グラフモデルを用いた OSS コミュニティ進化構造分析方法の提案と評価

2013SE050　稲垣　遥太　2013SE074　加藤　聖也

指導教員　青山　幹雄

1. 研究背景

OSS（Open Source Software）開発コミュニティ内の開発者ネットワークは複雑化している．OSS コミュニティを対象としたリポジトリマイニングの解析は既に提案されているが， OSS コミュニティの動的な構造変化の分析は確立されていない．また，グラフモデルを用いて OSS コミュニティ構造を明らかにする研究も未確立である．

2. 研究課題

本研究では，背景を踏まえ，以下の 2 点を研究課題とする．

(1) グラフモデル表現による OSS コミュニティ構造の明確化

(2) 動的な構造変化に対応できる OSS コミュニティ構造分析方法の確立

3. 関連研究

(1) OSS コミュニティ

OSS コミュニティは，OSS の開発や改善を目的とした，開発者や利用者から構成され，運営されている組織である．

コミュニティ構造を，中核，非中核的開発者として捉え，それを k-平均法クラスタリングを用いた分類が行われている[1]

(2) ソフトウェアリポジトリマイニング

ソフトウェアリポジトリマイニング(MSR)とは，ソフトウェアリポジトリの調査に関する幅広い研究である．ソフトウェア進化の研究において，リポジトリから適切な情報の抽出や，関係や傾向を明らかにする試みが行われている[3].

(3) グラフデータベース[5]

グラフデータベース(グラフ DB)は，グラフデータモデルを公開する CRUD メソッドを持った，オンライン DB 管理システムである．グラフデータモデルには，プロパティグラフ，ハイパーグラフ，トリプルなどの複数の異なるモデルが存在する．

4. アプローチ

研究課題を解決するためのアプローチを以下に示す．

(1) OSS コミュニティ構造分析へのグラフ DB の適用

OSS コミュニティ構造をグラフ DB を用いてモデル化する．

(2) グラフデータベースを用いた分析

グラフ DB が持つクエリを利用して可視化し，分析を行うことにより，OSS コミュニティの進化の特性を得る．

5. 提案方法

5.1. 提案プロセス

以下の図 1 に本稿で提案するプロセスを示す．

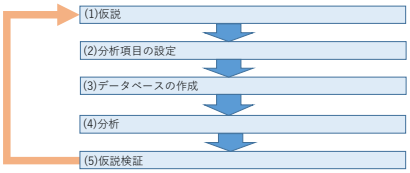

図 1 提案プロセス

5.2. 仮説設定

OSS コミュニティが，どのような要素，要因によって進化しているのか仮説を立てる．

5.3. 分析項目の設定

立てた仮説に基づいて，それぞれの仮説ごとに明らかにするべき項目を列挙する．明らかにするべき項目を統合して，分析項目を設定する．

5.4. グラフデータベースの作成

(1) モデリング

新規のノード，関係のデータを追加する必要がある時には，再モデル化を行い，分析すべき関係の意味が損なわれないよう注意しなければならない．

(2) データ収集

データの収集には主に，OSS の開発に使用されているプラットフォームが提供している API を使用する．OSS リポジトリから収集するデータとして，GitHub から得られるデータを想定する．

(3) 収集データの処理

収集するデータは，分析項目で設定した内容に沿う必要がある．取得したデータには，分析に不必要なものが存在する．そこで，DB へデータを挿入する前に，分析に必要なデータのみをプロパティとして付与する．

(4) データベースへの挿入

処理が完了した収集データを DB へ挿入する．挿入には，利用するグラフ DB で使用可能なクエリ言語を使用する．

5.5. 分析

設定した分析項目に従って分析し，結果を出力する．分析する際，開発コミュニティの性質が全て同じではないこと考慮する必要がある．本稿では，コミュニティの性質が近いと考えられる，いくつかのコミュニティを分析対象とした．

