

What the Fork: A Study of Inefficient and Efficient Forking Practices in Social Coding

ソーシャルコーディングにおける 非効率的なフォークプラクティスの研究

著者

Shurui Zhou, Bogdan Vasilescu, Christian Kästner
Carnegie Mellon University, USA

論文情報：

Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, 12 pages.

発表者：堀口 日向

1. 背景 & 目的

- OSS開発におけるフォークの目的
 - 昔：プロジェクトから独立しての開発．元プロジェクトと競合・取って代わる
 - 今：フォーク元にPRを提出し，変更をマージする
- フォークベースのOSS開発の非効率性
 - フォーク元に貢献されない変更
フォーク元はその変更が欲しい（かもしれない）
 - マージされない変更
拒否されると開発者は貢献したくなる
 - 重複した変更
他人の作業が見えにくいので，無駄な作業をしてしまう
 - ハードフォーク（昔タイプのフォーク）が現れる
コミュニティが断片化

研究目的

どうすれば
減らせる？

2. 方法

1. フォークベースのOSS開発の非効率性の要因になりそうな特性を調査

- OSS開発者へのインタビュー

特性が異なる4つのOSSプロジェクトの開発者計15人

→ Bitcoin, Marlin, Smoothieware, scikit-learn

- 文献調査

分散協調作業に関する理論を探す

2. 仮説を立てる

インタビュー，文献調査で得たデータをカテゴリ化して仮説を立てる

3. 重回帰分析で仮説を検証

3. 仮説を立てる (1/3)

1. モジュール性

- インタビュー
 - モジュール化することで、小規模でより一貫性のある変更が提出される
 - 高度なマージ作業が必要なプロジェクトに貢献をするつもりはない
- 文献の主張
 - 地理的に分散したソフトウェアプロジェクトには不可欠
 - モジュール性がないと、並行して作業することはできない

仮説1：モジュール性が高いと、フォーク元に貢献する割合が大きくなる

仮説2：モジュール性が高いと、PRがマージされる割合が大きくなる

3. 仮説を立てる (2/3)

2. 作業の事前調整

- インタビュー
 - どんなPRでも歓迎．事前にPRを調整してはいない
 - issueで前もって何をするか決定．それ以外のPRは無視
- 文献の主張
 - Linuxではプロジェクトを始めた人が課題を定義
 - どのissueを解決するかは個人の自由選択（調整されることはあまりない）

仮説3：issueで作業の事前調整を行うと，フォーク元に貢献する割合が大きくなる

仮説4：issueで作業の事前調整を行うと，PRがマージされる割合が大きくなる

仮説5：issueで事前にコミュニケーションをとると，重複PRの割合が小さくなる

3. 仮説を立てる (3/3)

3. コミュニティの断片化

- インタビュー
 - PR提出したけど拒否されたので、フォークを維持（ハードフォーク化）
 - モジュール性が高すぎると、コア部分の変更が拒否される
- 文献の主張
 - プロジェクトリーダーのビジョンとコミュニティのニーズの対立によってハードフォークが現れる

仮説6：PRがマージされる割合が小さいと、ハードフォークが現れやすくなる

仮説7：モジュール性が極端に高いと、ハードフォークが現れやすくなる

仮説8：issueで作業の事前調整を行うと、ハードフォークが現れやすくなる

4. 重回帰分析の準備 | 目的変数の決定

- フォーク元に貢献するフォークの割合
- PRがマージされる割合
- 重複PRの割合
 - 重複PRの見つけ方
PRについてのコメントに、ほかのPRへのリンク、
または重複に関連するキーワードがある場合、重複PRと判断
- ハードフォークが現れるかどうか
 - ハードフォークの定義（a, bどちらかを満たすフォーク）
 - a. オーナー以外から2つ以上PRが提出された
 - b. フォーク元にマージされていない変更が
100コミット以上あり、プロジェクト名が変更された

