

GitHub 上のソフトウェア開発者の貢献タイプの定量的分析

益田 拓実

岡山大学大学院環境生命自然科学研究科
pf3i4wr7@s.okayama-u.ac.jp

門田 暁人

岡山大学学術研究院環境生命自然科学学域
mondn@okayama-u.ac.jp

稲吉 弘樹

岡山大学学術研究院環境生命自然科学学域
inayoshi@okayama-u.ac.jp

二本柳 佑磨

株式会社オークネット
nihonyanagiy@ns.aucnet.co.jp

要旨

本稿では、GitHub 上の各開発者の業務実態を把握し、不足している人材の特定や開発活動の改善につなげるため、あるソフトウェア開発企業の開発履歴データを題材として、GitHub 上の開発者の貢献タイプの定量化を行う。定量化にあたっては、開発における課題を管理する GitHub Issues を対象とし、issue/pull request における議論や開発者割り当て、プログラム追加、変更、マージなどの開発行動を計測する。ケーススタディとなるプロジェクトを分析した結果、定量化可能な貢献タイプとして、(1) コーディング、(2) コーディング統括、(3) コミュニケーション、(4) issue マネジメントの 4 つを同定し、それらを定量化するための 6 つのメトリクスを定義した。この定義に基づいて、2 つのリポジトリにおける各開発者の貢献タイプの定量化を行った結果、リポジトリ毎に 10 名程度の開発者の特徴、貢献タイプ別の貢献度合いが明らかとなった。

1. はじめに

ソフトウェア開発を円滑に進め、成功に導くためには、様々な貢献を行う開発者が必要となる。今日では、バージョン管理システム Git のホスティングサービスである GitHub が広く開発に用いられ、GitHub 上で多数の開発者が様々な貢献を行っている。例えば、コーディングによる貢献、issue の open や assign といった管理的な貢献、issue にコメントを付与するといったコミュニケー

ション上の貢献などである。開発を潤滑に進めるためには、各開発者がどのような貢献を行っているのかについて、その実態を把握し、不足している貢献タイプを特定したり、不足している貢献を行う開発者をトレーニングによって育成していくことが望ましい。また、組織の管理者が想定している各開発者の貢献タイプと、実際の貢献内容にズレがある場合、何らかの是正措置が必要となる場合がある。ただし、多人数によるソフトウェア開発では、各開発者の貢献の実態を把握することは必ずしも容易でない。

そこで、本稿では、ある企業における GitHub を用いたアジャイル型のソフトウェア開発プロジェクトを対象とし、GitHub Issues の履歴、および、そこから辿れる commit の履歴に基づいて、各開発者の貢献タイプを定量化することを目的とする。ソフトウェア開発では、既存のプログラムに追加や変更を行うことが繰り返されるが、そのトリガとなるのが issue である。issue が open され、プログラムに対する変更要求が明らかとなると、開発者が assign され、pull request, add commit, push commit, comment, merge などの活動が行われ、最終的に close される。これらの活動は、開発に対する何らかの貢献に関連しており、貢献タイプの定量化につなげることができる期待される。

ただし、GitHub 上の活動を貢献タイプへと結びつけるためには、GitHub issue の運用方法（例えば、issue の open はどのようなタイミングで行われ、こういった立場の人が行うことになっているのかなど）や、組織が想定している各開発者の立場や役割、各活動の背景にある事

情（例えば、開発者の unassign ほどのような場合に行われるのか）を把握し、それに基づいた分析が必要である。そこで、本研究では、まず、GitHub Issues から各活動の履歴を計測し、頻度付きの活動遷移グラフを作成し、それを開発企業へ提示して議論することで、GitHub Issues の運用方法や各活動の背景にある事情を明確にする。また、GitHub Issues に登場する各開発者について、その立場や役割を開発企業に尋ねる。その結果、定量化可能な貢献タイプとして、(1) コーディング、(2) コーディング統括、(3) コミュニケーション、(4) issue マネジメントの4つを同定し、それぞれを定量化するためのメトリクスと紐づけることができた。各開発者の GitHub 上の活動をこれらのメトリクス値のレーダーチャートによって可視化することで、各開発者の貢献の詳細を知ることが可能となった。

以降、2章では、関連研究について述べる。3章では、活動遷移グラフの構築、および、その解釈について述べる。4章では、貢献タイプの定義とその分析結果について述べる。5章はまとめと今後の課題である。

