Vol.2015-DBS-162 No.1
2015/11/26

分散バージョン管理システムの共同開発履歴に基づいた開発者の特徴抽出手法の検討

李 書[†1]　鷹野 孝典[†1]

概要：ソフトウェア開発における分散バージョン管理システムの普及に伴い，様々なソフトウェア開発プロジェクトが Web 上で共有・管理されるようになっている．そこでは，プロジェクトのリーダーと協力者の関係を初めとして，重要な関数を開発する人，エラーやバグを修正する人，プロジェクト進展のために見通しの良いコードを提供してくれる人などによる，ソフトウェア共同開発に関する営みが日々行われている．本稿では、このようなソフトウェア開発者の役割やスキルに関する特徴を分析することを目的とした，分散ソースコード管理システムの共同開発履歴に基づいた開発者の特徴抽出手法について検討する．

キーワード：GitHub，協同開発，分散ソースコード管理システム，開発者，特徴抽出

A Method of Feature Extraction from Software Developers based on a History of Collaborative Development using Distributed Version Control System

Shu LI [†1]　Kosuke TAKANO [†1]

Abstract: With the widespread of distributed version control systems for the software development, a whole bunch of software development projects are shared and managed on the Web environment. There, many developer, such as project leaders and supporters, people who write important functions, people who solve the bugs and errors, and people who provide fundamental source code for relieving bottlenecks, are collaboratively progressing their projects. In this paper, for analyzing the software developers' characters such as their skills and rolls in the project, we present a method of feature extraction from software developers based on a history of collaborative development using distributed version control system.

Keywords: GitHub, collaborative development, distributed version control system, developer, feature extraction

1. はじめに

ソフトウェア開発における分散バージョン管理システムの普及に伴い，様々なソフトウェア開発プロジェクトが Web 上で共有・管理されるようになっている．そこでは，プロジェクトのリーダーと協力者の関係を初めとして，重要な関数を開発する人，エラーやバグを修正する人，プロジェクト進展のために見通しの良いコードを提供してくれる人などによる，ソフトウェア共同開発に関する営みが日々行われている．

GitHub が Git プロジェクトのソースコード・リポジトリを分析するために提供している API[1)]（以下，GitHub API）の普及により，オンライン上でのソフトウェア開発に関する様々な解析が盛んに行われるようになった．図 1 は，GitHub API で取得できる項目について示している．例えば，「開発プログラミング言語とその使用量」に着目して統計データを取ることで，プログラミング言語の人気ランキングを行う事ができる．また，「コミット時のコメント」を分析することで，開発者の感情抽出を行う試みもある [2)]．

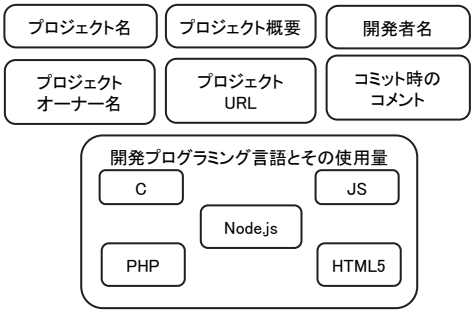

図 1　GitHub API で取得できる項目
Figure 1　Available items using GitHub API

本稿では，ソフトウェア開発者の役割やスキルに関する特徴を分析することを目的とした，分散ソースコード管理システムの共同開発履歴に基づいた開発者の特徴抽出手法について検討する．提案方式によって抽出した開発者の特徴を分析することにより，例えば，現在進行中プロジェクトの早期評価や新規プロジェクト立ち上げ時の人材獲得の支援が可能となる．

†1 神奈川工科大学 情報学部 情報工学科
Department of Information and Computer Sciences, Faculty of Information Technology, Kanagawa Institute of Technology

