5K-08

GitHubのデータを利用したコードレビュー時間の推定

大山義人† 大場みち子†
公立はこだて未来大学 システム情報科学部†

1. 研究背景

システム開発プロセスにおいて，コードレビューが多くの開発現場で取り入れられている．コードレビューとは，IPAによると「開発者担当者が書いたソースコードを閲読してセキュリティ脆弱性あるいはそのきざしを読み取る作業」[1]とされている．その中でもGitHubなどのバージョン管理サービスを使用したコードレビューが一般的である．GitHubはコードの変更内容を本番環境に反映する前にレビューできるPullRequest機能を持っており，コードレビューに適したサービスである．コードレビューは『レビュワーのスキル』やレビュー対象の『ソースコードの属性』など多くの要因が関係しているため，レビューにかかる時間などの『レビュー結果』を推定しにくい．『レビュワーのスキル』や『ソースコードの属性』，『レビュー結果』の関係を分析することで，コードレビューの質や所要時間の見積もりができる可能性が高い．コードレビューに関する指標化と，それを用いた推定は多くの研究でされている．一方で，コードレビュー時間に着目した研究は少なく，コードレビュー時間を事前に推定することは困難である．コードレビュー時間を推定できると，コードレビューに関するタスク管理が容易になる．

2. 目的と目標

本研究の目的は，ソースコードのレビュー時間を推定することである．これを達成するための目標として，レビュー時間に関係する要素を分析をする．

3. 関連研究

コードレビュー時間に関係する研究を述べる．

ソースコードの属性として，循環的複雑度[2]と呼ばれるソースコードの複雑さを測定する手法がある．1＋分岐（if，for，whileなど）の数で表すことができ，その数字の大小によってコードの構造がどれだけ複雑か推定できる．循環的複雑度はソースコードの可読性を数量的に表すことを表している．

GitHubとStackOverflowというエンジニアコミュニティサイトの活動実績から開発者のスキルをスコア化した研究[3]がある．開発者のGitHubでの活動やStackOverflowでの回答から，開発者がどの言語や技術ジャンルに詳しいか分析をしている．

コードレビュー分析のためにレビュー内容のカテゴリ分けした研究[4]がある．コードレビューのコメントをどのような種類のコメントか手動でタグ付けし，カテゴリ分けをしている．具体的な分類の種類には，バグや欠陥の発見，コードの改善，代わりのアプローチの提案などが考えられる．この手法を利用することにより，コードレビュー内容の分析をすることができる．

4. 研究課題

研究目的に対して，レビュー時間は，関連研究で述べたようなソースコードの属性やレビュワーの情報，レビューされる人の情報など多くの要素が関わっているという課題がある．

5. 解決アプローチ

GitHubを対象にどのような要素がコードレビュー時間に関係しているのかを分析する．要素としてレビュワーのフォロワー数やコミット数などのレビュワー情報やレビューされる人の情報，ソースコードの複雑度や変更行数などのソースコードの情報等を候補とする．

分析では，レビュー時間とそれに関係する要素との関係性を調査する．レビュー時間は，多変量解析の1つである重回帰分析を利用して推定する．目的変数であるレビューにかかる時間は数量データである．要因を分析する説明変数は数量データと質的データが混在している．質的データはダミー変数へ変換することで数量データとして扱うことができるため，重回帰分析を用いる．

6. 実験

GitHubでの対象のリポジトリはスター数が多く，コードレビューが行われているOSSリポジトリを選んだ． スター数とは，GitHub上で個人がリポジトリを覚えるため，メンテナに感謝を示すために使用されており，リポジトリの信頼度の目安として使用することができる．データ取得方法として，GitHub REST API v3を使用する．APIクライアントPyGitHubを使用して，

Estimating Code Review Time Using GitHub Data
†Yoshito Oyama †Michiko Oba
†School of System Information Science, Future University Hakodate

