# 卒 業 研 究 報 告 書

題　目　　異なる尺度で取得した
ソフトウェアリポジトリ間における
課題管理の差異の調査

指導教員　　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　16122056

氏　　名　　山本　凱

令和2年2月12日提出

# 異なる尺度で取得した ソフトウェアリポジトリ間における課題管理の差異の調査

令和 2 年 2 月 12 日

16122056　山本　凱

## 概　要

ソフトウェア工学において，ソースコードと過去の開発履歴を保持したリポジトリを研究対象とする研究者はそれを選定する必要がある．

GitHub にはリポジトリの人気や関心，貢献を示すいくつかの基準が存在している．これらの基準はリポジトリの機能の追加やバグの修正などに関係していると考えられるが，経験則によってリポジトリの選定尺度として利用されており根拠に乏しい．

本研究の目的は，ソフトウェアの品質に関わる研究に対して利用するリポジトリの選定の基準を提供することにある．

本研究では，いくつかの尺度でリポジトリを選択し，その管理方法と選定尺度の関連を検証する．またそれらの尺度によって選択されたリポジトリにどのような傾向があるかを調査する．具体的には Java 言語のパッケージ管理システム Maven で公開されているパッケージのうち，GitHub で管理されているリポジトリに関するデータを対象にする．

本研究によって，GitHub のリポジトリへの貢献者の数を示す Contributors の多さで選定したリポジトリは GitHub の課題解決機能に関して，課題の数が多く, さらに課題分類機能の利用率が高いことが示された．一方，パッケージが依存されている数の多さで選択したリポジトリは課題の数が少なく, さらに課題分類機能の利用率が低いため，課題を対象にする研究には向かないことを明らかにした．

# 目　次

**1. 緒言** **1**

**2. 準備** **3**

2.1 Libraries.io . . . . . 3

2.1.1 概要 . . . . . 3

2.1.2 Dependent Projects Count . . . . . 3

2.2 GitHub . . . . . 3

2.2.1 概要 . . . . . 3

2.2.2 Star . . . . . 3

2.2.3 Watch . . . . . 4

2.2.4 Fork . . . . . 4

2.2.5 Contributors . . . . . 4

2.2.6 Issue . . . . . 4

2.2.7 Label . . . . . 5

2.2.8 GitHub API . . . . . 5

2.3 Maven . . . . . 5

2.3.1 概要 . . . . . 5

2.3.2 pom.xml . . . . . 5

2.4 cloc . . . . . 5

**3. 研究の目的** **6**

3.1 研究の目的 . . . . . 6

3.2 研究設問 . . . . . 6

**4. 実験** **7**

4.1 データセット . . . . . 7

4.2 RQ1 に対する実験 . . . . . 7

4.2.1 準備：データセットの取得 . . . . . 7

4.2.2 準備：それぞれの尺度によるリポジトリの選定 . . . . . 7

4.2.3 実験 1 . . . . . 8

4.3 RQ2 に対する実験 . . . . . 8
4.3.1 準備：リポジトリ情報・コード総行数のデータを取得 . . . . . 8
4.3.2 実験 2 . . . . . 9
4.3.3 実験 3 . . . . . 9
4.3.4 実験 4 . . . . . 9
4.3.5 実験 5 . . . . . 10

**5. 実験結果** **11**
5.1 RQ1 に対する実験 . . . . . 11
5.1.1 実験 1 . . . . . 11
5.2 RQ2 に対する実験 . . . . . 11
5.2.1 実験 2 . . . . . 11
5.2.2 実験 3 . . . . . 11
5.2.3 実験 4 . . . . . 13
5.2.4 実験 5 . . . . . 13

**6. 考察** **18**
6.1 RQ1 に対する考察 . . . . . 18
6.1.1 実験 1 に対する考察 . . . . . 18
6.1.2 RQ1 への回答 . . . . . 21
6.2 RQ2 に対する考察 . . . . . 21
6.2.1 実験 2 に対する考察 . . . . . 21
6.2.2 実験 3 に対する考察 . . . . . 22
6.2.3 実験 4 に対する考察 . . . . . 24
6.2.4 実験 5 に対する考察 . . . . . 24
6.2.5 RQ2 に対する回答 . . . . . 25
6.3 リポジトリの選定基準の提供 . . . . . 25

**7. 妥当性への脅威** **26**
7.1 外的妥当性 . . . . . 26
7.2 内的妥当性 . . . . . 26

7.3 構成概念妥当性 . . . . . 26

**8. 関連研究** **28**

**9. 結言** **29**

**謝辞** **29**

**参考文献** **30**

# 1. 緒言

ソフトウェア工学において，コミット情報を利用した不具合予測や開発者コミュニティ分析などソースコードと過去の開発履歴を保持するシステム（Git）および，ソースコード管理サービス（GitHub）を対象とした研究は数多く存在する．これらの研究はソフトウェア中のバグの発見やソフトウェア開発手法の改善に貢献している．

こうした研究を行う上で，研究者はその研究の用途に適したGitHubリポジトリを選定するため，何らかの選定基準を用いる必要がある．GitHubにはリポジトリの人気や関心，貢献を示す複数のメトリクスが存在している．このうち，リポジトリへの関心や感謝などの肯定的意思を示すStarの数は人気や，バグの修正，機能の追加に関係していると考えられているため選定基準としてしばしば利用される [1, 2, 3]．しかしこれは暫定的に利用されているためリポジトリ選定の根拠として乏しい．

また，Vasilescuら [4] の研究によると，Star数の増加によって，外部の開発者からのバグの報告が増加すること，およびフォークの増加にしたがってバグの報告が増加することが示されている．一般的に，GitHubではバグの報告には課題解決機能であるIssueが用いられる．しかし，リポジトリの情報を表す複数のメトリクスと，IssueおよびそのIssueのドメインを示すLabelを利用した管理についてはまだ十分に研究されていない．

本研究の目的は，ソフトウェアの品質に関わる研究に対して利用するリポジトリの選定の基準を根拠とともに提供することにある．

本研究では，いくつかの尺度でリポジトリを選択し，GitHubのIssueおよびLabelに注目した管理方法と選定尺度の関連を調査する．また各尺度によって選択されたリポジトリ群にどのような傾向があるかを調査する．データセットはJava言語のパッケージ管理システムMavenで公開されているパッケージのうち，GitHubで管理されているリポジトリを対象にする．

本研究によって，リポジトリへの貢献者の人数で選定したリポジトリはGitHub上により多くのIssueが存在し，それに対するLabelの付与率も高いことが示された．一方，他のパッケージからの被依存数で選択したリポジトリは貢献者数で選択したものに比べてGitHub上でのIssue数は1/7程度であり，Issueに対するLabelの付与

