

What Makes a Code Change Easier To Review?

An Empirical Investigation on Code Change Reviewability

Achyudh Ram, Anand Ashok Sawant, Marco Castelluccio and Alberto Bacchelli

Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering
Conference and Symposium on the Foundations of Software Engineering
- ESEC/FSE 2018, 201–12.

どんなものか

ソフトウェアの品質を高めるため、目視によるコード検証
(コードレビュー)が行われている

レビューしやすいコード変更に必要なテーマは何か
以下の3ステップで調査した

- ①さまざまな文献からテーマを定義
e.g.) Code Style, Test Inclusion
- ②テーマをもとに開発者10人へインタビュー
- ③レビューしやすさを評価するツールを開発・利用

RQ

RQ1

文献によると，何が良いコード変更をもたらすか

RQ2

レビュワーはレビューしやすいコード変更と何に関連付けているか

a 一般的なレビュワーによる不慣れなシステムでレビューしやすさに問題がありそうなコード変更に対する認識の調査

b 特定のレビュワーが取り組んでいる実システムについてレビューしたことのフィードバック

RQ1.文献について

white/gray literatureをサーベイし，良いコード変更の
テーマを決める

white

各論文検索サイトやトップカンファレンスの関連論文

gray

プロジェクトのガイドラインや開発者のブログ

文献から決定したテーマ

テーマ	説明
Change description	変更点についての説明
Change scope	変更範囲の広さ
Code churn	コードがどれくらい編集されたか
Code quality	コードの品質
Code style	コーディング規約
Commit history	コミット履歴
Composite changes	複数の変更が混ざっている
Nature of the change	変更の種類
Subsystem hotness	サブシステムの活発さ
Test inclusion	テストの有無

a テーマをもとにインタビュー

以下の10名にインタビューを行う

Domain	Company / Project	Experience
Industry	Microsoft	4 years
Industry	HP	18 years
Industry	SET GmbH	8 years
Industry	SET GmbH	14 years
Industry	Milvum B.V. NL	5 years
Industry	Google	10 years
Industry	Google	5 years
Industry	Google	10 years
OSS	Spring Framework	9 years
OSS	Spring Framework	12 years

a インタビュー方法

実際のコード変更をレビューしてもらうため
Mockito projectから3つのPRを選択

- ①問題なし
- ②変更に関して明確な説明がなく，テストも含まれていない
- ③複数の問題を含み，不明瞭なメッセージを含んだ複数のコミット

b 評価ツールの使用

Pull Request Feedback

TEST-USER/WEBHOOK-TEST



*How would you rate the **necessity** of this patch?*

How long did you take to review this patch?

Number of minutes



*How would you rate the **reviewability** of the submitted patch?*

What aspects of this patch, if any, contribute to its reviewability?

Enter here...

What aspects of this patch, if any, should be improved to enhance its reviewability?

Enter here...

Change Description

開発者一人を除いてレビューしやすさに関連すると主張

“the first thing I want to see in a pull request is why you want [it].”

反対意見が一人

“the code cannot lie and users do lie.”

Code Churn

変更が少ないほどレビューしやすい

“because it is a really short change, one can read it fast.”

ただ上記の内容を一般化することはできないとの主張も
変更数は変更の種類に依存するため

Commit history

コミットメッセージは明瞭である必要があり,
“Fix typo” や “Update” のような表現ではダメ

“First I check if the commit history is normal . . . not ‘fix this’, ‘fix that’ in there because people can just use rebase to clean that up.”

単独で成り立つ単一のコミットの方が良い

“the developer to clean up the commit structure before the pull request”

