

OSSコミュニティへの参加支援を 目的とした新人開発者のGitHubと StackOverflowでの活動分析

オープンソースソフトウェア工学研究室

学籍番号 60216332

堀口 日向

1. 背景 | OSSについて

OSS (Open Source Software)



オペレーティングシステム



OpenJDK



GCC

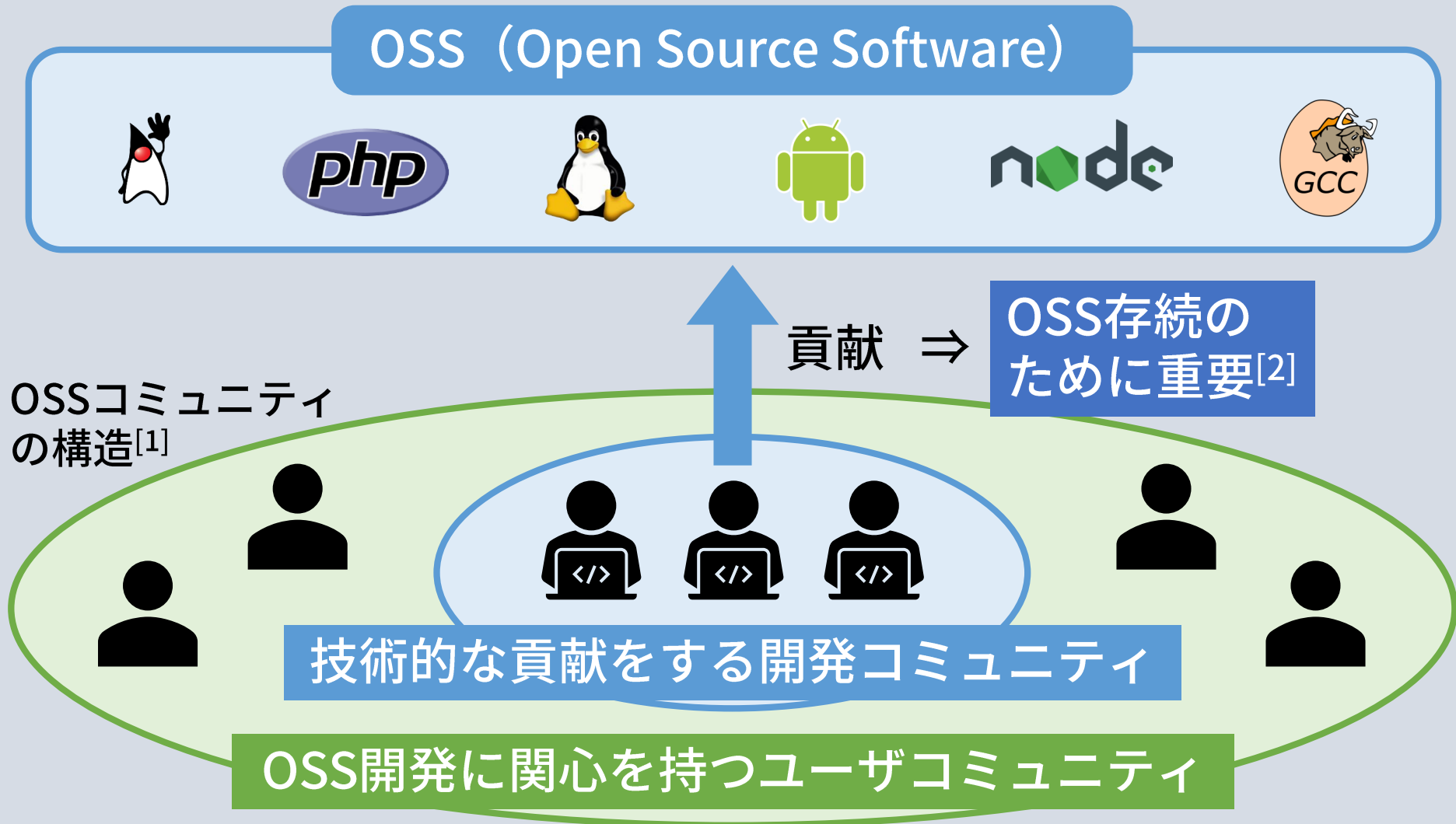
ソフトウェア開発環境



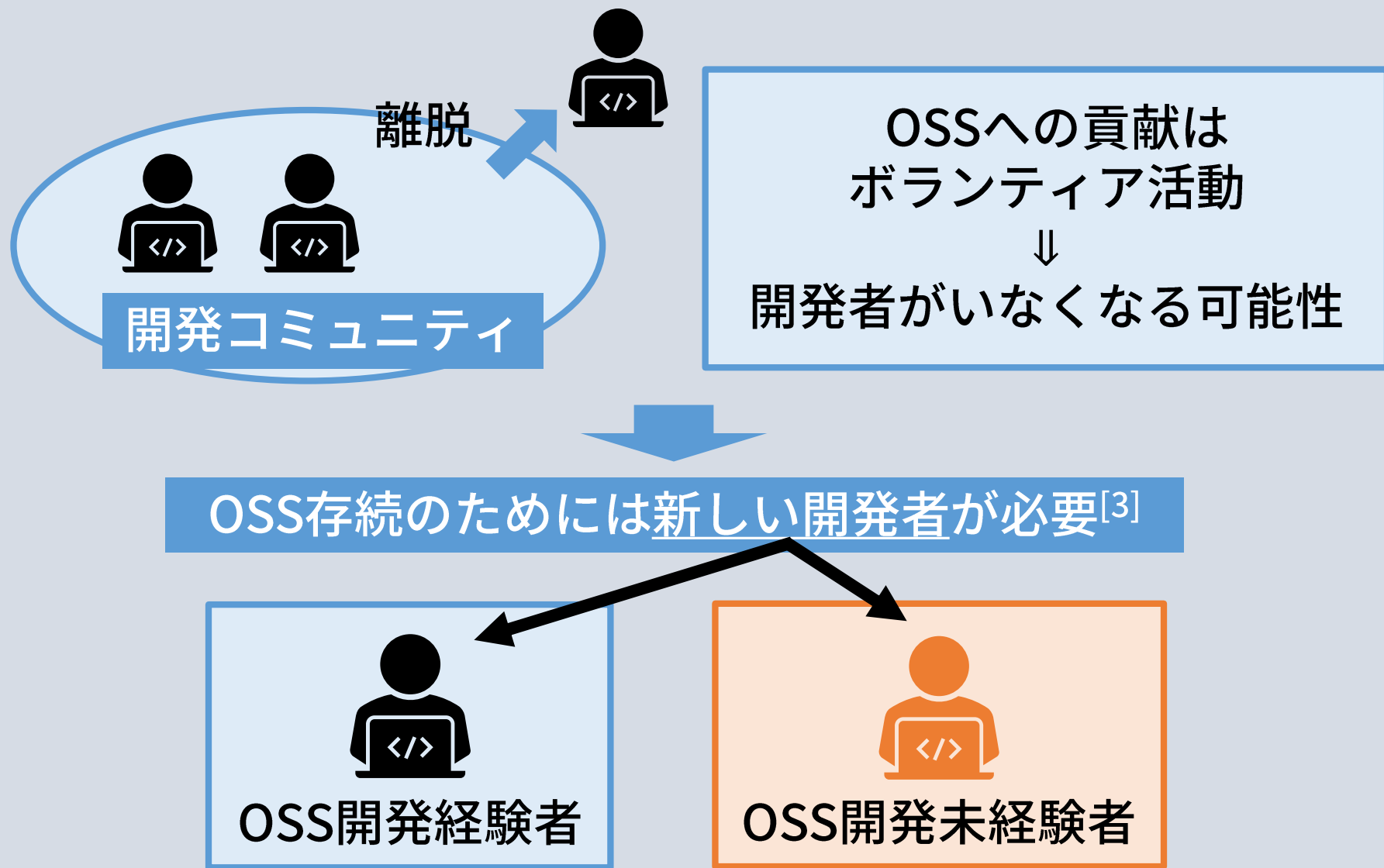
Webアプリケーション実行環境

ソフトウェア開発にOSSは必要不可欠

1. 背景 | OSSコミュニティ (1/2)