5.6. 仮説検証

分析で得られた結果から，立てた仮説を検証する．必要に応じ，得られた結果に基づいて，仮説の修正や追加を行い，はじめから一連のプロセスを繰り返す．

6. プロトタイプの実装

図 2 に，本稿で用いたシステムの構成を示す．OSS コミュニティとして GitHub，グラフデータベースに Neo4j，グラフの可視化に Neo4j Browser を用いた．

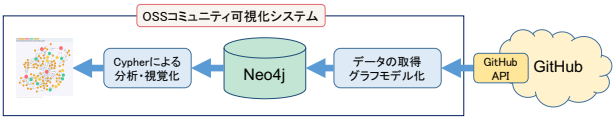

図 2 システム構成図

7. GitHub を用いた OSS コミュニティへの適用例

7.1. 分析対象の OSS コミュニティ

機械学習に関する OSS コミュニティ，TensorFlow，Caffe，Chainer，Jubatus を分析対象とした．本稿では主に，Chainer に関しての分析を取り上げる．

7.2. 仮説設定

OSS コミュニティが，どのような特性によって進化しているのかの問いに対して，以下の仮説を立てる．

(1) 定期的な新規開発者の参入
(2) 新規開発者の中核的開発者への成長
(3) 中核的開発者と新規開発者の更新を行うファイルの共通化
(4) コミュニティ進化の段階

7.3. 分析項目の設定

仮説より，以下の分析項目を設定する．

(1) 期間ごとの，コミュニティへ新しく参加した開発者
(2) 期間ごとの，コミュニティに対して貢献の多い開発者
(3) 新規開発者と貢献の多い開発者の関係
(4) 新規開発者と中核的開発者が共通して更新を行ったファイル
(5) 中核的開発者による貢献数の累計の推移

分析において，コミュニティ進化の時系列を期間ごとに分割する必要がある．コミュニティ発足日から 3 ヶ月ごとに分割し，それぞれを 1 期，2 期などとした．また，3 ヶ月に満たないデータは分析対象としないこととした．

7.4. グラフデータベースの作成

図 3 にグラフ DB の概要を示す．GitHub 上から取得したデータをノード，関係及びプロパティとして定義した．データは，コミュニティ発足時から 2016 年 12 月 1 日までとした．

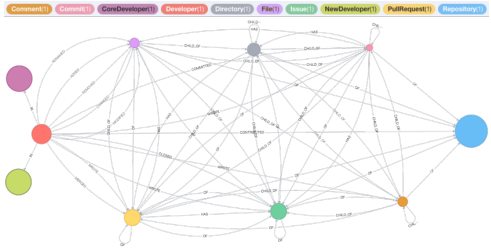

図 3 グラフデータベースの概要

7.5. 分析と仮説検証

7.5.1. 3 層で構成されるコミュニティ進化モデル

コミュニティ上の開発者の構造を以下の非中核，準中核，中核の3層に分類できる(図 4)．

(1) 中心的な役割を果たすまでに至らなかった，非中核的開発者
(2) 一定の期間においてのみコミュニティの中心になって貢献している準中核的開発者
(3) 長期間，コミュニティの中心になって貢献している中核的開発者

期間ごとに新規開発者が参入し，多数は非中核的開発者に留まるが，その一部はコミュニティにとって中心的な役割を果たす開発者として現れる．しかし，その中の多数は，一定の期間にのみ中心的な役割を果たし，その後はフェードアウトしていく傾向があり，中核的な開発者へと成長するものは少数である．