率も 10%以上低いことが示された．そのため，Label によるカテゴリ分類が行われた Issue を多く持つリポジトリのデータセットを必要とする場合には，貢献者数によるリポジトリの選定が適切であり，被依存数によってリポジトリを選定するのは適切でないといえる．

最後に以降の本報告の構成を述べる．第 2 章では本研究で利用するサービス等の用語について説明する．第 3 章では本研究の目的について説明し，その目的に沿った研究設問を述べる．第 4 章では研究設問に対応した実験について説明し，第 5 章では実験の結果を述べ，第 6 章では実験結果をもとに研究設問に対する考察を述べる．第 7 章では妥当性の検証をし，第 8 章では関連研究について述べ，第 9 章をもって本研究の結言とする．

# 2. 準備

## 2.1 Libraries.io

### 2.1.1 概要

Libraries.io[5] とは，異なるパッケージ管理システムのオープンソースパッケージについての情報を収集し提供しているプラットフォームである．39 のパッケージ管理システムと GitHub を含む 3 つの開発プラットフォームの様々な情報を収集の対象としている．

### 2.1.2 Dependent Projects Count

あるプロジェクトの他のプロジェクトからの被依存数を示す項目．Libraries.io で提供されている．本研究ではこの項目を尺度とし，被依存数の多いプロジェクトが属するリポジトリを選定するため用いた．以降このメトリクスを被依存数と呼称する．

## 2.2 GitHub

### 2.2.1 概要

GitHub とは，開発プラットフォームである．オープンソースプロジェクトから個人プロジェクトまで GitHub 上にソースコードをホスティングすることでチームでの開発を円滑に行うことができる．特徴として 2.2.2 項から 2.2.8 項に述べる機能を有している．バージョン管理は Git で行われる．Git は分散型バージョン管理システムの一つであり，開発履歴を保持する機能を備えている．

### 2.2.2 Star

GitHub 上のリポジトリやトピックに付すことができる機能．GitHub ユーザは Star を付すことで興味を持ったプロジェクトを追跡し，関連するコンテンツを見つけることができる．さらにリポジトリに Star を付けることはそのリポジトリメンテナに対してその作業についての感謝を示すことであるとされている [6]．GitHub では Star の数によるフィルタリング機能を有している．本研究では Libraries.io が提供す

るプロジェクトのStar数をプロジェクトの選定の尺度として用いた．以降，このメトリクスをスター数と呼称する．

### 2.2.3 Watch

GitHub上のリポジトリからの通知を受け取ることができる機能[7]．Watchした人はそのリポジトリについて新しく作成されたプルリクエストおよびIssueに関する通知を受け取ることができる．本研究ではLibraries.ioが提供するプロジェクトのWatch数をプロジェクトの選定の尺度として用いた．以降，このメトリクスをウォッチ数と呼称する．

### 2.2.4 Fork

GitHub上のリポジトリを自分のアカウントにコピーできる機能[8]．Forkすることでオリジナルのリポジトリに変更を加えることなくForkしたユーザが内容を改変できる．さらに，Forkしたユーザは，変更部分をオリジナルのリポジトリに反映することを提案できる．本研究ではLibraries.ioが提供するプロジェクトのFork数をプロジェクトの選定の尺度として用いた．以降，このメトリクスをフォーク数と呼称する．

### 2.2.5 Contributors

GitHub上のプロジェクトに対して変更を依頼し，外部から変更したユーザを指す．一般にContributorsはFork機能を利用してリポジトリの内容を変更し，その変更の提案が元のリポジトリの管理者に受け入れられた時に追加される．本研究ではLibraries.ioが提供するプロジェクトのContributors数をプロジェクトの選定の尺度として用いた．以降，このメトリクスをコントリビューター数と呼称する．

### 2.2.6 Issue

GitHubの課題解決機能[9]．対象のリポジトリへのアクセス権を持つ全てのGitHubユーザはそのリポジトリにIssueを投稿できる．Issueはバグの報告，追加機能の提案，タスク管理，質問とそれに対する返答など，様々な用途に使用される．Issueに

は Open と Closed という 2 つの状態が存在する．Open の状態はまだその Issue が終了しておらず，Closed の場合はその Issue が終了していることを指す．

### 2.2.7 Label

GitHub の Issue で利用されるカテゴリ分類機能 [10]．Label にはバグの報告に利用される bug や，機能の追加や提案をする enhancement などが存在し，リポジトリ管理者が自由に Label の種類を増やすこともできる．この Label がついた Issue は全てが直ちに解決されるわけではないものの，効率的に処理されることが先行研究により示されている [11]．

### 2.2.8 GitHub API

GitHub が提供する API．リポジトリの情報等を取得できる．本研究では，リポジトリのスター数，ウォッチ数，フォーク数を取得するために用いた．

## 2.3 Maven

### 2.3.1 概要

Maven とは Apache が提供するソフトウェアプロジェクト管理ツールである [12]．主に Java 用のプロジェクトを管理するために用いられ，プロジェクト間の依存解決のための機能が提供されている．

### 2.3.2 pom.xml

Project object model の考えを元に作成されるファイル．プロジェクトをビルドするために必要なパッケージ情報などを記述する．パッケージの依存情報がこのファイルから取得できる．

## 2.4 cloc

コード行数（LOC）の計測ツール [13]．cloc がサポートしている識別子が付けられたファイルの空白行，コメント行，コード行を識別しそれらの数を取得できる．本研究では，プロジェクトの総行数を測定するために利用した．

# 3. 研究の目的

## 3.1 研究の目的

本研究の大域的な目的は，ソフトウェアの品質に関わる研究に対して利用するリポジトリの選定の基準を提供することにある．

その一部として，リポジトリの課題管理に焦点を当てた研究でリポジトリ群を選定する際に，その研究に適当な選定基準を調査する．

そもそも，異なる尺度で選定したリポジトリ群がほぼ一致していれば，特に選定基準を設ける必要はないといえる．したがって最初に，異なる尺度で選定したリポジトリ群がどの程度一致しているかを調べる．

次に，異なる尺度で選定されたリポジトリ群が異なっていれば，どの尺度によって選定されたリポジトリ群が最も課題管理を綿密に実施しているかを調べることで，上述した目的に対して最も適切な選定基準を提供できる．

