オープンソース・ソフトウェア開発コミュニティにおける ライフサイクルの視覚化

増田 礼子
フェリカネットワークス株式会社
Ayako.Masuda@FeliCaNetworks.co.jp

松尾谷 徹
有限会社 デバッグ工学研究所
matsuodani@debugeng.com

要旨

オープンソース・ソフトウェア *(OSS : Open Source-Software)* は目覚ましい発展を遂げている．開発を推進する *OSS* 開発コミュニティは，*5* 年 〜 *10* 年という期間をかけて成長しており，ライフサイクルが長いという特徴がある．近年では，日々約 *5* 万件の *OSS* 開発プロジェクトが登録されているが，半年以上継続して開発活動を行っているプロジェクトは 5 % 程度であり，約 95 % は半年以内に開発を終了している．この要因の一つには，メンバのモチベーションの低下があると考えられる．本研究では，モチベーションを維持する要因を探るため，コミュニティの活動状況という観点から継続している *OSS* 開発コミュニティのライフサイクルの視覚化を試みた．視覚化は，コミュニティの開始から終了までをライフサイクルと定義し，*GitHub* をケーススタディ対象としてその変化を計測した．具体的には，メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求めた．ここでは，ライフサイクルの測り方と視覚化について報告する．この手法を用いて分析を行った結果，*OSS* 開発コミュニティが成長期にある場合には，ジニ係数の値が大きくなることが明らかとなった．本研究で提案する *OSS* 開発コミュニティのライフサイクルの視覚化手法を活用することにより，メンバのモチベーションなどについて新たな知見を得ることができる．また，*OSS* の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．

1. はじめに

オープンソース・ソフトウェア (OSS : Open Source Software) は目覚ましい発展を遂げている．Web サーバにおける LAMP (Linux, Apache, MySQL, PHP / Perl / Python) [1–6] や携帯端末における Android [7] などが核となり，さまざまなアプリケーションが開発され，無償で使用されている [8]．OSS は，普及の過程において標準化が進んでおり，企業にとっても，投資コストの低価格化という点で注目を浴びている．近年，情報産業のビジネスの中心が，ハードウェアからソフトウェアに，そしてさらにソリューションへと移行していることから，OSS が企業戦略の中に深く取り込まれたり，企業間ネットワークを形成する手段となったりしており，OSS と企業の関係は進化を続けている [9].

開発を推進する OSS 開発コミュニティは，5 年 〜 10 年という期間をかけて成長しており，ライフサイクルが長いという特徴がある．近年では，日々約 5 万件の OSS 開発プロジェクトが登録されているが，半年以上継続して開発活動を行っているプロジェクトは 5 % 程度であり，約 95 % は半年以内に開発を終了している．この要因の一つには，メンバのモチベーションの低下があると考えられる．本研究では，モチベーションを維持する要因を探るため，継続している OSS 開発コミュニティのライフサイクルの視覚化を試みる．

OSS 開発コミュニティは，企業などが主催し雇用や契約によって結束したチームとは異なる形態である．企業において一般的に用いられている「プロジェクト」とは，Project Management Body of Knowledge (PMBOK) において，独自のプロダクト，サービス，所産を創造する

ために実施する有期性のある業務と定義されている [10]. しかしながら，OSS 開発コミュニティにおけるプロジェクトは，明確な活動期限を持たないものが多く，割り当てられた業務でもないため，PMBOK の定義とは全く異なる性質を持つものである．OSS 開発は，継続的インテグレーション (Continuous Integration) により頻繁にリリースを行いながら，メンバのモチベーションによって開発が継続される．また，OSS 開発の主体も多様なコミュニティであり，開発プロジェクトの運用もコミュニティによってさまざまである．このようなことから，OSS 開発では，プロダクトの工程からライフサイクルを定義することはできない．そこで，OSS 開発を支えているオンライン上のコミュニティ・メンバの活動状況という観点からライフサイクルを捉えることとした．コミュニティのライフサイクルを視覚化することができれば，オンライン上のコミュニティ・メンバのモチベーションなどについて新たな知見を得ることができるようになると考える．また，OSS の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．

本研究では，OSS 開発コミュニティの開始から終了までをライフサイクルと定義し，GitHub [11] をケーススタディ対象とする．GitHub API v3 [12] を用いて，ライフサイクルの変化を測り，マクロとミクロの特性の 2 階層で視覚化を試みる．マクロな特性は，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求める．ミクロな特性は，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合を視覚化する．本研究では，ライフサイクルの測り方と視覚化について報告する．

2. OSS 開発コミュニティのライフサイクル計測に関連する先行研究

本章では，まず，OSS 開発の特徴を述べ，次にインターネット上のコミュニティにおける活動状況の計測に関連する先行研究をサーベイし，本研究における課題を示す．

2.1. OSS 開発の特徴

OSS の開発は，主に GitHub [11] や Bitbucket [13] といったインターネット上のソフトウェア開発のプロジェクトホスティングプラットフォームを利用して行われている．OSS 開発は，プラットフォームで OSS 開発プロジェクトを公開することにより，インターネット上で複数の開発者が並行して開発を進めていくことができる．プラットフォーム上では，多種多様なソフトウェア開発が行われるため，開発対象ごとに Repository と呼ばれる単位で一元的にデータの構成管理を行っている．OSS 開発プロジェクトの特色としては，予算を持たない，開発期間の設定がない，開発活動への参加義務がない，などがある．