中核的開発者は，コミュニティ発足時の一部の開発者が主体になって構成されている．この構造はコミュニティが進化しても，大きく変わることがない．

中核的開発者を中心に，非中核的開発者が参入し，その中から準中核的開発者があらわれ，それが入れ替わりながら進化する特性が明らかになった．

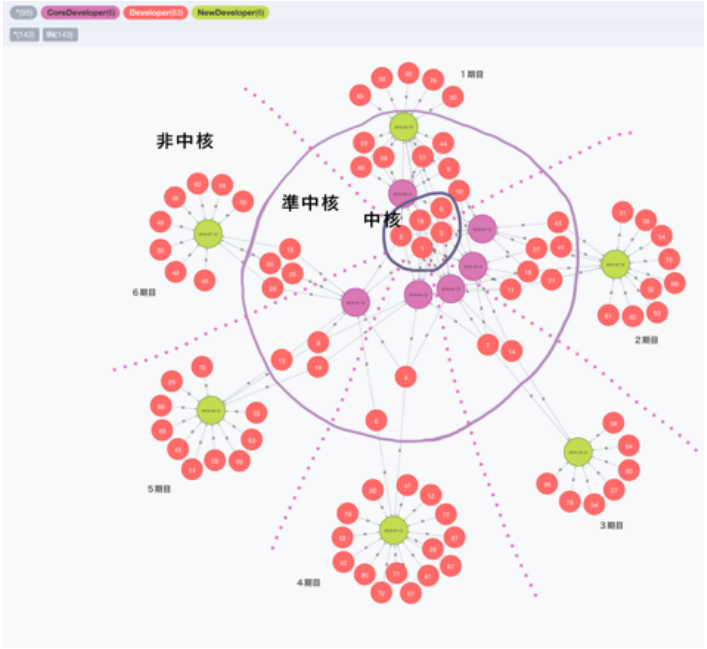

図 4 コミュニティの 3 層構造

7.5.2. 開発者の行動による 3 つの成長パターン

新規参入時からデータ取得時までの，貢献度に差がある開発者 3 人を抽出した．3 人の開発者が，新規参入時から半年間の行動と，その行動に対する他の開発者の行動をサブグラフとして可視化した(図 5)．

(1)　ノード番号 14 の開発者

コメントが多く，他の開発者とのコミュニケーションは活発であるが，Issue に比べて Pull Request の作成は少ない．

(2)　ノード番号 6 の開発者

コメントは少なく，他の開発者とのコミュニケーションも少ない．しかし，積極的に Pull Request を作成している．

(3)　ノード番号 4 の開発者

コメントが多く，他の開発者とのコミュニケーションは活発である．また，積極的に Pull Request も作成している．

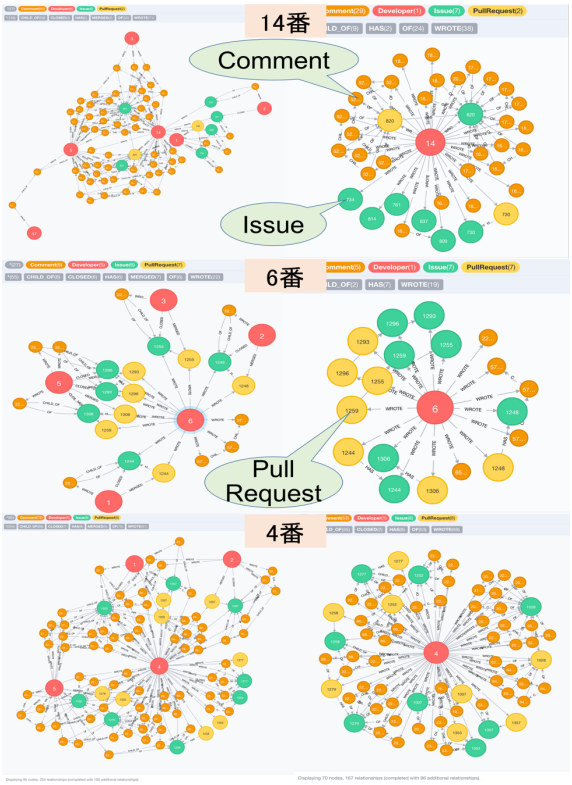

図 5 開発者行動のサブグラフ

また，3 人の開発者が，コミュニティに貢献したコミット数の累計を図 6 に示す．期間は 3 ヶ月ごとに区切り，開発者が新規参入してから3期分のデータをとった．このグラフから，14 番は初期からコミット数がほぼ変わらない．6 番と 4 番は増加しているが，増加率に差がある．

