

An Empirical Study of the Impact of Modern Code Review Practices on Software Quality

Empirical Software Engineering
October 2016, Volume 21, Issue 5, pp 2146-2189

60206052

岡坂亮汰

背景

- 現場での会議やレビューのチェックリストなどの厳格なレビュー基準が定められている正式なコードレビューはソフトウェアの品質を向上させるという先行研究がある
- ただし、そのような厳格なレビュー基準が定められていない近代的なコードレビュープロセスとソフトウェア品質との関係についての研究はほとんどない

目的

近代的なコードレビュープロセスとソフトウェア品質との関係を調査

- ・ 3つの観点から関係を調査
 - ソースコードカバレッジ
 - コードレビューの参加
 - レビューワの専門知識

リサーチクエスチョン

1. **コードレビューカバレッジ**とリリース後の欠陥には関係があるか？
2. **コードレビューの参加**とリリース後の欠陥には関係があるか？
3. **レビューワーズの専門知識**とリリース後の欠陥には関係があるか？

分析方法

対象システム

	Qt		VTK	ITK	Android	LibreOffice
Version	5.0	5.1	5.10	4.3	4.0.4	4.0
Tag name	v5.0.0	v5.1.0	v5.10.0	v4.3.0	4.0.4	4.0.0
Size (LOC)	5,560,317	5,187,788	1,921,850	1,123,614	18,247,796	4,789,039
Components w/ defects	254	187	15	24	-	-
Components total	1,339	1,337	170	218	-	-
Defective rate	19%	14%	9%	11%	-	-
Commits w/ reviews	10,003	6,795	554	344	1,727	1,679
Commits total	10,163	7,106	1,431	352	80,398	11,988
Review rate	98%	96%	39%	98%	2%	14%
# Authors	435	422	55	41	-	-
# Reviewers	358	348	45	37	-	-

表 1 : システム概要

Review rate が高い**Qt**, **VTK**, **ITK**を対象システムとする

メトリクス

	Metric	Description	Rationale
Product	Size	Number of lines of code.	Large components are more likely to be defect-prone (Koru <i>et al.</i> , 2009).
	Complexity	The McCabe cyclo-matic complexity.	More complex components are likely more defect-prone (Menzies <i>et al.</i> , 2002).
Process	Prior defects	Number of defects fixed prior to release.	Defects may linger in components that were recently defective (Graves <i>et al.</i> , 2000).
	Churn	Sum of added and removed lines of code.	Components that have undergone a lot of change are likely defect-prone (Nagappan and Ball, 2005, 2007).
	Change entropy	A measure of the distribution of changes among files.	Components where changes are spread among several files are likely defect-prone (Hassan, 2009).
Human Factors	Total authors	Number of unique authors.	Components with many unique authors likely lack strong ownership, which in turn may lead to more defects (Bird <i>et al.</i> , 2011; Graves <i>et al.</i> , 2000).
	Minor authors	Number of unique authors who have contributed less than 5% of the changes.	Developers who make few changes to a component may lack the expertise required to perform the change in a defect-free manner (Bird <i>et al.</i> , 2011). Hence, components with many minor contributors are likely more defect-prone.
	Major authors	Number of unique authors who have contributed at least 5% of the changes.	Similarly, components with a large number of major contributors, i.e., those with component-specific expertise, are less likely to be defect-prone (Bird <i>et al.</i> , 2011).
	Author ownership	The proportion of changes contributed by the author who made the most changes.	Components with a highly active component owner are less likely to be defect-prone (Bird <i>et al.</i> , 2011).