一般的に，OSS 開発の利用者，開発者，愛好者らの集団を OSS コミュニティと呼ぶ．OSS コミュニティは，開発を中心とした開発コミュニティと利用を中心としたユーザコミュニティに大別できる [14]．OSS 開発コミュニティは，複数のプロジェクトを並行して開発しているものもあり，必ずしもプロジェクト間に関係性があるとは限らない．そこで，本研究では，単体のプロジェクトに相当する GitHub の Repository を計量の単位とした．そのため，本研究における OSS 開発コミュニティは，Repository 単位での集団を指す．

2.2. インターネット上のコミュニティにおける活動状況の計測

本節では，インターネット上のコミュニティの活動構造を表す研究について概観する．

北山は，29 のメーリングリストにおける投稿数を個人毎に計測し，投稿数の分布をジニ係数 (Gini Coefficient) とローレンツ曲線 (Lorenz Curve) で表した [15]．ジニ係数は，分布の偏りを表す指標であり，その分布をグラフで表したものがローレンツ曲線である．ジニ係数とローレンツ曲線は，経済学では，国民の所得がどのように分配されているのかを示す指標とグラフとして利用されている [16, 17]．北山がジニ係数を用いたのは，メーリングリストにおける個人毎に集計した投稿数の分布に偏りがあったためである．

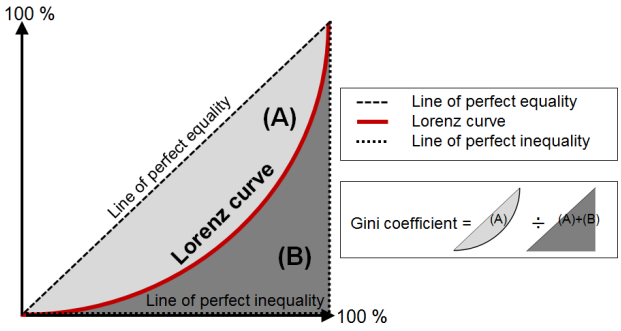

図 1: ジニ係数とローレンツ曲線

図 1 に，ローレンツ曲線の例を示し，国の世帯所得の分布を例にジニ係数とローレンツ曲線について説明する．ローレンツ曲線は，世帯を所得の低い順番に並べ，横軸に世帯数の累積比をとり，縦軸に所得の累積比をとって，世帯間の所得分布をグラフ化したものである．世帯間の所得格差が存在せず，全ての世帯の所得が同額であるならば，ローレンツ曲線は 45 度線 (Line of perfect equality) と一致する．世帯所得の分布に偏りがある場合には，ローレンツ曲線は下方 (Line of Perfect inequality) に膨らんだ形になる．ジニ係数は，ローレンツ曲線の下方への膨らみ具合を，図 1 の (A) の面積と (A) + (B) の面積の比で表したものである．ジニ係数の値は，0 と 1 の間をとるため，ジニ係数の値が 0 に近ければ分布の偏りは小さく，1 に近ければ分布の偏りが大きい．

北山の研究の他にも，ネットワーク上のデータの分析にジニ係数を用いた研究がある [18]．この Dewan の研究では，Web の閲覧行動における特定のサイトに集中する程度をジニ係数で表現し，比較している．このように，ジニ係数とローレンツ曲線は，初期の目的である所得の均等度合いを表す以外にも活用されている．ジニ係数は所得均等性の評価指標として提案された係数であるが，分布の偏りを評価したり，表したりする指標として，正規分布以外の分布を持つさまざまな事項へ応用が可能である．

ジニ係数は，計算が容易であり，データの特性をひとつの値で表すことができる．一方で，分布の形が異なっていてもジニ係数の値は同値となる場合があるため，この点には注意しなければならない．分布の偏りの程度は，ジニ係数によって表現することができる [15]．

2.3. OSS 開発における活動構造の計測

最近では，OSS 開発コミュニティの開発形態を分析した先行事例が多数報告されている．Jensen らは，3 つの OSS 開発コミュニティにおけるメンバの役割の変化を分析し，比較している [19,20]．伊原らは，OSS 開発コミュニティのメンバ間のコミュニケーションについて，メーリングリストのデータを用いてネットワーク分析を行い，開発者，ユーザ，バグ報告者の 3 者間を橋渡しする人物らが多数の参加者と常にコミュニケーションを行うことによってサブコミュニティ間の活動が円滑に行われている傾向が見られたと報告している [21]．Guimarães らによる OSS 開発コミュニティにおけるライフサイクルモデルの研究では，SourceForge [22] のダウンロード，バグ，フォーラムメッセージ，ソースコードアクティビティ情報などを用いて主成分分析を行い，スタートアップ，成長，成熟，衰退，死といった OSS 開発コミュニティのライフサイクルの段階 (Stage) によってコミュニティの有効性レベルと活動レベルに変化があることを示している [23]．

Mockus らは，Apache 開発プロジェクトの CVS データとバグ報告データを分析し，Apache コミュニティの 4 % の開発者が 88 % のソースコードを追加し，66 % の不具合修正を行っていることを明らかにした [21,24]．筆者らは，50 件の大規模 OSS 開発プロジェクトの開発記録を著作者単位で分析し，全開発量の 80 % が少数のメンバによって行われていることを示すと共に，活動実態はべき分布であり，開発規模が増加すると開発効率が向上することを示した [25]．