2. 関連研究

従来、GitHub 上の開発者の行動を定量化したり、貢献タイプを同定したりする研究が行われてきた [1][2][3][4][5]。

Nishiura ら [4] は、GitHub 上の OSS 開発者の信頼度 (trustworthiness) の評価を目的として、開発者の信頼度に関連する行動特性を定量化するためのメトリクスとして、issue 遂行率、issue 平均遂行時間、コメント付与数、issue 作成数、被 assign 数、コメント獲得率の6つを挙げている。例えば、issue 遂行率は、「仕事を投げ出さない」「仕事が丁寧である」といった信頼できる開発者の行動特性に関連していると仮定されている。同様に、issue 平均遂行時間は「業務遂行スピードが速い」、コメント付与数は「困っている人を助けることができる」および「相手の話を聞くことができる」といった行動特性と紐づけられている。これらのメトリクスに基づいて React プロジェクトと Python プロジェクトを分析した結果、メトリクス値には大きなばらつきがある、つまり、プロジェクト内には信頼度の高い開発者と低い開発者が混在していることなどを定量的に明らかにしている。ただし、企業における開発では、issue の遂行率は総じて高く、また、issue 完了の可否や完了までの時間は各 issue 固有の事情に依存するため、これらの計測結果から開発者の信

頼度を推定することは必ずしも適当でない。

池本ら [3] は、不足している貢献タイプの同定などを目的として、GHTorrent から取得された GitHub 上の 104,425 名の開発履歴データに対して原形分析を適用することで、7つの貢献タイプを同定している。この研究では、まず、コーディング志向/ディスカッション志向、コア/非コア、といった2種類のメトリクスを定義し、実データに適用して開発者を4つの貢献タイプに分類している。次に、各貢献タイプに含まれるデータに対して原型分析を適用し、貢献タイプの詳細化と整理を行った結果、最終的に7種類の貢献タイプを同定している。得られた貢献タイプを用いて、stars の多い bootstrap プロジェクトと stars の少ない portfolio プロジェクトの開発者を分析した結果、bootstrap プロジェクトでは多様な貢献タイプが開発者が網羅的に存在しているのに対し、portfolio プロジェクトでは貢献タイプに大きな偏りがみられることなどを明らかにしている。同様に、Cheng ら [1] は 29 の GitHub プロジェクトの 20,838 人の貢献者の活動データに対して、因子分析とクラスタリングを行うことで、4つの Active ロールと5つの Supporting ロールを同定している。ただし、これらの研究では、各貢献タイプやロールは統計的に求められたものであり、GitHub の運用ルールや組織固有の事情に即したものとなっておらず、企業における開発者の分析に当てはめることは必ずしも適当でない。

Emmanuel ら [2] は、5つの Eclipse プロジェクト (Tycho, Birt, Jetty, DeepLearning4J, AspectJ) を対象として、GitHub 上の開発者の用いるライブラリやフレームワークに基づいて開発者のスキルを分析している。今日のソフトウェア開発は多種多様なライブラリやフレームワークを用いることが必須であり、ライブラリやフレームワークを使いこなすことが開発者の重要なスキルとなっているためである。分析では、各 commit に含まれる Java の import 文に着目し、import されているライブラリを (フレームワークを含む) 同定している。さらに、ライブラリを UI, Tester, Debugger/Monitor, Networking/Server, I/O, Database, Security, Utility の8カテゴリに分類することで、各開発者がどのカテゴリのスキルを持っているか、また、各プロジェクトはどのカテゴリのスキルを必要とするかを定量化することを可能としている。一方、本研究では、コーディングに特化した開発者スキルではなく、管理やコミュニケーションなども含めた開発者の役割や貢献に焦点を当てている点

が異なる。

3. 活動遷移グラフの構築

3.1. 分析対象プロジェクト

本研究では、商品の売買を行うソフトウェアシステムの開発プロジェクトのうち以下の機能を持つ2リポジトリを対象とする。いずれも、競りシステムのフロントエンドの実装を担うリポジトリである。

- 買い手側フロントエンド (リポジトリ 1)
- 売り手側フロントエンド (リポジトリ 2)

本プロジェクトは、ソフトウェアの開発とそれに続く保守を行うプロジェクトであり、アジャイル開発が行われ、開発者は20名程度である。データ集計期間は開発初期(2021年4月)から現在(2025年1月時点)迄である。issueの数は(pull requestも含め)、各リポジトリ800程度である。