1. 背景 | OSSコミュニティ (2/2)



1. 背景 | OSS開発未経験者が抱える問題



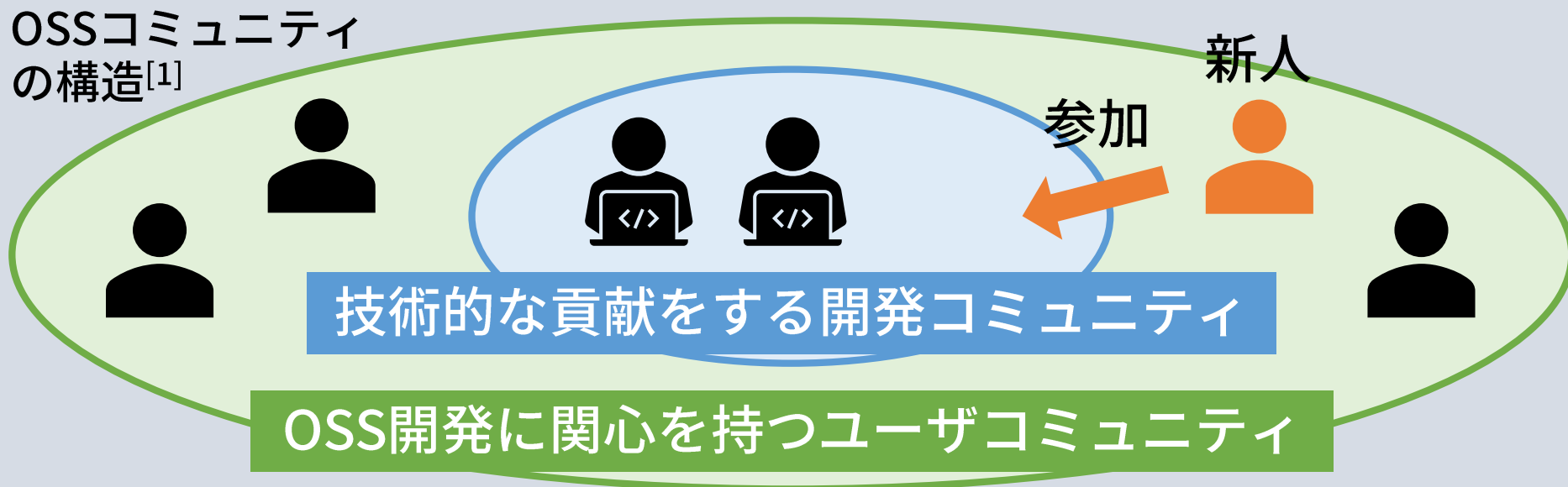
OSS開発未経験者 (以降，新人と呼称)

先行研究^{[4][5]}で指摘された新人が抱える問題

OSSへの貢献の際障壁に直面
貢献が開発コミュニティに受け入れられないことがあり
その影響で貢献をやめてしまうことも

本研究では，新人の支援に焦点を絞る

1. 背景 | 本研究の着眼点



新人が開発コミュニティに受け入れられるためには
ユーザコミュニティでの活動が役に立つ？

2. 目的

解決すべき問題

新人の貢献が開発コミュニティに受け入れられない

目的

新人にとって、開発コミュニティでの貢献が
受け入れられるためにユーザコミュニティが
どう役に立つのかを示す

3. 開発コミュニティとユーザコミュニティ

開発コミュニティ

ユーザコミュニティ

Webサービス



Bitbucket

等

デファクトスタンダード



slack

等

プログラマ専用のQ&Aサイト

機能

- ソースコードの公開
- Issue機能（課題管理）
- Pull Request機能（ソースコードの変更）

- Q&A
- チャット

4. GitHubの機能 (1/2)

- Issue機能

バグ報告や新機能の
提案(Issue)を投稿



コメント欄で議論

Bug with namespaced ActiveRecord models on 4.1.x (PostgreSQL) #15811

① Open

thelucid opened this issue on 19 Jun 2014 · 25 comments



thelucid commented on 19 Jun 2014

Contributor + 😊 ...

I've come up against a bug with ActiveRecord when using namespaces models and polymorphic associations, it's a common use-case so should probably be fixed before a 4.2 release.

Error:

```
NoMethodError: undefined method `relation_delegate_class' for Comment:Module
```



rafaelfranca commented on 20 Jun 2014

Member + 😊 ...

Which version are you using? Have you tried the latest 4.1.2.rc2? Could you create a reproduction script with this template

https://github.com/rails/rails/blob/master/guides/bug_report_templates/active_record_master.rb



rafaelfranca added the **activerecord** label on 20 Jun 2014



taish commented on 23 Jun 2014

+ 😊 ...

I think it is problem and should be fixed

I create an reproduction script [here](#)

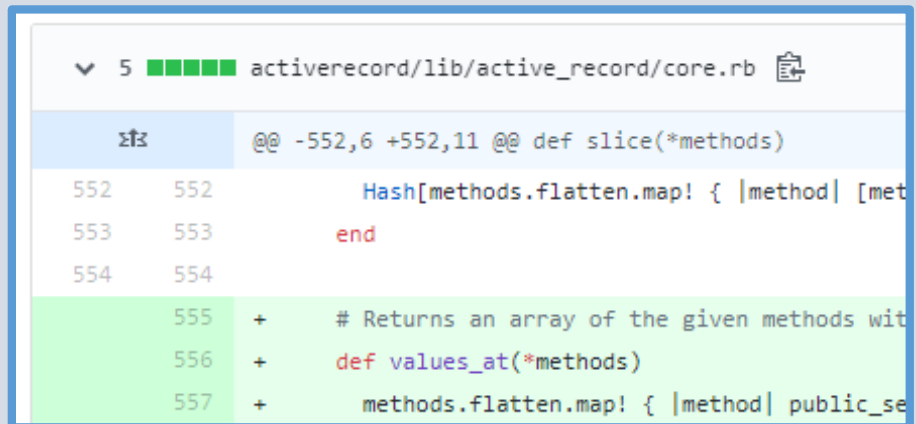
This test should be passed but actually failed

I'll try to solve

4. GitHubの機能 (2/2)

• Pull Request機能

開発者がバグ修正や機能拡張のため、ソースコードの変更を投稿(Pull Request)



The screenshot shows a diff view for the file `activerecord/lib/active_record/core.rb`. It highlights changes between two versions of the code. Line 555 is added, showing a comment and a new method definition `def values_at(*methods)`. Line 556 is added, showing the start of the method body. Line 557 is added, showing the end of the method body. The diff shows that the new method `values_at` is being added to the class.

プロジェクトの
中心メンバーがレビュー
追加の修正，議論を行う



The screenshot shows a conversation in a Pull Request. The title is "This conversation was marked as resolved". The first message is from **rafaelfranca** (Member) on 28 Jul, asking "Does not `flat_map` work here?". The second message is from **guillaumebriday** (Author) on 29 Jul, replying "it does not. Because: `a.flat_map(&b)` works exactly like `a.map(&b).flatten!(1)`. So if you send an array as parameter, it will try to `public_send` on an array. I added test for array as params if needed".

中心メンバー

投稿者

最終的に変更を統合するか判断

5. StackOverflowの機能



プログラミングに関する
質問・回答を行う
知識共有コミュニティ

開発者が質問



別の開発者が回答

Rails/Ruby - Format string into Date with locale

Asked today Active today Viewed 20 times

Is there a way in rails with localize or in plain ruby with Date.strptime to convert a string into a date with a specific months locales?

