OSS 開発における長期貢献者の特徴理解のための注目度と活動期間の関係分析

福井克法[1]　築山文香[2]　大平雅雄[1]

概要：GitHub に代表されるソーシャルコーディングサイトでは，Follow や Watch によって開発者やコミュニティを「注目」することができる．本研究では，他の開発者からの「注目」により開発者の自発性が向上すると仮定し，結果的に「注目」が開発者の活動継続性に寄与している可能性について分析する．分析の結果，(1) 3 ヶ月以内に 100 以上の「注目」を獲得した開発者（1,537 人）はその後の活動期間が長いこと，(2) プルリクエストなど技術的な活動の活発よりも，「自身の注目 (Following)」が「他者からの注目 (Watch)」に繋がることが多いことがわかった．

キーワード
GitHub，OSS，ソーシャルコーディング，注目度，長期貢献者

1. はじめに

近年では，集中型バージョン管理システム (CVCS: Centralized Version Control System) を基盤システムとして発展してきた OSS コミュニティにおけるソフトウェア開発の潮流がソーシャルコーディング [2] という概念によって大きく変貌を遂げつつある．ソーシャルコーディングとは，開発者同士の社会的な繋がりを強化してソースコードの共有を加速させるための考え方である．ソーシャルコーディングでは，CVCS の利用による課題を克服するために分散バージョン管理システム (DVCS: Distributed Version Control System) が基盤システムとして採用される．また，Facebook や Twitter からインスピレーションを受けた様々な仕組みが用意されており，他の開発者の開発内容の進捗や他のコミュニティの運営状況を効率的に把握できるようになっている．

ソーシャルコーディングを容易に実施することを可能にしたエポックメイキングな Web サービス（本論文では，ソーシャルコーディングサイトと呼ぶ）が Github である．Github[*1]では，DVCS の Git[*2]をベースとして，2.2 節で詳述する Follow や Watch，Star などの機能群により他の開発者やコミュニティを容易に「注目する」ことができる．他の開発者やコミュニティに今現在起きていることが把握しやすい，すなわち，アウェアネスが以前の OSS 開発に比べて大幅に向上しているため，他の開発者やコミュニティが必要としている事柄に対して開発者がより積極的に貢献しやすくなると言われている．実際，他の開発者から非常に高い注目を集めている（「注目されている」）Rockstar[2] と呼ばれる開発者は，自身が活発であるだけでなく，Rockstar を「注目している」開発者の活動量もそれ以外の開発者に比べて高くなる傾向がある [8]．

本研究では，「注目する・される」ことが可能になるという以前の OSS コミュニティの運営にはなかった状況が，今後主流となることが確実視されているソーシャルコーディングを採用した OSS コミュニティの安定運営においての鍵になると考えている．従来の OSS コミュニティでは，報告した不具合が修正されたり投稿したパッチが管理者によってコミットされるという形でしか自身の貢献を明示的に感じることができなかったが，Github を代表とするソーシャルコーディングサイトでは，他の開発者からの「注目」は貢献が認められたという明示的な証になるため，開発者の自発性がより高まる可能性があるためである．

そこで本研究では，OSS コミュニティの安定運営を可能にするためのメカニズムの解明へ向けて，開発者への注目度が活動の継続性に与える影響を分析する．具体的には，ソーシャルコーディングサイト Github に登録されている 1,000 万人規模の開発者のデータを用いて，(1) 他の開発者から注目されることと開発者の活動継続期間にどのような関係があるのか，(2) どのような活動によって開発者は他の開発者から注目されるのかを分析により明らかにする．

[1] 和歌山大学システム工学部
[2] 和歌山大学大学院システム工学研究科
[*1] Github，https://github.com/
[*2] Git，https://git-scm.com/

Copyright (c) 2017 by the Information Processing Society of Japan

2. 分散型バージョン管理システムとソーシャルコーディング

本章ではまず，従来の集中型バージョン管理システム (CVCS: Centralized Version Control System) の問題点を克服し開発の自由度を高めることに成功した分散型バージョン管理システム（DVCS: Distributed Version Control System）の特徴を説明し，DVCS が開発者の長期的に貢献に寄与している可能性について議論する．次に，OSS コミュニティにおける開発者同士の繋がりを強化する仕組みとして近年主流となりつつあるソーシャルコーディングについて説明し，ソーシャルコーディングサイトの具体的事例として Github を紹介する．最後に，Github をはじめとするソーシャルコーディングサイトにおいて，OSS コミュニティにおける開発者の継続的貢献を促すメカニズムの解明へ向けて，本研究で行う分析の着眼点について述べる．

2.1 分散型バージョン管理システム

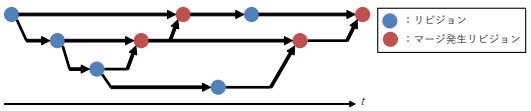

図 1 分散型バージョン管理システム (DVCS) における変更履歴の管理

集中型バージョン管理システムの CVCS の欠点を補うために近年主流となっているのが分散型バージョン管理システム (DVCS) である [1]．代表的な DVCS には，オープンソースの Git[*3]や Mercurial[*4] がある．

DVCS を用いたソフトウェア開発では，CVCS のような中央リポジトリは存在せず[*5]，開発者は個々人のローカル環境にリポジトリを持つ．すなわち，DVCS ではすべての開発者が自身のリポジトリに対するコミット権限を持つ．個々の開発者は他の開発者の最新の変更内容を取り込み（マージと呼ぶ）つつ，自身のローカルの環境で独自に開発を進めてリビジョンを管理する．そのため，それぞれの開発者が派生バージョン（ブランチと呼ぶ）を自身のローカルの環境で開発することになる．ただし，OSS コミュニティの成果物としては 1 つに集約する必要があるため，ブランチのマージが頻繁に発生する．そのため，図 1 に示すように，CVCS に比べるとリビジョンの構造が複雑になりがちである．

なお，一般開発者が変更内容をマスターリポジトリにマージしてもらうためには，CVCS の場合と同様，管理者の承認が必要である．一般開発者の貢献を反映するという作業プロセスは CVCS でも DVCS でもほぼ同様であるが，CVCS を用いた開発では個々の開発者が常に中央リポジトリの内容とローカル環境を同期させた状態で作業する必要があるため，自身が加えた変更は CVCS の中央リポジトリで反映してもらわなければ別の変更内容によって上書きされてしまうことになる．そのため，CVCS による従来の OSS 開発では，一般開発者が作成したパッチが中央リポジトリに反映されないという状況が長時間続くと，一般開発者にとっては開発効率が低下することになり，開発を継続するためのモチベーションを低下させる一因となっていた．

DVCS が現在主流となっている理由の 1 つは，CVCS による開発効率の低下を回避する手段として前述したようなブランチによる開発方法が行えるためである．DVCS を利用した開発では，マスターリポジトリとは独立して開発を進めることができるという大きなメリットがある．自身のブランチがすぐにマスターリポジトリにマージされる必要がないため，CVCS の利用によって発生していた前述のようなフラストレーションは感じにくい．結果的に，開発を継続するという観点では，DVCS を利用する最近の OSS コミュニティでは，CVCS を利用する従来の OSS コミュニティに比べて，一般開発者の離脱が起こりにくい環境になっている可能性がある．しかしながら，これまでの研究のほとんどは CVCS を利用した OSS コミュニティを分析の対象にしたものがほとんどであり，DVCS の利用による開発者の継続的貢献の効果については不明なままである．