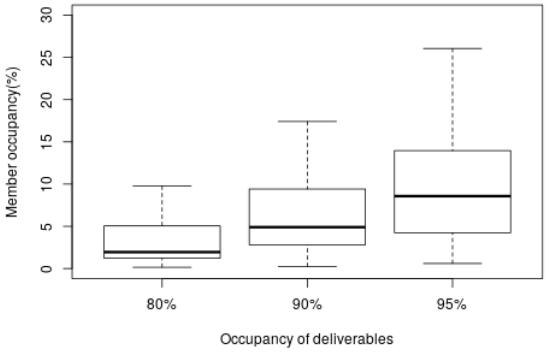

図 2: 50 件の OSS 開発 PJ における著作者毎の開発量

この研究では，著作者の活動分布に関して，図 2 に示す結果，すなわち，各 OSS 開発プロジェクトにおける全開発規模の 80 %，90 %，95 % の規模を登録した著作者数の割合の分布を測定した．図 2 より，全開発量の 80 % における著作者数の全体比率は，最小が 0.15 %，第 1 四分位が 1.2 %，第 2 四分位 (中央値) が 1.9 %，第 3 四分位が 5.1 %，最大が 9.8 % であり，極少数のメンバが開発を推進していることを示している．また，この研究では，著作者毎の開発量を時系列で集計し，順位データを作成した結果，平均値や分散などで特性を表現できないべき分布であることを示した．図 3 に，“RIOT” コミュニティ [26] の著作者毎の累積開発量の時系列グラフを示す．この時系列記録は，累積登録規模が多い上位 16 名の著作者を対象としている．図 3 のグラフ上で線が途切れていたり，途中から線が始まっていることから，活動期間 (以降，活動量と示す) がメンバ毎に異なることが分かる．

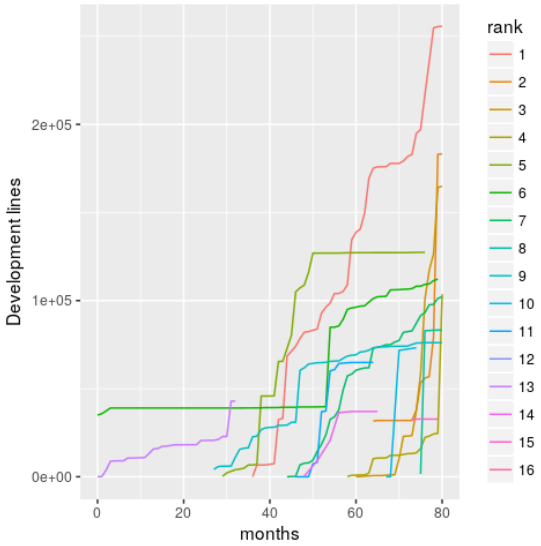

図 3: RIOT における著作者毎の累積開発量

この研究を受けて，筆者らは，開発活動の負荷分散に着目し，ジニ係数を用いて活動量の分布の計測を行った結果，OSS 開発コミュニティの活動構造の分析にジニ係数を用いることができることを示唆した [27]．しかし，この研究では，全期間の活動量を 1 つのジニ係数で表しており，ジニ係数の時系列推移を用いた分析が課題であるとしている．

2.4. 本研究における課題

本研究では，OSS 開発コミュニティの開始から終了までをライフサイクルと定義し，先行研究 [27] で示唆し，次の課題となっていたジニ係数を用いた OSS 開発コミュニティの活動構造の時系列変化をライフサイクルの変化として捉え，マクロとミクロの特性の 2 階層で視覚化を試みる．マクロな特性は，OSS 開発コミュニティメンバの活動量を Repository 単位で視覚化する．具体的には，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態をジニ係数の変化として求めることとした．また，先行研究 [24,25,27] が示す通り，ジニ係数の変化には，開発を推進するメンバ (以降，コアメンバと呼ぶ) の活動が影響している．そのため，ミクロな特性は，OSS 開発コミュニティのコアメンバに焦点を当て，コアメンバを中心とした活動割合を視覚化することとした．本研究では，ライフサイクルの計測手法と視覚化手法を提案する．

3. 分析対象と計測方法

本章では，本研究でケーススタディとして用いた分析対象と，ライフサイクルの計測手法について示す．

3.1. 分析対象の選定

ソフトウェア開発のプロジェクトホスティングプラットフォームの中でも，GitHub は最も勢いを増している [28]．そこで，本研究では，GitHub における Repository をケーススタディとして用いることとした．対象 Repository および各種データの抽出は，GitHub API v3 を用いた．

まず，2010 年 1 月 1 日から 2019 年 5 月 31 日までの期間において，乱数を用いて 200 日の特定の日を選定した．この特定の 200 日に登録された全 Repository を対象とし，次の条件に当てはまる Repository を抽出した．