表2：ベースラインメトリクス
(一般的なソフトウェアメトリクス)

メトリクス

	Metric	Description	Rationale
Coverage (RQ1)	Proportion of re-viewed changes	The proportion of changes that have been reviewed in the past.	Since code review will likely catch defects, components where changes are most often reviewed are less likely to contain defects.
	Proportion of re-viewed churn	The proportion of churn that has been reviewed in the past.	Despite the defect-inducing nature of code churn, code review should have a preventative impact on defect-proneness. Hence, we expect that the larger the proportion of code churn that has been reviewed, the less defect prone a module will be.
Participation (RQ2)	Number of self-approved changes	The proportion of changes to a component that are only approved for integration by the original author.	By submitting a review request, the original author already believes that the code is ready for integration. Hence, changes that are only approved by the original author have essentially not been reviewed.
	Number of hastily-reviewed changes	The proportion of changes that are approved for integration at a rate that is faster than 200 lines per hour.	Prior work has shown that when developers review more than 200 lines of code per hour, they are more likely to let lower quality source code slip through (Kemerer and Paulk, 2009). Hence, components with many changes that are approved at a rate faster than 200 lines per hour are more likely to be defect-prone.
	Number of changes without discussion	The proportion of changes to a component that are not discussed.	Components with many changes that are approved for integration without critical discussion are likely to be defect-prone.
	Typical review window*	The length of time between the creation of a review request and its final approval for integration, normalized by the size of the change (churn).	Components with shorter review windows may not be spending enough time carefully analyzing the implications of a change, and hence may be more defect-prone.
	Typical discussion length*	The length of the review discussion (i.e., # non-automated comments), normalized by the size of the change (churn).	Components with many short review discussions may not be deriving value from the review process, and hence may be more defect-prone.
Expertise (RQ3)	Number of changes that do not involve a subject matter expert*	The proportion of changes to a component that are not authored by nor reviewed by a subject matter expert, i.e., a major author.	Components with many changes that do not incorporate a subject matter expert are more likely to be defect-prone.
	Typical voter expertise*	The percentage of prior changes to a component that each voter has either authored or voted on.	Components with a high degree of voter expertise are likely less defect-prone.

RQ1

RQ2

RQ3

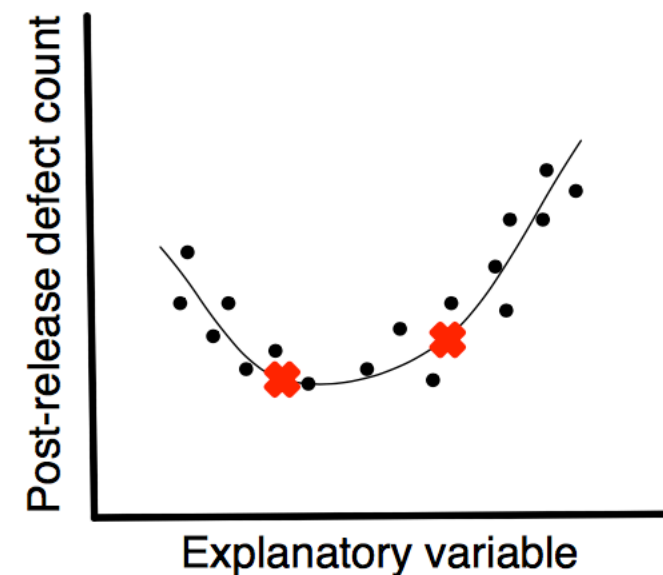
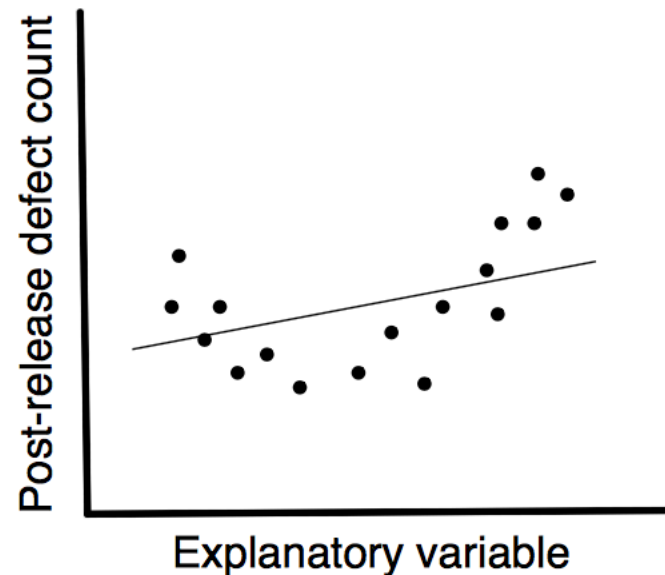
表3：コードレビューメトリクス