For example, I have the following need it to convert it to a Date object

ruby-on-rails ruby date

share edit

add a comment

New contributor

質問に関連する技術名や
プロダクト名でタグを付ける

1 Answer

active oldest votes

If you have the date month in english, you can use Date.strptime with this format:

```
date_str = '5 apr 19'  
date = Date.strptime(date_str, '%d %b %y')
```

repts english I would just translate the month, only a few months changes:

```
, 'abr' => 'apr', 'ago' => 'aug', 'dic' => 'dec'}  
gsub(/ene|abr|ago|dic/, months) # only replace the months that should  
e_str_aux, '%d %b %y')
```

share edit

answered 3 hours ago

arielljuod
10.5k • 2 • 15 • 27

6. 分析方法 | リサーチクエスチョン

ユーザコミュニティと開発コミュニティで
新人の扱いに違いはあるのか

RQ1

StackOverflowとGitHubで新人の
活動の受け入れられやすさに違いはあるか

ユーザコミュニティで活動を行うことで開発コミュニティでの
新人の貢献が受け入れられやすくなるのか

RQ2

事前にStackOverflowで活動を行った新人と行わなかった
新人では、貢献の受け入れられやすさに違いはあるか

6. 分析方法 | RQ1 (1/3)

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

活動の受け入れられやすさの指標 = 活動に対するコミュニティからの反応

StackOverflow

質問に対して

- 回答の有無 (名義尺度)
- 回答数 (比率尺度)
- 最初の回答が寄せられるまでの時間 (比率尺度)

GitHub

Issueに対して

- コメントの有無 (名義尺度)
- コメント数 (比率尺度)
- 最初のコメントが寄せられるまでの時間 (比率尺度)

Pull Requestに対して

- 統合されたかどうか (名義尺度)
- 統合されるまでの時間 (比率尺度)

6. 分析方法 | RQ1 (2/3)

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

活動の受け入れられやすさについて各コミュニティで

新人



新人以外

差があるかを仮説検定

⇒ 差がある = 新人と新人以外の扱いが異なる
新人に対するコミュニティからの反応が良い or 悪い

6. 分析方法 | RQ1 (3/3)

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

検定手法

- 名義尺度
カイ二乗検定 (有意水準：5%) ， 効果量はCohen's $w[x]$
- 比率尺度
マンホイットニーのU検定 (有意水準：5%) ， 効果量はCliff's $d[x]$

6. 分析方法 | RQ2 (1/2)

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

貢献の受け入れられやすさの指標 = 貢献に対するGitHubからの反応

Issueに対して

- コメントの有無 (名義尺度)
- コメント数 (比率尺度)
- 最初のコメントが寄せられるまでの時間 (比率尺度)

Pull Requestに対して

- 統合されたかどうか (名義尺度)
- 統合されるまでの時間 (比率尺度)

6. 分析方法 | RQ2 (2/2)

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

貢献の受け入れられやすさについて

事前にStackOverflowで
活動を行った新人



事前にStackOverflowで
活動を行わなかった新人

差があるかを仮説検定

⇒ 差がある = 貢献の受け入れられやすさが異なる
事前に活動を行ったことで貢献が受け入れられやすくなった
or 受け入れられにくくなった

検定手法や効果量はRQ1と同様

6. 分析方法 | 分析対象のOSSプロジェクトの決定

① StackOverflowの質問についているタグを質問数が多い順にソート

🏷 javascript

🏷 python

例: 🏷 c

🏷 ruby-on-rails

🏷 node.js

⋮

② GitHubのリポジトリと直接関係のあるタグを目視で抽出

🏷 javascript

🏷 python

🏷 c

🏷 ruby-on-rails

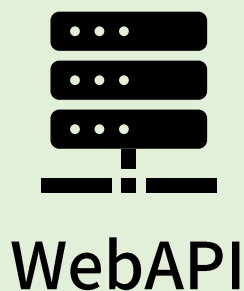
🏷 node.js

⋮

分析
対象

検定に十分な数のデータを確保するため
6個のプロジェクトを選択

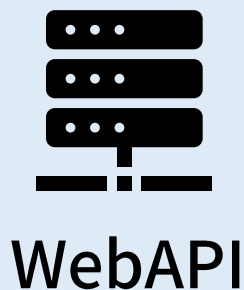
6. 分析方法 | 対象プロジェクトのデータを収集



質問, 回答
ユーザ名, ユーザID等



データベース

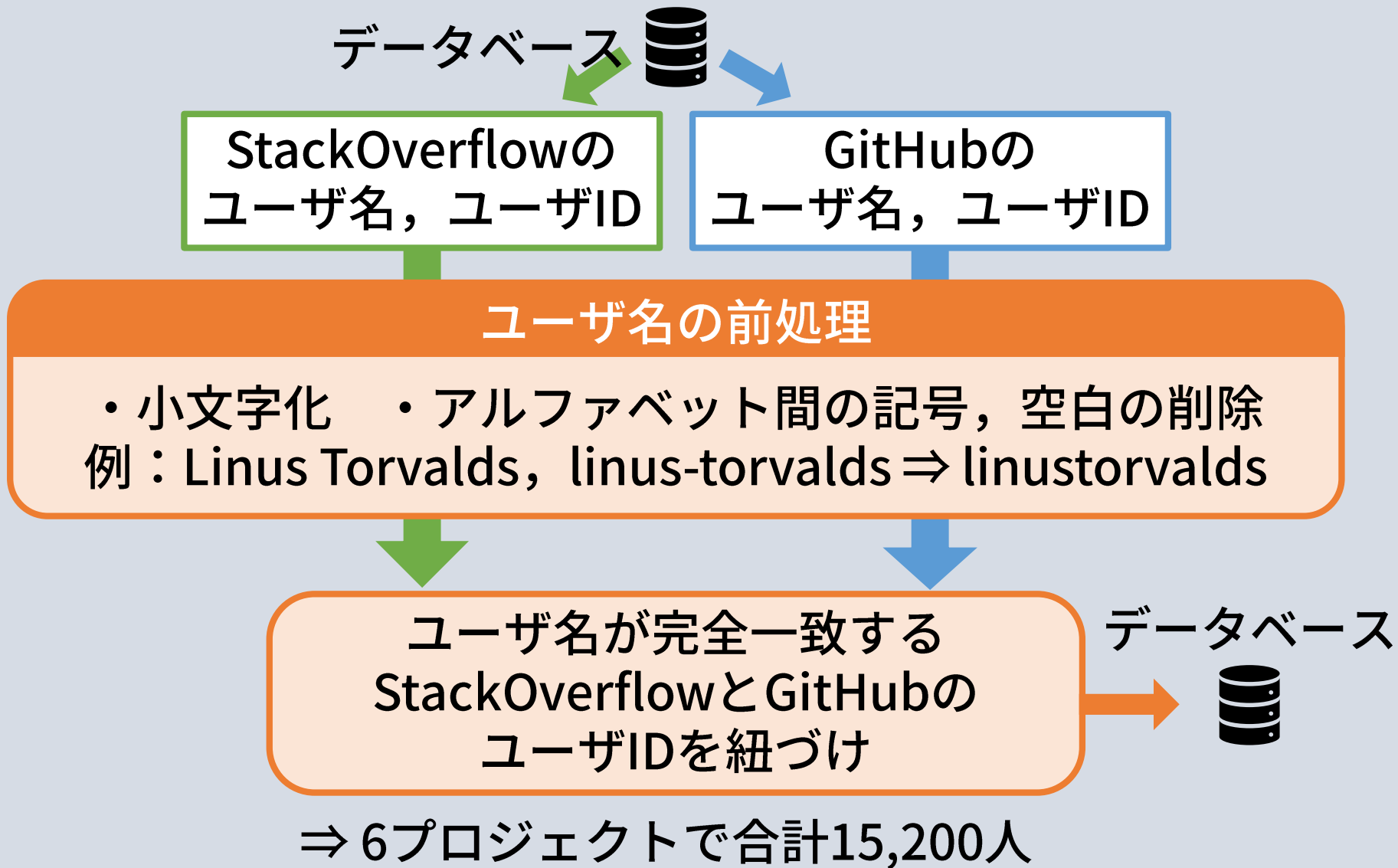


Issue, Pull Request,
ユーザ名, ユーザID等



データベース

6. 分析方法 | StackOverflowとGitHubで同じ開発者の紐づけ



6. 分析方法 | 各開発者の活動履歴の取得



データベース

StackOverflowと
GitHubで
紐づけられた
ユーザID

開発者ごとに
StackOverflowと
GitHubでの活動を
時系列データにまとめる

活動履歴の例：

```
"activities": [  
  {  
    "activityType": "question",  
    "datetime": "2015-01-14T20:59:04.000Z",  
    "activity_db": "so/node_js/questions",  
    "activity_id": 27952330,  
    "developer_layer": null  
  },  
  {  
    "activityType": "post issue",  
    "datetime": "2015-03-06T14:22:37Z",
```