本研究では，実際の Git プロジェクトを対象とした実験により，提案方式の実現可能性を検証する．

2. 研究動機

オンライン上でのソフトウェア開発リポジトリにおける開発者の行動分析に関する研究が活発に行われている．尾上等は，GitHub 上の活動履歴分析による開発者分類手法の提案し，GitHub で活発な OSS (Open Source Software) プロジェクトである homebrew と node に参加する開発者を活動履歴からクラスタリングし，その結果に基づいて開発者の分類を行っている [5]．文献 6)では，ソフトウェア開発におけるソーシャルコーディングに着目し,「プロジェクトへのバグの発見報告や新機能の提案」,「ソースコードの変更を伴わない提案」,「ソースコードの変更を伴う提案」といった有益な提案を GitHub から機械的に抽出する方法を示している．また，開発者のプロジェクトへの貢献度については，坂口等らが，複数のプロジェクトで活動する開発者が OSS プロジェクトに与える影響を明らかにすることを目的として，複数のオープンソースプロジェクトに参加する開発者による貢献度の分析手法を提案している [7]．

これらの研究に対して，提案方式は，GitHub 上で管理される Git プロジェクトに参加する開発者を対象として，個々の開発者の特徴を抽出する方式である．提案方式の分析対象となる Git プロジェクトは, Ruby on Rails, Java Spring, Laravel といった MVC フレームワークを対象としている．MVC フレームワークに着目することで，フォルダ構成，ファイル名，ファイル拡張子から，開発者が開発に携わっているソース・プログラムの機能性や重要性を把握することが可能となり，開発者のプロジェクトにおける役割，貢献度，リーダー性，サポート性に関する特徴を分析することができる（図 2）．

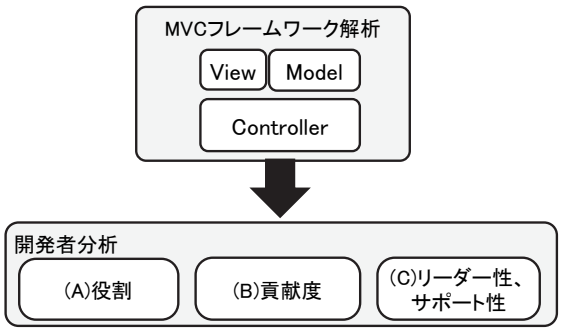

図 2　提案方式による分析項目
Figure 2　Items that can be analyzed by proposed method

提案方式により，PHP の MVC フレームワークである Laravel で開発したプロジェクトを対象とした分析結果の例を表 1 に示す．この例では，コントロール (ctrls)に関するソース・ファイルの作成者は，5 名おり，wu さんが多くのソース・ファイルを作成していることがわかる．ビュー (views) に関する開発者は 4 名おり, Black Lotus さんと leeke さんが開発に貢献している．モデル (models) に関する開発者は，6 名おり，wu さんと Black Lotus さんが開発に貢献している．これらのことから，wu さん，Black Lotus さん, leeke さんの 3 人がリーダー的役割となって各々得意なところを分担し，他の開発者の協力を得ながら，このプロジェクト開発が進行している様子がわかる

表 1　Laravel によるソフトウェア開発プロジェクトを対象とした分析結果の例
Table 1　Example of analysis result for software development project using Laravel.

```
------------ctrls------------
all file: 57
Black Lotus: 12 ==> 21.05%
wu: 24 ==> 42.11%
leeke: 14 ==> 24.56%
fyy5: 1 ==> 1.75%
linmingkun: 6 ==> 10.52%

------------views------------
all file: 14
leeke: 5 ==> 35.71%
wu: 2 ==> 14.29%
Black Lotus: 7 ==> 50.0%

------------models------------
all file: 41
Black Lotus: 11 ==> 26.83%
wu: 15 ==> 36.59%
linmingkun: 9 ==> 21.95%
leeke: 5 ==> 12.20%
ttmk008: 1 ==> 2.44%
```