モデル構築

- 重回帰分析

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

(y : 従属変数、各 x_i : 説明変数)

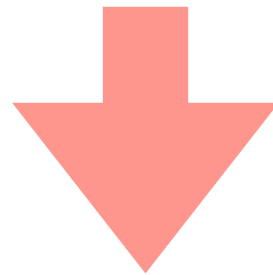


より適切に関係を近似できるようにknot（結び目）を割り振る

モデル構築

- knot (結び目)

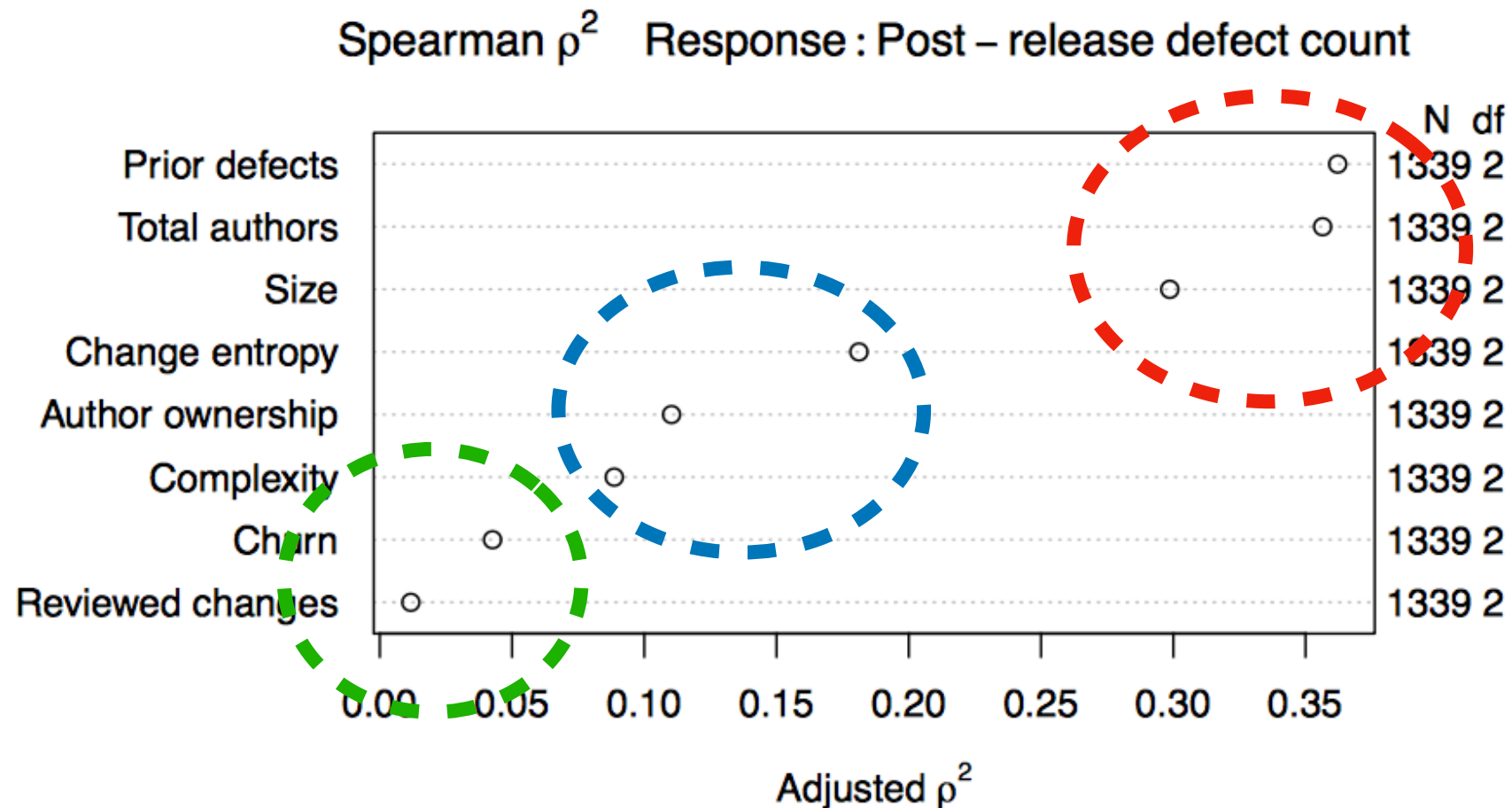
従属変数と説明変数の間に非線形, 非単調な関係がある場合, knot を割り当てることでより適切に近似



