DBSJ Journal, Vol.12, No.3
February 2014

ソーシャルメディア上の協働: ソーシャルコーディングにおける成功するプロジェクトの要因分析

Collaboration on Social Media: Analyzing Successful Projects on Social Coding

吉川 友也♥　岩田 具治♦　澤田 宏♣

Yuya YOSHIKAWA　Tomoharu IWATA
Hiroshi SAWADA

ソーシャルコーディングサイト(SCS)は，ソフトウェア開発をWeb上で共有するためのサービスである．SCSの特徴は，ソーシャルな環境で開発が行われる点である．すなわち，SCSでは開発者をつなぐソーシャルネットワークが構築されており，それぞれの開発者が行ったプロジェクト作成やお気に入り登録等の行動が開発者間で共有される．また，プロジェクト外部の開発者もプロジェクトを改善できる．このようなソーシャルな環境がプロジェクトの人気度や活発度に影響を与える可能性がある．では，どのようなプロジェクトがSCSで成功するのだろうか．本論文ではGitHubのデータを使い，お気に入り登録の多いプロジェクトやコードの更新頻度の高いプロジェクトなどに見られる特徴を分析する．

Social Coding Sites (SCSs) are services for sharing software development processes on the Web. The SCSs enable their users to develop softwares in a social environment, i.e., the SCSs have social networks which developers are connected and activities such as project creation and bookmarks of softwares are shared among the developers. Moreover, developers outside projects contribute to the projects by reporting and fixing bugs. Such a social environment could influence the popularity and activity of projects. What are differences between successful and unsuccessful projects on SCSs? In this paper, we use GitHub data to investigate the characteristics of successful projects which are frequently updated by inside developers, bookmarked and revised by outside developers.

1. はじめに

ソーシャルコーディングサイト(SCS)は，ソフトウェア開発プロジェクトをWeb上で共有するためのサービスである．代表的なSCSとしてGitHubとBitbucketなどがあり，これらのサービスは，Web上での開発やプロジェクト管理を支援する機能を携えている．利用者数は年々増加し，GitHubでは300万ユーザ（2013年1月16日時点），Bitbucketでは100万ユーザ（2013年6月4日時点）が利用している．

SCSでは，開発者は自らがオーナーとなるプロジェクトを作り，その中でソフトウェア開発を行う．SCSの特徴は，ソーシャルな環境で開発が行われる点である．すなわち，SCSでは開発者をつなぐソーシャルネットワークが構築されており，プロジェクト作成やお気に入り登録等の行動が開発者間で共有される．

本論文の目的は，ソーシャルコーディングにおけるプロジェクト成功の要因を見つけ出すことである．成功の指標として，プロジェクトの活発度を表すコードの更新頻度，プロジェクトの人気度を表すお気に入り数，プロジェクトのソーシャル度を表す外部からのコード変更要求回数を用いる．成功に繋がる要素を発見するために，1)プロジェクトチームの構成，2)プロジェクト外部の開発者への対応，3)開発するソフトウェアの種類に着目する．1)プロジェクトチームの構成では，プロジェクトメンバーが構成するネットワークの性質と成功の指標との相関を見る．また，メンバーの人数と活発度との関係からプロジェクトを進める上で効率的なメンバー数に関して議論する．さらに，仕事量の偏りと成功の関係についても議論する．2)プロジェクト外部の開発者への対応では，外部からのコード変更要求への対応の仕方が成功とどのように関係するのかを分析する．3)開発するソフトウェアの種類では，プロジェクトが作るソフトウェアと成功の関係を分析するためにREADMEテキストをトピック解析し，成功するプロジェクトとトピックの関係を分析する．

本研究は，ソーシャルコーディングにおける成功するプロジェクトを様々な観点から分析した最初の研究である．我々の研究では，30万以上の個人や組織から成る幅広いプロジェクトのデータを使い分析を行っているため，本論文の結果は，ソーシャルコーディングに限らず，他の多くのプロジェクトに対しても有益な知見である．

以下では，2節でソーシャルコーディングの説明と使用するデータセットの説明を行う．3節では，ソーシャルコーディングの世界の概観を知るための基本分析を行い，4節から6節で成功に繋がる要因を分析する．7節では関連研究を紹介し，最後の8節で結論を述べる．