これらを踏まえ，設定した目的の達成に向けて次の研究設問を立てる．

## 3.2 研究設問

前節の研究目的の達成に向けて，2 つの研究設問（RQ）を立てる．

RQ1 複数の尺度を用いて GitHub で管理されているリポジトリ群をそれぞれ選定した場合，それらのリポジトリ群はどの程度一致するか．

RQ2 異なる尺度によって選定されたリポジトリ群で Issue の規模や Label を用いた課題管理状況はどれほど異なるか．

# 4. 実験

## 4.1 データセット

実験には，GitHub に存在するリポジトリのうち，Maven によってパッケージ管理されている Java プロジェクトに含まれるものをデータセットとして使用する．Java はよく知られた歴史の長い言語であり，プロジェクト数が多くそれを対象とした研究事例も多いことから対象とした．また Maven の管理対象であるものに焦点を絞ることで，高い公共性と有益性を前提として作られたプロジェクトを対象とした．

Libraries.io が提供するデータを用いることで，各プロジェクトが GitHub および Maven に登録されているかを調査できる．

## 4.2 RQ1 に対する実験

### 4.2.1 準備：データセットの取得

Libraries.io が提供しているオープンソースプロジェクトに関するデータのうち，実験開始時に最新のリリースであったバージョン 1.4.0 をダウンロードした．これは 2018 年 12 月 22 日に公開されたものであり，ここには 3,494,424 個のプロジェクトのデータが含まれている．一つのリポジトリ中に複数のプロジェクトが属している場合も存在するため，リポジトリの数とプロジェクトの数は必ずしも一致しない．

ここから，Maven で公開されており，かつ GitHub にホスティングされているプロジェクトを抽出した．結果，106,671 個のプロジェクト情報が得られた．これらのプロジェクトが属するリポジトリは合計で 31,429 個となる．

### 4.2.2 準備：それぞれの尺度によるリポジトリの選定

4.2.1 項で取得したデータセットから各プロジェクトのスター数，ウォッチ数，フォーク数，コントリビューター数，被依存数が取得できる．これらを本研究における選定尺度とする．4.2.1 項で得られたプロジェクトを選定尺度で降順にソートした．さらにそれぞれの尺度に対してリポジトリに重複がないよう上位 500 件のリポジトリを抽出した．ただし，あるリポジトリに属するプロジェクトが複数ある場合，それらのプロジェクトの被依存数の最大値をそのリポジトリの被依存数とする．

#### 4.2.3 実験 1

それぞれの選定尺度で取得した上位 500 件のリポジトリ群がどの程度一致しているかを調べる．ここでいう一致とは，別々の尺度で選定したリポジトリ群同士に同じリポジトリがどの程度存在するかを指す.

### 4.3 RQ2 に対する実験

#### 4.3.1 準備：リポジトリ情報・コード総行数のデータを取得

4.2.2 項で取得したそれぞれの尺度における上位 500 件のリポジトリについて，RQ2 に対する実験のために必要な情報を追加で取得する．取得する情報は次の通りである．4.3.2 項からの実験の準備として，Issue Page の有無， Closed な Issue 数， 各 Issue に付与されている Label の数を取得した．さらに，リポジトリを取得し，cloc によってそのリポジトリ内の cloc にサポートされているファイルの LOC の合計を算出した．また，使用した cloc のバージョンは 1.84 である．これらとは別に，4.3.5 項で行う実験のため GitHub API を用いて，各リポジトリのスター数，ウォッチ数，フォーク数，Closed な Issue 数を取得した．Closed な Issue は主にリポジトリ管理者によって何らかの形でその Issue が解決されたことを示し，その数によって管理者による課題管理の程度を測ることができると考えられるためである.

RQ2 で，本研究で利用する Libraries.io のデータのリリース日以前のデータが正確に取得できない GitHub API のバグを発見したため，研究開始時の最新リポジトリの Closed な Issue の情報と各メトリクス（スター数，コントリビューター数，ウォッチ数）を利用することとした．さらに，コントリビューター数については，GitHub API で正確に値を取得できないバグを発見した．したがって，コントリビューター数については，急激な増加はしないと仮定し，Libraries.io のデータを利用した．被依存数に関しては，急激な増加をする可能性があるため，Closed な Issue 数との相関係数を取る実験 5 で利用しないこととした．また，発見したバグは GitHub の運営団体に報告した.

#### 4.3.2 実験 2

それぞれの尺度で選ばれたリポジトリ群がどれほど GitHub 上で Issue Page を持っているかを調べる．これは，リポジトリが GitHub の Issue 機能を用いずに GitHub 以外のバグトラッキングシステムやメーリングリストを使用する場合があることから，それぞれの尺度で選定されたリポジトリにおける Issue 機能の利用状況の差異を示すためである．

#### 4.3.3 実験 3

それぞれの尺度で選ばれたリポジトリ群のうち Issue Page を持っているものについて Closed な Issue 数を調べる．次に，Closed の Issue を 1 つ以上持っている場合，Label を利用していないリポジトリがどれほど存在するかを調べる．また，Label がどの程度の割合で Issue に付与されているかを調べる．さらに，Closed な Issue 数と Label に付与された Issue の数の相関をピアソンの相関係数を用いて調べる．これは，リポジトリが GitHub の Issue 機能を用いずに GitHub 以外のバグトラッキングシステムやメーリングリストを使用する場合があることから，それぞれの尺度で選定されたリポジトリにおける Issue 機能の利用状況の差異を示すためである．

#### 4.3.4 実験 4

それぞれの尺度で選ばれたリポジトリ群のうち Issue Page を持っているものについて，Closed な Issue 数とリポジトリ中のファイルの LOC の合計との相関をピアソンの積率相関係数を使って調べる．なお，識別子は特に指定せず，空白行，コード行，コメント行の合計を行数として扱った．Closed な Issue 数は課題管理の綿密さを表していると考えられる一方で，単純にプロジェクトの規模が多いことで増加する可能性も考えられる．コードの規模と Closed な Issue 数との相関が小さければ，そういった可能性を排除でき，Closed な Issue 数を課題管理の綿密さを表すメトリクスとして使用できる．

#### 4.3.5 実験 5

それぞれの尺度で選ばれたリポジトリ群のうち Issue Page を持っているものについて，GitHub API を用いて取得した GitHub 中のメトリクスであるフォーク数，スター数，ウォッチ数，および Libraries.io によって取得したコントリビューター数のそれぞれの相関，およびメトリクスとさらにそのリポジトリの持つ Closed な Issue 数のそれぞれの間の相関をピアソンの積率相関係数を使って調べる．これは実験 4 までの結果から得られるリポジトリ選定基準の妥当性をさらに検証するものである