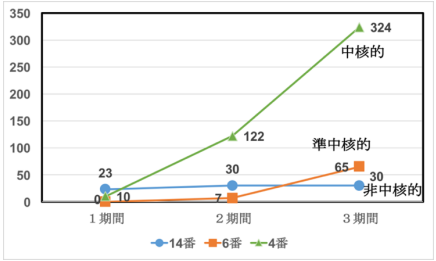

図 6 開発者の成長パターン

これらの結果より，開発者個人の行動が開発者の成長，コミュニティへの貢献に影響を及ぼすかを分析した．開発者行動によって以下の成長パターンが明らかになった．

(1)　バグ報告や機能提案を中心に行動する開発者は，非中核的開発者に留まる傾向がある

(2)　実装を中心に行動する開発者は中核的な役割を果たすようになる傾向がある．

さらに，準中核，中核的開発者に成長するパターンの違いは，他の開発者とのコミュニケーション量が異なっている．ある期間に継続して中核的開発者の役割を果たすものは，機能提案，実装を積極的に行うため，他の開発者との交流も必然的に多くなっている．コミュニケーションの中心は継続的に中核的開発者になっているものが多く，概ねコミュニティの立ち上げに貢献した開発者と考えられる．

7.5.3. 中核的開発者と新規開発者のファイル更新による相互作用の推移

プロジェクト及び，リポジトリの成長に伴い，中核的開発者が更新を行うファイルと，新規開発者が更新を行うファイルに，ハイプ曲線のような特徴[2]が表れると考えた．他の 3 つのコミュニティ分析と合わせ，開発の段階について，以下のことが明らかとなった．

(1)　初期

更新が共通するファイルが一時的に増加している．これは，新しいプロダクトへの期待，関心から開発者が積極的に更新を起こすためだと考えられる．また，共通してファイルの更新が行われた時は，ルート直下に存在するフィル，ドキュメントやセットアップ，コンフィグファイルなどが同時に更新されている可能性が高い．

(2)　中期

一定数の更新を続けたあと，共通するファイルの更新が大きく増加する様子が見られた．これは，新規開発者によって，機能提案などが行われ，実装を行ったためだと考えられる．

(3)　後期

共通するファイルの更新はほぼ存在しなかった．これは，プロジェクトの完成によってファイルの更新がなくなるため，共通するファイルがそもそも存在しないことになるためである．

図 7 に開発段階が顕著に表われた Jubatus のグラフを示す．2 期から 5 期が初期，5 期から 8 期が中期，8 期以降が後期となっている．

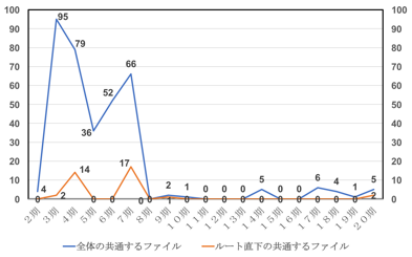

図 7 共通更新ファイル数の推移

以上より，開発の段階によって，ファイル更新に中核的開発者及び新規開発者間の相互作用がみられた．推移のグラフは，初期と中期の双峰性が見られ，山による波形はハイプ曲線での波形と似た意味を持つと考えられる．

7.5.4. コミュニティ進化の3段階モデル

仮説では，立ち上げ期，成長期，成熟，減衰期の 3 つによって開発スピードが変わると考えていたが，立ち上げ期からすぐに成長期と同様の開発スピードによって開発が行われていたコミュニティが存在した．これは，企業が主導する OSS コミュニティであることから，立ち上げ段階で中核的開発者が確保されていたためと考えられる．

コミュニティ発足時から 3 年経っているコミュニティでは，成長期から成熟，減衰期への推移を発見することができた．

8. 考察

8.1. グラフモデルによる OSS コミュニティ構造

本稿で提案したプロセスを実際のコミュニティに適用し，設定した仮説に対して，適切に分析項目を設定できた．そのため、OSS コミュニティの動的な構造変化の分析に必要なデータを十分に収集でき，グラフモデル表現によるコミュニティのモデリングが可能となった．