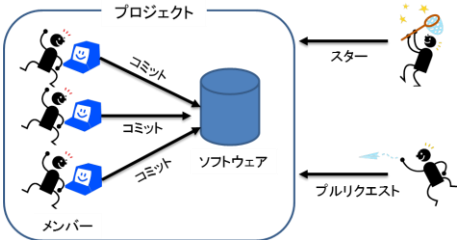


図1: ソーシャルコーディングの概略図

2. ソーシャルコーディング

2.1 ソーシャルコーディングとは

ソーシャルコーディングはWeb上のソーシャルな環境において行われるソフトウェア開発である．図1は，ソーシャルコーディングの概略図を示す．ソーシャルコーディングにおける開発は以下のように行われる．開発者（もしくは組織）は自らがオーナーとなるプロジェクトを作成する．そして，開発者はコードの追加・変更を行い，リポジトリと呼ばれる

♥ 学生会員　奈良先端科学技術大学院大学情報科学研究科
yuya-y@is.naist.jp
♦ 非会員　NTTコミュニケーション科学基礎研究所
iwata.tomoharu@lab.ntt.co.jp
♣ 非会員　NTTサービスエボリューション研究所
sawada.hiroshi@lab.ntt.co.jp

データベースにコードを追加して，ソフトウェアを作り上げる．このコードを追加する作業を**コミット**と呼ぶ．ソーシャルコーディングにおける特徴の一つは，コミットの履歴などが Web 上で可視化されている点である．これによって，開発者間のやり取りやコードの変更が理解しやすくなっている．ここではオーナーが一人のケースを考えたが，オーナーと同等の権限を持つ開発者を複数人設定することも可能である．さらに，ソーシャルコーディングでは，プロジェクトのメンバーでない人も開発に参加できる．開発者は追加したいコードと共に**プルリクエスト**を送ることによって，自らがオーナーではないプロジェクトのソフトウェアの開発に参加できる．ただし，そのコードがプロジェクトに取り込まれるかどうかは，プロジェクトのメンバーの意思決定によって決まる．

2.2 データセット: GitHub

本論文では，代表的なソーシャルコーディングサイトである GitHub のデータを使用する．データは GitHub Archive と GitHub API を利用し取得した．GitHub Archive から取得したデータには，2011 年 2 月から 2013 年 5 月までに行われたコミットやプルリクエストなどの公開されているイベントがすべて含まれている．その中から 30 回以上コミットされたプロジェクトを抽出し，各プロジェクトのメンバー情報やコミット，プルリクエスト，プロジェクトのブックマークを意味するスターなどの必要な情報を収集した．また，GitHub では好きな開発者の行動を追うために，フォローと呼ばれる仕組みが存在する．この仕組みによって作られたネットワークを構築するために，一回以上コミットかフォローした経験のある開発者の情報を収集した．加えて，ソフトウェアの特徴を知るために，各プロジェクトの README テキストを取得した．表 1 は収集した GitHub データセットの統計量を示す．

表 1: 収集した GitHub データセットの統計量

	数量
プロジェクト数	317,077
総コミット数	41,720,139
総スター数	5,164,934
総プルリクエスト数	1,903,595
開発者数	1,381,121
総フォロー数	1,885,266

2.3 成功を表す指標

プロジェクトの成功を表す指標として以下の3種類を使用する．これらは基本的にプロジェクトの経過時間と共に増えるため，プロジェクトの経過日数で割り正規化する．

コミット数(Commit)．コミット数はコードやドキュメントなどの追加や変更の回数を表す．従って，プロジェクトの活発度を示す指標である．

スター数(Star)．スター数はプロジェクトにスターを付けた開発者の数を示す．スターは基本的にブックマークの役割を果たしているため，スター数の多さはプロジェクトの人気度を示す．

プルリクエスト数(PullReq)．プルリクエスト数はプロジェクト内部のメンバー以外からのコードの変更要求の回数を表す．従って，プロジェクトのソーシャル度を表している．

3. 基本分析

まず，ソーシャルコーディングの世界の概観をつかむため，以下の問いに関する分析を行う．

1. プロジェクトはどれくらいの頻度で更新されるのか？
2. プロジェクトはどれくらいソーシャルから見られているのか？
3. プロジェクトはどのように成長するのか？
4. 開発者間はどのように繋がっているのか？

3.1 プロジェクトの統計量

問(1),(2)に答えるために，プロジェクトの Commit, Star, PullReq の統計量の分布を調査する．

表 2: 一日平均コミット数，スター数，プルリクエスト数の統計量

	Commit	Star	PullReq
平均	0.62	0.05	0.02
標準偏差	3.47	0.76	0.66
中央値	0.22	0.00	0.00
最大値	1201.67	184.84	298.03
最小値	0.04	0.00	0.00