従属変数と説明変数間のSpearmanの順位相関係数を計算し, 結果の値が大きければ非線形, 非単調な関係がある

モデル構築

<例>

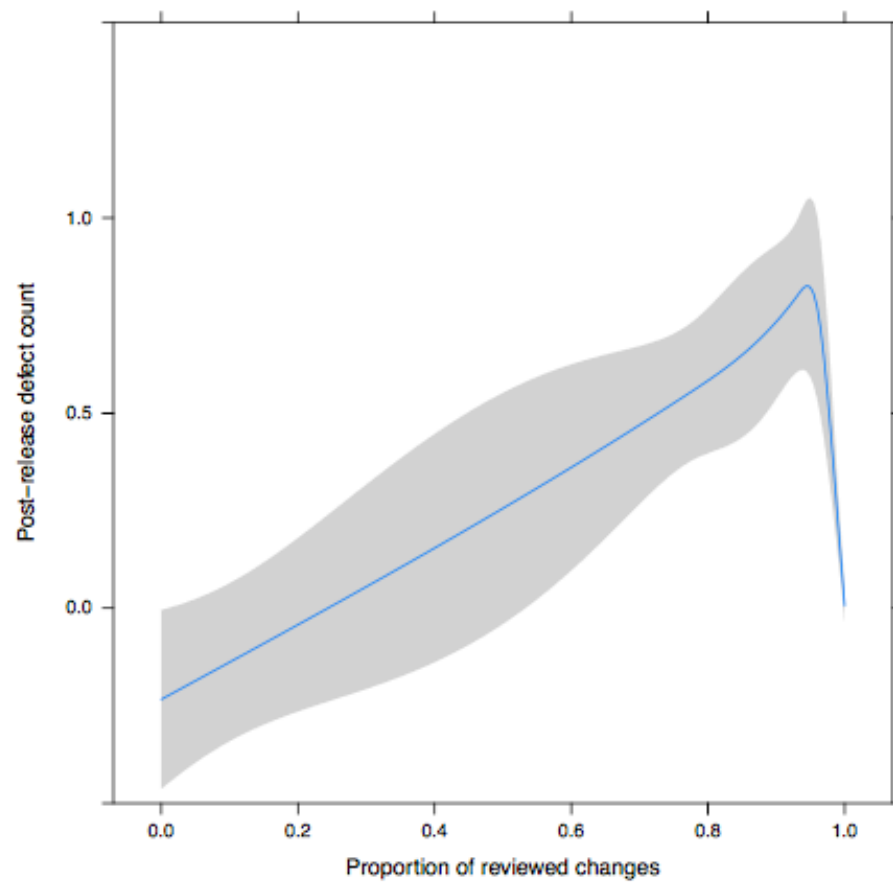


- ρ^2 の値によって説明変数を 3 つのグループに分ける
- 値の大きいグループ（赤）には 5 つの結び目,
2 つ目のグループ（青）には 3 つの結び目,
3 つ目のグループ（緑）には結び目は割り当てない.