2.2 ソーシャルコーディング

2.2.1 OSS 開発のためのコミュニティポータル

財団や寄付による支援を受けて運営している一部の人気コミュニティを除き，大多数の OSS コミュニティは従来，SourceForge[*6]などのポータルサイトを利用して OSS 開発を行ってきた．OSS 開発のためのコミュニティポータルでは，Subversion などの集中型バージョン管理システム，不具合管理システムやメーリングリストなど，OSS コミュニティの運営に必要となるシステム群を無償で利用できる．SourceForge には以前は 10 万件以上のコミュニティが登録されており，OSS 開発の主流として利用されてきた．

しかしながら，前述したように CVCS を用いた開発には開発効率の点において制約があるため，近年では DVCS を基盤システムとして採用した OSS コミュニティが主流になっている．特に，Github のようなソーシャルコーディングサイトの登場により，OSS コミュニティの多くはすでに，中央集権型の組織から個々人の自発性を重視した有機的組織として運営されている．

ソーシャルコーティングサイトとは，開発者同士の繋がりを強化する仕組みを取り入れた OSS 開発のための

*3 Git，https://git-scm.com/
*4 Mercurial，https://www.mercurial-scm.org/
*5 実際の運用においては，すべての開発者のリファレンスとなるマスターリポジトリが存在し，マスターリポジトリをローカル環境に複製（クローンと呼ぶ）してから開発を行うことになる．
*6 SouceForge，https://sourceforge.net/

Copyright (c) 2017 by the Information Processing Society of Japan

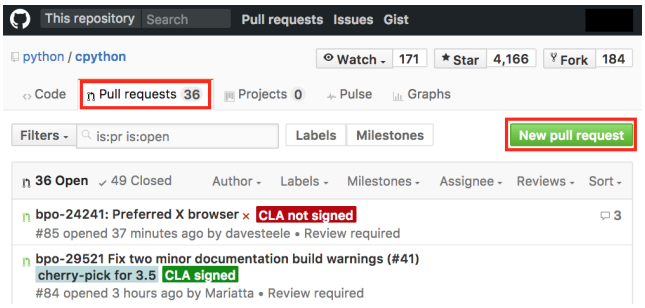

図 2　Github におけるプルリクエストの画面例

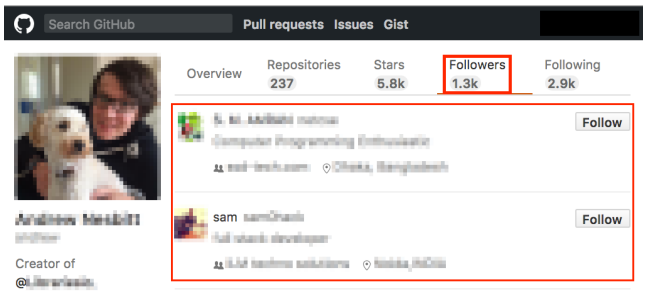

図 3　GitHub の Follower ページ例

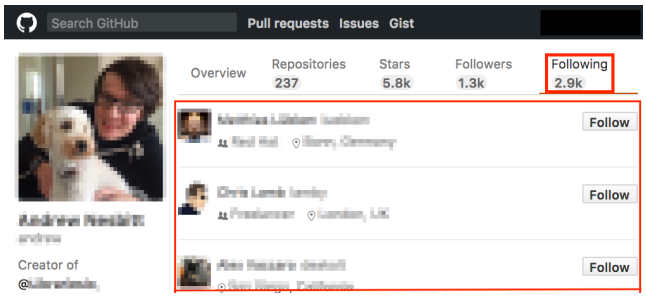

図 4　GitHub の Follow ページ例

新たなコミュニティポータルである．Github の他には，Bitbucket[*7]や GitLab[*8]などがある．次項では，ソーシャルコーティングサイトの特徴を説明するために，具体的な事例として Github を取り上げる．

2.2.2　Github におけるソーシャルコーディング

Github をはじめとするソーシャルコーディングサイトでは，バージョン管理システムとして Git や Mercurial を採用し開発の自由度を高めている．また，開発者同士の活発な交流を通じたソフトウェア開発，すなわち，ソーシャルコーディングと呼ばれる所以となっている以下のような機能を提供することにより，開発効率の向上および開発者同士の社会的繋がりの強化を図っている．

- プルリクエスト
- ソーシャルネットワーキング
- アウェアネス

以下ではそれぞれの機能について詳しく説明する．

プルリクエスト

プルリクエストとは，個々の開発者が自身のローカル環境（ブランチ）で進めた開発内容（変更内容）をマスターブランチに取り込んでもらうための依頼である．従来はパッチとして変更差分をまとめた上でメーリングリストに投稿するなどの手間が必要であったが，Github 等のソーシャルコーディングサイトでは図 2 右上のボタンから数クリックで変更内容のマージを依頼することができる．

登録されたプルリクエストは課題 (Issue) として課題管理システムで管理され，プルリクエストの内容の不備や改善点などについて，マスターリポジトリの管理者と開発者が直接議論できる．プルリクエストの内容に問題がなければ管理者によりマスターリポジトリにマージされる．たとえプルリクエストが承認されなくても，開発者のローカル環境ではブランチとして開発をそのまま継続することができるため変更内容が無駄になることはない．このように，パッチ方式よりも手軽に変更内容を申請できフィードバックが得やすく，さらに，変更内容がマージされなくとも開発を継続することができることから，以前よりも開発者の自発的な貢献が促進されるとされている．プルリクエスト方式のソフトウェア開発は，ソーシャルコーディングを実現するための重要な要素の 1 つとなっている．

ソーシャルネットワーキング

Github をはじめとするソーシャルコーディングサイトでは，Facebook や Twitter のようなソーシャルネットワーキングサービス (SNS) を参考にした，開発者同士の社会的繋がりを促進するための様々な機能（ここでは，SNS 機能と呼ぶ）が提供されている．以下に，Github における SNS 機能を紹介する．

自分の興味や関心に合った活動を行なっている注目すべき他の開発者がいれば，Follow [*9]することができる．また，自分が注目されている（Follow されている）こともでき，図 4 および図 3 に示したように，自身のアカウントのプロフィール欄から Follow/Follower の一覧を確認することができる．

アウェアネス

SNS 機能に加え，個人やコミュニティ（リポジトリ）の開発状況を素早く把握できるようにするための機能として，Star および Watch という機能が提供されている．Star はブックマークとしての機能しかないが，Star が多く付けられているリポジトリは現在多くの開発者から注目されていることを意味する．一方 Watch は，Watch したリポジトリに変更が加えられたなどのイベントが発生する度に開発者にメールで通知を送るための機能である．注目（Follow）している開発者の

*7　Bitbucket，https://bitbucket.org/
*8　GitLab，https://gitlab.com/
*9　Twitter の Follow 機能と同様のものであり，Facebook の友達申請の様に承認の必要はない．