表 2 はそれぞれの指標における統計量を示す．一日あたりのコミット数，スター数，プルリクエスト数の平均値はそれぞれ，0.62，0.05，0.02 であった．しかし，標準偏差や最大値，最小値を見て分かるように，これらはプロジェクトによるばらつきが大きい．また，スター数とプルリクエスト数の中央値が 0 となっている．実際，約 53%のプロジェクトはスター数が 0，約 70%はプルリクエスト数が 0 であり，オーナー以外の開発者によって評価されたり変更されるプロジェクトは少ないことが分かる．

3.2 プロジェクトの成長パターン

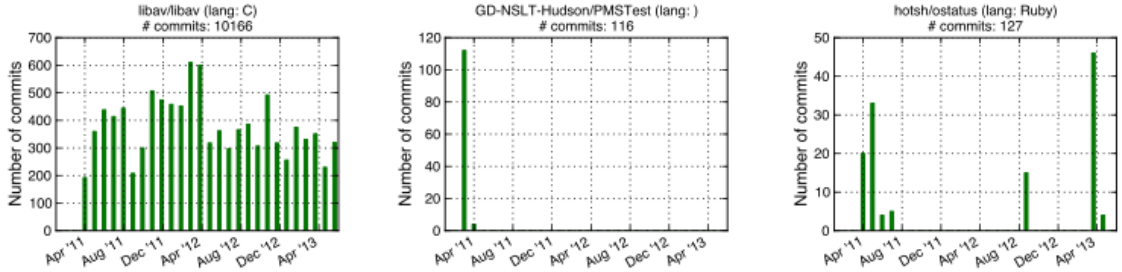
次に，問(3)に答えるために，プロジェクトがどのように成長していくのかを分析する．分析のために，200 のプロジェクトをランダムに抽出し，3 種類の成長パターンに手作業で分類した．図 2 はそれぞれの成長パターンの具体例を示す．以下では，それぞれの成長パターンの特徴とパターン出現の割合を示す．

- 定期更新型（10%）：常にコミットが行われているプロジェクト．
- 一回集中型（45%）：プロジェクト開始から 1，2 ヶ月でプロジェクトの成長が止まるプロジェクト．
- 複数回集中型（45%）：何度か集中してコミットが行われるプロジェクト．

なお，定期更新型はメンバー数の多いプロジェクトでよく見られ，一回集中型や複数回集中型はメンバーが 1 人のプロジェクトで多く見られる傾向がある．

次に，プロジェクトはどれくらい続くのかを分析する．図 3 は，総コミットの 90%に到達するまでのプロジェクトの経過月の分布を示す．プロジェクトの約 20%は開始から 1 ヶ月以内，約 40%は 3 ヶ月以内でコミットの 90%を終えていることが分かる．一方で，約 20%は 20 ヶ月以上掛かって総コミットの 90%に到達しており，長期間に渡って更新されるプロジェクトも一定数存在することが分かる．

3.3 フォローネットワーク



(a) 定期更新型　(b) 短期集中型　(c) 複数回集中型

図 2: プロジェクトの成長パターンの具体例

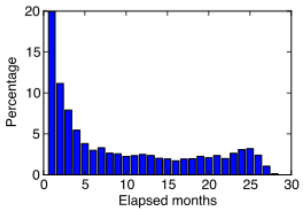


図 3: 総コミットの 90%に到達するのにかかった月数

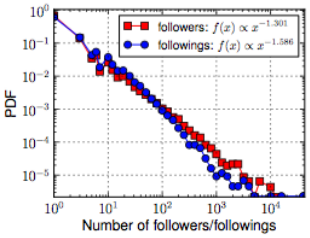


図 4: フォローネットワークの次数分布

最後に，フォローネットワークを分析することによって開発者の繋がり方の特徴を分析する．

次数分布. 図 4 はフォローネットワークの次数分布を示す．横軸は次数，縦軸は正規化した頻度(PDF)を表す．また，赤の四角点はフォロワー数（入次数）を表し，青の丸点はフォロー数（出次数）を表す．ソーシャルネットワークの次数分布はべき乗則 $f(x) \propto x^{-\alpha}$ に従うとされており，多くのネットワークで α の値が計算されている．フォローネットワークでは，フォロワー数分布は α =1.301，フォロー数分布は α =1.586 である．先行研究では，Twitter のフォローネットワークは α =2.276[7]，Mixi の友人ネットワークは α =2.4 程度[14]と報告されており，GitHub のフォローネットワークの α は比較的小さいことが分かる．これは，芸能人のような誰もが知っているユーザがネットワークに存在しないことが一因であると考えられる．また，$x=10^2$ 付近からフォロワー数の PDF がフォロー数の PDF よりも大きくなっている．多くのユーザをフォローするためには大きな労力を必要とするため，$x=10^2$ 以降では，フォロー数の PDF がべき乗則から下に外れている．一方，フォロワー数は所謂 The rich get richer 効果により，多数のフォローを獲得するユーザが自然と発生する．このような現象は Twitter のフォローネットワークでも見られる[7].