3.2. データ計測

本研究では、GitHub API を利用し、データを計測する。GitHub API には GitHub REST API と GitHub GraphQL API の2種類が存在するが、本研究では GitHub REST API を利用する。本 API を利用した理由は、多種類のキーを含むデータを一括して取得でき、データ取得後に分析を行うのに適していると判断したためである。次に、GitHub REST API の使用方法について示す。本 API では、HTTP メソッドと URL(エンドポイントと呼ばれる)を指定することで、レスポンスを JSON 形式で得ることができる。本研究では issue のデータを取得するため、エンドポイントは次のように設定する。

```
https://api.GitHub.com/repos/<owner>
/<repository>/issues
```

curl コマンドや Python 上で Request ライブラリを用いて実行可能である。curl による実行例を以下に示す。なお、実行例にある `personal_access_token` は、API のリクエスト可能数及び権限の範囲を拡張するために GitHub 側であらかじめ生成しておく必要がある。

```
> curl -o "<filename>.json"
-H "<personal_access_token>" <url>
```

本研究ではこれに加えて、issue 毎のイベント情報と、それに付随するコミットの情報を API を用いて取得した。issue イベント情報の取り出しには、イベントを issue 番号別に取り出す「event エンドポイント」と、イベントを実行時間順に取得する「timeline エンドポイント」の二つを利用する。

実行結果(レスポンス)は多数のキーと値からなり、この値から有用なものを選択し集計する。データの選択、計算方法については4章で述べる。

3.3. 活動遷移グラフの構築方法

issue イベントの種類と活動傾向を明らかにするために、活動遷移グラフを構築する。まず、Python を用いて API から得たデータから issue のイベント情報を取り出し、すべての遷移を確認した。しかし、issue イベントの種類が多く、傾向が読み取れなかった。ただし、各イベントの内容は知ることが出来たため、イベントの厳選を行うこととした。以下に選定したイベントの一覧と内容を示す。

1. opened: issue/pull request がオープンされた。
2. referenced: コミットが issue と紐づけられた (add commit), もしくは紐づけられたコミットがプッシュされた (push commit)。
3. assigned: 開発者が業務を割り当てられた。
4. self-assigned: 開発者が自身で業務を割り当てた。
5. unassigned: 担当者が担当を外された。
6. committed: issue に関するコミットが追加された。
7. commented: コメントが投稿された。
8. merged: pull request がマージされた。
9. closed: issue/pull request が閉じられた。

上記に含まれないイベントとしては、reopened (issue の再オープン) や milestoned (マイルストーンの設定) などの出現回数が少なかったもの、labeled(ラベルの付与)のように、本プロジェクトで必ずしも明確なルールに沿っ