3. 提案方式

3.1 概要

図 3 は提案方式の概要を示している．提案方式は，(1) MVC フレームワーク構造分析用のプロバイダ，(2)プロジェクト解析機能群，および (3)開発者の特徴抽出機能群より構成される．MVC フレームワーク構造分析用のプロバイダは，個々の MVC フレームワークにカスタマイズして分析をするためのプラグイン・モジュールである．また，プロジェクト解析機能では，MVC 層ごとに，新規ソース・プログラム作成数の分析，他者開発ソース・プログラム修正数の分析，共通関数作成数の分析を行う．これらの分析に基づいて, プロジェクトにおける開発者の役割, 貢献度, サポート性，リーダー性に関する特徴抽出を行う．

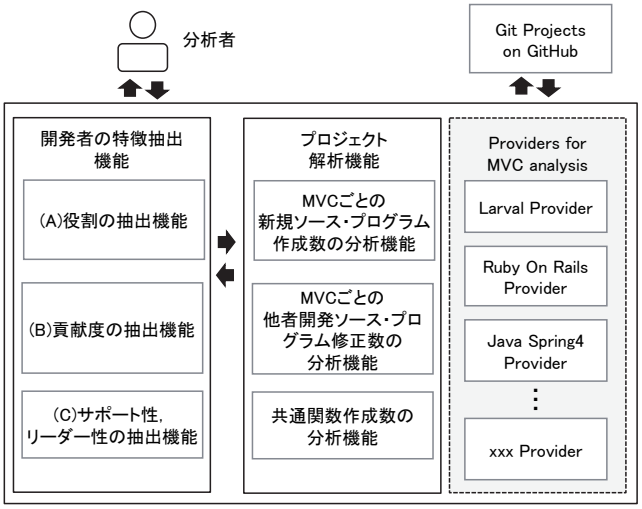

図 3　提案方式の概要

Figure 3　Overview of proposed method

3.2 開発者の特徴抽出

本節では，提案方式による MVC フレームワーク解析に基づいた，開発者の役割，貢献度，リーダー性，サポート性の抽出方法について述べる．

[(A)役割の抽出]

MVC フレームワークを対象として，コントロール，ビュー，モデルの各開発層の作成・修正ファイル数を数えることにより，開発者の役割を抽出する (図 4)．例えば，コントロール層の開発が多い人は，フロントエンジニアの役割を担っていると考えられる．コントロール，ビュー，モデルの各層ごとに，新規作成したソース・プログラム，および他者開発ソース・プログラムを修正した割合 a が，しきい値 α %より大きい場合に，その開発層の中心的役割を担っていると判断する．

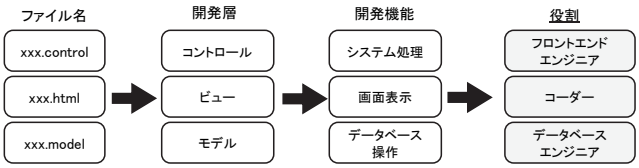

図 4　役割の抽出フロー

Figure 4　Extraction flow of Role

[(B)貢献度の抽出]

開発者の貢献度は，質的な貢献度と量的な貢献度に分けて抽出する．

質的な貢献度は，プロジェクト中の重要なモジュール群の開発量に応じて算出する（図 5）．例えば，他の関数からの参照回数が多いライブラリ関数を開発している開発者は，質的な貢献度が高いと考えられる．そこで，ライブラリ関数のような参照回数の多い関数の作成数 b がしきい値 β より大きい場合，そのライブラリ関数の開発者はプロジェクトの質的な貢献度が高いと判定する．

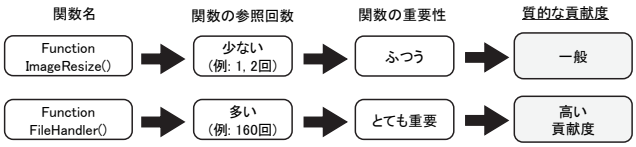

図 5　質的な貢献度の抽出フロー

Figure 5　Extraction flow of qualitative contribution

量的な貢献度は，開発者が作成・修正したソース・ファイル数に応じて算出する（図 6）．プロジェクト中の多くのソース・ファイルについてコミットしているほど，量的な貢献度が高いと考えられる．コントロール，ビュー，モデルの各層全体において，新規作成したソース・プログラムおよび他者開発ソース・プログラムを修正した割合 c に応じて，量的な貢献度を算出する（表 2）．