相互リンク率. 相互リンク率はネットワーク上の双方向リンクの割合を表す．GitHub のフォローネットワークにおける相互リンク率は 19.1%であり，これは Twitter の 22.1% [7] よりも小さい値である．これはフォローをコミュニケーションというよりは，情報収集のために行っているためだと考えられる．

4. チーム構成と成功

この節では，プロジェクトのチーム構成や人数がプロジェクトの成功にどのように繋がるかを分析する．ここで，プロジェクトメンバーは，表 1 で示した全開発者の中で，そのプロジェクトの変更権限を持つ開発者とする．従って，プルリクエストを介してプロジェクトにコミットした開発者はメンバーには含まれない．これはプロジェクトのオーナーが調整できる要素のみを研究対象とするためである．

プロジェクトメンバー間の仕事上の繋がりを表現するために，二つ以上の同じプロジェクトに参加した開発者間でリンクを張り，各プロジェクトの協働ネットワークを構築する．上記の手順で構築したプロジェクトの協働ネットワークは，そのプロジェクト以外での協働関係を反映したものとなっている．この協働ネットワークを使って，プロジェクト内のチーム構成の特徴量を抽出する．表 3 は抽出した特徴量の一覧を示す．

表 3: チーム構成を表す特徴量

特徴量	説明
Member	変更の権限を持つメンバー数
EdgeDense	協働ネットワークにおけるリンク数をノードペアの組み合わせの数で割った値
Cluster	協働ネットワークのクラスター係数[13]
FracComp	協働ネットワークのノード数に対する連結成分の数の比
PathLen	協働ネットワークの平均最短パス長

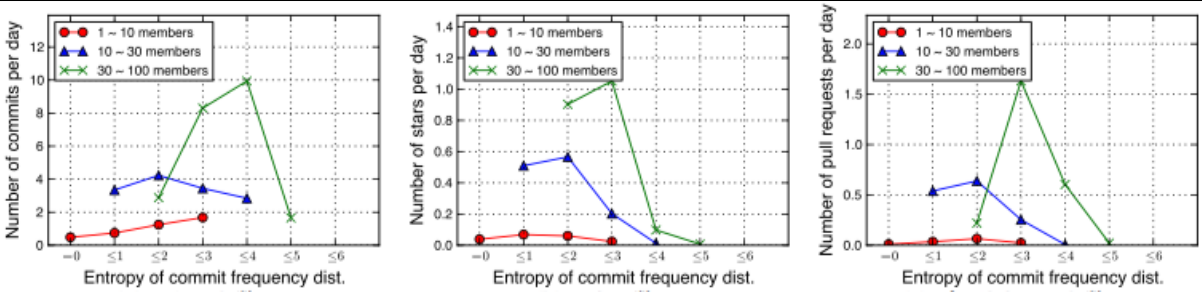
4.1 相関分析

表 4 はチーム構成の特徴と成功の指標の間のケンドールの順位相関係数を示す．

表 4: チーム構成と成功の間のケンドールの順位相関係数

	Commit	Star	PullReq
Member	0.16	0.11	0.18
EdgeDense	0.09	0.10	0.16
Cluster	-0.04	0.08	0.13
FracComp	-0.09	-0.11	-0.17
PathLen	0.11	0.15	0.16

まず，内部メンバーの数 Member の多さと成功の各指標はある程度相関があることが見て取れる．メンバー数が多いことによってコミット数が増えることは自然ではあるが，外部からの評価であるスターやプルリクエストの獲得にも繋がることが示唆された．次に，EdgeDense との相関を見る．EdgeDense はプロジェクト内の繋がりの強さを表す指標である．表より，EdgeDense は全ての成功の指標において正の相関を持ち，比較的強い相関があったのは PullReq であることが分かる．この結果は，FracComp が成功と負の相関を持つことからも説明できる．FracComp は協働ネットワークがどれだけ分断されているかを示す．従って，協働ネットワークが分断されていない方が成功するという結果になって