# 5. 実験結果

## 5.1 RQ1 に対する実験

### 5.1.1 実験 1

実験 1 に対する結果を表 5.1 に示す．各尺度によって選定したリポジトリ群同士で最も一致率が高かったのは，フォーク数とウォッチ数についての，75.2%であった．最も一致率が低かったのは，被依存数とスター数でリポジトリ群を選定した場合の，12.8%であった．フォーク数，ウォッチ数，スター数で選定したリポジトリ群同士では 7 割程度の一致率であり，これら 3 つとコントリビュータ数との一致率は 5 割程度に留まる．被依存数と他の尺度で選んだリポジトリの一致率は 2 割程度と低い．

## 5.2 RQ2 に対する実験

### 5.2.1 実験 2

実験 2 に対する結果を表 5.2 に示す．リポジトリが Issue Page を持っている割合を *%IP*，エラーによって取得できなかった数を *#Error* とする．なお，*%NoIP* の算出に際して，エラーの出たリポジトリは排除している．エラーの要因としては，GitHub 上のリポジトリが削除や非公開といった措置によってその情報が取得できなかったことや cloc のエラーが挙げられる．ここで取得できなかったリポジトリ情報については以降の実験 3，実験 4 でも含めない．

スター数で選定したリポジトリ群は GitHub で Issue Page を持っている割合が最も高かった．一方，被依存数で選定したリポジトリ群は GitHub で Issue Page を持っている割合が最も低かった．全体としては，どの選定尺度を用いても各リポジトリ群の 8 割以上が Issue Page を持っていると分かる．

### 5.2.2 実験 3

Issue Page を持っているリポジトリの持つ Closed な Issue 数の結果を表 5.3 に示す．

最も Closed な Issue 数の中央値が大きかったのはコントリビューター数で選ばれたリポジトリ群であった．一方で最も小さかったのは被依存数で選ばれたものであっ

表 **5.1**　異なる尺度で選別したリポジトリ群の一致率

| | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|
| フォーク数 | 53.4% | 73.4% | 75.2% | 16.6% |
| コントリビューター数 | - | 51.2% | 47.0% | 24.4% |
| スター数 | - | - | 68.4% | 12.8% |
| ウォッチ数 | - | - | - | 17.6% |

表 **5.2**　異なるメトリクスで選別されたリポジトリが **Issue Page** を持っている割合

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| *#Error* | 5 | 4 | 1 | 3 | 4 |
| *%IP* | 93.1% | 88.1% | 95.6% | 94.2% | 83.7% |

た．また，その尺度で選ばれたリポジトリ群の持つ Closed な Issue 数の中央値が最高値となったコントリビュータ数と二番目に高いフォーク数との間において，Closed な Issue 数の差異についてマンホイットニーの U 検定によって P 値を求めたところ，0.00475 となった．

次に，それぞれの選定尺度で Closed な Issue を 1 つ以上持ちながら，Label を全く利用していないリポジトリの数を表 5.4 に示す．*#hasIssueRepo* は Closed な Issue を 1 つ以上持っているリポジトリ数を，*#hasLabel* は Closed な Issue を 1 つ以上持っているリポジトリのうち Label を利用しているリポジトリ数を，*#hasLabelPerAll* は，*#hasIssueRepo* 中の *#hasLabel* の割合を示す．最も Label を付与していなかったのは被依存数で選定されたリポジトリ群であった．一方，最も多く Label を付与していたのはコントリビューター数で選定したリポジトリ群であった．

また，Closed な Issue 全体に対して Label が付与されている Issue 数の割合の最小値，最大値，四分位数を表 5.5 に示す．この割合の中央値が最も大きかったのはコントリビューター数で選定したリポジトリ群であった．一方，最も小さかったのは被依存数で選定したリポジトリ群であった．

さらに，各リポジトリにおける Closed な Issue 数と Label の付けられた Closed な Issue 数の相関について表 5.6 に示す．全てのリポジトリ群で相関係数が 0.90 以上であった．

### 5.2.3 実験 4

実験 4 の結果を表 5.7 に示す．スター数で選択したリポジトリ群の Closed な Issue 数とコード行数との相関係数は 0.424 であった．スター数以外で選定された尺度はいずれの尺度も 0.15 程度である．

### 5.2.4 実験 5

実験 5 の結果を表 5.8 から表 5.9 に示す．実験 5 では cloc を利用していないため，実験 2 から実験 4 では cloc のエラーによって実験対象から排除されていたリポジトリが含まれている．使用したリポジトリ数は表 5.8 に示される通りである．また表 5.9 中の表記として，例えば Fork-Watch はフォーク数とウォッチ数の相関係数であることを示す.「フォーク数」列の Fork-Watch の値である 0.527 は，フォーク数で選定

表 5.3　異なる尺度で選別したリポジトリ群の Closed な Issue 数の最大値・最小値・四分位数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| 最小値 | 4 | 0 | 24 | 0 | 0 |
| 第一四分位数 | 368.0 | 579.0 | 306.0 | 216.8 | 37.0 |
| 中央値 | 912.0 | 1063.0 | 809.0 | 773.0 | 143.0 |
| 第三四分位数 | 2049.0 | 2193.0 | 1762.0 | 1899.5 | 553.5 |
| 最大値 | 20655 | 20654 | 20678 | 20665 | 20137 |

表 5.4　異なる尺度で選別したそれぞれのリポジトリ群のうち，Closed な Issue を持っているものについて Label を利用しているリポジトリの割合

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| *#hasIssueRepo* | 461 | 434 | 477 | 453 | 401 |
| *#hasLabel* | 434 | 427 | 455 | 417 | 321 |
| *hasLabelPerAll* | 94.1% | 98.4% | 95.4% | 92.1% | 80.0% |

表 5.5　異なる尺度で選別したリポジトリ群の Closed な Issue に対する Label の付与率の最大値・最小値・四分位数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| 最小値 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| 第一四分位数 | 11.4% | 18.9% | 11.9% | 9.7% | 1.6% |
| 中央値 | 28.1% | 36.5% | 27.3% | 27.5% | 22.2% |
| 第三四分位数 | 50.4% | 56.3% | 47.7% | 50.4% | 54.0% |
| 最大値 | 100.0% | 100.0% | 99.9% | 100.0% | 100.0% |

表 5.6　Closed な Issue の数と Label のついた Closed な Issue の数の相関

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| 相関係数 | 0.91 | 0.91 | 0.91 | 0.91 | 0.96 |

表 5.7　Closed な Issue 数とコード行数との相関係数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| 相関係数 | 0.161 | 0.145 | 0.424 | 0.160 | 0.160 |