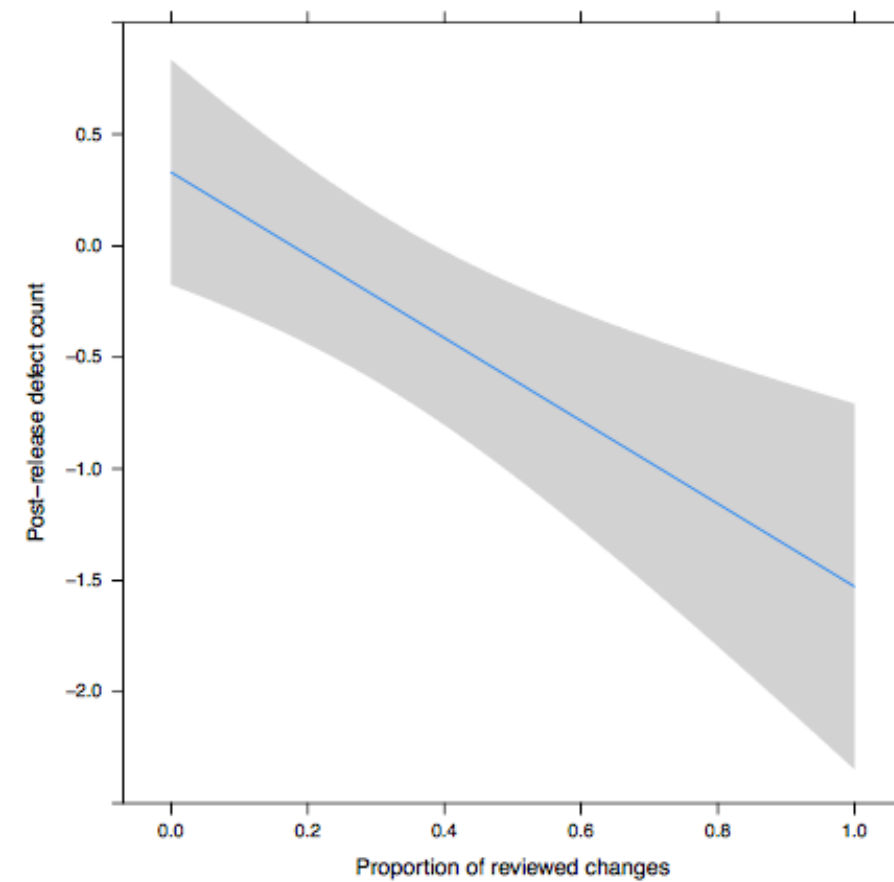
結果

結果

RQ1：コードレビューカバレッジとリリース後の欠陥には関係があるか？



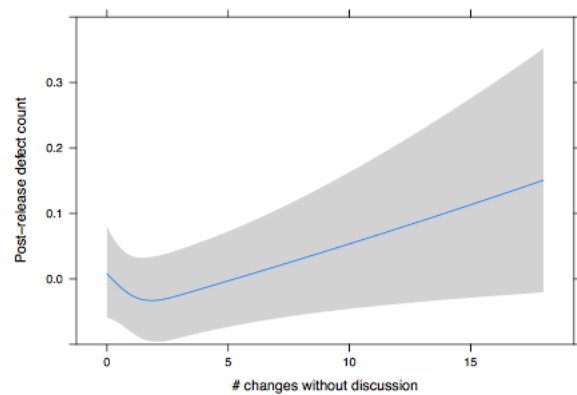
(a) Qt 5.1



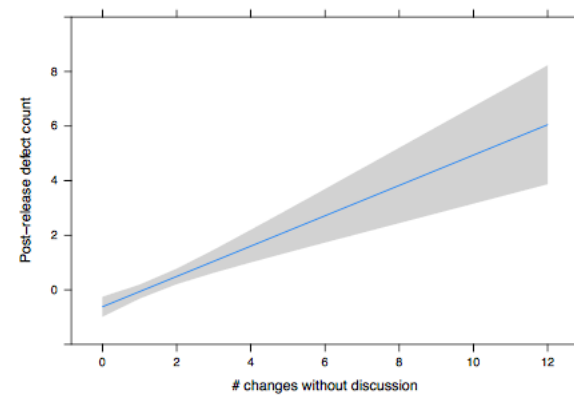
(b) VTK

結果

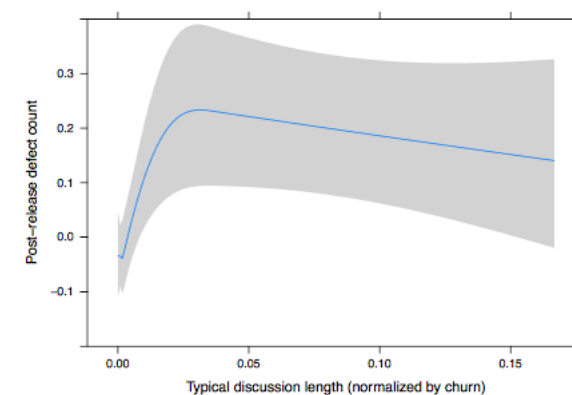
RQ2：コードレビューの参加とリリース後の欠陥には関係があるか？