(a) コミット数　(b) スター数　(c) プルリクエスト

図 5: メンバーの貢献分散度と成功の関係

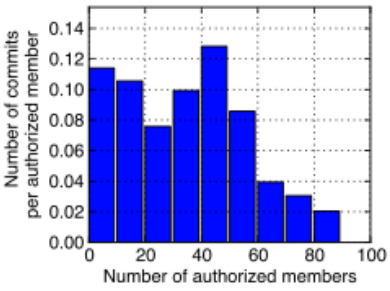


図 6: メンバー数とコミット数の関係

おり，繋がりの強さが成功にはある程度重要であることが示唆される．その一方で，PathLen と成功の各指標は正の相関を持つことは興味深い．

PathLen は全ての開発者ペア間のネットワークにおける平均距離を示すため，EdgeDense が大きいプロジェクトでは PathLen は小さくなる傾向がある（EdgeDense と PathLen の相関係数は-0.34）．PathLen が大きいネットワークの特徴の一つは，多くのメンバーと繋がるハブがいないことである．以上の知見をまとめると，プロジェクト内でハブとなるような開発者はできるだけ作らず，ネットワークは分断させず開発者間の繋がりを保つことが重要であると考えられる．

4.2 効率的なメンバー数

チームによる共同作業における最適なメンバー数の研究が多くなされている[8,5,6]．ここでは，プロジェクトの活発度を表すコミット数に焦点を当てて，ソーシャルコーディングのプロジェクトにおける効率的なメンバー数の分析を行う．図 6 はメンバー数とコミット数の関係を表す．横軸はプロジェクト内のメンバー数で，縦軸はメンバー1 人あたりのコミット数の中央値である．また，1 つのビンの幅は 10 人である．まず，全体的な傾向として，メンバー数が増えることによって 1 人あたりのコミット数が減少することが分かる．この結果の原因として，1)仕事が分散してメンバー1 人 1 人の仕事量が減った，2)仕事をしないメンバーが現れたことが考えられる．いずれの理由であっても，メンバーの力を効率的に生かせていないことになる．特にメンバー数が 60 人以降で，顕著な減少傾向が見られる．すなわち，プロジェクトの活発度を上げるために 60 人以上のメンバーを登用することは効率的ではないと考えられる．

4.3 メンバーの貢献の偏り

次に，プロジェクト内でのメンバーのコミット数の偏りが成功に対して与える影響を分析する．コミット数の偏りに関して，2 つの極端なケースが想定できる．

1. メンバーの一部がほとんどのコミットを行う
2. 全てのメンバーで平等に分担してコミットを行う

では，どちらのケースに当てはまるプロジェクトが成功するのだろうか？各プロジェクトのコミットの偏りを定量化するために，以下のエントロピーを計算する．

$$\mathrm{Entropy} = -\sum_{m \in M} \theta_m \log \theta_m$$

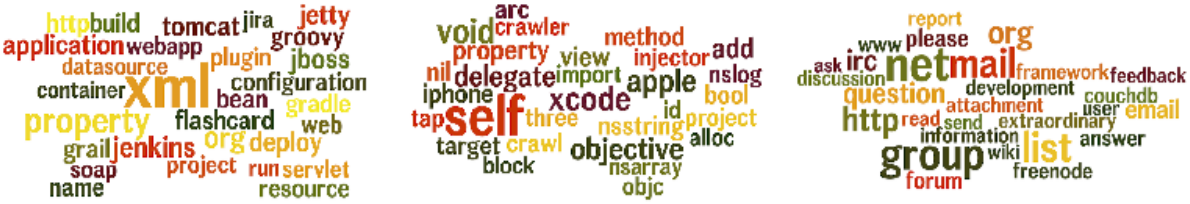
ここで，M はプロジェクトのメンバー集合，θ_m はそのプロジェクトにおけるメンバー m のコミットの割合を表す．Entropy は一部のメンバーにコミットが集中するとき 0 に近い値となり，コミットが分散するとき大きな値となる．

図 5 はメンバーの貢献分散度 Entropy と成功の各指標の関係を示す．図 5(a)はコミット数との関係を示す．メンバーが 1-10 人と小さいプロジェクトでは，コミットが分散していた方が全体としてのコミット数を増加させることが見て取れる．これは，メンバーが 1 人から 10 人に増加しても一人あたりのコミット数はあまり減少しないため，その規模のプロジェクトではメンバー全員で協力することによってプロジェクトの活発度を上げることができると考えられる．メンバーが 10-30 人の中規模プロジェクトを見ると，メンバーが 10 人以下のプロジェクトよりもコミット数が多くなるものの，Entropy が大きくなってもコミット数が増えないことが分かる．さらに，メンバーが 30--100 人の大規模プロジェクトでは，一部のメンバーにコミットが集中する場合にはその一部のメンバーのコミットの頻度も低く，反対にコミットが分散する場合でも全員のコミット数が少なくなり，中規模プロジェクトのコミット数よりも少なくなる結果が得られた．