したリポジトリ群において，各リポジトリのフォーク数とウォッチ数の相関係数が0.527であったことを示す．下線が引かれた部分については，選定尺度として利用したメトリクスとClosedなIssue数の相関係数を示す部分である．

表5.9からの4つのことが分かる．

- Fork-Star，Fork-Watch，Star-Watchのそれぞれについて，どの尺度でリポジトリ群を選定した場合でも，相関係数が0.80以上となっている．
- 被依存数によって選定したリポジトリ群は，Watch-Starが0.956，Fork-Watchが0.921，Fork-Starが0.880であった．これはいずれも，他の尺度によって選定されたリポジトリ群の同じ相関係数の中で最も大きい．
- 被依存数によって選定したリポジトリ群のWatch-ClosedIssueと，Fork-ClosedIssue，およびフォーク数によって選定したリポジトリ群のContributors-ClosedIssueは0.70以上であった．しかし，他の尺度によって選定したリポジトリ群で，ClosedなIssue数とGitHubメトリクスとの相関係数が0.70を超えているものは存在しない．
- フォーク数で選定したリポジトリ群のフォーク数のように選定尺度と同じメトリクスとClosedなIssue数の相関係数はいずれも0.60以上である．

表 **5.8**　実験 **5** で使用したリポジトリの数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| リポジトリ数 (件) | 463 | 439 | 477 | 468 | 416 |

表 **5.9**　各尺度で選定したリポジトリ群のメトリクス間の相関係数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| Fork-Contributors | 0.527 | 0.550 | 0.551 | 0.568 | **0.687** |
| Fork-Star | 0.812 | 0.825 | 0.823 | 0.821 | **0.882** |
| Fork-Watch | 0.908 | 0.917 | 0.914 | 0.872 | **0.923** |
| Fork-ClosedIssue | 0.653 | 0.663 | 0.685 | 0.673 | **0.763** |
| Contributors-Star | 0.544 | 0.539 | 0.557 | 0.591 | **0.745** |
| Contributors-Watch | 0.497 | 0.526 | 0.520 | 0.483 | **0.732** |
| Contributors-ClosedIssue | 0.677 | 0.622 | 0.699 | 0.702 | **0.753** |
| Star-Watch | 0.896 | 0.914 | 0.900 | 0.840 | **0.956** |
| Star-ClosedIssue | 0.608 | 0.616 | 0.641 | 0.632 | **0.666** |
| Watch-ClosedIssue | 0.644 | 0.666 | 0.677 | 0.617 | **0.726** |

# 6. 考察

## 6.1 RQ1 に対する考察

### 6.1.1 実験 1 に対する考察

フォーク数, ウォッチ数, スター数で選定したリポジトリ群同士では 7 割程度の一致率であった．このため，これら 3 つの選定尺度によって得られるリポジトリ群はある程度類似しているといえる．しかしながら，3 割程度は異なるため，完全に一致しているとは言い難い．被依存数で取得したリポジトリ群と他の尺度で選定したリポジトリ群の一致率はいずれも 25%を超えておらず，ほとんど一致しない．またコントリビューター数で選定したリポジトリ群と他の尺度によって選定したリポジトリ群の一致率は 5 割程度であることから，大きく一致しているとは言い難い．

それぞれの尺度で選定したリポジトリ郡のみに含まれるリポジトリの特徴について考察する．

**被依存数** で選定したリポジトリ群のみに含まれていたリポジトリには，

- mysql/mysql-connector-j
- pgjdbc/pgjdbc
- webjars/jquery
- jboss/jboss-parent-pom

などが挙げられる．

mysql/mysql-connector-j については，そもそも GitHub のサービス開始以前からの古いプロジェクトであることから，バグを修正したリリースは発表されるもののメジャーバージョンの更新が 2 年以上行われていない．したがって，リポジトリに関するトピックや機能の追加などに関心を払う必要性がなく，Watch の需要がない．また，外部の開発者が大きく機能の変更を加える必要がないことからも Fork されることがなく，Contributors の数も比較的少ないと考えられる．pgjdbc/pgjdbc についても，同様の状態であることが確認された．

webjars/jquery や jboss/jboss-parent-pom のリポジトリの内容を確認したところ，ソースコードがほとんど存在していないことが確認された．これらはあるソフト

ウェアが別のリポジトリやソフトウェア，ライブラリを利用するために必要となる設定が記述されたファイルを保有しているリポジトリに当たる．こうしたリポジトリは，ほとんどソースコードが存在していないためバグの発生や機能の追加がなく，ユーザの関心を引くリポジトリではないため，Star や Watch を付与するユーザが少ないと考えられる．

被依存数に関するこれらの考察をさらに支持する研究として，Borges ら [14] によると，スターを付けた理由として，実際にリポジトリを利用していることを挙げたユーザは 4 割弱程度であった．それ以外の過半数のユーザは自身のプロジェクトの参考にするためであることや，単純な興味であることを挙げており，積極的にリポジトリに介入しているユーザとはスターの使用目的が異なっている．こうしたことから，スターをつけるユーザとそのリポジトリを使用するユーザは必ずしも一致しない．

**ウォッチ数** で選定したリポジトリ群のみに含まれていたリポジトリには，

- uber/react-map-gl
- wix/wix-restaurants-availability
- Netflix/Priam

などが挙げられる．

uber/react-map-gl に関しては，頻繁にメジャーバージョンのリリースが行われていた．また，wix/wix-restaurants-availability は 2020 年 2 月 2 日現在でスター数が 2 であるにも関わらず，ウォッチ数は 352 であ った．このリポジトリの README ファイルを確認したところ，Usage の部分が未だ記述されておらず開発が初期の段階であった．こうした変更があるかもしれないリポジトリに対して変更の通知を希望するユーザが Watch をつけていると考えられる．また，ウォッチ数で選定したリポジトリ群のみに含まれていたリポジトリには，Netflix が管理するリポジトリが 29 件あり，uber が管理するリポジトリもこれに 9 件含まれていた．このように，ユーザは人気のある企業が開発する OSS には開発初期から関心を持っており，Watch を付けていると考えられる．

**フォーク数** で選定したリポジトリ群のみに含まれていたリポジトリには，

- BlackrockDigital/startbootstrap-simple-sidebar

- mathiasbynens/jquery-placeholder
- bingoogolapple/BGARefreshLayout-Android

などが挙げられる．

BlackrockDigital/startbootstrap-simple-sidebar はコントリビューター数が 1 人であった．他のリポジトリに関してもコントリビューター数が比較的少なかった．Fork されているにも関わらずコントリビューター数が少ないことは，元となるリポジトリに変更を加える目的で Fork するわけではなく，ユーザが別のプロジェクトとして利用するためにそのリポジトリを Fork している可能性が高い．こうしたことが，コントリビューター数で選定したリポジトリ群との差異に影響を与えていると考えられる．