Copyright (c) 2017 by the Information Processing Society of Japan

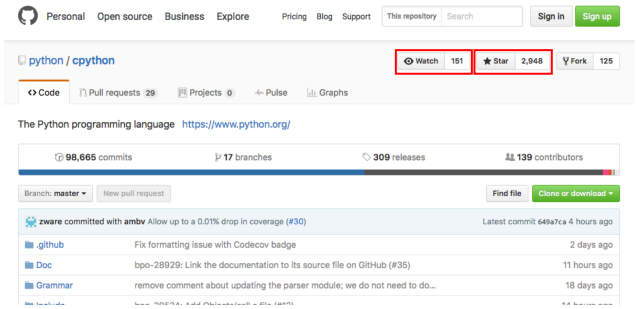

図 5 Github における Watch と Star の画面例

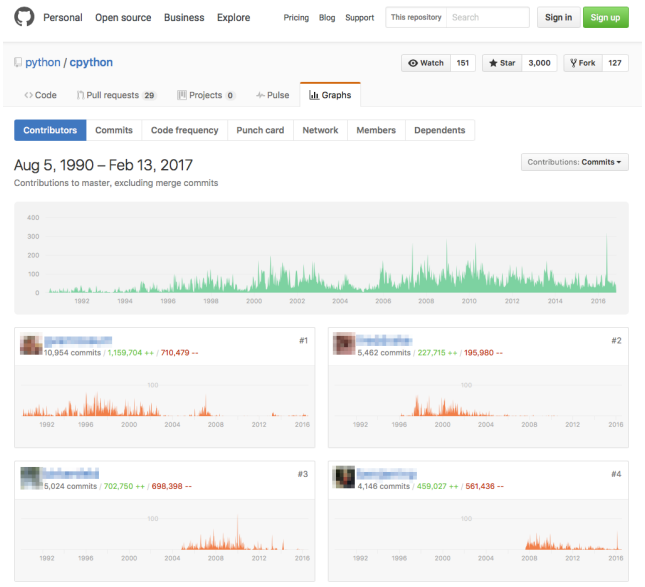

図 6 Github における活動履歴の画面例

プロファイルページからは，Follow している開発者が注目（Watch/Star）しているリポジトリを確認できる．図 5 は，Github のリポジトリの Watch（赤枠左）の数と Star（赤枠右）の数が表示されている画面の例である．それぞれのボタンを押下すると，該当リポジトリが誰から Watch/Star されているのかを確認することができる．さらに，図 6 に示すように，注目しているリポジトリ内の活動の履歴も閲覧できる．

このように，注目する開発者やリポジトリの開発状況を容易に把握するための機能が提供されておりアウェアネスが向上するため，どの開発者 b のどのリポジトリにどのような貢献が望まれているのかをタイムリーに知ることができる．結果的にこれらの機能は，開発者の自発的な貢献を促す仕組みとしても重要な役割を果たしていると考えられる．

2.3 本研究の着眼点

Zhou らの研究 [10] は長期的に活動する OSS 開発者の特徴を，(1) 自発性 (Willingness) と (2) コミュニティの巨視的・微視的雰囲気（Micro/Micro Climate）の観点から分析し，(1) により活動初期に自ら報告した不具合が修正されることが長期的活動に寄与することを明らかにしている．また反対に，(2) により人気の高いプロジェクトへの参加 (Micro Climate) やレスポンスの早すぎるコミュニケーション (Micro Climate) は長期的活動を阻害する要因となることを明らかにしている．Zhou らの研究は，CVCS として Subversion を利用していた時代の Mozilla と Gnome プロジェクトを対象としたものである．したがって，前述したように，DVCS を採用したソーシャルコーディングサイトが主流となっている現在の OSS 開発では，開発者の自発性が促される状況やコミュニティの雰囲気に対する開発者の捉え方が大きく異なっている可能性がある．

ソーシャルコーディングサイトでは，Follow や Watch，Star といった「注目する」ための機能により，開発者はより能動的に他の開発者やリポジトリの状況を把握できる．また，同時に，自身あるいは自身のリポジトリが「注目されている」という状況もリアルタイムに知ることができる．先行研究では分析の対象外であるため，ソーシャルコーディングが主流となっている現在，開発者が「注目する・される」状況と OSS コミュニティへの長期的貢献との関係を分析することは，開発者が早期に離脱する前に長期貢献者となりうる候補者を予測したり，早期離脱を防ぐための施策の策定を行うための有益な知見を提供できる可能性がある．

次章では，開発者の注目度に着目して本研究が行うソーシャルコーディングサイトにおける長期開発者の分析について述べる．

3. 開発者の注目度と活動期間の関係分析

本章では，開発者の注目度と活動期間の関係を分析する目的について述べるとともに，分析の前提となるベースモデルについて説明する．また，具体的な分析に際して設定するリサーチクエスチョンについて説明する．

3.1 目的

本研究の目的は，近年主流となっているソーシャルコーディングサイトにおいて開発者の「注目度」が OSS コミュニティへの長期的貢献にどのように寄与するのかを明らかにすることである．DVCS の利用によるプルリクエストを主体とした開発方法がソフトウェア開発の実践にもたらす影響については，管理者と開発者（貢献者）のそれぞれの観点から分析 [3], [4] が行われている．しかしながら，Github のようなソーシャルコーディングサイトにおいてプルリクエストの他に特徴的な SNS 機能やアウェアネスの向上がどのような影響を与えるのかについてはほとんどまだ研究が行われていない．特に，本研究が着目する「注目する・される」状況が開発者の長期的活動にどのような関係にあるのかを明らかにすることができれば，開発者が早期に離脱する前に長期貢献者となりうる候補者を予測したり，早

Copyright (c) 2017 by the Information Processing Society of Japan

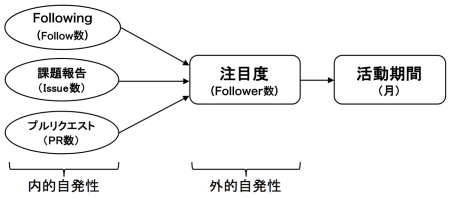

図 7 本研究における分析モデル

期離脱を防ぐための施策の策定を行うための有益な知見を提供できる可能性がある．そこで本研究では，いくつかのリサーチクエスチョンを設定し，開発者の「注目度」と「活動期間」の関係を明らかにすることを試みる．次節では，リサーチクエスチョンを設定しケーススタディを実施するための前提なる本研究の分析モデルについて述べる．

3.2 分析のためのモデル

図 7 は，本研究が前提としている分析のためのベースモデルである．本モデルでは，OSS コミュニティへの長期貢献を支える開発者の自発性を，大きく内的自発性と外的自発性の 2 種類に分けて考える．Facebook や Twitter の成功は主に，人間の自己承認要求を容易に満たすことを可能にした「いいね」機能の導入によるところが大きい．これは，投稿を行うユーザの外的な自発性を促進するための仕組みであると言える．Facebook や Twitter での投稿が「長続き」するためには，ある一定数の「いいね」を他者から獲得する必要があるため，「いいね」と「活動の継続性」には関連があるものと思われる．

一方，これら SNS を参考に発展したソーシャルコーディングサイトでは，どれくらい Follower がいるか（Follower 数），あるいは，自らのリポジトリがどれくらい Watch/Star されているかが「いいね」に相当する．本研究は特に開発者個人の特性に絞って分析を行いたいため，前者の Follower 数を「注目度」と解釈して「活動の期間」との関係を調査するが，後者の自身のリポジトリに対する Watch/Star も重要な因子であることが予想されるため，本研究の今後の課題する．