8.2. OSS コミュニティの動的な構造変化の分析方法の確立

コミュニティの進化を時系列に分割して，グラフとして可視化してすることにより，以下の OSS コミュニティの動的な構造を明らかにした．

(1) 非中核，準中核，中核的開発者の 3 層コミュニティ進化モデル
(2) 開発者の行動による成長パターン
(3) 開発段階による開発者間の相互作用の変化

8.2.1. コミュニティ構造と進化モデル

関連研究では，OSS コミュニティの構造を，中核，非中核的開発者として分類していた[1]．本稿では，開発者を中核，非中核に加えて準中核的開発者に分類が可能であることを明らかにした．これより，コミュニティ上の開発者の役割構造をより明確にすることが可能になった．

進化モデルとして，中核的開発者を中心に開発が進み，新規開発者の参入と，その中から準中核的開発者が現れ，それが入れ替わりながら進化していることを明らかにした．これはコミュニティの運用の支援に活用が期待できる．

8.2.2. 開発者の行動による成長パターンの特性

開発者個人の行動に着目して分析を行うと，開発者が中心とする行動から，開発者がコミュニティ内でどのような役割を担う可能性があるかを，パターン化できると考えられる．

行動パターンは，2 つに分類できる．

(1) バグ報告や機能提案を中心とする
(2) 機能の実装を中心とする

非中核的開発者は(1)の行動をとり，準中核，中核的開発者は(2)の行動をとっている可能性が高い，

成長パターンは，3 つに分類できる．

(1) 貢献数がほぼ増加せず，成長率が低い
(2) 貢献数は増加するが，成長率が低い
(3) 貢献数が増加し，成長率も高い

非中核的開発者は(1)，準中核的開発者は(2)，中核的開発者は(3)の成長パターンとなっている．準中核と中核的開発者の違いは，他の開発者とのコミュニケーション量の違いであり，継続的に機能提案，実装を行う開発者は，他の開発者とのコミュニケーションも多いと考えられる．これらの成長パターンから開発者の成長の予測へ活用することが期待できる．

8.2.3. 開発段階による開発者間の相互作用の変化

開発段階によって，ファイル更新に中核，新規開発者間の相互作用がみられた．開発は初期，中期，後期の 3 つに定義でき，開発の初期と中期で双峰性が見られ，ハイプ曲線[2]の波形と似た意味を持つと考えられる．

8.2.4. 計算量の評価

本研究の分析では，実用的な時間でクエリの結果を得ることができた．しかし，グラフデータベースを探索する経路が多いクエリほど，実行時間がかかり，計算量が増加していく．そのため，データベースの規模増大に伴い，グラフの可視化が困難になる可能性が考えられる．

9. 今後の課題

(1) 分析結果の妥当性検証
(2) コミュニティ進化モデルの拡張
(3) グラフモデルの改善
(4) 分析方法の改善

10. まとめ

本研究では，OSS コミュニティ構造にグラフモデル表現が適していることを明らかにし，時間変化に伴う OSS コミュニティの動的な構造変化の分析方法を提案した．OSS コミュニティ構造を分析することにより，以下の 3 つの特性を発見した．

(1) 非中核，準中核，中核的開発者の 3 層で構成されるコミュニティ進化モデル
(2) 開発者の行動による成長パターン
(3) 開発段階による開発者間の相互作用の変化

これらの特性を用いることで，OSS コミュニティの運用や開発者の成長の支援への活用が期待できる．

11. 参考文献

[1] A. Bosu, et al., Impact of Developer Reputation on Code Review Outcomes in OSS Projects: An Empirical Investigation, Proc. of ESEM '14, Article No. 33, Sep. 2014.
[2] J. Fenn, et al., Mastering the Hype Cycle, Harvard Business School Pr. 2008
[3] M. L. Collard, et al., A Survey and Taxonomy of Approaches for Mining Software Repositories in the Context of Software Evolution, J. of SMERP Vol.19, No.2, Mar. 2007, pp. 77-131.
[4] I. Robinson, et al., Graph Databases, O'Reilly, 2015.