Test inclusion

レビューしやすさにあまり影響しない

テストの有無とカバレッジに影響していないかのみ確認しており、テストの中身はあまり見ていない

※accept/rejectには影響している

インタビューのまとめ

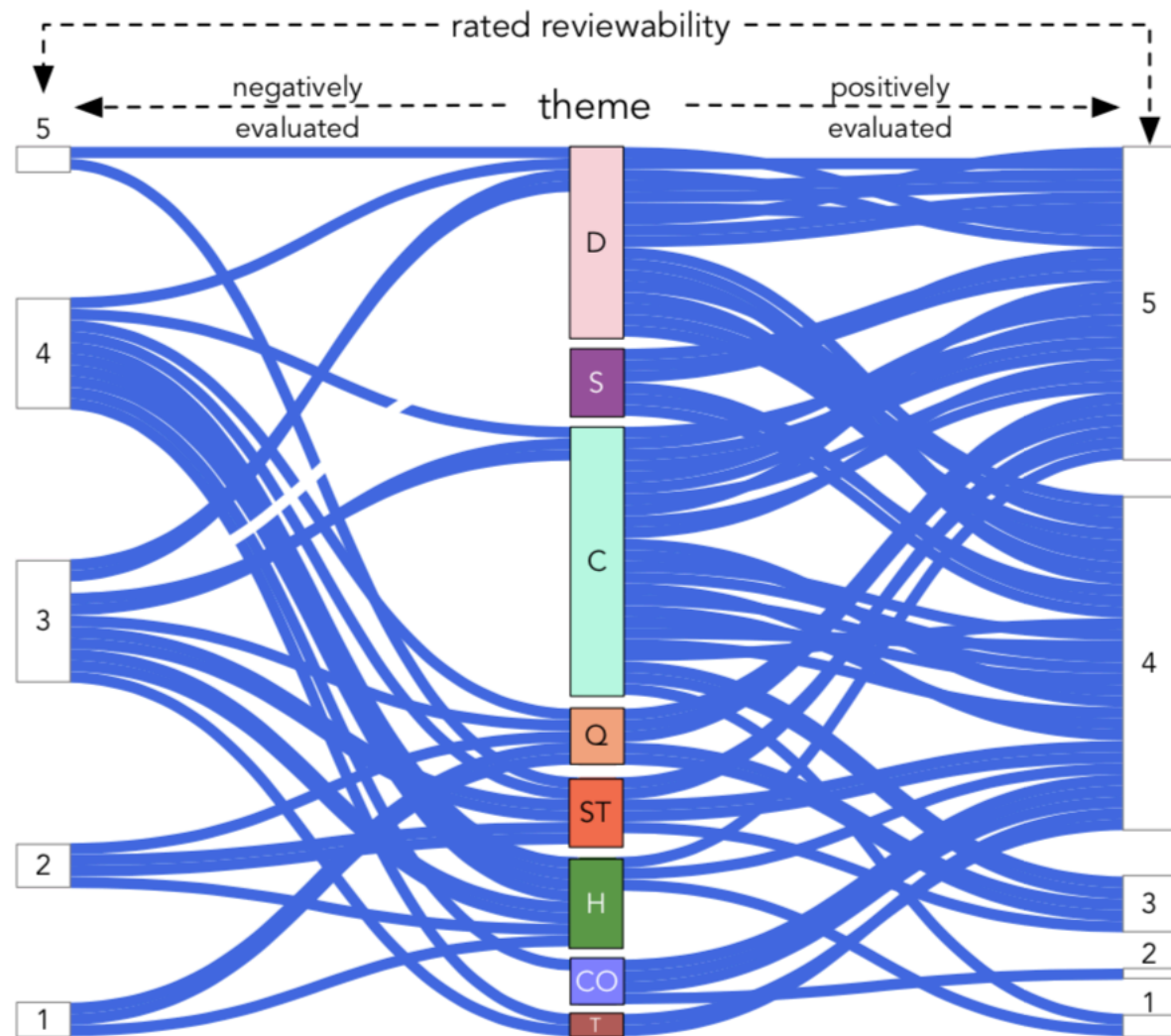
インタビューによると
code churn
change description
commit history
に重点を置いている