興味深いことに，メンバーの貢献分散度は外部評価であるスター数やプルリクエスト数とも関係がある．図 5(b)は獲得したスター数との関係を示す．これを見ると，プロジェクトの規模によらず，Entropy が大きくなると獲得スター数が減ることが分かる．図 5(c)のプルリクエスト数では，スター数で見られた傾向とともに，メンバー数 30 以上で一部のメンバーにコミットが集中したプロジェクトはプルリクエストが得られていないことが分かる．

5. プロジェクト外部の開発者への対応と成功

4 節では，プロジェクト内部のメンバーをどのように構成すれば良いかについて言及した．この節では，プロジェクトメンバーがどのような活動をし，プロジェクト外（ソーシャル）からのプルリクエストにどのように対応すると成功するのかについて分析する．表 5 は，本研究で扱うプロジェクト外部の開発者への対応を表す特徴量の一覧を示す．ここで，プルリクエストへの反応とは，プルリクエストの是非に関して議論することやプルリクエストを取り込む（マージする）ことを意味する．



(a) コミット数　(b) スター数　(c) プルリクエスト

図 7: 最大の回帰係数を持つトピックに含まれる単語．単語の文字サイズは各トピックにおける頻度確率を表す．

表 5: プロジェクト外部の開発者への対応を表す特徴量

特徴量	説明
ResponseRate	プルリクエストに対してメンバーが 1 ヶ月以内に反応した割合
MergeRate	プルリクエストをプロジェクトに取り入れた割合
ResponseTime	プルリクエストに対してメンバーが最初に反応した経過時間の平均
PullReqByInside	メンバーによるプルリクエストの数

表 6 はプロジェクト外部の開発者への対応と成功の間のケンドールの順位相関係数を示す．まず，プルリクエストへの対応の良さを表す ResponseRate とスター数やプルリクエスト数は相関があることが分かる．すなわち，プロジェクト外の開発者にプルリクエストをしてもらうためには，プルリクエストされたときには無視せず対応することが重要であり，そのことによってプロジェクトの人気度も上昇することが示唆される．では，プルリクエストへの反応の早さは重要なのだろうか．表 6 における ResponseTime の行を見ると，対応の遅さとスター数やプルリクエスト数の間には比較的小さいが正の相関があるとの結果が得られている．プルリクエストに対して遅く対応することが望ましいとは言えないが，意識的に素早く対応する必要はなさそうである．

表 6: プロジェクト外部の開発者への対応と成功の間のケンドールの順位相関係数

	Commit	Star	PullReq
ResponseRate	0.01	0.39	0.83
MergeRate	0.09	0.37	0.68
ResponseTime	-0.04	0.20	0.14
PullReqByInside	0.14	0.27	0.31
Commit	1.00	0.06	0.03

プルリクエストはプロジェクトメンバーでも行うことができる．このようなことを行うメリットは，コードの変更がオープンな議論の下で行うことができる点である．ソーシャルコーディングにおいては，プロジェクトメンバー以外が開発に参加するケースがあるため，コード変更の透明性は重要だと考えられる．このような開発の仕方は，日本ではチケット駆動開発と呼ばれている．PullReqByInside と成功の関係を見ると，Star や PullReq と正の相関があることが分かる．なお，PullReq はソーシャルからのプルリクエスト数のみを計算している．Commit は，Star や PullReq と相関がないことから，内部でもプルリクエストを利用して開発を行うことによりソーシャルからの人気度を高めたり，多くのバグ報告やコードの修正を受けることができる．

6. 開発するソフトウェアの種類と成功

最後に，開発するソフトウェアがプロジェクトの成功に与える効果を分析する．残念ながら，GitHub はソフトウェアの特徴を表すカテゴリやタグ情報は提供していない．従って，我々は各プロジェクトの README テキストに基づいて，プロジェクトを特徴付ける．README には，プロジェクトの説明やプロジェクトが作成するソフトウェアの使い方などが記述されている．GitHub では，README はプロジェクトのトップページに表示され，プロジェクト外部の開発者がプロジェクトで何を行っているのか理解するために不可欠な情報である．