GitHub REST API v3 を利用しデータを対象リポジトリから取得する．

実験は GitHub からデータ取得，データの前処理，重回帰分析を使用してコードレビュー時間との関係性の分析及び推定の順番で行う．

取得したデータは統計的解析手法で分析しやすいように，ノイズデータを取り除くなどの前処理をする．その後，目的変数をコードレビュー時間として，重回帰分析し，コードレビュー時間に関する要因分析・推定する．分析に使用する目的変数はコードレビューにかかった時間，説明変数はレビューを受ける人の過去のレビュー時間の平均値，レビュワーの過去のレビュー時間の平均値，リポジトリの過去のレビュー時間の平均値，ソースコードの変更行数の 4 つである．推定モデルの精度を測るためにデータの25%を切り分けておき，テストデータとして利用する．

実験の結果を表１，表 2 に示す．説明変数の中では特にレビューを受ける人の過去のコードレビュー時間の平均値に相関関係がみられた．コードレビュー時間の推定モデルを重回帰分析を利用して作成したが，決定係数が 0.25 となり十分な精度を得ることは出来なかった．

表 1　レビュー時間(目的変数)と説明変数の相関関係

	レビューされる人の過去のレビュー時間平均	レビュワーの過去のレビュー時間平均	リポジトリの過去のレビュー時間平均	ソースコードの変更行数
相関係数	0.56	0.24	0.08	0.01

7. 考察

コードレビュワーの過去のレビュー時間の平均値がコードレビュー時間と関係があることから，コードレビュー時間はコードレビュワーの信頼度やレビュワーのコメント能力に依存すると考えられる．また，vscode の推定精度が最も高く react が最も低かった．これは，vscode のリポジトリでは PullRequest の説明に関連する issue を利用して説明されることが多かったが，react では少なかったことから PullRequest を作成する際のメッセージのわかりやすさが関係している可能性がある．また，フォーク数が多いリポジトリほど制度が低かったことから，活発なリポジトリは多くの人が関わるためレビューされる人の数が増え，精度が低下した可能性がある．

表 2　リポジトリごとの実験結果

	Total	react	vscode	laravel	vue
決定係数（学習データ）	0.45	0.13	0.62	0.29	0.15
決定係数（テストデータ）	0.25	0.17	0.33	0.25	0.25
データ数	5097	1904	1877	718	601
平均レビュー時間	414.49	220.50	353.22	164.92	414.49
フォーク数(千)	136	27.2	13.6	17.6	23.3

8. 結言

本研究では GitHub の情報を利用して，コードレビュー時間の関係性を重回帰分析によって分析した．実験では 4 つのリポジトリのデータを使用し，コードレビュー時間と関係している要素の分析を行った．その結果，コードレビュー時間にはコードレビューを受ける人の過去の平均レビュー時間が最も関係が深いことがわかった．しかし，コードレビュー時間を十分な精度で推定することはできなかった．

今後の展望として，今回明らかになった課題の解決とコードレビュー計画支援システムの開発がある．今回は，コードレビューの際のコメントはリポジトリごとに大きく異なるため要素として扱わなかったが，特徴語抽出を使用することで説明変数として利用できる可能性がある．コードレビュー時間の推定とその要因を表示するコードレビュー計画支援システムを開発していく．

参考文献

[1] IPA:IPA ISEC セキュア・プログラミング講座：C/C++言語編 第 2 章 脆弱性回避策とソフトウェア開発工程：ソースコードレビュー，IPA(オンライン)，入手先 <https://www.ipa.go.jp/security/awareness/vendor/programmingv2/contents/c103.html>(参照 2019-11-8)

[2] Yang, Cheng, Xun-hui Zhang, Ling-bin Zeng, et al.: RevRec: A Two-Layer Reviewer Recommendation Algorithm in Pull-Based Development Mode, Journal of Central South University, Vol.25, No.5, pp.1129-43 , (2018).

[3] Bacchelli, Alberto, and Christian Bird.: Expectations, Outcomes, and Challenges of Modern Code Review, Proc. Proceedings of the 2013 International Conference on Software Engineering(ICSE '13), USA: IEEE Press, pp.712-721, (2013).

[4] McCabe, T. J.: A Complexity Measure, IEEE Transactions on Software Engineering, Vol.2, No.4, pp.308-20, (1976).