**スター数** で選定したリポジトリ群のみに含まれていたリポジトリは，

- kristoferjoseph/flexboxgrid
- facebookarchive/Keyframes
- svgdotjs/svg.js
- lipis/flag-icon-css

などが挙げられる．

kristoferjoseph/flexboxgrid は，CSS の描画に関するツールであり，Web ページをユーザが用意に作成できるようするツールである．facebookarchive/Keyframes や svgdotjs/svg.js，lipis/flag-icon-css といったリポジトリについても，アニメーションや CSS に関するものであり，こうした，Web やアプリケーションでの UI や表示に関して簡単に作成できるようなツールやフレームワークがユーザが Star をつける動機づけになっていると考えられる．また，これらのリポジトリについては，決して少ないフォーク数でなかったことから，一定の人気を持ちつつ，そのリポジトリを利用した開発もされていることが分かった．したがって，本研究では上位 500 件のリポジトリを調査したが，500 件以上のリポジトリを選定した場合にフォーク数で選択したリポジトリ群との一致率が向上する可能性がある．

**コントリビューター数** で選定したリポジトリ群のみに含まれていたリポジトリは，

- gatling/gatling
- qunitjs/qunit

- kulshekhar/ts-jest
- testcontainers/testcontainers-java
- spalantir/tslint
- languagetool-org/languagetool
- scalameta/scalafmt

などが挙げられる．

gatling/gatling は，qunitjs/qunit，kulshekhar/ts-jest，testcontainers/testcontainers-java はテストに関連したツールである．テストツールはその性質上，テスト対象のバージョンアップやツール利用者による未知のバグの発見など，外的要因によって新たなテスト項目の需要が発生すると考えられる．このため，外部開発者のプルリクエストによる変更の割合が大きくなり，コントリビューター数が多くなっていると考えられる．また，spalantir/tslint や languagetool-org/languagetool, scalameta/scalafmt は Lint ツールやフォーマッタである．これらは，ユーザが書いたコードや文章を訂正することなどを目的にしたツールである．こうしたツールはルールベースで記述されることが多く．新たなルールが必要になった時に，ルール部分を追記するだけで済む．したがって，外部の開発者がリポジトリに貢献しやすくコントリビューター数が多くなっていると考えられる．

### 6.1.2 RQ1 への回答

異なる選定尺度で取得したリポジトリ群は必ずしも一致しない．また，使用した選定尺度によって他の尺度で選定した場合の取得結果との一致率も異なる．したがって，ある尺度に基づいてリポジトリ群を選択する場合には，他の尺度でリポジトリ群を選択した場合とは異なるリポジトリ群が得られていることを承知する必要がある．

## 6.2 RQ2 に対する考察

### 6.2.1 実験 2 に対する考察

実験 2 について，Issue Page を最も多く持っていたのはスターで選定したリポジトリ群であった．一方，Issue Page が最も少なかったのは被依存数で選定したリポジト

リ群であった．Issue Page を持たないリポジトリは GitHub 以外でバグトラッキングシステムをしばしば利用しており，被依存数が高くなる傾向にある古くからのプロジェクトはこうしたシステムを取り入れられていると考えられる．

Issue Page の利用率が全体で 8 割以上であり，これは，おおよそどの選定基準でも，Issue Page を持っているリポジトリを選定できると考えられる．

### 6.2.2 実験 3 に対する考察

実験 3 については，コントリビューター数で選定したリポジトリ群の Closed な Issue 数の中央値が最も大きかった．しかし，コントリビューター数で選定したリポジトリは，表 5.2 に示される通り，フォーク数，スター数，ウォッチ数で選定したそれぞれのリポジトリ群と比べて Issue Page を持っていない割合が大きく，比較しているリポジトリ数が異なるため，対等な評価結果とは言い難い．

そこで，コントリビューター数でリポジトリ群を選定した場合と同数の上位 437 件のリポジトリをフォーク数，スター数，ウォッチ数で選定した場合の Closed な Issue 数に関する結果を，表 6.1 に示す．この結果から，同数のリポジトリを選択した場合でも，Closed な Issue 数の中央値が最も小さかったのは，被依存数で選択したリポジトリ群であったのに対し，最も大きかったのはコントリビューター数で選択したリポジトリ群であった．

Closed な Issue に対する Label の付与率の中央値は，コントリビューター数で選定したリポジトリ群が最も大きく，被依存数で選択した場合が最も小さかった．しかし，この結果の妥当性に対しても同様の懸念がある．そこで，被依存数で選定した場合に Closed な Issue を 1 つ以上持っているリポジトリ数の 401 件に揃えて，その他の尺度を用いてリポジトリを選定した結果を表 6.2 に示す．この場合も変わらず，選定するリポジトリ数を揃えた場合でもコントリビューター数で選定したリポジトリ群の Closed な Issue に対する Label の付与率の中央値が最も大きかった．また，この場合でも，被依存数で選定したリポジトリ群の Closed な Issue に対する Label の付与率の中央値が最も小さかった．

最後に，Closed な Issue 数と Label の付与された Closed な Issue 数の相関について表 5.6 に示される通り，いずれの結果についても高い相関があると言える．

こうした結果に影響を与えた要因として，被依存数で選定したリポジトリ群の

表 6.1　異なる尺度で選別した同数のリポジトリ群の Closed な Issue 数の最大値・最小値・四分位数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 |
|---|---|---|---|---|
| 最小値 | 4.0 | 0.0 | 24.0 | 0.0 |
| 第一四分位数 | 376.0 | 579.0 | 337.0 | 230.0 |
| 中央値 | 946.0 | 1063.0 | 864.0 | 811.0 |
| 第三四分位数 | 2075.0 | 2193.0 | 1883.0 | 1957.0 |
| 最大値 | 20655.0 | 20654.0 | 20678.0 | 20665.0 |

表 6.2　異なる尺度で選別した同数のリポジトリ群の Closed な Issue 数に対する Label の付与率の最大値・最小値・四分位数

| | フォーク数 | コントリビューター数 | スター数 | ウォッチ数 | 被依存数 |
|---|---|---|---|---|---|
| 最小値 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| 第一四分位数 | 11.4% | 20.1% | 12.8% | 12.3% | 1.6% |
| 中央値 | 28.1% | 37.7% | 28.5% | 29.9% | 22.2% |
| 第三四分位数 | 50.4% | 56.6% | 47.8% | 50.9% | 54.0% |
| 最大値 | 100.0% | 100.0% | 99.9% | 100.0% | 100.0% |