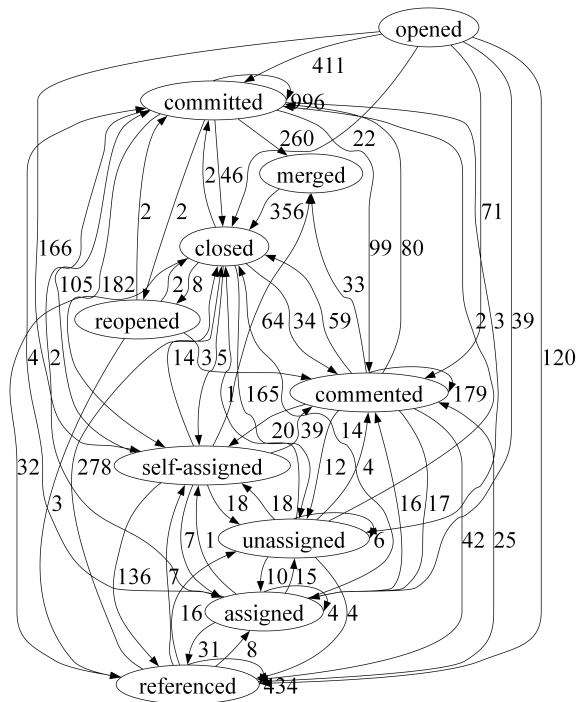


図 1. リポジトリ 1 における活動遷移グラフ

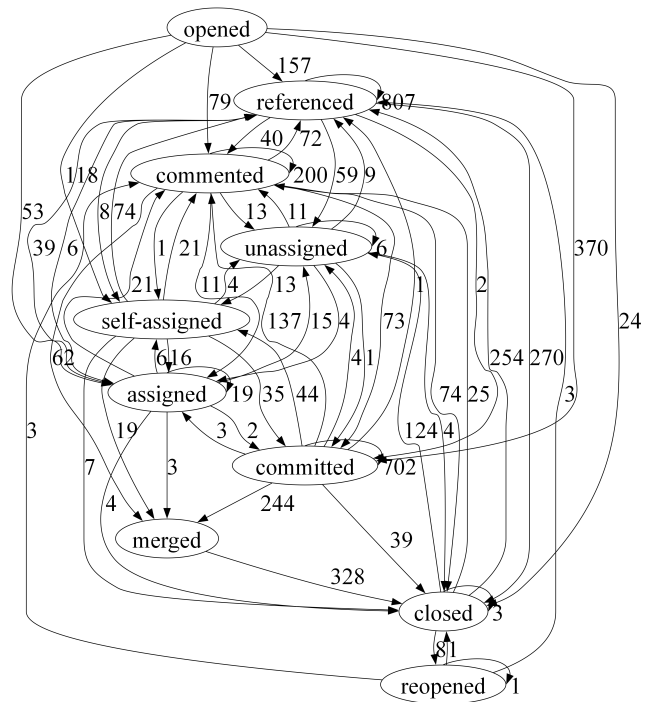


図 2. リポジトリ 2 における活動遷移グラフ

た運用がされていないと判断されたもの、renamed(タイトル変更)など、開発作業に直接関係しないと判断されたものがある。これらは、本稿において分析対象外としている。

3.4. 結果とその解釈

リポジトリ 1 の活動遷移グラフを図 1 に、リポジトリ 2 の活動遷移グラフを図 2 にそれぞれ示す。図中、楕円はイベントを表し、数値は遷移した回数を表す。例えば、図 1 では、opened の次の遷移先としては committed が最も多く、その回数は 411 となっていることが読み取れる。

これらの活動遷移グラフを精査するとともに、開発企業に対する質問の結果、次に示すような開発の実態（各作業の背景にある事情）が明らかとなった。

1. 大まかな流れは、opened → (assigned, commented, committed, referenced) → merged → closed である。
2. opened の後、assigned せずに closed していることが多い。

- プログラム変更後に issue を立てているケースがある。

3. 自分自身への assigned(self-assigned) が多い。

- 管理者が assign を振り分けているとは限らない。

4. close 後に unassign しているケースが多い。

- 離任によってシステムから自動で unassign されている。

これらの結果をもとに、4 章では、定量化可能な貢献タイプの同定を行う。

4. 貢献タイプの分析

4.1. 貢献タイプとメトリクスの定義

3.4 節の結果から、close 数、unassign 数、self-assign 数は有用でないと判断した。また、issue は管理者から割り当てられるとは限らないこと、および、プログラム変更後に issue を立てることも少なくないことから、issue の遂行時間や遂行率といったメトリクスは有用でないと

表 1. リポジトリ 1 の開発者ごとのメトリクス値

開発者	commit 数	新規機能追加率	merge 数	comment 数	open 数	assign 数	event 参加回数
A	139	0.73	252	139	204	62	2267
B	225	0.68	20	35	228	3	863
C	76	0.75	2	107	41	6	540
D	73	0.91	86	129	56	11	723
E	113	0	0	56	83	0	458
F	51	0.70	0	29	25	0	224
G	80	0.32	0	9	13	0	189
H	88	0.61	0	4	46	0	163
J	20	0.01	0	0	43	2	141
K	0	0.00	0	68	47	2	228
O	6	0.70	0	41	20	0	222

表 2. リポジトリ 2 の開発者ごとのメトリクス値

開発者	commit 数	新規機能追加率	merge 数	comment 数	open 数	assign 数	event 参加回数
A	136	0.37	227	254	142	49	2559
C	42	0.34	4	69	18	17	515
D	101	0.47	100	112	54	15	1023
F	261	0	0	46	120	4	843
I	0	0	5	37	43	2	403
M	700	0.28	0	302	334	2	1489
N	0	0	0	37	0	5	222
P	83	0	0	30	22	11	682
Q	0	0	0	0	3	0	225