目的変数ごとに
計4つの
重回帰モデル
を作成

4. 重回帰分析の準備 | データ収集 (1/2)

GHTorrentから20フォーク以上あるプロジェクト（計137,424）を以下の条件でフィルタリング

- 課題提出用のプロジェクトを削除
「homework」, 「assignments」, 「course」などのキーワードを含む
- ドキュメント集を削除
「awesome-」を含む, または, ソースコードがない
- 開発履歴の少ないプロジェクトを削除
 - コミットが10個より少ない
 - フォーク後にコミットが1個以上あるフォークが10個より少ない
 - 外部からのPR提出が1つもない
- 閉じられたissueが1つもないプロジェクトを削除
issueで作業の事前調整がされているかどうか確認しようがないため

4. 重回帰分析の準備 | データ収集 (2/2)

フィルタ後，フォーク数で階層分けした結果

Table 1: How we stratified our sample.

Group	Num. forks	Num. projects on GITHUB	Num. projects in sample
A	[3,000 , +]	231	200
B	[1,000 , 3,000)	847	300
C	[20 , 1,000)	116,532	1300

ランダム
サンプリング

サンプルから，以前インタビューしたプロジェクトと，重複プロジェクトを削除
→ 重回帰分析に使うのは，計1,131プロジェクト

5. 重回帰分析の結果 | 表の見方 (1/2)

Table 2: Contributing forks model ($R^2 = 17\%$).

説明変数	決定係数		
	係数（平均値）	標準偏差	Contributing forks
	Coeffs (Errors)		LR Chisq
CentralizedMngmtIndex	6.03	(0.06)***	868.03***
ModularityIndex	1.23	(0.03)***	35.72***
AdditiveContributionIndex	0.97	(0.11)	0.09

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

N = 1131

有意水準

プロジェクト数 結果は省略

5. 重回帰分析の結果 | 表の見方 (2/2)

Table 2: Contributing forks model ($R^2 = 17\%$).

	Ratio contributing forks	
	Coeffs (Errors)	LR Chisq
CentralizedMngmtIndex		868.03***
ModularityIndex		35.72***
AdditiveContributionIndex	0.97 (0.11)	0.09

説明変数を
モデルから
除外したときの
残差の平方和
(厳密にはAnova
Type- II の平方和)

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

N = 1131

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | フォーク元に貢献するフォークの割合 (1/3)

Table 2: Contributing forks model ($R^2 = 17\%$).

Contributing forks		
	(errors)	LR Chisq
クローズドPRの中で、issueへのリンクがあるPRの割合 高いほど、作業の事前調整がされている		
CentralizedMngmtIndex	6.03 (0.06)***	868.03***
ModularityIndex	1.23	72***
AdditiveContributionIndex	0.97	09

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

N = 1131

※切片，制御変数の結果は省略

強い正の相関
(仮説3を支持)

5. 重回帰分析の結果 | フォーク元に貢献するフォークの割合 (2/3)

Table 2: Contributing forks model ($R^2 = 17\%$).

Contributing forks		
	(forks)	LR Chisq
プロジェクト内のすべてのファイルペアのうちあるコミットで一緒に変更されたファイルペアの割合を最近の50コミットで測定した際の中央値の逆数	0.06)***	868.03***
高いほど、モジュール性が高い		
ModularityIndex	1.23 (0.03)***	35.72***
AdditiveContributionIndex	0.97 (0.09)	0.09

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

正の相関
(仮説1を支持)

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | フォーク元に貢献するフォークの割合 (3/3)

Table 2: Contributing forks model ($R^2 = 17\%$).

	Ratio contributing forks	
	fs (Errors)	LR Chisq
Commitごとに変更されたすべてのファイルから新たに追加されたファイルの割合をすべてのコミットで測定した際の中央値	0.03 (0.06)***	868.03***
高いほど、極端なモジュール性が高い	0.23 (0.03)***	35.72***
AdditiveContributionIndex	0.97 (0.11)	0.09

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

相関なし 1

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | マージされたPRの割合 (1/2)

Table 3: External PR merge ratio model ($R^2 = 27\%$).