Closed な Issue 数が，他の尺度に比べて少なかったことから Label による Issue のカテゴリ分類の必要性がないほど少ない Issue 数では Label が利用されていない可能性が考えられる．一方，コントリビューター数で選定したリポジトリ群は Closed な Issue 数も他の尺度に比べて多かったことから，Label による Issue のカテゴリ分類によって円滑に Issue 管理を進めようとしているものと考えられる．

これらのことから，特にコントリビューター数でリポジトリ群を選定した場合に，Closed な Issue 数が多く，またそれに対する Label も付与されている数が多いものを選定できる．

### 6.2.3 実験 4 に対する考察

実験 4 については，いずれの結果についてもリポジトリの総コード行数と Closed な Issue 数の間に高い相関があるとは言い難い．したがって，Closed な Issue 数はリポジトリ内のコードの規模によらず，純粋にリポジトリの管理者による課題管理の綿密さを表していると考えられる．ただし，本研究では識別子を区別せずコード行数を取得したため，識別子を区別した場合には異なる結果が得られる可能性もある．

### 6.2.4 実験 5 に対する考察

実験 5 は各尺度によって選定したリポジトリ群で，表 5.9 の通り，ウォッチ数とフォーク数，ウォッチ数とスター数のメトリクスとの相関係数が大きい．ウォッチ数とフォーク数の関係は Sheoran ら [15] によって同様にして示されているが，本研究ではウォッチ数とスター数との相関係数がいずれも 0.80 以上であることから高い相関があることがわかった．こうした傾向が実験 1 でフォーク数，スター数，ウォッチ数でそれぞれ選ばれたリポジトリ群の高い一致率に影響を与えたと考えられる．また，Closed な Issue 数について，被依存数で取得したリポジトリ群ではフォーク数，ウォッチ数，スター数それぞれと Closed な Issue 数の相関係数が全て 0.70 以上であることから，強い相関であるとは言えないものの，ある程度の相関を示しているといえる．また，被依存数で選定したリポジトリ群の Star と Closed な Issue 数の間の相関係数が 0.666 であることから，リポジトリの内容の変更に関係する Watch，Fork，Contributors が Star よりも Closed な Issue と関係している可能性がある．さらに，他の尺度で選定したリポジトリ群以外でも，各メトリクスと Closed な Issue 数との相

関が全くないとは言えず，各メトリクスに加えて本研究では対象にしなかった他の要因が影響を与えている可能性がある．

### 6.2.5 RQ2 に対する回答

以上のことから，本研究で対象にしたデータでは異なる尺度で選定したリポジトリの課題管理については，コントリビューター数で選択した場合に，Closed な Issue 数や Label の付与率が他の尺度で選択したリポジトリ群よりも大きい．被依存数によって選定したリポジトリ群は，コントリビューター数で選定したリポジトリ群に比べて，Closed な Issue 数の中央値は 1/7 程度であり，Label の付与率も 10%以上低い．

## 6.3 リポジトリの選定基準の提供

異なる尺度で選定したリポジトリ群は必ずしも一致せずその管理の程度も異なることが示された．コントリビューター数で選定したリポジトリ群は他の尺度で選定されたリポジトリ群と比較してより Closed な Issue 数が多く Label の利用率が高かった．さらに，コントリビューター数と Closed な Issue 数との相関もある程度確認された．したがって，Issue の情報を多く必要とする研究にはコントリビューター数をリポジトリ選定の基準として採用することが有用であると考えられる．

# 7. 妥当性への脅威

## 7.1 外的妥当性

6.1.1 項では，各尺度で選定したリポジトリ群のみに含まれる特徴的と思われるリポジトリのグループのみについて記述したが，全てのリポジトリを確認したわけではない．したがって，各尺度で取得できるリポジトリ群の全てのリポジトリについて一般化できるとは限らない．

また，実験に利用するリポジトリを上位 500 件のみに制限したため，それよりも下位のリポジトリも含めた場合について本研究の結果の全てを一般化できない可能性がある．実験で使用した GitHub API のレート制限によってある個人が一定期間で取得できるリポジトリの情報が制限，全ての Issue の Label など長期にわたるデータの取得には非常に時間がかかる．したがって，本研究では時間の制約上，リポジトリの取得数は 500 件までとした．

## 7.2 内的妥当性

本研究で，データの取得のために作成したプログラムに不具合がある可能性があるが，いくつかのリポジトリについてプログラムから得られた結果と手動で取得した結果を照合することで不備がないことを確認した．

## 7.3 構成概念妥当性

本研究では，Closed な Issue と各メトリクスの相関を取得するためにコントリビューター数について Libraries.io によって取得されたデータを利用した．これに対して，Closed な Issue 数や Label の付けられた Closed な Issue 数, スター数, ウォッチ数, フォーク数については最新のデータを利用した．これは，Libraries.io の提供するデータ項目には Closed な Issue 数や Label の付けられた Closed な Issue 数が存在しなかったこと，および GitHub API の不具合によりコントリビューター数の正しい値が取得できないことを確認したためである．いくつかのリポジトリを目視で確認し最新のコントリビューター数と Libraries.io によって取得されたデータを比較したとこ

ろ，急激な変化は見られなかった．したがって，Libraries.io から取得できるコントリビューター数を実験 5 で利用することは妥当であると考えられる．

# 8. 関連研究

本研究でリポジトリの選定尺度として利用したメトリクスの性質について，Sheoran ら [15] は Watch を付けた人が Contributors とどのような関係を持つかについて調査しており，Watch を付けた人が Contributors になる割合はそれほど高くないことを示している．さらに，Watch を付与した人で Contributors になった人は他の Contributors に比べて長期に渡ってリポジトリに貢献することが示されている．Borges ら [14] によると，スター数の増加の仕方は広告などのマーケティング活動によって急激な伸び方をすることが示されている．さらに，リポジトリに Star を付与したからといって必ずしもそのリポジトリを使用しているわけではなく，ブックマークとして利用している人が 5 割程度であった．Borges ら [16] の別の研究では，本研究で使用したデータセットとは異なるデータセットでスター数とフォーク数には強い相関があること，およびスター数とコントリビューター数には弱い相関があると示されており，これらの事実は本研究の結果を支持するものである．