(a) 更新期間が半年以上

(b) Fork 数が 1 以上

(c) Star 数が 1 以上

コミュニティのライフサイクルの計測が目的であることから，継続して開発が行われている Repository を分析

対象とするため，(a) の条件を設定した．GitHub 上に登録されている Repository には，更新頻度が低く，Contributor 数が少ないものが多いことから，個人的な入れ物として利用しているものが含まれていると推察される．Fork 数と Contributor 数の相関を確認したところ，両者には相関があり，Fork が行われていない Repository は，個人での開発であるものが多いことが明らかとなった．本研究の分析対象は OSS 開発コミュニティであることから，Contributor 数が 少ない Repository を対象から除外するため (b) を設定し，開発のためにサーバ上で Repository の複製を作成する行為である Fork が全く行われていない Repsitory を対象から外すこととした．また，OSS 開発ではプロダクト自身に価値があるものであるため (c) を設定し，OSS の利用者が評価する Star が全く付いていない Repository を対象から外すこととした．これらの条件に当てはまる Repository は，合計で 91,787 件であった．コミュニティのライフサイクルを計測するためには，ある程度の人数以上が開発に参加している Repository である必要がある．そこで，この 91,787 件の Repository から，コミュニティの人数が 30 人以上の Repository を抽出した．コミュニティの人数が 30 人以上であった Repository を抽出した結果，1,707 件となった．

続いて，OSS 開発コミュニティの多様性を考慮するため，OSS コミュニティを分類する観点を検討した．コミュニティへの参加状況は，対象 Repository に何らかの貢献を行った Contributor 数から見ることができる．開発プロダクトへの評価状況は，Star 数から見ることができる．そこで，コミュニティの人数が 30 人以上である 1,707 件を次の 2 つの観点で 4 段階に分類した．

総 Contributor 数 ：開発活動への参加状況

総 Star 数 ：開発プロダクトへの評価状況

分類には，各観点に対する 1,707 件の四分位を用いた．本研究は OSS 開発コミュニティの分析であることから，開発活動への参加状況の観点を重要視し，まず，総 Contributor 数の各カテゴリから 20 件ずつ，合計 80 件をサンプリングした．次に，サンプリングした Rpository において，もう一つの観点である開発プロダクトへの評価状況として総 Star 数のカテゴリ毎の分布を確認した所，大きな偏りなく分類できていたため，この 80 件の Repository を分析対象とした．80 件のサンプリング Repository のカテゴリ毎の分布を表 1 に示す．各観点毎のランクに記載されている値は，Rank 1 が全体の 25 % 未満，Rank 2 が全体の 25 % 以上 50 % 未満，Rank 3 が全体の 50 % 以上 75 % 未満，Rank 4 が全体の 75 % 以上に分類された Repository 数である．

表 1: 80 件のサンプリング Repository のカテゴリ分布

		Contributors			
Rank		1	2	3	4
Stars	1	4	5	5	3
	2	5	6	3	8
	3	8	5	6	5
	4	3	4	6	4

3.2. 計測手法

OSS 開発では，労務管理そのものが存在しないため，開発に費やした時間の記録は存在しない．それゆえ，OSS 開発においては，企業におけるソフトウェア開発で一般的に用いられている工数 (Person-Months) を単位として計測することは難しい．分析を行うためには，工数に相当する単位を定義する必要がある．そこで，OSS 開発コミュニティに参加し，何らかの著作物を Repository に Commit した Contributor を計測対象とし，開発期間を 1 ヶ月単位で区切り，その区間内で 1 回以上の Commit 記録があれば，1 単位として定義した．Person-Months と区別するため，ここでは Effort Person-Months と呼ぶ．

OSS 開発コミュニティは，要員を雇用していないため，稼働率をマネージできない．それゆえ，稼働率に代わる計測項目が必要となる．そこで，ジニ係数と呼ばれる指標に着目した．稼働率は，Effort Person-Months や Effort Person-Hours を Total Months か Total Hours で割った値である．Occ: 稼働率 (Occupancy rate) は，Ef: 全 Effort Person-Months の平均，To: Total Months とすれば，式 (1) で表される．

$$Occ = \frac{Ef}{To} \tag{1}$$

雇用関係が存在しない OSS 開発コミュニティでは，Total Months を固定的に決めることが出来ないので，Effort Person-Months の分布の偏りを表すジニ係数を

用いる．*Gini*: ジニ係数は，Effort の分布を $L(F)$ とすると式 (2) で表される．

$$Gini = \frac{\frac{1}{2} - \int_0^1 L(F)dF}{\frac{1}{2}} = 1 - 2\int_0^1 L(F)dF \quad (2)$$

ジニ係数を用いて稼働率に相当する値を出すとすると，式 (3) で表される．

$$Occ \fallingdotseq 1 - Gini \quad (3)$$

$L(F)$ の値は，OSS 開発コミュニティのある測定期間における，Contoributor の Effort Person-Months を用いる．i 番目の Contributor C_i の Effort Ef_{C_i} は，測定期間において活動が記録されている月数とする．たとえば，1 年間の測定期間において，3 ヶ月間に 5 回の Commit 行った場合，Commit 回数の “5” ではなく，Commit 記録のある 3 ヶ月の “3” を Effort Person-Months とする．活動量は，GitHub API v3 で取得した Commit 記録から求める．全 Contributor の Ef_{C_i} が $L(F)$ に対応する．ジニ係数は，Ef_{C_i} のベクトルから R の統計パッケージを用いて求める．

GitHub では，GitHub に登録した日より前から開発が始まっている Repository が存在し，取得したデータには登録日以前の記録も含まれている．そのため，分析対象とする Repository の開発期間はさまざまである．そこで，本研究では，それぞれの開発開始時点から 12 ヶ月単位で期間を区切り，1 年を単位として分析データを整形した．