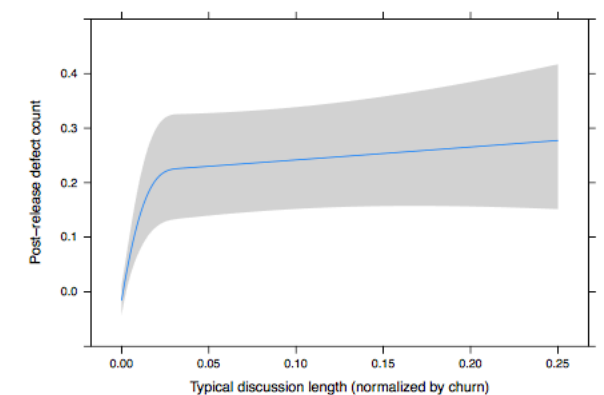
(a) No discussion (Qt 5.0)



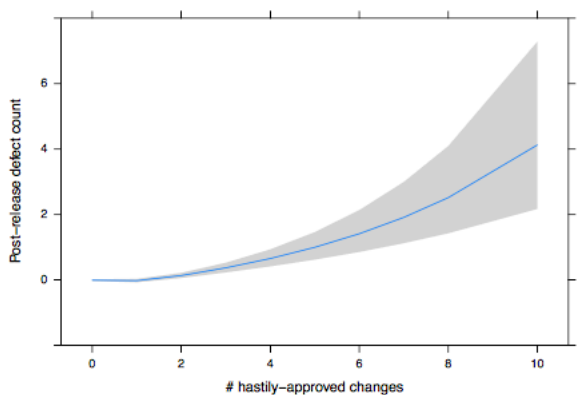
(b) No discussion (ITK 4.3)



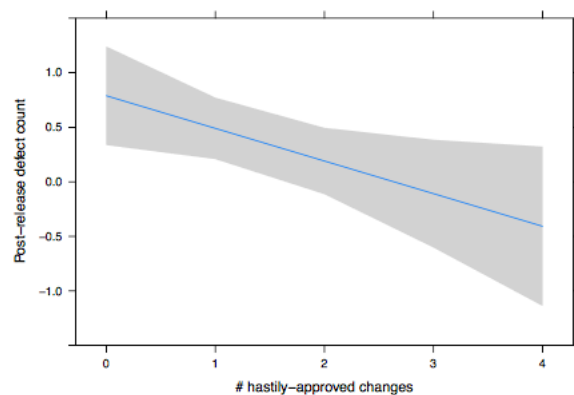
(c) Typical discussion length (Qt 5.0)



(d) Typical discussion length (Qt 5.1)



