレンジングを行う，もしくは行わないデータをそれぞれ作成し，分析・比較した．比較の結果，データクレンジングの有無はレビュア，レビュー期間の両方に影響を与えることが分かった．さらに，具体的なコードレビュー分析としてレビュー経験とレビュー期間の関係（相関）を対象に，レビュア，レビュー期間の各々についてデータクレンジングを実施した，もしくは実施しなかったデータを作成し，分析結果を比較した．比較の結果，レビュアへのクレンジングは一部の条件下で相関係数に影響を与えることが，レビュー期間へのクレンジングは総じて相関係数に大きな影響を与えることが分かった．また 2 つのクレンジングを組み合わせることで相関係数が変化する可能性があることも示唆された．

今後の課題として，他の OSS プロジェクトに対しても同様の分析を行い，データクレンジングの影響を調査すること，特に今回レビュー経験とレビュー期間の関係について得られた知見が他プロジェクトにおいても成立しうるのかを明らかにすることがあげられる．それに加え，データクレンジングがそもそも必要ない出力を与える開発管理システムの開発も課題である．今回実施したクレンジング，特にレビュー期間の補正は，2 つの管理システムから取得した日付型データを組み合わせることで実際のレビュー期間を算出する試みであり，これは 2 つの開発管理システムを組み合わせた結果生じた齟齬の解消といえる．この齟齬の解消には開発システムからの出力の変更が必要であり，開発管理システムを開発する側へのリポジトリマイニングにおけるデータクレンジングについての問題の周知や問題意識の浸透が必要と考えられる．

参考文献

[1] Boehm, B. and Basili, V.: Software Defect Reduction Top 10 list, *Computer*, Vol.34, No.1, pp.135–137 (2001).

[2] Melo, W., Shull, F. and Travassos, G.H.: Software Review Guidelines, *COPPE/UFRJ Systems Engineering and Computer Science Program Technical Report ES-556/01* (2001).

[3] Bacchelli, A. and Bird, C.: Expectations, outcomes, and challenges of modern code review, *Proc. 2013 International Conference on Software Engineering*, pp.712–721, IEEE Press (2013).

[4] Kononenko, O., Baysal, O. and Godfrey, M.W.: Code review quality: How developers see it, *Proc. 38th International Conference on Software Engineering*, pp.1028–1038, ACM (2016).

[5] Gousios, G.: The GHTorrent dataset and tool suite, *Proc. 10th Working Conference on Mining Software Repositories*, *MSR '13*, pp.233–236 (2013).

[6] Hamasaki, K., Kula, R.G., Yoshida, N., Cruz, A., Fujiwara, K. and Iida, H.: Who does what during a code review? datasets of OSS peer review repositories, *Proc. 10th Working Conference on Mining Software Repositories*, pp.49–52, IEEE Press (2013).

[7] Rahm, E. and Do, H.H.: Data cleaning: Problems and current approaches, *IEEE Data Eng. Bull.*, Vol.23, No.4, pp.3–13 (2000).

[8] Zimmermann, T. and Weißgerber, P.: Preprocessing CVS data for fine-grained analysis, *Proc. 1st International Workshop on Mining Software Repositories*, pp.2–6 (2004).

[9] Mockus, A.: How to run empirical studies using project repositories, *4th International Advanced School of Empirical Software Engineering* (*IASESE 2006*) (2006).

[10] Xin, Y., Yoshida, N., Kula, R.G. and Hajimu, I.: Peer

review social network (PeRSoN) in open source projects, *IEICE Trans. Information and Systems*, Vol.99, No.3, pp.661–670 (2016).
[11] Mukadam, M., Bird, C. and Rigby, P.C.: Gerrit software code review data from android, *2013 10th IEEE Working Conference on Mining Software Repositories* (*MSR*), pp.45–48, IEEE (2013).
[12] 戸田航史，亀井靖高，濱崎一樹，吉田則裕：Chromium プロジェクトにおけるレビュー・パッチ開発経験がレビューに要する時間に与える影響の分析，コンピュータソフトウェア，Vol.32, No.1, pp.1_227–1_233 (2015).
[13] Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Ihara, A. and Matsumoto, K.: The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models, *Proc. 37th International Conference on Software Engineering – Volume 1*, *ICSE '15*, Piscataway, NJ, USA, pp.812–823, IEEE Press (online), available from ⟨http://dl.acm.org/citation.cfm?id=2818754.2818852⟩ (2015).
[14] Gousios, G., Pinzger, M. and Deursen, A.v.: An exploratory study of the pull-based software development model, *Proc. 36th International Conference on Software Engineering*, pp.345–355, ACM (2014).
[15] Thongtanunam, P., Tantithamthavorn, C., Kula, R.G., Yoshida, N., Iida, H. and Matsumoto, K.: Who should review my code? A file location-based code-reviewer recommendation approach for modern code review, *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering* (*SANER*), pp.141–150, IEEE (2015).
[16] Thongtanunam, P., McIntosh, S., Hassan, A.E. and Iida, H.: Revisiting Code Ownership and Its Relationship with Software Quality in the Scope of Modern Code Review, *Proc. 38th International Conference on Software Engineering*, *ICSE '16*, New York, NY, USA, pp.1039–1050, ACM (online), DOI: 10.1145/2884781.2884852 (2016).
[17] McIntosh, S., Kamei, Y., Adams, B. and Hassan, A.E.: An empirical study of the impact of modern code review practices on software quality, *Empirical Software Engineering*, Vol.21, No.5, pp.2146–2189 (online), DOI: 10.1007/s10664-015-9381-9 (2016).
[18] Adams, B. and McIntosh, S.: Modern Release Engineering in a Nutshell – Why Researchers should Care, *Leaders of Tomorrow: Future of Software Engineering, Proc. 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering* (*SANER*), Osaka, Japan (2016).
[19] Jiang, Y. and Adams, B.: Co-evolution of Infrastructure and Source Code: An Empirical Study, *Proc. 12th Working Conference on Mining Software Repositories*, *MSR '15*, Piscataway, NJ, USA, pp.45–55, IEEE Press (online), available from ⟨http://dl.acm.org/citation.cfm?id=2820518.2820527⟩ (2015).

戸田 航史 （正会員）

2004 年大阪大学基礎工学部卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学同研究科博士研究員．2012 年より福岡工業大学情報工学科助教，博士（工学）．ソフトウェアメトリクス，ソフトウェアリポジトリマイニングの研究に従事．電子情報通信学会，IEEE 各会員．

亀井 靖高 （正会員）

2005 年関西大学総合情報学部卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年日本学術振興会特別研究員（PD）．2010 年カナダ Queen's 大学博士研究員．2011 年九州大学大学院システム情報科学研究院助教．2015 年より同大学院准教授．博士（工学）．ソフトウェアメトリクス，マイニングソフトウェアリポジトリの研究に従事．電子情報通信学会，IEEE 各会員．

吉田 則裕 （正会員）

2004 年九州工業大学情報工学部知能情報工学科卒業．2009 年大阪大学大学院情報科学研究科博士後期課程修了．同年日本学術振興会特別研究員（PD）．2010 年奈良先端科学技術大学院大学情報科学研究科助教．2014 年名古屋大学大学院情報科学研究科附属組込みシステム研究センター准教授．2017 年より同大学大学院情報学研究科附属組込みシステム研究センター准教授（改組による）．博士（情報科学）．コードクローン分析手法やリファクタリング支援手法に関する研究に従事．