また， 外的自発性としてどれくらいの開発者に「注目」されていることを調べるだけでは不十分である．OSS 開発においてコミュニティに参加しているだけで他の開発者から注目されることはまず起こり得ないためである．そこで，コミュニティ内での何らかの活動が他の開発者から「注目」される理由となっていると想定した．特に，前章で述べたように，ソーシャルコーディングサイトは開発者の内的自発性を促進するいくつもの仕組みが採用されている．本研究ではまず，以下の 3 つの活動を内的自発性による活動と定義して，外的自発性，すなわち「注目度」の獲得に寄与する因子として分析を行う．

- Following：開発者が自発的に「注目した」他の開発者の数 （Follower の数）である．Marlow らの研究 [7] で，Web での開発において Follow された時に相手のプロフィールを見にいく場合があると示していた．Github に参加した開発者の中でも他の開発者の活動状況の把握を重視している開発者は，他の開発者の開発方法やコードの内容を参考にして今後も積極的にコミュニティに関与する意図があるものと考えられる．したがって，Follower の多い開発者はより効率的にコミュニティとの関わりを持てるようになる可能性が高いため，他の開発者から「注目される」ことにも繋がりやすいと思われる．
- 課題報告：Zhou らの研究 [10] でもコミュニティ参加初期の不具合報告は開発者の自主性を示すものとされている．OSS コミュニティの参加初期にはコード寄贈は技術レベルの高さから行えないことが多い．多くの開発者は通常，不具合報告を始めることによりコミュニティへの関わりを持つ．したがって，活動初期の課題報告数は直接的・間接的に長期的活動と関係している可能性がある．
- プルリクエスト：プルリクエストはマスターブランチへ自身の変更内容を反映してもらいたいという内的な自発性により提案されるものと考えるのが自然である．課題報告よりも高い技術レベルを求められるものの，開発者の直接的な貢献として他者から「注目される」理由として最も可能性の高い活動である．プルリクエストの数は内的自発性の高さとして解釈でき活動期間と直接関係がある可能性もあるが，本研究では単純化のために「注目度」に寄与する因子として定義する．

なお，本研究における分析モデルは，あくまでの分析の基本的枠組みを与えることが狙いであり，分析結果を踏まえてモデルを修正することもありえる．

3.3 リサーチクエスチョン

本研究では，Github を対象としたケーススタディを行う．ケーススタディに際して以下の 4 つのリサーチクエスチョンを設定し，開発者の注目度と活動期間の関係を明らかにする．

3.3.1 RQ1: 注目度の高い開発者（Rockstar）と一般開発者の平均活動期間に違いはあるか？

図 7 に示した分析モデルのコアとなる注目度と開発期間との関係について，まず，Rockstar と呼ばれる注目度が高い開発者 [2] と一般開発者の平均活動期間の違いについて調べ基本的な前提知見を得る．Rockstar とは 1,000 人以上の Follower を持つ極めて注目されている開発者である．Github には 1,000 万人以上のユーザが登録されているが，2016 年 2 月現在，Rockstar の数は 516 人であり極めて少数の存在である．したがって，Rockstar の平均活動期間と一般開発者の平均活動期間には大きな違いがあるはずであ

Copyright (c) 2017 by the Information Processing Society of Japan

る．Rockstar の平均活動期間と一般開発者の平均活動期間には大きな違いがない場合は，本研究の分析モデルの前提が成り立たないため，まず RQ1 について調査する．

3.3.2 RQ2: 注目度の高さと平均活動期間に関係はあるか？

RQ1 では，極めて少数かつ注目度の高い Rockstar とランダムに抽出した一般開発者との平均活動期間の違いを調査する．しかしながら，1,000 万人以上のユーザからランダムに抽出した一般開発者のほとんどはほとんど何も活動していない可能性が高く [9]，Rockstar と一般開発者の平均参加期間に大きな違いが見られるのは当然である．そこで RQ2 では，100 人以上の Follower を持つ開発者グループ（上位群）と，100 人未満の Follower 数を持つ開発者を注目度の高い順に並べて上位群と同数になるように構成した開発者グループ（中位群）とを比較することで，中位群よりも注目度の高い上位群の平均参加期間に違いがあるかどうかを調査する．RQ2 の調査により，上位群の方が中位群よりも平均参加期間が長ければ，外的自発性としての注目度が参加期間と関係があると言える．

3.3.3 RQ3: 注目は活動前期に多く獲得するのか？活動後期に多く獲得するのか？

RQ2 により注目度が高い方が活動期間が長くなる傾向を確認することができたとしても，活動期間の長さが注目度の高さに寄与するのであれば本研究の分析モデルの前提が成り立たない．すなわち，本研究の分析モデルでは，注目度が活動期間に寄与するという因果関係を想定しており，活動期間が注目度に寄与するという因果関係は想定していない．しかしながら，活動期間が長く多くの貢献を行った開発者に注目が集まるということも十分可能性としては考えられるため，注目度と活動期間という 2 変数の因果関係を明らかにしておく必要がある．そこで RQ3 では，一般開発者よりも長い間活動を継続している開発者を抽出し，他の開発者からの注目を活動初期に獲得したのか，あるいは，活動後期に獲得したのかについて分析する．前者であれば，活動初期に外的自発性が刺激された結果，活動期間が長くなるという解釈が成り立つ．一方，後者であれば，本研究の分析モデルを見直す必要がある．

3.3.4 RQ4: 高い注目を集める活動は何か？

RQ3 により活動期間を長くする要因が活動初期に集めた注目であることがわかれば，注目されるに至った内的自発性に基づく開発者の活動について調べる意味がある．本研究の分析モデルでは，Following（Follow した人数），課題報告（Issue 数），プルリクエスト（PR 数）の 3 種類を少なくとも注目を集めた要因として想定している．これらの 3 つの要因を調査して，注目度を集めるのに寄与した要因はいずれのものか，また，それらの要因は RQ2 における上位群と下位群とで違いがないかについて調査する．RQ4 で特定の要因が活動期間と最も深く関係していることが明らかになれば，一般開発者の活動初期にそれらの要因が突出している者がいれば将来長期的に貢献を行うことのできる資質を備えた開発者を発見することが可能になると考えられる．OSS コミュニティとしては，そのような開発者が早期離脱しないように早めに管理者権限を与えるなどの措置を講じることができるようになるため，本研究の成果はコミュニティの安定的な運営を支援するための有益な知見となる可能性がある．

4. ケーススタディ

本章では，3 章で設定したリサーチクエスチョンに答えるためのケーススタディを行う．ケーススタディでは，GHTorrent[*10][6] の 2016 年 11 月 1 日に公開されたデータ（2016 年 10 月までとする）を使用する．2016 年 10 月現在ユーザ登録総数はすでに削除されたものも含めて 14,380,149 人，Following 総数は全部で 11,616,754 人である．

4.1 RQ1: 注目度の高い開発者（Rockstar）と一般開発者の平均活動期間に違いはあるか？

4.1.1 分析方法

注目度の高い開発者（Rockstar）と一般開発者の平均活動期間を算出する．本研究では注目度を Follower 数でカウントする．活動期間は，課題報告・プルリクエスト・コミットを抽出する．統計的に集団の平均活動期間に対して差があるかどうかを判断するのに外れ値に強いマン・ホイットニーの U 検定（有意水準 5%）を行う．