自動的にプロジェクトの特徴を知るために，README テキストに対して潜在的ディリクレ配分法(LDA)[2]を適用する．LDA を適用することにより，各プロジェクトのトピック分布が得られる．その後，トピック分布を説明変数，成功指標を目的変数として特徴選択付き線形回帰モデル Lasso[12]を学習し，各トピックの回帰係数を調べることにより，成功するプロジェクトと関係のあるトピックを抽出する．分析にあたり，LDA のトピック数 K=80，Lasso の正則化パラメータ $\alpha=10^{-4}$ と設定した．

図 7 は，各成功指標に対して最大の回帰係数を持つトピックに含まれる単語を示す．コミット数の結果を見ると，Web アプリケーション特有の単語（例: webapp, servlet, tomcat）がよく表れることが分かる．加えて，run, build, configuration といった単語もあり，これはコマンドラインツールのインストールの仕方を示すものである．スター数の結果では，apple, iPhone, objc といった iOS や MacOS のためのソフトウェアを表す単語が出現しており，Apple 製品上で動くソフトウェアを開発するプロジェクトが人気を集めていることが分かる．興味深いことに，プルリクエスト数の結果では，mail, ask, discussion, question のようなコミュニケーションの方法を表す単語がよく含まれる．これは，プロジェクト外部の開発者に内部とのコミュニケーションの取り方を記載することによって，多くのプルリクエストが得られることを示している．

7. 関連研究

ソフトウェア工学では，GitHub から取得したデータを使った研究がいくつか存在する．例えば，Allamanis らは，GitHub から取得した大量のソースコードを使って言語モデルを学習することにより，コードの提案の性能が向上することを示した[1]．Pham らは，GitHub ユーザに対してインタビューとアンケート調査を行い，GitHub における開発プロセスの透明性が開発者のテストの仕方に影響を与えていると報告した[9]．Ell は，GitHub 特有の大規模な協働活動によって起きるバグを発見するために，コードの変更履歴から作成した開発者のネットワークに基づいて間違ったコードの変更を行う可能性のある開発者ペアを発見する方法を提

案した[4]．データマイニング的視点では，Thung らは，GitHub 上の約 10 万プロジェクトから協働関係から開発者ネットワークとプロジェクトネットワークを作成し，これらのネットワーク的性質の調査と PageRank を使ったプロジェクトと開発者のランキングを行った[11]．しかし，GitHub 上のプロジェクトのチーム構成や開発者行動に焦点を当ててプロジェクトの成功要因を分析した研究は本論文が初めてである．

ソーシャルコーディングと類似するサービスとして，SourceForge.net などのオープンソースホスティングサービスがある．SourceForge.net 上のオープンソースプロジェクトを対象に，人気度やバグ修正にかかる時間を決める要因を分析した研究が存在する[10, 3]．しかし，ソーシャルコーディングは SourceForge.net とは異なる環境である．第一に，ソーシャルコーディングはソースコードだけでなく開発過程もオープンである．また，プロジェクトの背後には開発者のソーシャルネットワークがあるため，開発者間の相互作用によってプロジェクトの活発度や人気度が左右される可能性がある．従って，SourceForge.net のデータを使った分析とは異なる．

8. おわりに

ソーシャルコーディングにおけるプロジェクト成功の要因を見つけ出すために，GitHub のデータ分析を行った．特に，1)プロジェクトチームの構成，2)プロジェクト外部の開発者への対応，3)開発するソフトウェアの種類に着目し，それぞれと成功の関係を調べた．なお，本論文ではプロジェクトの活発度を一日あたりのコミット数，プロジェクトの人気度を一日あたりのスター数，ソーシャル度を一日あたりのプルリクエスト数とし，これらを成功の指標として用いた．

本研究を通じて得られた知見を簡単に以下にまとめる．

1. プロジェクトメンバーの多いプロジェクトは活発度，人気度，ソーシャル度も大きくなる傾向があるが，メンバー数が 60 人を超えると，一人一人の活発度は下がっていく．
2. メンバー間の繋がりの強いプロジェクトは，ソーシャル度が高くなる傾向がある．
3. メンバー数が 10 人以下ならば，仕事が分散しているときに活発度が高い．メンバー数が 30 人以上ならば，適度に分散しているときに活発度が高い．
4. プルリクエストへの対応の良いプロジェクトは人気度とソーシャル度が高い．
5. プロジェクト外部にも開発のやり取りが可視化されることにより，人気度やソーシャル度が高くなる．
6. プロジェクト内部とのコミュニケーションの方法を記述するプロジェクトはソーシャル度が高い．

本研究は，ソーシャルメディア上でのプロジェクトのより良い進め方の理解に向けての最初の一歩である．今後の研究では，共同作業によって百科事典を作成する Wikipedia などの他の協働活動プラットフォームとの比較分析を行う．

[謝辞]

本研究は JSPS 特別研究員奨励費の助成を受けたものです．

[文献]

[1] Allamanis, M. and Sutton, C.: Mining Source Code Repositories at Massive Scale Using Language Modeling, MSR, pp. 207–216 (2013).