リポジトリやリポジトリ中のファイルの品質に関わる研究として，Wu ら [2] や Cito[1] ら，Cohen ら [17] ら，Jarczyk ら [18]，Guzman ら [3] の研究ではスター数をリポジトリを選定する際にリポジトリの人気の根拠として利用している．これらはリポジトリやリポジトリ中のファイルの品質について調査しており，Issue について研究した本研究と密接に関連している．いずれの研究も何らかのリポジトリ群を選定する際には，選定尺度を必要としており，実際に利用される尺度の一つにスター数が存在していることを示している．本研究では，これまで十分に調査されなかったメトリクスと Issue の関係についてスター数以外のメトリクスでも調査をした．

また，本研究では Label に注目したが，Gyimesi ら [19] は GitHub のソースコード中の不具合の特徴を示すため，バグに関する Label をデータの収集のために利用している．こうしたことから Label による分類がなされたリポジトリ群が必要となる場面が存在し，本研究ではその基準を提供できる．

# 9. 結言

本研究では，リポジトリの課題管理に焦点を当てた研究でリポジトリ群を選定する際の適当な選定基準を提供するために，Closed な Issue の数やその Label の付与率，各メトリクスとの Issue の相関関係について調査した．

調査の結果，コントリビューター数が多いリポジトリ群を選定対象とした場合，Closed な Issue の数が多く，さらに Label の付与率も他の尺度で選定したリポジトリに比べて高いことが示された．一方で，被依存数で選定したリポジトリ群は，Closed な Issue の数が少なく，さらに Label の付与率も他の尺度で選定したリポジトリに比べて低いことが示された．したがって，Issue を研究の対象にする場合に，十分なデータ数とその分類がされているデータが必要な場合には，コントリビューター数の多さで選択することが適当である．また，リポジトリ群を被依存数の多さで選択することは極力避けるべきである．

今後の課題としてはより正確に特定の研究内容に適したリポジトリを収集できるよう，Issue の分類のために使われる Label の種類を各尺度で調査することや，Issue の数に影響を与える因子を特定することなどが考えられる．

# 謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系水野修教授，崔恩瀞助教に厚く御礼申し上げます．本報告書執筆にあたり貴重な助言を多数頂きました，設計工学専攻西浦生成先輩，近藤将成先輩，本学情報工学専攻 國領正真先輩，塩津拓真先輩，脇上幸洋先輩，若林奎人先輩，舟山優先輩，情報工学課程 杉浦智基君，里形洋道君，上北裕也君をはじめとする，ソフトウェア工学研究室の皆さん，いつも研究室前モニターに黄色い数字を見せてくれたあくあたん，熱い漫画を世に送り出しモチベーションを高めて下さった集英社週刊少年ジャンプ作家のみなさん，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

# 参考文献


[1] J. Cito, G. Schermann, J.E. Wittern, P. Leitner, S. Zumberi, and H.C. Gall, “An empirical analysis of the docker container ecosystem on github,” 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pp.323–333, May 2017.

[2] Y. Wu, J. Kropczynski, R. Prates, and J. Carroll, “Understanding how github supports curation repositories,” Future Internet, vol.10, p.29, 03 2018.

[3] E. Guzman, D. Azócar, and Y. Li, “Sentiment analysis of commit comments in github: An empirical study,” Proceedings of the 11th Working Conference on Mining Software Repositories (MSR), p.352–355, MSR 2014, Association for Computing Machinery, New York, NY, USA, 2014.

[4] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, “Quality and productivity outcomes relating to continuous integration in github,” Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), p.805–816, ESEC/FSE 2015, Association for Computing Machinery, New York, NY, USA, 2015.

[5] Libraries.io, Libraries.io, Libraries.io（オンライン）, 入手先〈https://libraries.io/〉（参照 2020-02-07）.

[6] GitHub，Star を付けてリポジトリを保存する，GitHub（オンライン），入手先〈https://help.github.com/ja/github/getting-started-with-github/saving-repositories-with-stars#about-stars〉（参照 2020-02-07）.

[7] GitHub，通知について，GitHub（オンライン），入手先〈https://help.github.com/ja/github/receiving-notifications-about-activity-on-github/about-notifications〉（参照 2020-02-07）.

[8] GitHub，フォークについて，GitHub（オンライン），入手先〈https://help.github.com/ja/github/collaborating-with-issues-and-pull-requests/about-forks〉（参照 2020-02-07）.

[9] GitHub，Issue で作業を管理する，GitHub（オンライン），入手先

〈https://help.github.com/ja/github/managing-your-work-on-github/managing-your-work-with-issues〉（参照 2020-02-07）.

[10] GitHub，ラベルについて，GitHub（オンライン），入手先〈https://help.github.com/ja/github/managing-your-work-on-github/about-labels〉（参照 2020-02-07）.

[11] Z. Liao, D. He, Z. Chen, X. Fan, Y. Zhang, and S. Liu, "Exploring the characteristics of issue-related behaviors in github using visualization techniques," IEEE Access, vol.6, pp.24003–24015, 2018.

[12] Apache, Apache Maven Project, Apache Maven Project（オンライン）, 入手先〈https://maven.apache.org/〉（参照 2020-02-07）.

[13] AlDanial, AlDanial/cloc, GitHub（オンライン）, 入手先〈https://github.com/AlDanial/cloc〉（参照 2020-02-07）.

[14] H. Borges and M.T. Valente, "What ' s in a github star? understanding repository starring practices in a social coding platform," Journal of Systems and Software, vol.146, pp.112–129, 2018.

[15] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding " watchers " on github," Proceedings of the 11th Working Conference on Mining Software Repositories (MSR), p.336–339, MSR 2014, Association for Computing Machinery, New York, NY, USA, 2014.

[16] H. Borges, A. Hora, and M.T. Valente, "Understanding the factors that impact the popularity of github repositories," 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.334–344, Oct. 2016.

[17] E. Cohen and M.P. Consens, "Large-scale analysis of the co-commit patterns of the active developers in github ' s top repositories," Proceedings of the 15th International Conference on Mining Software Repositories (MSR), p.426–436, MSR ' 18, Association for Computing Machinery, New York, NY, USA, 2018.

[18] O. Jarczyk, Błażej Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, GitHub Projects. Quality Analysis of Open-Source Software, pp.80–94, Springer International

Publishing, Cham, 2014.

[19] P. Gyimesi, G. Gyimesi, Z. Tóth, and R. Ferenc, "Characterization of source code defects by data mining conducted on github," in Computational Science and Its Applications (ICCSA), eds. O. Gervasi, B. Murgante, S. Misra, M.L. Gavrilova, A.M.A.C. Rocha, C. Torre, D. Taniar, and B.O. Apduhan, pp.47–62, Springer International Publishing, Cham, 2015.