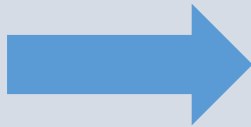
活動

日時

6. 分析方法 | 開発者が新人かどうかの判定

新人の定義

分析対象以外のプロジェクトで過去に Pull Request を投稿したことがない



ユーザが投稿した
一番古い Pull Request

YES



新人
1,624人

分析対象
プロジェクトの
Pull Request ?

NO



新人以外
13,576人

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

StackOverflowでの新人と新人以外の活動の受け入れられやすさの違いの検定
新人と新人以外の質問に対して、

指標	p値	効果量
回答の有無	0.000（有意差あり）	0.03（無視できる程小さい）

		回答があった	回答がなかった	合計
新人の質問	度数	4217	33642	37859
	期待度数	4391.70	33467.30	37859.00
	調整残差	-5.96	5.96	
新人以外の質問	度数	1418	9300	10718
	期待度数	1243.30	9474.70	10718.00
	調整残差	5.96	-5.96	
合計		5635	42942	48577

新人の質問
に対して
回答がない
傾向

7. 分析結果 | RQ1

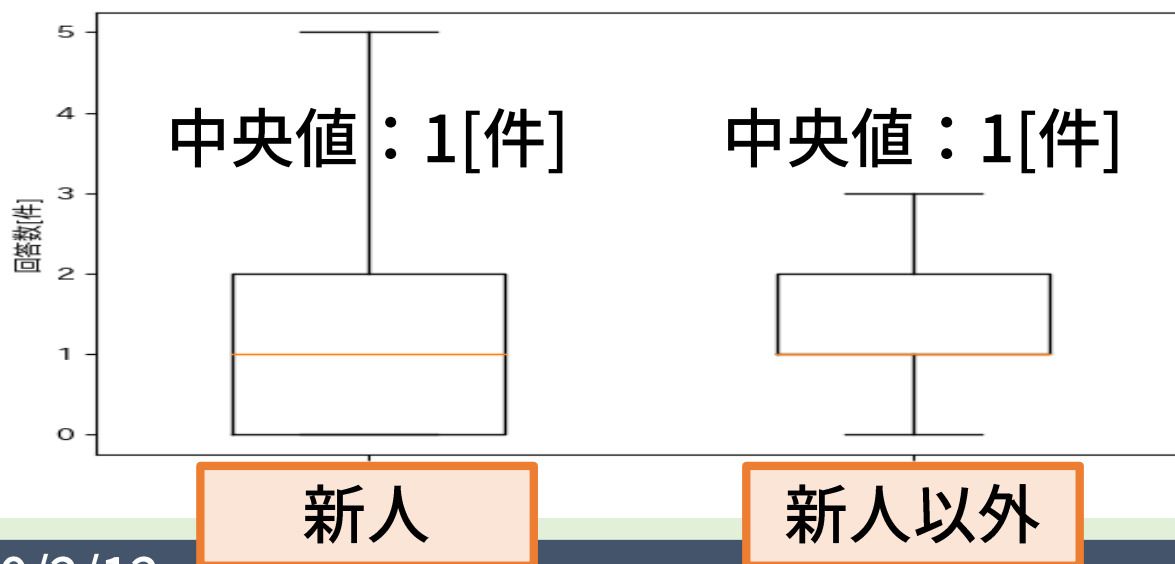
RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

StackOverflowでの新人と新人以外の活動の受け入れられやすさの違いの検定
新人と新人以外の質問に対して、

指標	p値	効果量
回答数	0.000（有意差あり）	0.06（無視できる程小さい）

※外れ値除去済み



回答数の
分布に
大きな違い
はない

7. 分析結果 | RQ1

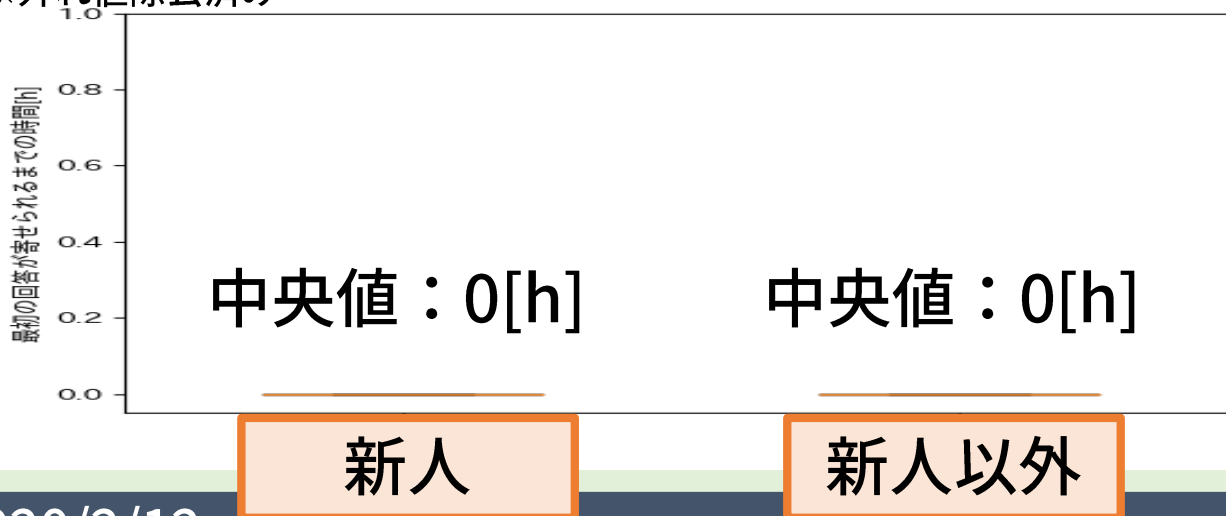
RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

StackOverflowでの新人と新人以外の活動の受け入れられやすさの違いの検定
新人と新人以外の質問に対して、

指標	p値	効果量
最初の回答が 寄せられるまでの時間	0.005（有意差あり）	0.01（無視できる程小さい）

※外れ値除去済み



時間の
分布に
違いはない

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

StackOverflowでの新人と新人以外の活動の受け入れられやすさの違いの検定
新人と新人以外の質問に対して、

指標	p値	効果量
回答の有無	0.000 (有意差あり)	0.03 (無視できる程小さい)
回答数	0.000 (有意差あり)	0.06 (無視できる程小さい)
最初の回答が 寄せられるまでの時間	0.005 (有意差あり)	0.01 (無視できる程小さい)