対象開発者

まず，開発者の Follower 数を注目度として Rockstar と一般開発者に分類する．

Rockstar ：Rockstar は Follower 数が 1,000 人以上いる開発者である [2]．GHTorrent のユーザ情報には一般ユーザと組織ユーザが存在する．2016 年 2 月現在，フォロワー数が 1,000 人以上存在するユーザは 522 人であったがそのうち 6 件は jQuery や Ruby on Rails といった著名な組織ユーザであった．組織ユーザは個人ではなく組織としてリポジトリを管理するためのものであるが，一般ユーザとは使える機能が異なる．今回はこれらの組織ユーザを除いた 516 人を Rockstar とした．

一般開発者 ：Rockstar に対して一般開発者を以下のように定義する．

(1) Follower 数が 1,000 人に満たないユーザ
(2) 組織ユーザではない一般ユーザ
(3) 最低でも 1 人のフォロワーがいる開発者ユーザ
GitHub にアカウント登録はしているがほとんど活動していないユーザが多いためである．

[*10] http://ghtorrent.org/

Copyright (c) 2017 by the Information Processing Society of Japan

表 1 活動期間を決めるメトリクス

メトリクス	説明
課題報告	自分または他人が所有するリポジトリに不具合などの課題を報告する
プルリクエスト	自分または他人が所有するリポジトリに不具合修正などのソースコードを提出する
コミット	自分のリポジトリに変更を反映するまたは他人が所有するリポジトリに変更が反映される

表 2 RQ1 対象開発者

区分	ユーザ登録した期間	人数	Follower 獲得数		
			最小値	中央値	最大値
Rockstar	2007/10〜2016/02	516 人	1,001	1,677.5	39,821
一般開発者	2008/01〜2013/05	939,705 人	1	2	999

表 3 RQ1 対象開発者の活動期間のデータ

対象開発者	取得した期間	メトリクス	総数	最小値	中央値	最大値
Rockstar（516 人）	2008/01〜2016/10	課題報告	140,168 件	0 件	94 件	4,588 件
		プルリクエスト	88,996 件	0 件	46 件	3,803 件
		コミット	2,901,321 件	0 件	3,108.5 件	91,873 件
		合計	3,130,485 件	0 件	3,365 件	91,874 件
一般開発者（516 人抽出後）	2008/01〜2016/10	課題報告	4,120 件	0 件	0 件	554 件
		プルリクエスト	2,978 件	0 件	0 件	527 件
		コミット	93,278 件	0 件	23 件	7,828 件
		合計	100,376 件	0 件	25 件	8,517 件

実際に 2016 年 2 月現在，組織ユーザを除く 10,955,757 人のうち Follower が 1 人以上いるユーザは 1,789,742 人である

(4) 2013 年 5 月までに登録したユーザ
登録直後など急に 1,000 人もの Follower を獲得するのは困難だと考えられるため，Rockstar で “ない” と判断するために Rockstar である 516 人の開発者が 1,000 人の Follower を獲得するまでの期間の中央値である 32 ヶ月（2 年 8 ヶ月）を 2016 年 2 月から逆算した．

上記の 4 つの条件に当てはまった 939,705 人のユーザを一般開発者とし，この中から Rockstar と同数の 516 人をランダムに抽出する．

活動期間

活動期間を決めるのに取得するメトリクスと活動期間の条件を示す．

取得するメトリクス ：OSS 開発に対する直接的な貢献活動であると考えられる課題報告・プルリクエスト・コミットの 3 つの活動を取得する．開発者が期間中（ユーザ登録した日から，2016 年 10 月まで）にプロジェクトのオーナーになっている可能性も考えられるため，活動期間で取得する各メトリクスは自分のリポジトリに対する活動も対象とする．

活動期間の条件

3 ヶ月間，上記 3 つの活動のいずれかを 1 回もしていないと活動を停止したとみなす．活動期間が複数ある場合は，より長く活動した方をその開発者の活動期間とする．

データセット

表 2 と表 3 に対象開発者の基本的なデータを示す．

4.1.2 結果

結果を表 4 に示す．平均活動期間について，一般開発者と Rockstar の間に有意な差があった．また，活動期間についても Rockstar の中央値が 66 ヶ月，一般開発者の中央値が 4 ヶ月であることから Rockstar の方が活動期間が長いと言える．

表 4 Rockstar と一般開発者の注目度と活動期間の関係

区分	活動期間の中央値	有意差	p 値
Rockstar	66 ヶ月	あり	> 0.01
一般開発者	4 ヶ月		

表 5 RQ2 対象開発者

Follower を獲得するまでの期間（月）	ユーザ登録した期間	開発者区分	対象開発者	Follower 獲得数		
				最小値	中央値	最大値
1 ヶ月	2008/01〜2016/01	上位群	1,246 人	100	161	9,061
		中位群	1,481 人	60	74	99
3 ヶ月	2008/01〜2015/11	上位群	1,529 人	100	157	9,957
		中位群	1,722 人	60	74	99
12 ヶ月	2008/01〜2015/02	上位群	2,857 人	100	160	16,900
		中位群	3,161 人	60	73	99

表 6 RQ2 対象開発者の活動期間のデータ

Follower を獲得するまでの期間（月）	対象開発者	取得した期間	メトリクス	総数	最小値	中央値	最大値
1 ヶ月	上位群（1,246 人）	2008/01〜2016/10	課題報告	277,737 件	0 件	34 件	80,786 件
			プルリクエスト	147,645 件	0 件	20 件	3,803 件
			コミット	3,908,240 件	0 件	1,342 件	91,873 件
			合計	4,333,622 件	0 件	1,438 件	91,874 件
	中位群（1,481 人）	2008/01〜2016/10	課題報告	192,245 件	0 件	38 件	3,368 件
			プルリクエスト	149,869 件	0 件	21 件	2,850 件
			コミット	3,492,978 件	0 件	1,065 件	62,513 件
			合計	3,835,092 件	0 件	1,162 件	64,143 件
3 ヶ月	上位群（1,529 人）	2008/01〜2016/10	課題報告	322,322 件	0 件	38 件	80,786 件
			プルリクエスト	176,512 件	0 件	22 件	3,803 件
			コミット	4,741,262 件	0 件	1,361 件	91,873 件
			合計	5,240,096 件	0 件	1,455 件	91,874 件
	中位群（1,722 人）	2008/01〜2016/10	課題報告	220,891 件	0 件	37 件	4,588 件
			プルリクエスト	176,799 件	0 件	23 件	2,850 件
			コミット	4,206,553 件	0 件	1,056.5 件	65,743 件
			合計	4,604,243 件	0 件	1,146.5 件	65,768 件
12 ヶ月	上位群（2,857 人）	2008/01〜2016/10	課題報告	519,564 件	0 件	43 件	80,786 件
			プルリクエスト	326,608 件	0 件	26 件	3,803 件
			コミット	8,825,124 件	0 件	1,381 件	91,873 件
			合計	9,671,296 件	0 件	1,496 件	91,874
	中位群（3,161 人）	2008/01〜2016/10	課題報告	355,505 件	0 件	32 件	5,470 件
			プルリクエスト	284,985 件	0 件	23 件	2,561 件
			コミット	7,064,169 件	0 件	985 件	232,531 件
			合計	7,704,659 件	0 件	1,092 件	233,123 件