作業の事前調整	Ratio merged PRs	
	Coeffs (Errors)	LR Chisq
CentralizedMngmtIndex	1.67 (0.03)***	226.64***
ModularityIndex	1.50 (0.00)***	200.46***
AdditiveContributionIndex	1.47	191.28***
*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$		N = 1125

正の相関
(仮説4を支持)

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | マージされたPRの割合 (2/2)

Table 3: External PR merge ratio model ($R^2 = 27\%$).

	Ratio merged PRs	
	Coeffs (Errors)	LR Chisq
モジュール性	1.67 (0.03)***	226.64***
ModularityIndex	1.50 (0.02)***	308.46***
AdditiveContributionIndex	1.47 (0.07)***	30.28***

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

正の相関
(仮説2を支持)

結果は省略

5. 重回帰分析の結果 | 重複PRの割合

Table 4: Duplicate PR ratio model ($R^2 = 4\%$).

PR提出前に、issueを使って
コミュニケーションをとったPRの割合
高いほど、作業の事前調整がされている

決定係数が低い

Ratio duplicate PRs
Coeffs (Errors) LR Chisq

(Intercept)	切片	0.01 (0.09)***	
NumForks		1.16 (0.01)***	245.03***
Size	制御変数	0.97 (0.01)***	19.03***
ProjectAge		1.00 (0.00)***	29.45***
RatioPrecommunicatedPRs		0.84 (0.06)**	7.81**

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

残差平方和が低い、相関があるといえない
(仮説5の棄却)

5. 重回帰分析の結果 | ハードフォークが現れるかどうか (1/3)

Table 5: Hard forks model ($R^2 = 10\%$).

作業の事前調整	Has hard forks (T/F)	
	Coeffs (Errors)	LR Chisq
CentralizedMngmtIndex	4.92 (0.58)**	7.39**
ModularityIndex	0.66 (	1.57
AdditiveContributionIndex	4.32 (	2.43
PRMergeRatio	0.14 (0.42)***	22.24***

正の相関
(仮説8の支持)

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

N = 1131

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | ハードフォークが現れるかどうか (2/3)

Table 5: Hard forks model ($R^2 = 10\%$).

	Has hard forks (T/F)	
	Coeffs (Errors)	LR Chisq
CentralityIndex	4.92 (0.58)**	7.39**
ModularityIndex	0.66 (0.32)	1.57
AdditiveContributionIndex	4.32 (0.93)	2.43
PRMergeRatio	0.14 (	2.24***

モジュール性

有意差なし
(仮説7の棄却)

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

※切片，制御変数の結果は省略

5. 重回帰分析の結果 | ハードフォークが現れるかどうか (3 / 3)

Table 5: Hard forks model ($R^2 = 10\%$).

	Has hard forks (T/F)	
	Coeffs (Errors)	LR Chisq
CentralizedMngmtIndex	4.92 (0.58)**	7.39**
MergeRatio	0.66 (0.32)	1.57
PRMergeRatio	4.32 (0.93)	2.43
PRMergeRatio	0.14 (0.42)***	22.24***

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

強い負の相関
(仮説6の支持)

結果は省略

6. 結果まとめ

フォークベースのOSS開発の**非効率性を減らすには**...

- **モジュール性を高くする！**

- フォーク元に貢献するフォークの割合が高くなる (仮説1)
- PRがマージされる割合が高くなる (仮説2)
- ハードフォークの現れる可能性が (間接的に) 低くなる (仮説6)

- **issueで作業の事前調整を行う！**

- フォーク元に貢献するアクティブフォークの割合が高くなる (仮説3)
- PRがマージされる割合が高くなる (仮説4)
- ただし, ハードフォークが現れる可能性は高くなる (仮説8)
開発者にとって, issueでの事前調整は煩わしい？