有意差はあるが効果量は無視できる程小さい



StackOverflowでは新人と新人以外の
活動の受け入れられやすさに大きな違いはない

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のIssueに対して

指標	p値	効果量
コメントの有無	0.000（有意差あり）	0.05（無視できる程小さい）

		コメントがあった	コメントがなかった	合計
新人の Issue 投稿	度数	8915	24483	33398
	期待度数	9325.10	24072.90	33398.00
	調整残差	-10.86	10.86	
新人以外の Issue 投稿	度数	2923	6077	9000
	期待度数	2512.90	6487.10	9000.00
	調整残差	10.86	-10.86	
合計		11838	30560	42398

新人のIssue
に対して
コメントが
ない傾向

7. 分析結果 | RQ1

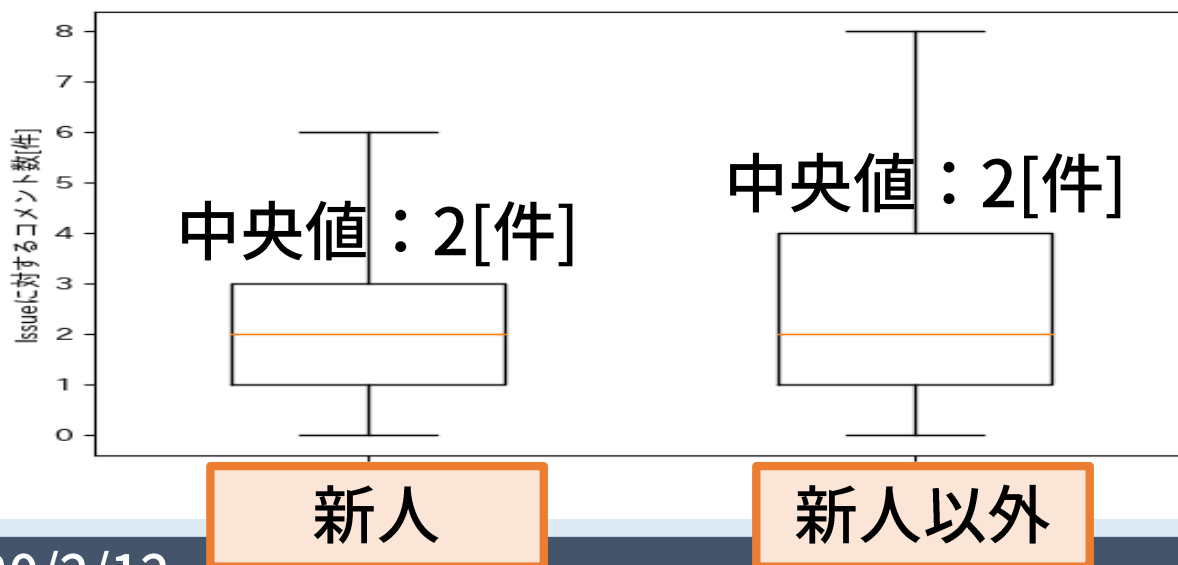
RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定
新人と新人以外のIssueに対して

指標	p値	効果量
コメント数	0.000（有意差あり）	0.07（無視できる程小さい）

※外れ値除去済み



コメント数
の分布に
大きな違い
はない

7. 分析結果 | RQ1

RQ1

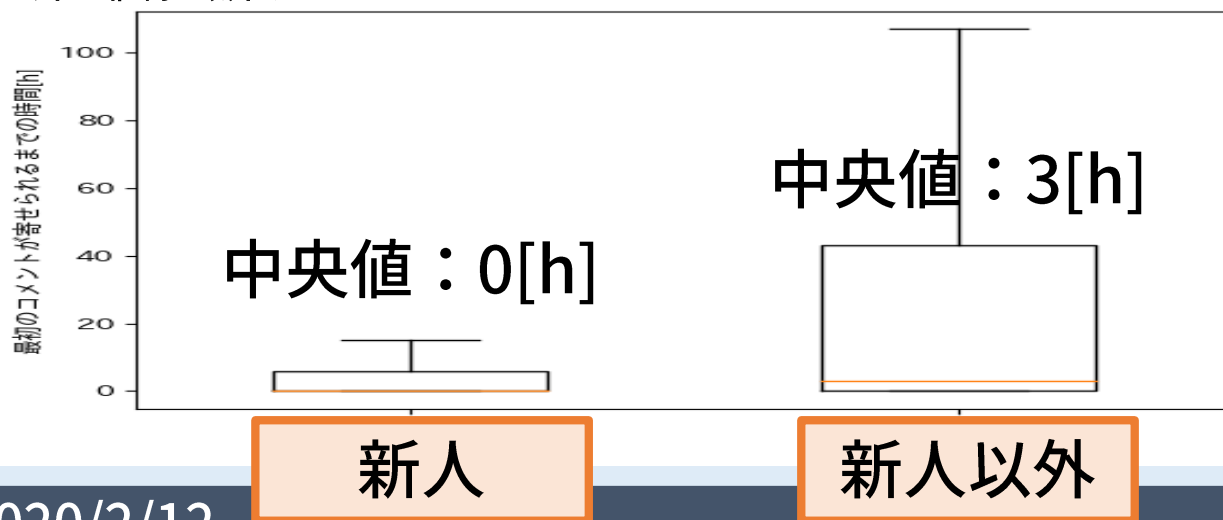
StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のIssueに対して

指標	p値	効果量
最初のコメントが 寄せられるまでの時間	0.000（有意差あり）	0.27（小程度）

※外れ値除去済み



新人のIssue
の方が最初の
コメントまで
の時間が短い

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のPull Requestに対して

指標	p値	効果量
統合されたかどうか	0.000（有意差あり）	0.05（無視できる程小さい）

		マージされた	マージされなかった	合計
新人の PR 投稿	度数	1048	1548	2596
	期待度数	857.00	1739.00	2596.00
	調整残差	8.36	-8.36	
新人以外の PR 投稿	度数	8312	17445	25757
	期待度数	8503.00	17254.00	25757.00
	調整残差	-8.36	8.36	
合計		9360	18993	28353

新人のPR
の方が統合
されやすい
傾向

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のPull Requestに対して

指標	p値	効果量
統合されるまでの時間	0.058 (有意差なし)	0.04 (無視できる程小さい)

※外れ値除去済み



新人のPRの
方が統合され
るまでの時間
が短い

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のIssueに対して

指標	p値	効果量
コメントの有無	0.000 (有意差あり)	0.05 (無視できる程小さい)
コメント数	0.000 (有意差あり)	0.07 (無視できる程小さい)
最初のコメントが 寄せられるまでの時間	0.000 (有意差あり)	0.27 (小程度)

Pull Requestに対して

指標	p値	効果量
統合されたかどうか	0.000 (有意差あり)	0.05 (無視できる程小さい)
統合されるまでの時間	0.058 (有意差なし)	0.04 (無視できる程小さい)

7. 分析結果 | RQ1

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

GitHubでの新人と新人以外の活動の受け入れられやすさの差の検定

新人と新人以外のIssueに対して

指標	p値	効果量
コメントの有無	0.000（有意差あり）	0.05（無視できる程小さい）
コメント数	0.000（有意差あり）	0.07（無視できる程小さい）
最初のコメントが 寄せられるまでの時間	0.000（有意差あり）	0.27（小程度）

Pull

指標

統合

統合

有意差はあるが効果量が無視できるほど小さい



GitHubでは新人と新人以外の
活動の受け入れられやすさに大きな違いはない

さい)

さい)

7. 分析結果 | RQ1まとめ

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

Answer

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはない

新人は開発コミュニティには受け入れられにくい^[X]
という先行研究での調査とは異なる結果

7. 分析結果 | RQ2

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Issueに対して、

指標	p値	効果量
コメントの有無	0.000（有意差あり）	0.08（無視できる程小さい）

		コメントがあった	コメントがなかった	合計
事前に SO で活動を行った新人の Issue	度数	5218	3796	9014
	期待度数	5002.92	4011.08	9014.00
	調整残差	9.20	-9.20	
事前に SO で活動を行わなかった新人の Issue	度数	1415	1522	2937
	期待度数	1630.08	1306.92	2937.00
	調整残差	-9.20	9.20	
合計		6633	5318	11951