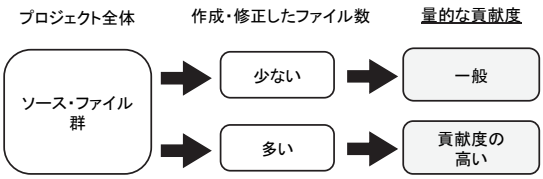

図 6　量的な貢献度の抽出フロー

Figure 6　Extraction flow of quantitative contribution

表 2　量的な貢献度のレベル

Table 2　Level of quantitative contribution.

c の範囲	貢献レベル
$0 < c \leqq \gamma_1$	1
$\gamma_1 < c \leqq \gamma_2$	2
$\gamma_2 < c \leqq 100$	3

[(C)サポート性とリーダー性の抽出]

サポート性は，他の開発者への援助の度合を表す性質であり，他者開発ソース・プログラムのうち，修正したソース・プログラムから算出する（図 7）．他者開発ファイルの修正数 d がしきい値 δ 以上ならば，その開発者はサポート性があるとする．

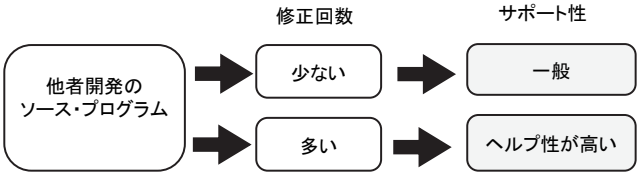

図 7　サポート性の抽出フロー

Figure 7　Extraction flow of role of support

また，リーダー性については，システム環境構築の貢献度とサポート性の両方を満たす場合にリーダー的役割を担っているとする．前者のシステム環境構築の貢献度については, 例えば, 設定ファイルやライブラリ定義ファイル等, システム環境構築に必要なソース・プログラムを率先して作成することは，プロジェクト全体の開発を効率よく進める上で重要なリーダー性を示していると考えられる．このため，システム環境構築に必要なソース・プログラムのコミット数が多い開発者は, リーダー性が高いと考えられる．表 3 に，例として，システム環境構築を効率化するための管理ツール，およびそれに関連付けられるソース・ファイル群を示す．

リーダー性の抽出条件は，サポート性がある，かつシステム環境構築ファイルの作成数 e がしきい値 ε 以上なことである．

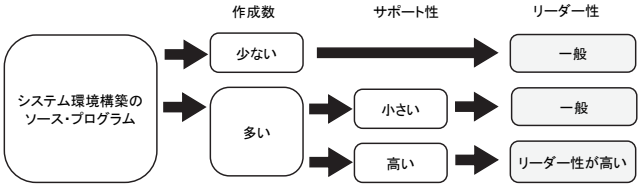

図 8　リーダー性の抽出フロー

Figure 8　Extraction flow of role of leader

表 3　システム環境構築のための
ソース・プログラム群の例

Table 3　Example of source programs for the construction of system environment.

管理ツール名	ファイル名	用途
bower	bower.json	Github に載っている javascript/css などライブラリを管理
npm	package.json	nodejs ライブラリの管理
grunt	gruntfile	Node.js をベースとした作業を自動化
gulp	gulpfile.js	Grunt と同じ、Node.js をベースとした作業を自動化
composer	composer.json	PHP ライブラリ管理
pip	requirements.txt	Python　パッケージの管理
maven	pom.xml	Java 自動化構築
gradle	build.gradle	Java 自動化構築
gem	gemfile	Ruby パッケージを管理
godep	godeps.json	Go 言語のライブラリが依存しているパッケージの管理
cordova	config.xml	HTML5 でのスマートフォンアプリの開発

4. 実験

実際の Git プロジェクトを対象とした実験により，提案方式の実現可能性を検証する．

4.1 実験環境

表 4 に示す, GitHub 上で公開されている 4 つのソフトウェア開発プロジェクト P_1～P_4 について分析を行う．各プロジェクトで用いられているフレームワークは, Laravel (php) と Ruby on Rails のいずれかである．プロジェクト P_1 については，著者ら自身のプロジェクトであるため，URL は非公開としている．

表 4　分析対象プロジェクト

Table 4　Target projects to be analyzed.