ジニ係数を算出するための Effort Person-Months は，前述の定義に従い，次の手順で集計する．

1. Repository 単位で全期間の Contributor 毎の Commit 記録を収集
2. 収集した Commit 記録を年単位で分割
3. 年毎に月単位で Contributor 毎の Effort Person-Months を算出
4. Contributor 単位で Effort Person-Months を集計

年単位で集計，ジニ係数を算出した結果を時系列で視覚化することにより，OSS 開発コミュニティのライフサイクルの変化を観測する．

4. ライフサイクルの計測結果の視覚化

本章では，3 章で示した分析対象に対して計測，算出した結果を視覚化する手法を事例を基に示す．4.1 節で，マクロな観点からの計測結果の視覚化の事例を示し，4.2 節でミクロな観点からの計測結果の視覚化の事例を示す．

4.1. マクロな観点からの視覚化

本節では，マクロな観点からの視覚化の事例として，“compiler-explorer” コミュニティ [29] のデータを例に視覚化手法について述べる．この Repository は，総 Contributor 数，総 Star 数の両観点において，第 3 四分位以上のカテゴリに属しており，分析対象とした 1,707 件のデータの上位 25 % に属している．

表 2 に，3.2 節で示した手法により集計した値の総数と，算出したジニ係数の値を示す．

表 2: compiler-explorer コミュニティ

Year	Contributors	Total Commits	Gini Coefficient
2011	2	102	0.4803922
2012	3	32	0.5208333
2013	5	104	0.7115385
2014	3	114	0.6315789
2015	9	618	0.8137361
2016	21	1,184	0.8072716
2017	42	953	0.8189427
2018	51	756	0.8312066
All	99	3,863	0.9038247

表 2 より，2011 年は，2 人の Contributor が 102 件の Commit を行っており，ジニ係数は，約 0.48 であった．2 人の Contributor が同数の Commit をしていた場合には，ジニ係数の値は，0 となるため，片方の Contributor の開発活動が活発であったと読み取れる．また，2016 年には，Contributor は 21 人に増え，総 Commit 数は 1,184 件と増加している．この時のジニ係数は，約 0.81 と高い値になっており，Contributor が増えることにより，開発活動のばらつきが大きくなっていることが読み取れる．このように年単位で算出した値を，横軸に開発期間 (年) を取り，縦軸にジニ係数と Contributor 数を取り視覚化したグラフを図 4 に示す．

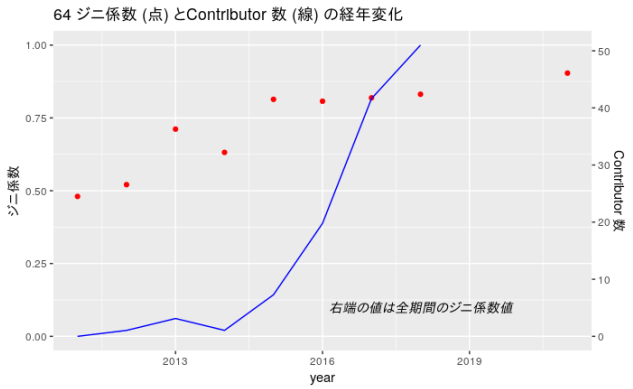

図 4: compiler-explorer コミュニティのライフサイクル

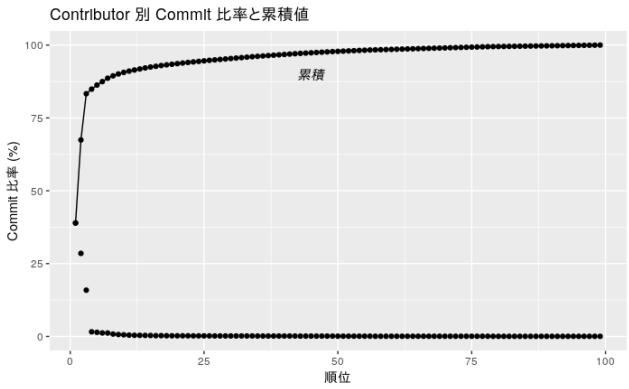

図 5: compiler-explorer コミュニティの Contributor 別の Commit 比率と累積値

図 4 の点が年毎に算出したジニ係数の値を示し，折れ線グラフは Contributor 数の変化を示している．右端の点は，全期間を対象として算出したジニ係数の値を示している．本手法により，OSS 開発コミュニティのライフサイクルをマクロな観点で視覚化することができる．

図 4 からは，compiler-explorer コミュニティの Contributor が 2015 年以降増加していることが読み取れる．一方で，ジニ係数の値は，Contributor が増加し始めた 2015 年に上昇して以降大きな変化は見られない．マクロな観点での視覚化だけでは，OSS 開発コミュニティの活動構造の詳細を読み取ることはできない．そのため，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合をミクロな観点から視覚化し，ジニ係数の変化の背景を探ることとした．次節にて，ミクロな観点からの視覚化について述べる．

4.2. ミクロな観点からの視覚化

本節では，ミクロな観点からの視覚化の事例として，前節同様に compiler-explorer コミュニティのデータを例に視覚化手法について述べる．ミクロな観点からの視覚化は，コアメンバを中心とした活動割合という観点から視覚化する．