事前に活動していた方がコメントがある傾向

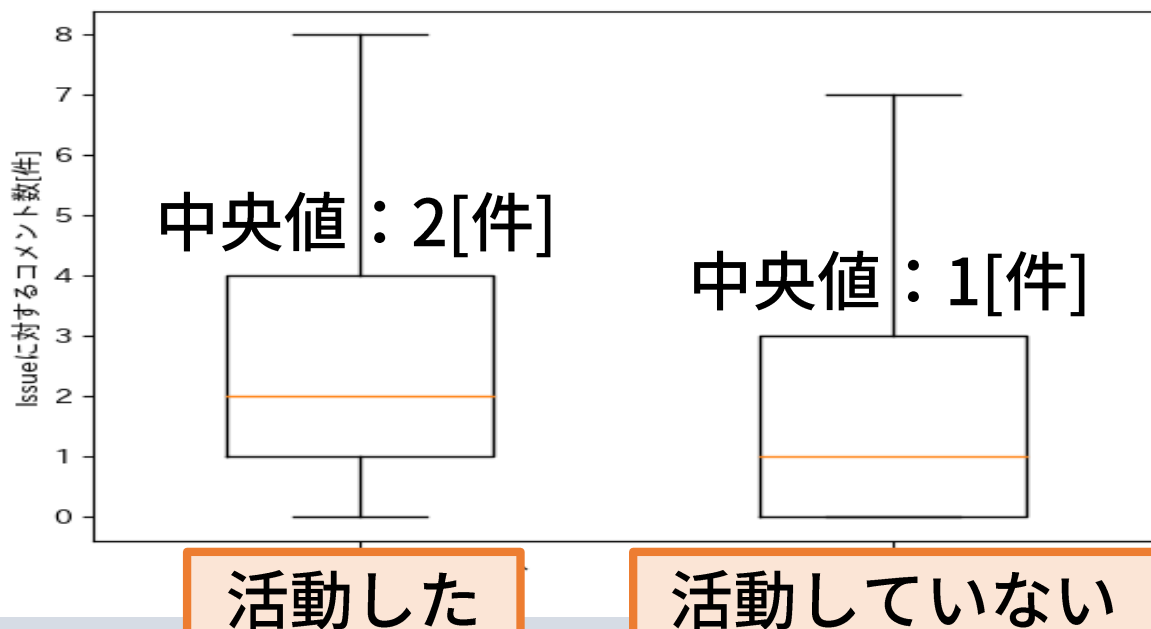
7. 分析結果 | RQ2

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Issueに対して、

指標	p値	効果量
コメント数	0.000（有意差あり）	0.13（無視できる程小さい）



事前に活動していた方がコメントを多くもらえる傾向

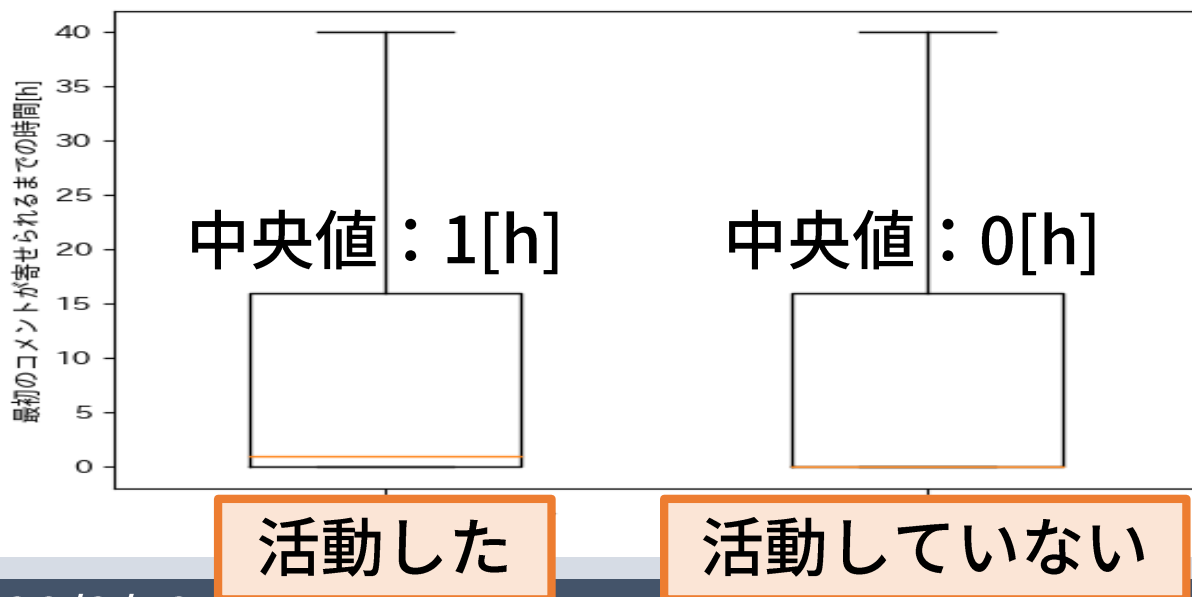
7. 分析結果 | RQ2

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Issueに対して、

指標	p値	効果量
最初のコメントが寄せられるまでの時間	0.000 (有意差あり)	0.06 (無視できる程小さい)



時間の
分布に大きな
違いはない

7. 分析結果 | RQ2

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Pull Requestに対して

指標	p値	効果量
統合されたかどうか	0.111（有意差なし）	0.02（無視できる程小さい）

		マージされた	マージされなかった	合計
事前に SO で活動を行った 新人の Pull Request	度数	552	498	1050
	期待度数	576.69	473.31	1050.00
	調整残差	-1.62	1.62	
事前に SO で活動を行わなかった 新人の Pull Request	度数	4619	3746	8365
	期待度数	4594.31	3770.69	8365.00
	調整残差	1.62	-1.62	
合計		5171	4244	9415