[2] Blei, D. M., Ng, A. Y. and Jordan, M.: Latent Dirichlet Allocation, The Journal of Machine Learning Research, Vol. 3, No. 4-5, pp. 993–1022 (2003).

[3] Crowston, K., Howison, J. and Annabi, H.: Information Systems Success in Free and Open Source Software Development: Theory and Measures, Software Process: Improvement and Practice, Vol. 11, No. 2, pp. 123–148 (2006).

[4] Ell, J.: Identifying Failure Inducing Developer Pairs within Developer Networks, ICSE (2013).

[5] Fan, Z.-P., Feng, B., Jiang, Z.-Z. and Fu, N.: A Method for Member Selection of R&D Teams Using the Individual and Collaborative Information, Expert Systems with Applications, Vol. 36, No. 4, pp. 8313–8323 (2009).

[6] Hoegl, M.: Smaller teams–better teamwork: How to keep project teams small, Business Horizons (2005).

[7] Kwak, H., Lee, C., Park, H. and Moon, S.: What is Twitter, a social network or a news media?, Proceedings of The 19th International Conference on World Wide Web, pp. 591–600 (2010).

[8] Mueller, J. S.: Why Individuals in Larger Teams Perform Worse, Organizational Behavior and Human Decision Processes, Vol. 117, No. 1, pp. 111–124 (2012).

[9] Pham, R., Singer, L. and Liskin, O.: Creating a Shared Understanding of Testing Culture on a Social Coding Site, ICSE (2013).

[10] Sen, R., Singh, S. S. and Borle, S.: Open source software success: Measures and analysis, Decision Support Systems, Vol. 52, No. 2, pp. 364–372 (2012).

[11] Thung, F., Bissyande, T. F., Lo, D. and Jiang, L.: Network Structure of Social Coding in GitHub, 17th European Conference on Software Maintenance and Reengineering, IEEE, pp. 323–326 (2013).

[12] Tibshirani, R.: Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society. Series B (Methodological), Vol. 58, No. 1, pp. 267–288 (1996).

[13] Watts, D. J. and Strogatz, S. H.: Collective dynamics of 'small-world' networks., Nature, Vol. 393, No. 6684, pp. 440–2 (1998).

[14] 松尾豊, 安田雪: SNS における関係形成原理：mixi のデータ分析,人工知能学会論文誌, Vol. 22, pp. 531-541 (2007).

吉川 友也 Yuya YOSHIKAWA

平成 25 年奈良先端科学技術大学院大学 情報科学研究科博士前期課程修了．現在，同大学博士後期課程在籍中．平成 25 年より日本学術振興会特別研究員(DC1)．データマイニング，機械学習の研究に従事．日本データベース学会会員．

岩田 具治 Tomoharu IWATA

平成 13 年慶應義塾大学 環境情報学部卒業．平成 15 年東京大学大学院 総合文化研究科修士課程修了．同年 NTT 入社．平成 20 年京都大学大学院 情報学研究科博士課程修了．博士（情報学）．平成 24 年より 1 年間 Cambridge 大学客員研究員．現在，NTT コミュニケーション科学基礎研究所研究主任．機械学習，データマイニング，情報可視化の研究に従事．平成 16 年船井ベストペーパー賞，平成 21 年情報処理学会論文誌論文賞等受賞．電子情報通信学会，情報処理学会各会員．

澤田 宏 Hiroshi SAWADA

1991 年，京都大学工学部情報工学科卒業．1993 年，同修士課程修了． 同年，日本電信電話株式会社(NTT)入社． 以来，同社コミュニケーション科学基礎研究所にて，VLSI 向け CAD および計算機アーキテクチャの研究に従事．2000 年より，信号処理，特にブラインド音源分離の研究に従事．2009 年より，知能創発環境 研究グループグループリーダ．2013 年より，サービスエボリューション研究所ヒューマンアナリシスプロジェクト グループリーダ，現在に至る．2001 年，京都大学博士(情報学)．電子情報通信学会，日本音響学会，IEEE 各会員．