判断した。一方、commit 数は開発者のコーディング作業量を表す指標として利用可能であり、その内容を定量化する一手段として、追加行数や削除行数が利用できると判断した。なお、本プロジェクトでは、pull request は commit を行った者が行うという運用となっていたため、pull request を独立した開発行動として区別しないこととしたが、その merge については、コーディングを統括する行動として区別することとした。また、issue を他者へ assign することは、管理的側面がある行動であり、他の行動と区別して扱うこととした。さらに、issue に対する comment によって議論が行われていたことから、comment 数をコミュニケーションのメトリクスとして用いることとした。

以上の検討をもとに、4 種類の貢献タイプと、それらに関係する 6 つの指標（メトリクス）を定義した。以下にその一覧を示す。

1. コーディング：コーディングによる貢献を行う開発者である。commit 数が多いほど貢献が大きいと判断する。また、コーディング内容の定量化の一手段

として、「新規機能の追加率」を用いる。この値が小さい場合には、新規機能ではなく既存機能の変更による貢献が大きいと判断する。

(a) commit 数

(b) 新規機能の追加率

$$\left(= \frac{\max((\text{追加行数} - \text{削除行数}), 0)}{(\text{追加行数} + \text{削除行数})} \right)$$

2. コーディング統括：コーディング統括するリーダー的な立場の開発者である。Pull request の merge 数が多いほど統括的な役割が大きいと判断する。