傾向なし

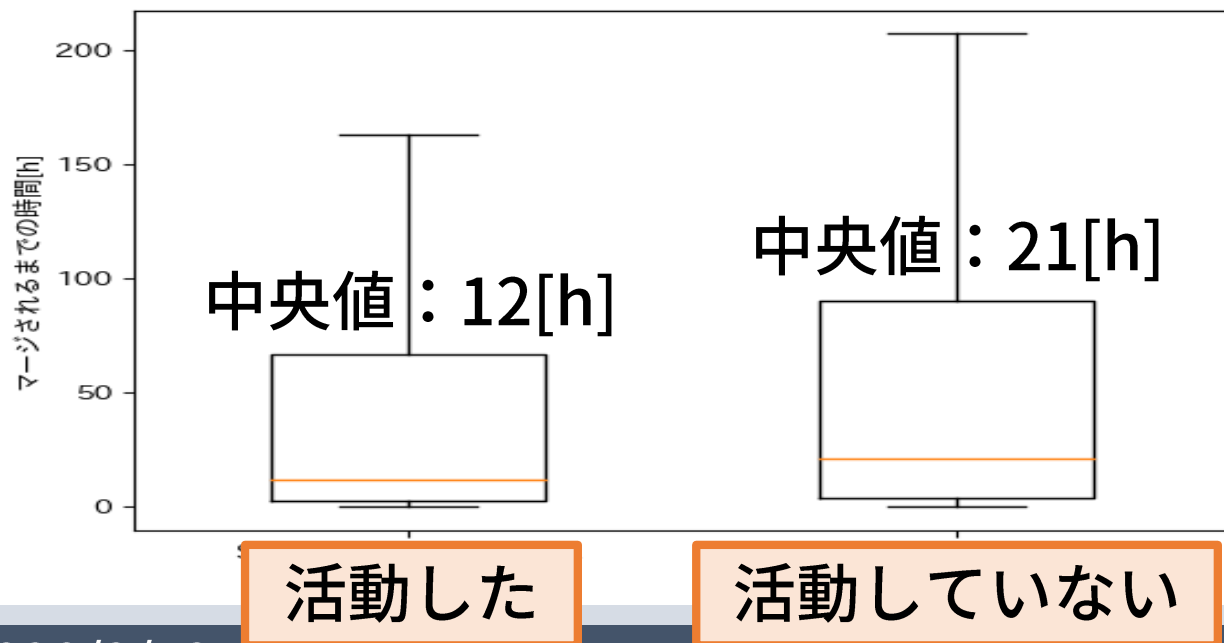
7. 分析結果 | RQ2

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Pull Requestに対して

指標	p値	効果量
統合されるまでの時間	0.003（有意差あり）	0.11（無視できる程小さい）



事前に活動していた方がマージまでの時間が短い

7. 分析結果 | RQ2まとめ

RQ2

事前にStackOverflowで活動を行った新人とそうでない新人では、貢献の受け入れられやすさに違いはあるか

Answer

事前にStackOverflowで活動を行った新人の方が
活動を行わなかった新人と比べて
貢献が受け入れられやすい

しかし、その効果は無視できる程小さい

8. 本研究のまとめ（目的の再掲）

解決すべき問題

新人の貢献が開発コミュニティに受け入れられない

目的

新人にとって、開発コミュニティでの貢献が
受け入れられるためにユーザコミュニティが
どう役に立つのかを示す

8. 本研究のまとめ

仮説検定の結果わかったこと

- ユーザコミュニティと比べ，開発コミュニティは新人を受け入れないということはない
- 事前にユーザコミュニティで活動していた新人の方が活動しなかった新人と比べて開発コミュニティでの貢献が受け入れられやすい．しかし，効果は無視できるほど小さい

6．分析方法 | 分析対象のOSSプロジェクトの決定（補足資料）

分析対象のプロジェクトの表

プロジェクト名	リポジトリ名	タグ名	対象期間
Ruby on Rails	rails/rails	ruby-on-rails	2008/08/02 ~ 2019/11/01
Node.js	nodejs/node	node.js	2008/11/11 ~ 2019/11/07
Angular.js	angular/angular.js	angularjs	2009/10/19 ~ 2019/11/11
Swift	apple/swift	swift	2009/01/19 ~ 2019/11/13
Django	django/django	django	2008/08/03 ~ 2019/11/14
Angular	angular/angular	angular	2013/04/15 ~ 2019/11/18

要旨

囲ってまとめたい情報

題名

題名が必要な何かまとめたい情報

必要な何かまとめたい情報

1. 背景 (2/3)

OSS (Open Source Software)



貢献

プロジェクトを持続させる
ためには新人が必要^[1]

バグ報告や
機能の要望^[2]

コミュニティに参加



ボランティアの
開発コミュニティ



新人OSS開発者

1. 背景 | OSSコミュニティ (1/2)

OSS (Open Source Software)



貢献 ⇒

OSS存続の
ために重要^[2]

技術的な貢献をする開発者層

OSS開発に関心を持つユーザ層

[1]

1. 背景 | OSSコミュニティ (2/2)

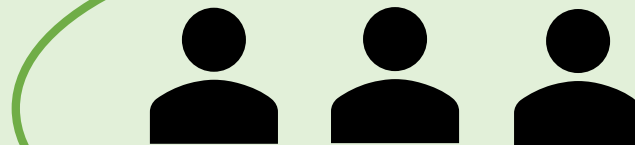


技術的な貢献をする開発者層

(以降, 開発コミュニティと呼称)

技術的な貢献

- バグの報告や修正
- 機能拡張



OSSに関心を持つユーザ層

(以降, ユーザコミュニティと呼称)

技術的な貢献

しない (将来的に
開発コミュニティに移る
可能性はある)



1. 背景 | 本研究の着眼点 (2/2)



開発コミュニティ

IssueやPull Request
を使った議論

新人にとって不慣れ



ユーザコミュニティ

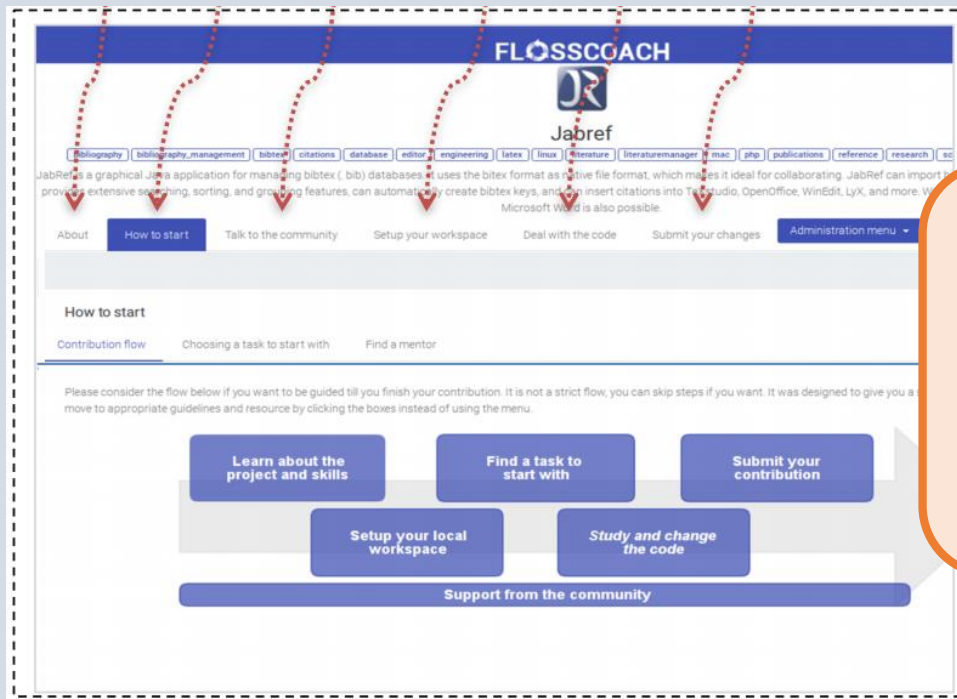
Q&A形式など

新人に馴染みやすい

新人にとっては開発コミュニティよりも
ユーザコミュニティの方が参加しやすい

2. 既存研究 (1/2)

- OSSプロジェクトに新人が参加しやすくするための研究
Steinbacherらは、OSSプロジェクトの新人を支援するためのポータルサイトを設計[1]



疑問

新しく新人を支援するサイトを設計
⇒ 既存のサイトについて

[1]Steinmacher, I., Conte, T., Treude, C., and Gerosa, M. 2016. Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In Proceedings of the 38th International Conference on Software Engineering (pp. 273–284).

2. 既存研究 (2/2)

- Pull Request (ソースコード変更の貢献) の受け入れられやすさに関連する研究
 - Soaresらは, Pull Requestの受け入れに影響のある要因をPull Requestに含まれるコミット数や追加されたファイル数などの観点から分析[2]

Table 1: Relevant association rules and their measures of interest.

#	Antecedent	Sup (%)	Conf (%)	Lift
1	language = Scala $\wedge$ commits_pull = 1 commit	4.3	77	1.43
2	language = C# $\wedge$ commits_pull = 1 commit	2.0	71	1.31
3	commits_pull = 1 commit $\wedge$ files_added = no file $\wedge$ developer_type = main_team $\wedge$ first_pull = false	10.5	76	1.40
4	project = Akka $\wedge$ commits_pull = 1 commit	1.5	92	1.70
5	language = Python $\wedge$ developer_type = main_team $\wedge$ first_pull = false	1.0	85	1.57

不足点

開発活動に焦点
コミュニティ活動については
考慮されていない

[2] Soares, D., Lima J'unior, M., Murta, L., and Plastino, A. 2015. Acceptance Factors of Pull Requests in Open-Source Projects. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (pp. 1541–1546).