図 5 は，横軸に，Contributor を Commit 数の多い順に上位 1 位から 100 位までを並べ，縦軸に Commit 数とその累積をパーセンテージで示したものである．

図 5 からは，全体の Commit 数の約 80% を 3 人の Contributor が行っていることが読み取れる．このことから，compiler-explorer コミュニティにおいて，Contributor 数が増加がジニ係数に大きな影響を及ぼさなかった要因が，Commit の多くは極少数のコアメンバが行っており，増加したメンバの Commit 数が少なかったためであることが明らかになった．

このように，本手法を用いてマクロな観点とミクロな観点での視覚化を組み合わせることにより，OSS 開発コミュニティのライフサイクルを確認することができる．

5. 考察

本研究では，ケーススタディとして用いた全 80 件の OSS 開発コミュニティについて，第 章で示した計測と視覚化を行った．本章では，ライフサイクルの例を示し，ライフサイクルの分類について考察する．

5.1. ライフサイクルの例

本節では，多様なライフサイクルの例として 2 件の OSS 開発コミュニティの事例を示す．

まず，“docs.larastars.cn” コミュニティ [30] の事例を図 6 と図 7 に示す．docs.larastars.cn コミュニティは，総 Contributor 数，総 Star 数の観点からは，いずれも下位 25 % 未満に属する Repository である．

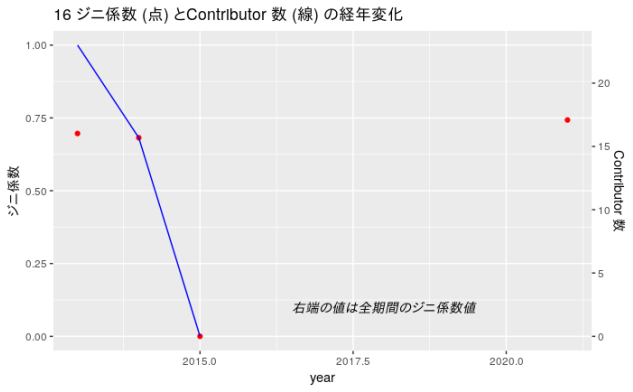

図 6: cocs.larastars.cn コミュニティのライフサイクル

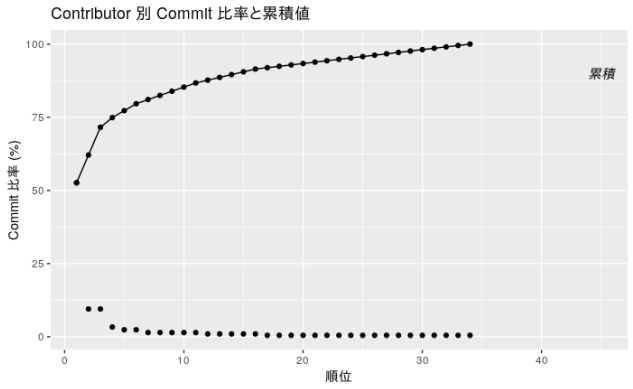

図 7: cocs.larastars.cn コミュニティの Contributor 別の Commit 比率と累積値

このコミュニティは，2013 年に開発を開始した当初は 23 人の Contributor が 107 件の Commit を行っており，ジニ係数は，約 0.70 であった．2014 年には Contributor が 16 人に減少し，総 Commit 数は 90 件と減少したが，ジニ係数は約 0.68 と大きな変化は見られない．図 7 より，cocs.larastars.cn コミュニティは 1 人の Contributor が総 Commit 数の半分を行っており，主に 3 人のコアメンバが開発を推進していたことが分かる．2015 年には，Contributor は 1 人となり，総 Commit 数も 13 まで減少し，図 6 の折れ線グラフが示す通り，以降の開発記録は見られない．

続いて，“u-boot” コミュニティ [31] の事例を図 8 と図 9 に示す．u-boot コミュニティは，総 Contributor 数は上位 25 % に属し，総 Star 数では，全体の 25 % 以上 50 % 未満に属する Repository である．