(a) merge 数

3. コミュニケーション：コミュニケーションによって多人数による開発を円滑に進める開発者である。comment 数によって判断する。

(a) comment 数

4. issue マネジメント：issue マネジメントを行う開発者である。issue を open したり、open した issue を

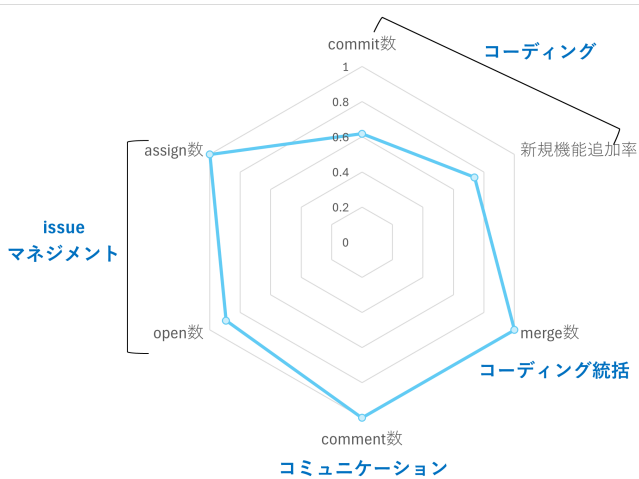


図 3. リポジトリ 1 における開発者 A

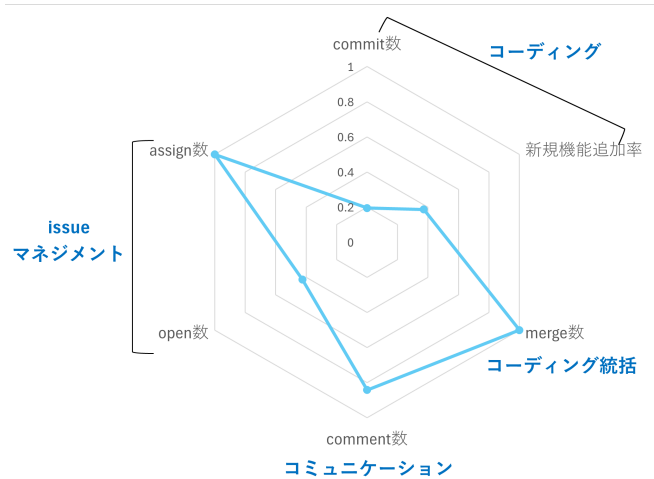


図 4. リポジトリ 2 における開発者 A

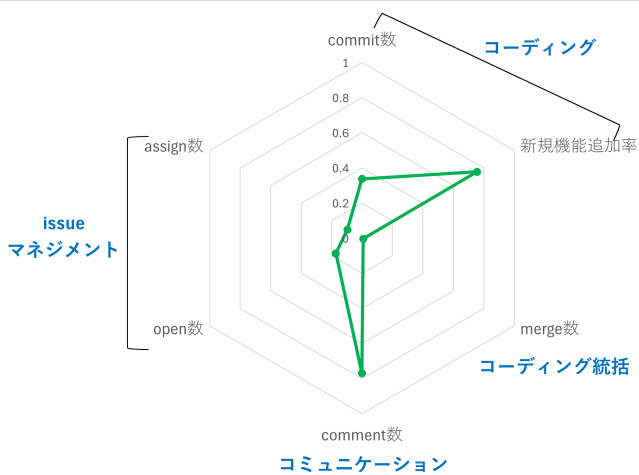


図 5. リポジトリ 1 における開発者 C

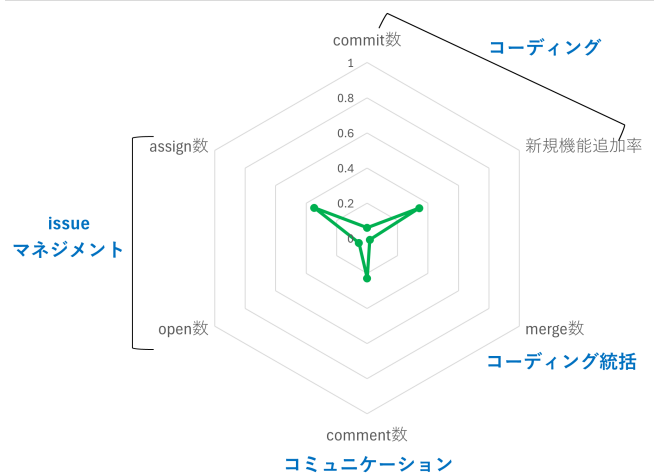


図 6. リポジトリ 2 における開発者 C

他者へ assign することが想定されている。

- (a) open 数
- (b) assign 数 (自身による assign を除く)

これらのメトリクスに加え、開発者ごとの event 参加回数も集計する。

4.2. 分析結果と考察

4.2.1. 開発者ごとのメトリクスの集計結果

集計結果を表 1, 表 2 に示す。なお、「event 参加回数が 100 に満たない」もしくは「commit 数と comment 数の

両方が 0 である」開発者は、開発に十分携わっていない者として本論文では省略している。これらの表より、各開発者の活動量、および、貢献タイプには大きなばらつきがあることが分かる。以降では、レーダーチャートを用いて、詳細な分析を行う。