ID	プロジェクト名 (フレームワーク)	URL	分析対象人数 / 開発者人数
P_1	Chain convenience stores management system (Laravel)	非公開	6 名 / 6 名
P_2	Fullycms (Laravel)	https://github.com/sseffa/fullycms	2 名 / 2 名
P_3	Caravel-Blog (Laravel)	https://github.com/FbF/Laravel-Blog	3 名 / 6 名
P_4	Gitlabhq (Ruby on Rails)	https://github.com/gitlabhq/gitlabhq	16 名/ 753 名

4.2 実験方法

プロジェクト P_1～P_4 の各開発者を対象として，3 章で示した方式により(A)役割，(B)貢献度（量的のみ），(C)リーダー性とサポート性の抽出を行い，結果を考察する．

(A)のしきい値については，P_1～P_3 については，25%以上とした．P_4 については，人数が 700 人以上と非常に多いため 1%とした．(B)のしきい値は，新規作成率と修正率の合計 50%以上でレベル 3，25%~50%でレベル 2 ，25%以下でレベル 1 とした．また，サポート性のしきい値は，P_1～P_3 については 25%以上，P_4 については 5%以上とした．さらに，リーダー性については，サポート性がある，かつシステム環境構築ファイルの作成数が 1 つ以上ある場合にリーダーの役割を担っているとした．

4.3 実験結果

図 9～図 12 に，表 4 のプロジェクト P_1～P_4 の解析結果を示す．また，表 5～表 8 に，この解析結果に基づいた開発者の特徴抽出結果を示す．これらの結果より，提案方式を用いて, MVC フレームワーク解析に基づいた, 開発者の役割，貢献度，リーダー性，サポート性の抽出が可能であることが確認できる．