4.2 RQ2: 注目度の高さと平均活動期間に関係はあるか？

4.2.1 分析方法

RQ1 に対して，対象開発者を Follower 数を指定した期間内に 100 人以上獲得した開発者とそうでない開発者に分類し活動期間に差があるかどうかを検定する．活動期間および検定については RQ1 に従う．

対象開発者

本研究では 100 人の Follower を獲得するまでの期間を 1 ヶ月，3 ヶ月，12 ヶ月に分けそれぞれの期間で獲得できたグループと獲得できなかったグループに分類する．Follower 獲得期間はユーザが GitHub にアカウント登録してから，2016 年 2 月までのデータから，それぞれの期間を逆算する（登録日時から最低限それぞれの期間を確保するため）．また獲得できなかったグループは，上限を 99 とし，下限を Follower 数を 100 人獲得できたグループと同程度になるように設定する

Copyright (c) 2017 by the Information Processing Society of Japan

表 7　開発者の注目度と活動期間の関係

期間（t）	開発者区分	活動期間の中央値	有意差	p 値
1 ヶ月	中位群	37 ヶ月	なし	0.2629
	上位群	39 ヶ月		
3 ヶ月	中位群	**11 ヶ月**	**あり**	＞ 0.01
	上位群	**40 ヶ月**		
12 ヶ月	中位群	**38 ヶ月**	**あり**	＞ 0.01
	上位群	**44 ヶ月**		

と 60 になった．

データセット

表 5 と表 6 に対象開発者の基本的なデータを示す．

4.2.2　結果

結果を表 7 に示す．3 ヶ月に 100 人以上の Follower 数を獲得した開発者とそうでない開発者，12 ヶ月に 100 人以上の Follower 数を獲得した開発者とそうでない開発者間に有意な差があった．

4.3　RQ3: 注目は活動前期に多く獲得するのか？活動後期に多く獲得するのか？

4.3.1　分析方法

RQ2 で有意差があり，活動期間の差が大きかった 3 ヶ月以内に 100 以上のフォロワーを獲得した開発者群に対して，注目度を活動前期に獲得したのか活動後期に獲得したのかを調査する．

対象

RQ2 で有意差があった開発者群のうち，100 以上の Follower を獲得した開発者を対象とする．複数あった場合は，平均活動期間に最も差が多い開発者群を一意に決める．

分類方法

対象開発者の期間内（ユーザ登録月から 2016 年 2 月まで）の Follower 獲得数をそれぞれ 1 として月毎に積み上げグラフを作成する．期間を 2 分割して，50%の Follower を前半で獲得するか後半で獲得するかで判断する．

データセット

RQ2 の表 5 のうち，3 ヶ月で 100 以上の Follower を獲得した開発者群（上位群）に従う．

4.3.2　結果

RQ2 の結果から，有意差がありかつ平均活動期間の差が大きかった 3 ヶ月以内に 100 人以上の Follower 数を獲得した開発者群の上位群 1,529 人に対して分類を行った．結果を表 8 に示す．ほとんどの開発者（97.8%）が登録してからの期間のうち前期に Follower の半数を獲得していることがわかった．図 8 に前期に Follower を多く獲得する開発者の例を，図 9 後期に Follower を多く獲得する例を示す．

表 8　Follower を多く獲得する時期

時期	人数
前期	1,496 人（97.8%）
後期	33 人（2.2%）
合計	1,529 人（100%）

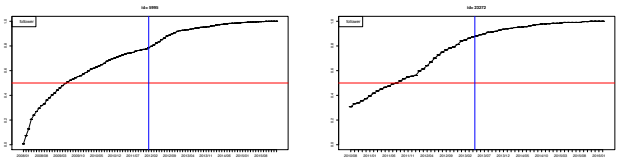

図 8　前期に Follower を多く獲得する開発者の例

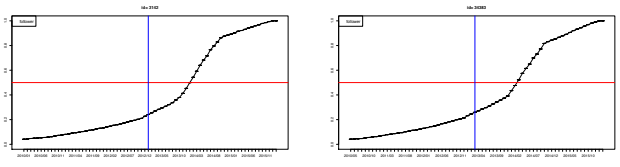

図 9　後期に Follower を多く獲得する開発者の例

表 9　開発者が注目される理由として取得するメトリクス

メトリクス	説明
Following	他の開発者を Follow する
課題報告	他人が所有するリポジトリに不具合などの課題を報告する
プルリクエスト	他人が所有するリポジトリに不具合修正などのソースコードを提出する

4.4　RQ4: 高い注目を集める活動は何か？

4.4.1　分析方法

RQ2 で有意差があり，活動期間の差が大きかった 3 ヶ月以内に 100 以上のフォロワーを獲得した開発者群と獲得できなかった開発者群を対象に，注目度を獲得する前にどのような活動を行なっていたかを調査する．

取得するメトリクス

分析モデルに基づいて内的自発性に対応するメトリクスを取得する．本ケーススタディでは，Following，課題報告，プルリクエストとする．表 9 に取得するメトリクスを示す．

分類方法

Follower 獲得，Following，課題報告，プルリクエストの活動に対して登録してからの活動期間（最大で 2008 年 1 月から 2016 年 2 月まで）のそれぞれの全数を 1 として積み上げグラフにし，Follower 獲得より前の段階で割合が大きくなっている活動を Follower 獲得の要因としてカウントした．複数ある場合は，より多くの割合が先行している活動をその開発者が Follower 獲得の主な原因とし一意に決める．

データセット

表 10 に対象開発者の基本的なデータを示す．

4.4.2　結果

RQ3 と同様，RQ2 の結果から，3 ヶ月以内に 100 人以上の Follower を獲得した開発者群 1,529 人と，獲得できなかった開発者群 1,722 人を対象にした．結果を表 11 に

Copyright (c) 2017 by the Information Processing Society of Japan

表 10　対象開発者の活動期間のデータ

Follower を獲得するまでの期間（月）	対象開発者	取得した期間	メトリクス	総数	最小値	中央値	最大値
3 ヶ月	上位群（1,529 人）	2008/01〜2016/02	Follower 獲得	1,644,200 件	104 件	490 件	39,821 件
			Following	911,757 件	0 件	15 件	164,129 件
			課題報告	235,056 件	0 件	16 件	79,127 件
			プルリクエスト	147,305 件	0 件	18 件	3,239 件
	中位群（1,722 人）	2008/01〜2016/02	Follower 獲得	517,071 件	60 件	198.5 件	6,839 件
			Following	163,614 件	0 件	14 件	12,816 件
			課題報告	141,806 件	0 件	17 件	3,779 件
			プルリクエスト	147,006 件	0 件	19 件	2,626 件

表 11　開発者が注目される理由

Follower を獲得した要因	60〜99（中位群）	100 以上（上位群）
Following	1,084 人（62.9%）	942 人（61.3%）
課題報告	26 人（1.5%）	33 人（2.1%）
プルリクエスト	38 人（2.2%）	28 人（1.8%）
その他	574 人（33.3%）	526 人（34.4%）
合計	1,722 人（100%）	1,529 人（100%）