4.2.2. レーダーチャートによる分析結果

開発者個人の特徴を見るため、リポジトリ 1 と 2 に共通する開発者 2 名、及び傾向が顕著な開発者を対象としてレーダーチャートによる分析を行った。

図 3, 図 4 は両リポジトリに共通する開発者 A のそ

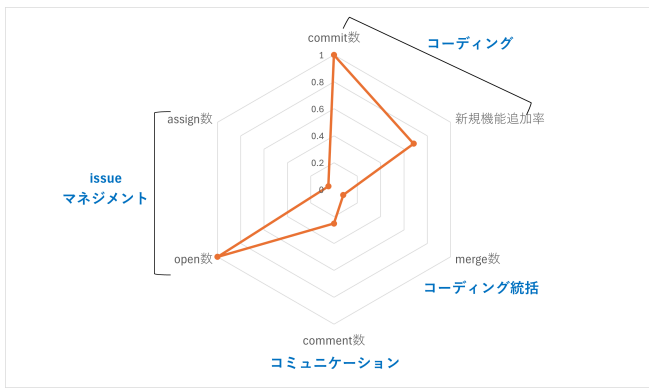


図 7. 寡黙な開発者

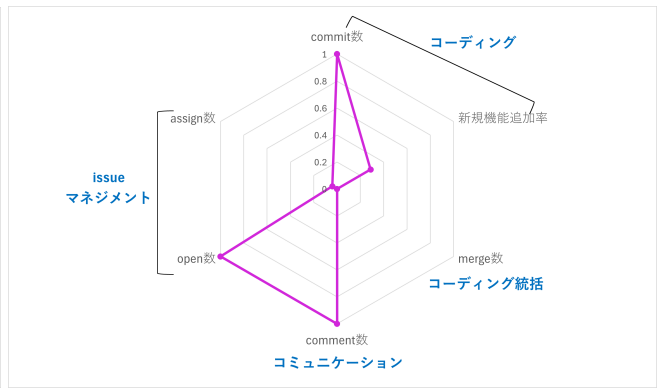


図 8. 内容改修を主体とした開発者

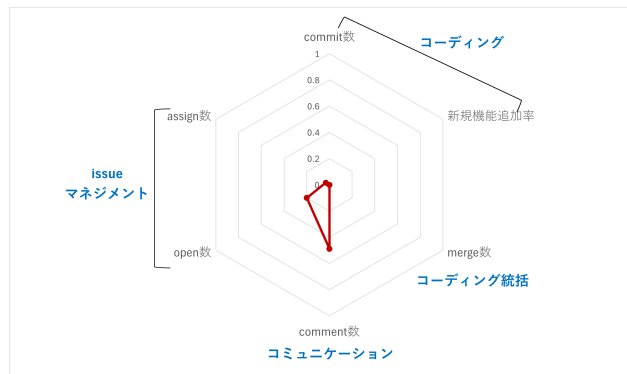


図 9. コミュニケーション主体の開発者

それぞれのリポジトリにおける貢献タイプを定量化した結果である。merge 数、comment 数がどちらも高いことから、「コーディング統括」「コミュニケーション」タイプの要素が強いことが共通しているものの、commit 数、open 数についてはリポジトリ 1 での値がリポジトリ 2 よりも高いため、リポジトリ 2 では主要なコーディング貢献者ではなくなっていることが推察される。

図 5、図 6 も両リポジトリに共通する開発者 C のそれぞれのリポジトリにおける貢献タイプを定量化した結果である。「merge 数」が小さいことから、コーディング統括者ではないことが分かる。また、リポジトリ 1 の方が全体的に貢献が大きいことが分かる。「comment 数」が両リポジトリでそれなりにあることから、コミュニケーションによる貢献を行っていると言える。なお、リポジトリ 1 では他者への assign を行わなかったのに対し、リポジトリ 2 では assign を行っていることから、issue マネジメントにおいてリポジトリによって果たす役割に違いがあると読み取れる。

図 7 に示す開発者は、コーディングを多く行うものの、コミュニケーションが少ない。すなわち一人で開発を進めていく寡黙なタイプであると分かる。このことから、この開発者は、コミュニケーションに改善の余地がある可能性がある。

また、図 8 に示す開発者は、コーディングが多く新規機能追加率が少ないことから、内容改修を主体とした開発者であると分かる。この開発者はコミュニケーションも多く、周りの開発者と連携しながら改修を進めていくことができていることが示唆される。

図 9 に示す開発者は、コーディングを行わず、コミュニケーションを主体としている。issue マネジメントに関するメトリクスの値も低いことから、開発には直接携わらないものの、開発を側面から支援していると推察される。貢献タイプの分析により、このような人材の存在も明らかにできた。

5. まとめ

本論文では、ソフトウェア開発企業の特定のプロジェクトを対象として GitHub 上の issue に関するデータを計測し、活動遷移グラフの構築と貢献タイプとそれを定量化するメトリクスの定義、および、2つのリポジトリを対象とした定量的分析を行った。活動遷移グラフからは、プログラム変更後に issue を立てているケースが少なくないことや、管理者が assign を振り分けているとは限らないことなどが明らかとなった。さらに、2つのリポジトリに共通する開発者や、特定の特徴を持った開発者についてレーダーチャートによる分析を行い、開発活動の詳しい特徴を明らかにできた。今後の課題として、貢献タイプのより詳細な分類やメトリクスの追加、多種多様なプロジェクトへの適用が挙げられる。

6. 謝辞

本研究は JSPS 科研費 JP20H05706 の助成を受けた。

参考文献

- [1] J. Cheng, J. L. C. Guo, “Activity-based analysis of open source software contributors: roles and dynamics,” Proc. 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE’19), pp. 11-18, Aug. 2019.
- [2] W. C. Emmanuel, A. Monden, “Human Resource Analysis Based on Used Libraries in Eclipse Projects on GitHub,” Proc. 22nd IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, No. WP1-07, pp.1-4, Nov. 2021.
- [3] 池本和靖, 門田暁人, “原型分析による OSS 開発者の貢献タイプの分析,” コンピュータソフトウェア, Vol. 37, No. 4, pp.17-23, Nov. 2020.
- [4] K. Nishiura, K. Ikeda, M. Sasakura, A. Monden, “Exploring Behavioral Trustworthiness of GitHub

Developers,” Proc. 5th World Symposium on Software Engineering (WSSE2023), pp. 92-95, Sep. 2023.

- [5] S. L. Vadlamani, O. Baysal, “Studying Software Developer Expertise and Contributions in Stack Overflow and GitHub,” Proc. 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 312-323, 2020.