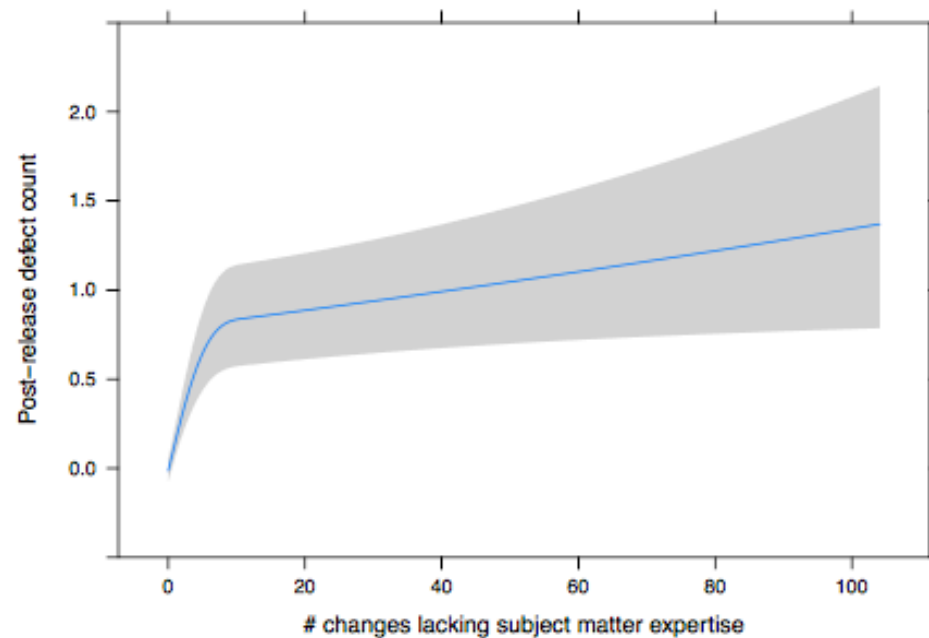
(e) Hastily-reviewed (Qt 5.1)



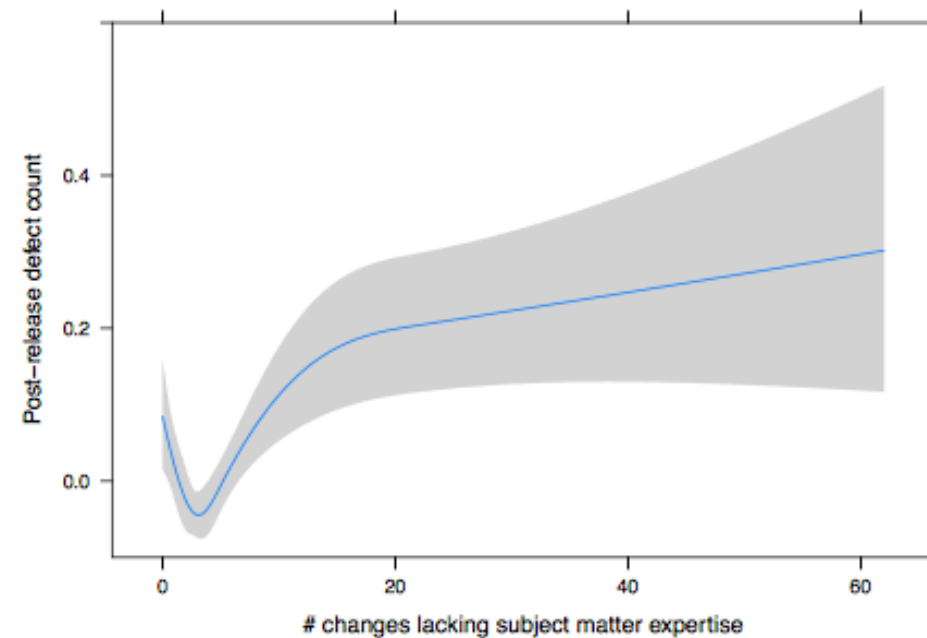
(f) Hastily-reviewed (ITK 4.3)

結果

RQ3：レビュワーの専門知識とリリース後の欠陥には関係があるか？



(a) Lacking subject matter expertise (Qt 5.0)



(b) Lacking subject matter expertise (Qt 5.1)

まとめ

- コードレビューカバレッジだけではリリース後の不具合の数を説明することは難しい
- レビューへの参加はソフトウェア品質に影響を与えるため、レビューを行う際には議論の数を考慮すべき
- レビューワーカーの専門知識が欠如している場合、リリース後の不具合の数は増加する

所感

- 議論の長さが不具合数にそれほど影響を与えないのは意外だった
- 今後、不具合の原因を調査する際に必要な観点がわかり、参考になった