2. 既存研究 (3/3)

- Pull Request (ソースコード変更の貢献) の受け入れられやすさに関連する研究
 - Padhyeらは、Pull Requestを分類し機能拡張よりもバグ修正の方がマージされる割合が高いことを示した[2]

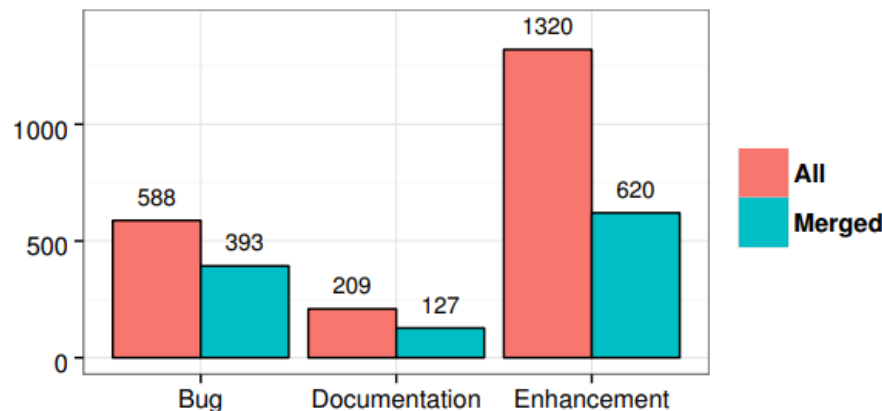


Figure 6: Number of pull requests merged, classified by types derived from labels.

疑問

機能拡張も
マージしてもらう
にはどうすればいい？

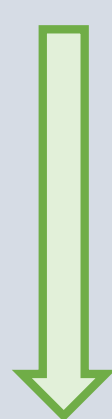
[2] Padhye, R., Mani, S., and Sinha, V. 2014. A Study of External Community Contribution to Open-Source Projects on GitHub. In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 332–335).

1. 背景 (ボツスライド)

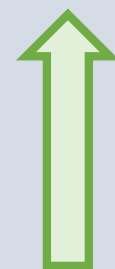
OSS (Open Source Software)



利用



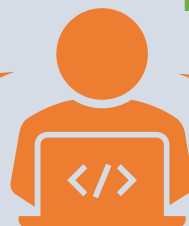
受け入れるかは
コアな開発者らが判断



- ・ バグ報告/機能の提案
- ・ ソースコード変更

⇒ OSSに貢献

仕事，研究，趣味
で使いたい



OSS利用者

バグや機能要望
がある

1. 背景 (ボツスライド)

OSS (Open Source Software)



APACHE



受け入れてもらえないと
利用時に困る

どうすれば
貢献を受け入れて
もらいやすくなる？



受け入れるかは
コアな開発者らが判断



・ バグ報告/機能の提案
・ ソースコード変更
⇒ OSSに貢献

仕事，研究，趣味
で使いたい



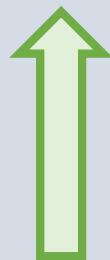
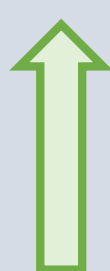
OSS利用者

バグや機能要望
がある

1. 背景 (ボツスライド)

加の

OSS (Open Source Software)



貢献

た貢献

開発コミュニティ
(ボランティア)

新人

貢献を受け入れられやすくするために、新人はどういった活動をするべきなのか考える必要がある。

IssueやPull Requestなどの
OSS開発の慣習に不慣れ

2. 開発コミュニティの慣習 (1/3)

開発コミュニティを支えるWebサービス



分散開発作業の支援や
Gitリポジトリのホスティングを行う

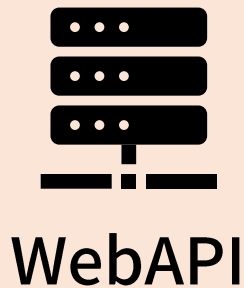
GitHubの機能

- Issue機能を使った議論
- Pull Request機能を使ったソースコード投稿

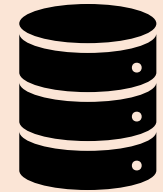
3. 方法（不要？）

1. 分析対象のOSSプロジェクトの決定
2. 対象プロジェクトのデータを収集
3. StackOverflowとGitHubで同じ開発者の紐づけ
3. 各開発者の活動履歴の取得
4. 活動履歴から仮説検定に必要なデータを抽出

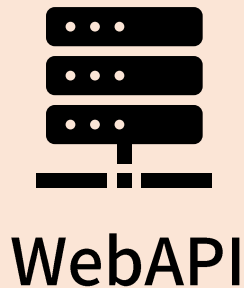
3.2. 対象プロジェクトのデータを収集（ボツスライド）



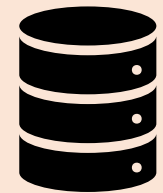
質問，回答
ユーザ名，ユーザID等



データベース



Issue, Pull Request,
ユーザ名，ユーザID等



データベース

1. 背景 | OSSコミュニティ (2/3)



OSS開発に関心を持つユーザ層

(以降、ユーザコミュニティと呼称)

技術的な貢献

しない (将来的に
開発コミュニティに移る
可能性はある)

利用されているWebサービス



OSSに関する質問, 話題など

- ・ Q&A形式
- ・ チャット形式

1. 開発コミュニティについて

開発コミュニティを支えるWebサービス



分散開発作業の支援や
Gitリポジトリのホスティングを行う

GitHubの機能

- Issue機能を使った議論
- Pull Request機能を使ったソースコード投稿

1. 背景 | OSSコミュニティ (2/3)



技術的な貢献をする開発者層

(以降、開発コミュニティと呼称)

技術的な貢献

- バグの報告や修正
- 機能拡張

OSSへの貢献を支えるWebサービス



- ソースコードの公開
- Issue機能 (課題管理)
- Pull Request機能 (第三者によるソースコードの変更)

8. 結果 | RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

Issueに対して最初のコメントが寄せられるまでの時間についての追加の質的調査

のコメントの内訳

説明を求めるコメント（例を求めるなど）：4件
間で議論する余地がないような投稿内容で
れた問題に対する解決策が提示されている：4件
投稿内容について議論があったのが：2件

新人以外のIssueに対する最初のコメント
・ Issue の投稿内容に関する追加の質問
・ 議論があったのが5件
・ 投稿内容の重複を示すコメントが

8. 結果 | RQ1

RQ1: StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

なぜ新人のIssueに対する最初のコメントが
寄せられるまでの時間が短いのか？

質的調査

- ① 代表的なIssue（最初のコメントが寄せられるまでの時間が中央値と一致）を抽出
- ② 新人，新人以外が投稿したIssueを各10件ランダムに抽出して目視

8. 結果 | RQ1

RQ1: StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

なぜ新人のIssueに対する最初のコメントが
寄せられるまでの時間が短いのか？

最初のコメントの特徴

新人のIssue

- 不完全な内容を補うための質問
- 即座に解決策の提示

短時間で投稿可能

新人以外のIssue

- 投稿内容についての議論

コメント内容を考えるのに
時間がかかる

新人は開発コミュニティからケアされている

⇒ 本研究の想定とは異なる結果

8. 結果 | RQ1まとめ

RQ1

StackOverflowとGitHubで新人の活動の
受け入れられやすさに違いはあるか

StackOverflow

新人と新人以外の扱いに
大きな違いなし

GitHub

新人と新人以外の扱いに
大きな違いなし

(ただし、Issueへの最初のコメントは
新人の方が早めにもらえる)

新人にとっては開発コミュニティには
受け入れられにくい^[X]という先行研究での調査とは異なる結果