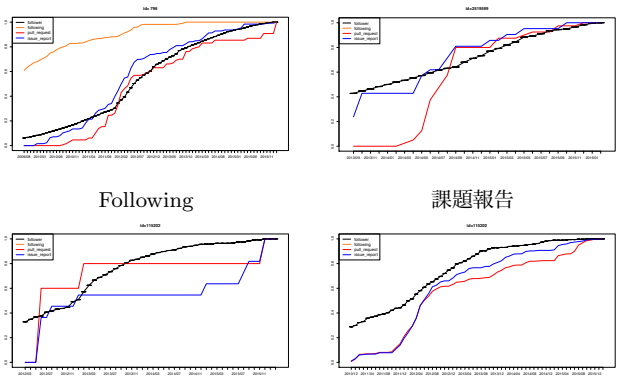

図 10　上位群例

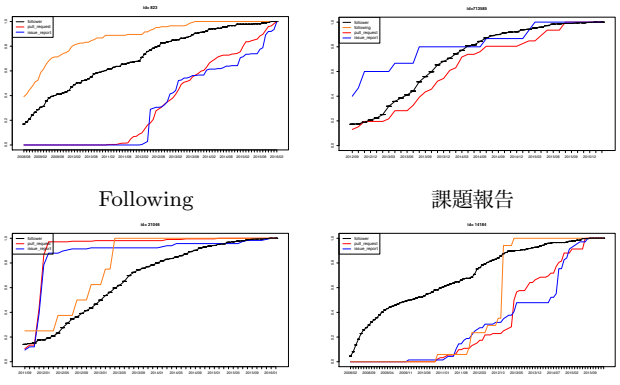

図 11　中位群例

示す．

3 ヶ月以内に 100 人以上の Follower 数を獲得した開発者群と獲得できなかった開発者群で分布に大きな差を見つけることができなかった．また，その他には Follower 獲得が一番先であった開発者の他に，今回取得した 3 つの要因を全く行っていなかった開発者（それぞれ中位群に 168 人（9.7%），上位群に 176 人（11.5%））を含む．図 10 に上位群の活動の時系列の一部の具体例を，図 11 に中位群の活動の時系列の一部の具体例を示す．

5. 考察

本研究では，ソーシャルコーディングによって OSS 開発における開発者のモチベーションがどう変化し，開発者の活動期間にどのような影響を与えるのか理解するために，外的自発性，内的自発性の観点から分析モデルを構築・分析した．考察を以下に述べた後，本研究の制約について述べる．

5.1 RQ1

注目度の高い開発者（Rockstar）と一般開発者の間に有意な差があり，平均活動期間も Rockstar が 66 ヶ月，一般開発者が 4 ヶ月と注目度の高い開発者（Rockstar）がより長い期間活動していることを示した．このことから，注目度が高い開発者は一般開発者より平均活動期間が長いことを示した．しかし，注目度が高いと活動期間が長いことを示しただけで，活動期間が長いため注目度を獲得したという逆の関係も成り立つ可能性がある．RQ1 では，本研究の前提として注目度が高い開発者は活動期間が長いということを示した．

5.2 RQ2

RQ2 では，注目度が高いと活動期間が長いという RQ1 の前提から，短期間に注目度を獲得した場合の結果について検証した．3 ヶ月以内に 100 以上の Follower を獲得した開発者と 12 ヶ月以内に 100 以上の Follower 数を獲得した開発者はそうでない開発者と比べて有意な差があった．また，その後の活動期間について短期間により多くの注目度（Follower 数）を獲得した開発者群の方が大きいことを示した．このことから短期間でより多くの注目度を獲得した開発者は活動期間が長くなるということを示した．

5.3 RQ3

RQ2 では短期間で 100 以上の注目度を獲得した開発者はその後の平均活動期間が長いことを示した．しかし，条件に当てはまる開発者の中には RQ1 で対象にした Rockstar などの非常に注目度の高い開発者も存在する可能性がある．これでは，注目度が高いため活動期間が長くなるという因果関係を十分に示せたとは言えない．Rockstar の平均活動期間が 66 ヶ月であることから，活動期間が長いため注目度が高くなったという逆の関係もあり得る．当然活動期間が長いために注目度を獲得した開発者も存在する可能性はあるが，本リサーチクエスチョンでは RQ3 で対象にした開発者の上位群に対して，3 章で提示した分析モデルが成り立つかどうかを検証した．

結果はほとんどの開発者が開発期間の前半に半数以上の Follower を獲得していた．このことから活動期間が長いた

Copyright (c) 2017 by the Information Processing Society of Japan

めたくさんの Follower 数を獲得したという逆の関係を否定し，注目度を獲得したために，活動期間が長いということを示したと考える．

5.4 RQ4

開発者が注目される理由について，Follower 獲得数中位群と上位群で両者の分布に大きな差を示せなかった．上記より長期的な活動につながる注目度を獲得した理由を示せなかったが，中位群の開発者も最低でも 60 の Follower 数を獲得していたことから，注目を獲得する前に行う活動を分析できると推測する．また分析方法として，Follower 獲得の理由を Following，課題報告，プルリクエストのうちから一意に決めたことから，Following の割合が圧倒的に多い結果になったのではないかと推測する．しかし，少数でも Following の前に課題報告，プルリクエストを先に行う開発者が存在した．本研究では，分析モデルの単純化のため技術レベルの異なるメトリクスを Following，課題報告，プルリクエストの 3 つに限定したため，各メトリクスについて以下で考察する．

- Following：技術的な能力が必要のない Following は，Follower 獲得前から行っていることが多い．Marlow らの研究 [7] で Follow 獲得時に相手の開発者について知ろうとするとインタビューされた開発者が述べていたことから，開発者が自発的に Following した相手から Follow が返ってきたのではないかと推測できる．また中位群と上位群，共に 60%以上の割合で Follower 獲得前に Following を行なっていたことから，Following は開発者がまず行う活動だということを示した．従来の OSS 開発では，メーリングリストへの登録が OSS コミュニティの参加する第一歩であったが，ソーシャルコーディングサイトでは Following のような「注目する」というような活動が他のユーザやコミュニティと関係を持つ始まりではないかと推測する．
- 課題報告： コミュニティの開発活動に直接貢献する活動の中でも，最も行いやすい活動だと考えられる不具合報告などの課題報告は Follower を獲得してから行う開発者が多く，今回対象とした開発者群の中位群よりわずかに上位群が多い．Gharehyazie ら [5] は開発者が開発コミュニティに開発者として参加するのにまず，プロジェクトのソースコードに慣れたり，プロジェクトの文化的背景を理解しようとすると述べている．課題報告は，今回取得したメトリクスの中で実際にソースコードを提出しないので，プルリクエストほど技術的な能力が必要ないと考える．よって，開発者は Follow していないが，プロジェクトのリポジトリへの Watch などを先に行ってから，不具合を報告したのではないかと推測する．また，その後の活動期間が長い上位群がわずかに多いことから，先にプルリクエストを行う開発者より，課題報告を行う開発者の方が長期貢献者になる可能性が高い可能性がある．
- プルリクエスト：技術的な能力が求められるプルリクエストは Follower を獲得してから行う開発者が多く，わずかに中位群（Follower 獲得数が少なかった開発者群）が多い．全体的に活動を行う割合の順番は Following，Follower 獲得，課題報告，プルリクエストであった．上記のことから，突然プルリクエストなどをマスターブランチに送るなどしてコミュニティに入ろうとする開発者より順を追って周りの環境を観察しながらコミュニティに入っていく開発者の方が活動期間が長くなるのではないかと推測する．また，プルリクエストは実際にソースコードを相手のマスターブランチへと送信するので，内的自発性は非常に高くモチベーションも高いと考えらる．しかし，上記の理由から支援が得られずにモチベーションが高い分，すぐにやめてしまう可能性があるのではないかと推測する．しかし，この議論を行うためには一度プルリクエストから行った開発者の活動期間を取得する必要がある．また，他の要因も特定した上で行う必要があると考えられる．