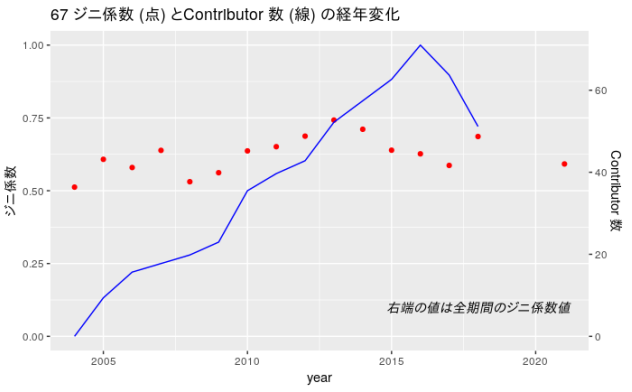

図 8: u-boot コミュニティのライフサイクル

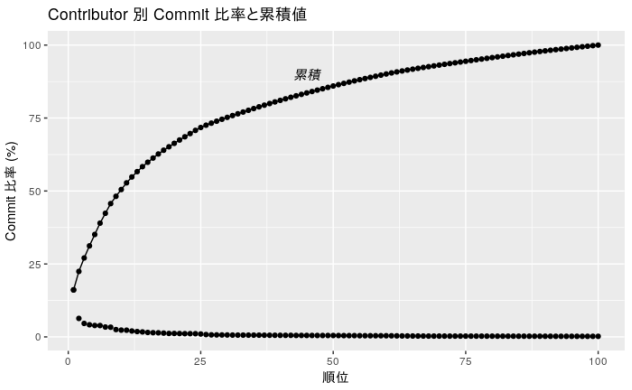

図 9: u-boot コミュニティの Contributor 別の Commit 比率と累積値

このコミュニティは，2004 年に 3 人の Contributor によって始まり，総 Commit 数は 108，ジニ係数は約 0.51 であった．図 8 からは，その後，Contributor が年々増加し，それに伴い総 Commit 数も増加していることが分かる．一方で，Contributor 数は 3 人から最大で 53 人まで増加しているが，ジニ係数の値は必ずしも Contributor 数の増減と共に高い値になっているわけではないことが分かる．図 9 より，これまでに示した OSS 開発コミュニティと比較すると，Contoributor 間の Commit 数の差異が小さいことが分かる．このコミュニティは，1 人の Contributor が全体の約 15 % の Commit を行い，それに続く Contributor が継続的に開発活動を続けていると読み取れる．

GitHub のサイト上で容易に閲覧可能な Commit 数，Contributor 数，Star 数などの値から，OSS 開発コミュ

ニティのライフサイクルや活動構造を推察することはできるが，本研究で提案した手法により，多様な OSS 開発コミュニティのライフサイクルと活動構造を視覚化することができることが明らかになった．

5.2. ライフサイクルの分類

本節では，視覚化情報を基にし，ライフサイクルパターンの分類を試みる．本研究で用いた 80 件の OSS 開発コミュニティは，継続して成長し続けるものや，成長後に衰退するもの，一度衰退して再び成長するものなど，多様であった．まず，これらの共通点を基に，OSS 開発のライフサイクルのステージを「開始期」,「成長期」,「維持期」の 3 つに分類できると考えた．

開始期 ：コミュニティの開始時期であり，参加メンバは少ない．この時期のジニ係数の値は小さい．

成長期 ：メンバが増加する時期であり，コミュニティへのメンバの参加・離脱が多く発生する．ジニ係数の値は大きくなる．

維持期 ：コミュニティへの新規参画者の減少，離脱者の増加により，持続的に活動しているコアメンバが残っている時期．ジニ係数の値は成長期より小さくなる．

本研究では，コミュニティのライフサイクルを対象としているため，開発開始後に Contributor が参加し，コミュニティと成長した OSS 開発プロジェクトである．OSS 開発プロジェクトの多くは，開始メンバ以外の Contributor が参加せず，コミュニティへと成長することなく，開始期を継続している．開始期は，Repository の登録者を中心とした開始メンバが中心となって開発しているため，ジニ係数の値は小さい傾向が観測された．成長期の新規参画者は，継続して参加し続けるタイプと，一時的に参加して離脱するタイプの 2 つのタイプに分類できる．成長期には，一時的に参加するメンバが大量に発生するため，全体のメンバ数が増加する．しかし，一時的に参加するメンバの一人当たりの活動量は少ないため，ジニ係数の値が大きくなる傾向が観測された．維持期は，新規参画者が減少し，離脱者が増加するため，開発を推進するコアメンバが中心となり活動を継続している．活動量の少ないメンバが減少するため，成長期よりジニ係数の値は小さくなる傾向が観測された．開発の終了に至る経緯は，必ずしも開始期，成長期，維持期と推移する訳ではなく，いずれのステージにおいても開発記録が見られなくなり，開発終了に至ったと想定されるライフサイクルが観測された．いずれの場合においても，開発を推進するコアメンバの存在が観測されなくなったタイミングで開発が終了している．このことから，OSS 開発コミュニティが終了するかどうかは，コアメンバが開発を持続するかどうかが大きく関係していると推察する．一方で，維持期であっても，新規参画者が増加し，再び成長期となるケースも多く観測された．これは，開発プロダクト利用者の増加などによる新たなサービスの提供や新規機能の開発着手などの要因が考えられる．OSS 開発コミュニティは，活動期限を持たず，また，業務ではないため，常に開発をし続けなければならないわけでもない．そのため，維持期や終了を迎えたとしても，メンバの知的好奇心や，利用者の要望などがモチベーションとなり，再び成長期を迎えることがあるのだと考える．

6. おわりに

本研究では，モチベーションを維持する要因を探るため，コミュニティの活動状況という観点から継続している OSS 開発コミュニティのライフサイクルの視覚化を試みた．視覚化は，コミュニティの開始から終了までをライフサイクルと定義し，GitHub をケーススタディ対象とし，GitHub API v3 を用いて，ライフサイクルの変化を測り，マクロとミクロの特性の 2 階層で視覚化を試みた．マクロな特性は，コミュニティ・メンバの活動記録を期間に分割し，活動量の不均衡状態を経済学の分野で用いられるジニ係数の変化として求め，視覚化した．ミクロな特性は，ジニ係数の変化には，開発を推進するコアメンバの活動が影響していることから，コアメンバを中心とした活動割合を視覚化した．本研究で提案する OSS 開発コミュニティのライフサイクルの視覚化手法を活用することにより，メンバのモチベーションなどについて新たな知見を得ることができる．また，OSS の利用者にとっては，視覚化した情報を活用することで，より安定したサービスやプロダクトを選択する際の一助になることが期待できる．本研究では，提案した手法で OSS 開発コミュニティを視覚化できることを示すと共に，視覚化により得られた観測値から，ライフサイクルのステージとして「開始期」,「成長期」,「維持期」といった分類を行った．次の課題は，考察したライフサイクル