テストの有無はレビューしやすさに影響せず,
レビューしやすさと受諾率の相関は弱い

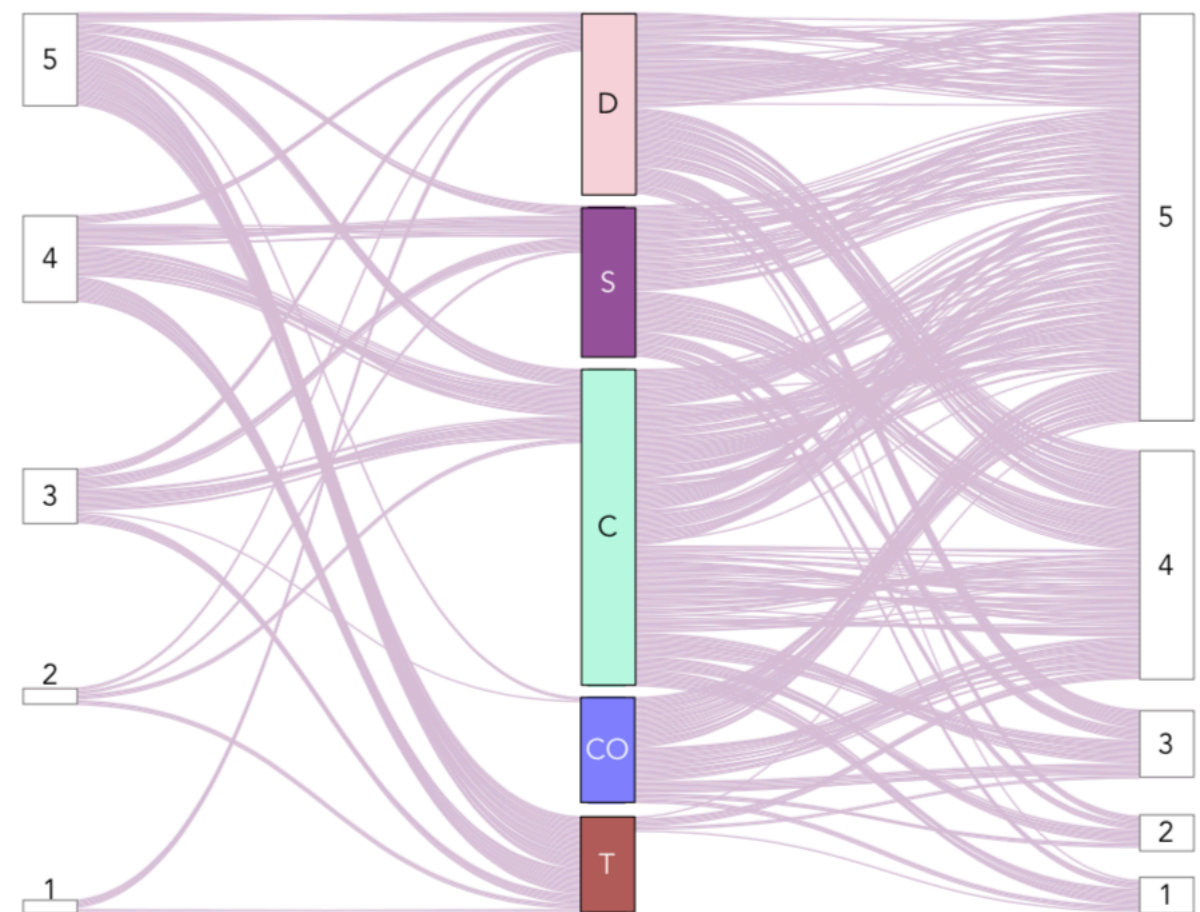
分析

本ツールで収集したフィードバック
一般的なレビューツールへのコメント
ソースコードの変更内容と付随する情報

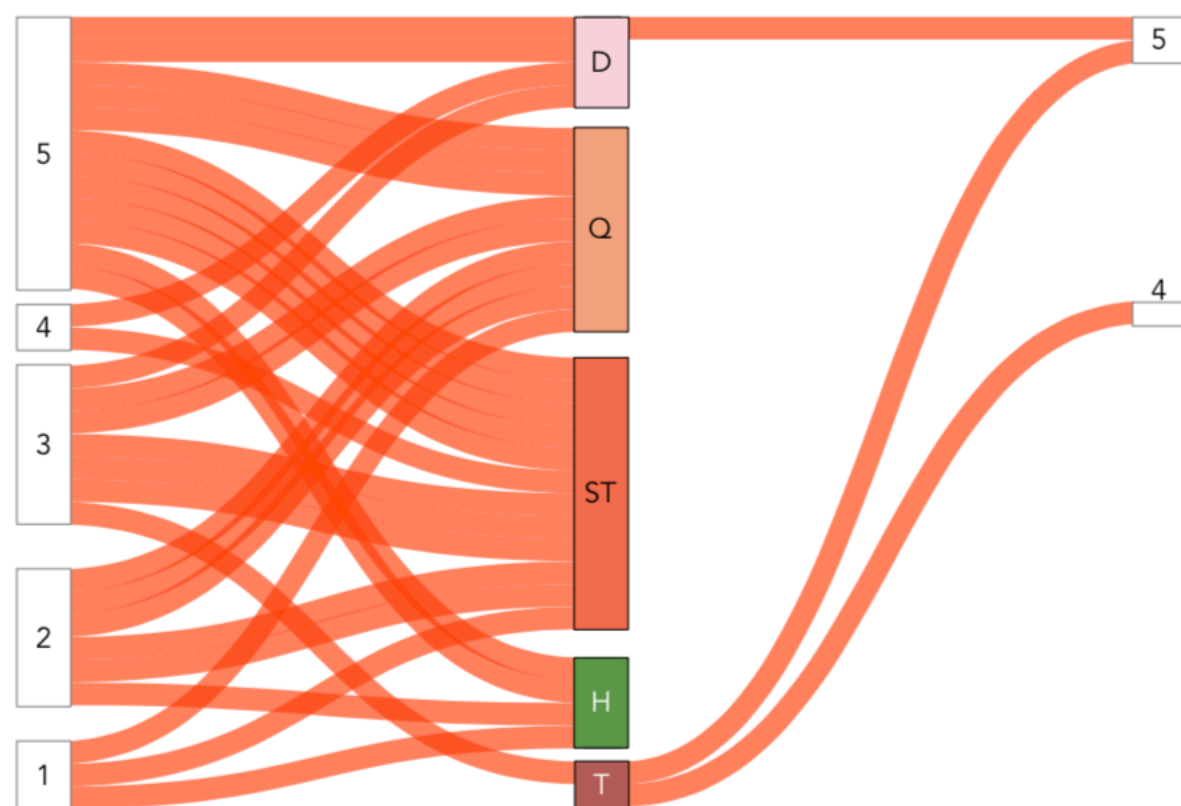
からpositive(+1),negative(-1),no comment(0)で各テーマを評価



A1 - Feedback



A3 - Content



A2 - Review comments

レビューコメントはnegativeなものが多いが、レビューしやすさとの関連性は顕著ではない

良いコード変更に必要なテーマ

①コード変更が説明されている

change description , **commit history**

②適切なサイズかつ整理されている

code churn , **composite changes**

③プロジェクトに合わせられている

code style

結論と所感

レビューしやすさを定義した経験的なものであり、レビューしやすさの自動評価ツールの作成などに使える

レビューしやすさと受諾率との関連性についての経験的証拠

👍 テーマ決めからレビューツール作成までしっかりと既存研究を参考にしていた点