RQ4 では「注目される」理由を見つけることができなかったが，Following 活動から Follower を獲得する開発者がいることを数値的に示した．また，課題報告，プルリクエストは注目された後に行う開発者が多いことから，注目される理由ではない場合が多いことがわかった．本研究では，他者と関わりを持つと考える活動に限定した（高品質なソフトウェアを作成しても見つけてもらえないと意味がない）が，ソーシャルコーディングサイトにおいては発見・注目される理由が他にある可能性がある．内的自発性のモデルはソーシャルコーディングの特徴から開発者がどうのように情報を得るのかを考慮して従来の OSS 開発の長期貢献者予測モデルのメトリクスにとらわれず，取得するメトリクスを検討する必要がある．以下に具体的な因子を示す．

- プロジェクトの作成：直接的に注目度を高める活動とは考えにくいが，Marlow らの研究 [7] でプルリクエストを送られた時に開発者が信用できるか判断するのにプロフィールのプロジェクト情報を見るとインタビュワーの開発者が述べていた．プロジェクトを作成しておくと，Folloeing，課題報告，プルリクエストなどの活動をした際に Follow が返ってくる場合が多くなる可能性がある．また，Watch /Star から同じようなプロジェクトに興味がある開発者に見つけてもらえる可能性や注目度の高い開発者に Watch または Following されたとき，Follower を多く獲得する可能性がある．
- リポジトリに対する Watch：リポジトリへの Watch/Star も相手を知ろうとするきっかけである

Copyright (c) 2017 by the Information Processing Society of Japan

と Marlow らの研究 [7] では示していた．Watch された時点で，相手を見に行き，それが注目されるきっかけになる可能性がある．また，同じリポジトリを Watch する開発者同士でお互いに興味が近いことがわかるのではないかと推測する．

- “誰” に Follow / Watch されたか：注目度の高い開発者は Follower に活動の影響を与える [8] ことから，高い開発者に Follow されたら注目度が高まる可能性や，同じプロジェクトに所属する複数の開発者に Follow されたら同じプロジェクトの開発者から Follow が増える可能性がある．

5.5 本研究の制約

GHTorrent のデータセットのうち Follow 情報にはいつどのユーザがどのユーザを Follow したかという情報しかなく，“Follow を外した” という情報がないため，Follow を外された場合などその時点での正確な Follower 数を見ることはできない．

GHTorrent のデータセットは 2016 年 11 月 1 日に公開されたものを使用したが，Follow 情報は 2016 年 2 月までのものしかなかったため，RQ1，RQ2 では Follow 情報は 2016 年 2 月までのもの，活動期間は 2016 年 10 月までのもの，RQ4 では各メトリクスの期間を揃えるために 2016 年 2 月までのものをデータとして使用している．

また， GHTorrent のユーザ情報には 2017 年 10 月に登録した開発者が数人含まれている（2017 年 11 月，2017 年 12 月は存在しない）が，これは GitHub 関係者のものと考えられる（プロフィールの所属が github になっている）．Rockstar には 2017 年 10 月に登録した GitHub 関係者が 2 人含まれているが，他の対象開発者は 2008 年 1 月から登録した開発者に限定している．

6. まとめと今後の課題

本研究では，ソーシャルコーディングサイトにおいて，内的自発性と外的自発性からなる分析モデルを設定し，OSS 開発での開発者の注目度（Follower 数）と活動期間の関係を分析した．

本研究のまとめを以下に示す．

- 「注目される」ことで活動期間が長くなることを示した．具体的には，3 ヶ月，12 ヶ月で 100 以上の Follower 数を獲得した開発者群とそうでない開発者群はその後の活動期間に有意な差があり，多くの Follower 数を獲得した開発者群が活動期間がより長いことを示した．
- 「注目される」理由について基礎的な分析を行い今後詳細に取得するための知見を得た．特に，Following のように活動初期の開発者が比較的簡単に行うことができる活動を積極的に行うかどうかが重要であることが分かった．

今後は引き続き他の因子について分析を進めるとともに，活動継続性を予測するための予測モデルの構築に取り組む予定である．

謝辞

本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参考文献

[1] Brindescu, C., Codoban, M., Shmarkatiuk, S. and Dig, D.: How do centralized and distributed version control systems impact software changes?, *Proceedings of the 36th International Conference on Software Engineering (ICSE'14)*, pp. 333–333 (2014).

[2] Dabbish, L., Stuart, C., Tsay, J. and Herbsleb, J.: Social coding in GitHub: transparency and collaboration in an open software repository, *Proceedings of the 2012 conference on Computer Supported Cooperative Work (CSCW '12)*, pp. 1277–1286 (2012).

[3] Dabbish, L., Willem, V. and Williams, L.: Work practices and challenges in pull-based development: the contributor's perspective, ICSE '16 Proceedings of the 38th International Conference on Software Engineering (ICSE '16)，pp. 285–296 (2016).

[4] Georgios, G., Andy, Z., Margaret-Anne, S. and Arie, van, D.: Work practices and challenges in pull-based development: the integrator's perspective, ICSE '15 Proceedings of the 38th International Conference on Software Engineering（ICSE '15)，pp. 358–368 (2015).

[5] Gharehyazie, M., Posnett, D. and Filkov, V.: Social Activities Rival Patch Submission For Prediction of Developer Initiation in OSS Projects, *Proceedings of the 29th International Conference on Software Maintenance (ICSM'13)*, pp. 340–349 (2013).

[6] Gousios, G. and Spinellis, D.: GHTorrent: Github's Data from a Firehose, *9th IEEE Working Conference on Mining Software Repositories (MSR)* (Lanza, M., Penta, M. D. and Xie, T., eds.), IEEE, pp. 12–21 (online), DOI: 10.1109/MSR.2012.6224294 (2012).

[7] Jennifer, M., Laura, D. and Herbsleb, J.: Impression formation in online peer production: activity traces and personal profiles in github, *Proceedings of the 2013 conference on Computer Supported Cooperative Work (CSCW '13)*, pp. 117–128 (2013).

[8] Lee, M. J., Ferwerda, B., Choi, J., Hahn, J., Moon, J. Y. and Kim, J.: GitHub developers use rockstars to overcome overflow of news, *Extended Abstracts on Human Factors in Computing System (CHI'13)*, pp. 133–138 (2013).

[9] Ohira, M., Ohsugi, N., Ohoka, T. and ichi Matsumoto, K.: Accelerating cross-project knowledge collaboration using collaborative filtering and social networks, *International Workshop on Mining Software Repositories (MSR2005)*, pp. 111–115 (2005).

[10] Zhou, M. and Mockus, A.: What make long term contributors: willingness and opportunity in OSS community, *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, pp. 518–528 (2012).

Copyright (c) 2017 by the Information Processing Society of Japan