ステージのさらなる検証とライフサイクルパターンの分類である．特に，OSS 開発コミュニティが成長する仕組みに着目して，引き続き調査，分析を行っていく所存である．

参考文献

[1] Linux, https://www.linux.com/ (accessed:2020/03/16)

[2] Apache, https://www.apache.org/ (accessed:2020/03/16)

[3] MySQL, https://www.mysql.com/ (accessed:2020/03/16)

[4] PHP, http://php.net/ (accessed:2020/03/16)

[5] Perl, https://www.perl.org/ (accessed:2020/03/16)

[6] Python, https://www.python.org/ (accessed:2020/03/16)

[7] Android, https://www.android.com/ (accessed:2020/03/16)

[8] Lee, James and Ware, Brent, “Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP,” Addison-Wesley Professional, 2003

[9] 竹田昌弘,「オープンソース・ソフトウェアとビジネスとの関係に関する考察」, 立命館経営学, Vol.44, No.3, pp.49–66, 立命館大学経営学会, 2005

[10] Project Management Institute，プロジェクトマネジメント知識体系ガイド (PIMBOK ガイド) 第 4 版，Project Management Institute，2009

[11] GitHub, https://github.com/ (accessed:2020/03/16)

[12] GitHub API v3, https://developer.github.com/v3/ (accessed:2020/03/16)

[13] Bitbucket, https://bitbucket.org/ (accessed:2020/03/16)

[14] 石井達夫,「Let’s Postgres：OSS の開発コミュニティってどんなところ？」，https://lets.postgresql.jp/documents/tutorial/oss_community/1 (accessed:2020:0523)

[15] 北山聡, 「組織内コミュニティの計量: ジニ係数とべき分布の視点から」, 東京経済大学 コミュニケーション学会, 2009

[16] Gastwirth, Joseph L, “The estimation of the Lorenz curve and Gini index,” The review of economics and statistics, pp.306–316, JSTOR, 1972

[17] 中村和之, 「経済指標の見方・使い方：所得格差を測る指標－ジニ係数とローレンツ曲線－」, http://www.pref.toyama.jp/sections/1015/ecm/back/2005apr/shihyo/ (accessed:2020/03/16)

[18] Dewan, Rajiv M and Freimer, Marshall L and Seidmann, Abraham and Zhang, Jie, “Web portals: Evidence and analysis of media concentration,” Journal of Management Information Systems, Vol.21, No.2, pp.181–199, Taylor & Francis, 2004

[19] Jensen, Chris and Scacchi, Walt, “Modeling recruitment and role migration processes in OSSD projects,” ProSim05, Vol.39, 2005

[20] Jensen, Chris and Scacchi, Walt, “Role migration and advancement processes in OSSD projects: A comparative case study,” roceedings of the 29th international conference on Software Engineering, pp.364–374, IEEE Computer Society, 2007

[21] 伊原彰紀, 大平雅雄, まつ本真佑, 亀井靖高, 松本健一, 「複数のサブコミュニティを有する OSS コミュニティを対象としたネットワーク分析」, 情報処理学会 マルチメディア, 分散, 協調とモバイル (DICOMO2008) シンポジウム, pp.297–303, 2008

[22] SourceForge, https://sourceforge.net/ (accessed:2020/03/16)

[23] Guimarães, André LS and Korn, Helaine J and Shin, Namchul and Eisner, Alan B, “The life cy-

cle of open source software development communities," Journal of Electronic Commerce Research, Vol.14, No.2, pp.167, 2013

[24] Mockus, Audris and Fielding, Roy T and Herbsleb, James D, "Two case studies of open source software development: Apache and Mozilla," ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.11, No.3, pp.309–346, ACM New York, NY, USA, 2002

[25] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦, 「大規模オープンソース・ソフトウェアプロジェクトにおける開発効率の計測」, 電気学会論文誌 C (電子・情報・システム部門誌), Vol.138, No.8, pp.1011–1019, 一般社団法人 電気学会, 2018

[26] RIOT-OS/RIOT, https://github.com/RIOT-OS/RIOT/ (accessed:2020/05/23)

[27] Ayako Masuda, Tohru Matsuodani, Kazuhiko Tsuda, "Team Activities Measurement Method for Open Source Software Development Using the Gini Coefficient," 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp.140–147, IEEE, 2019

[28] Thung, Ferdian and Bissyande, Tegawende F and Lo, David and Jiang, Lingxiao, "Network structure of social coding in GitHub," 2013 17th European Conference on Software Maintenance and Reengineering, pp.323–326, IEEE, 2013

[29] mattgodbolt/compiler-explorer, https://github.com/mattgodbolt/compiler-explorer/ (accessed:2020/03/16)

[30] larastarscn/docs.larastars.cn, https://github.com/larastarscn/docs.larastars.cn (accessed:2020/03/16)

[31] fourkbomb/u-boot, https://github.com/fourkbomb/u-boot (accessed:2020/03/16)