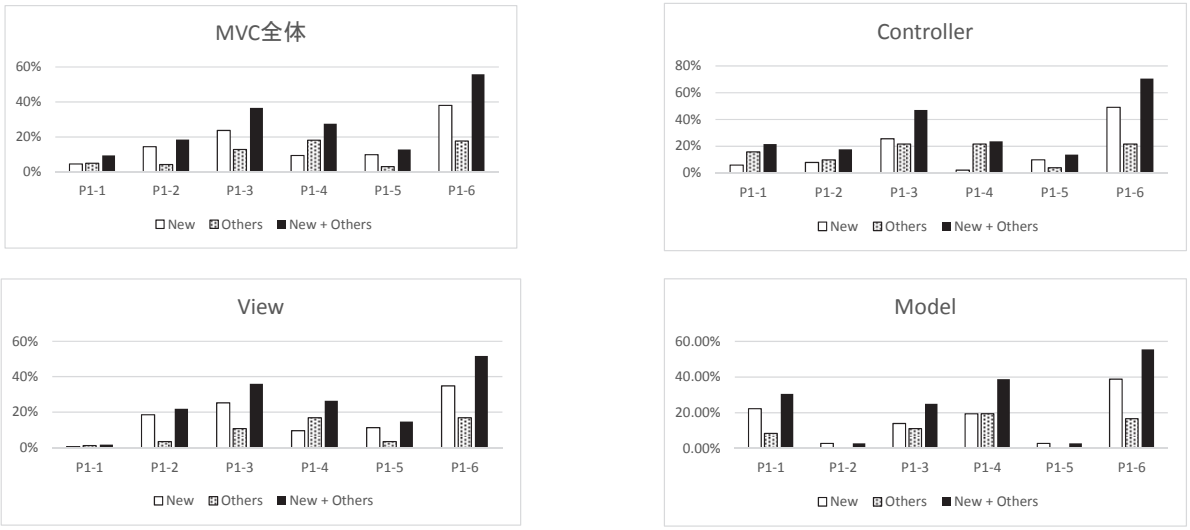

図 9　プロジェクト P_1 の解析結果

Figure 9　Analysis result of project P_1

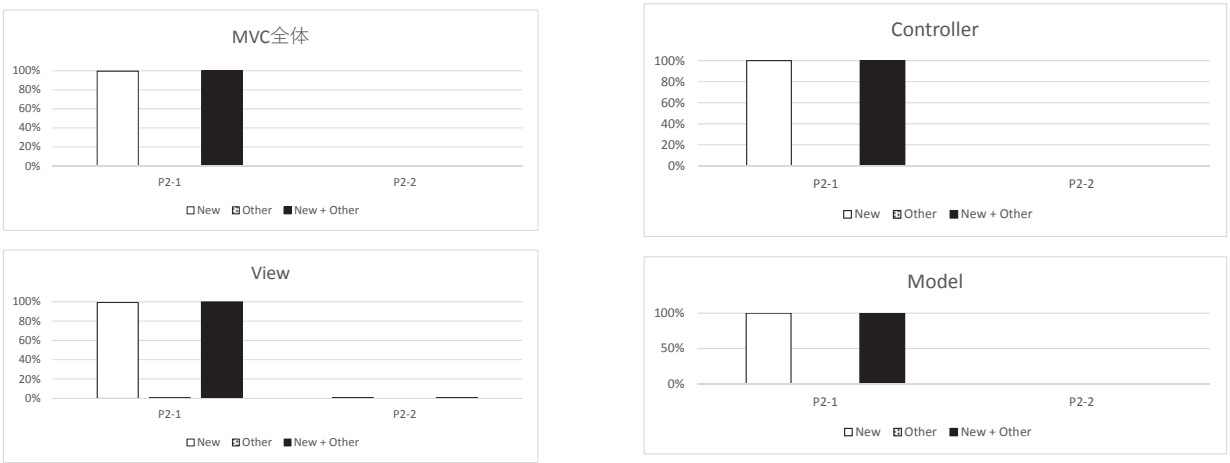

図 10　プロジェクト P_2 の解析結果

Figure 10　Analysis result of project P_2

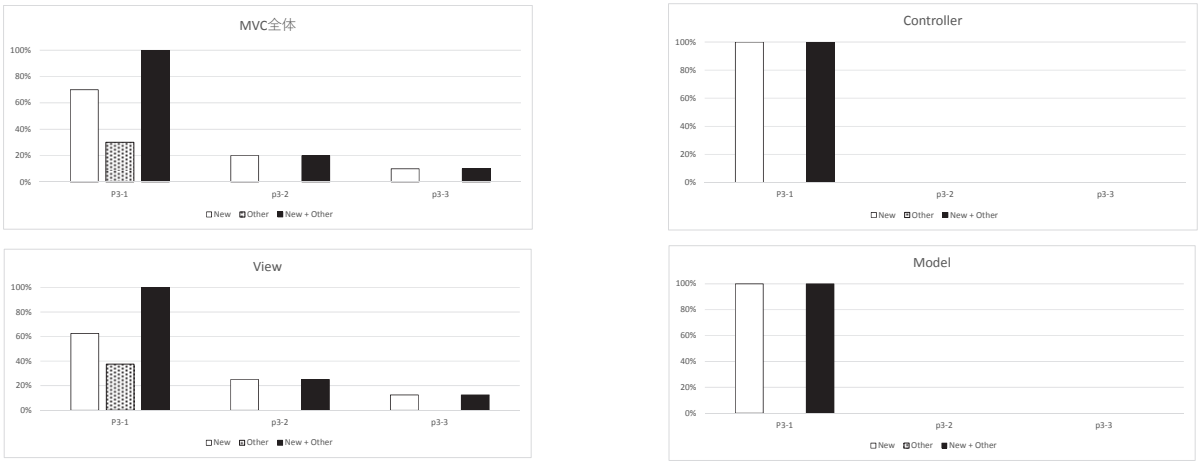

図 11　プロジェクト P_3 の解析結果

Figure 11　Analysis result of project P_3

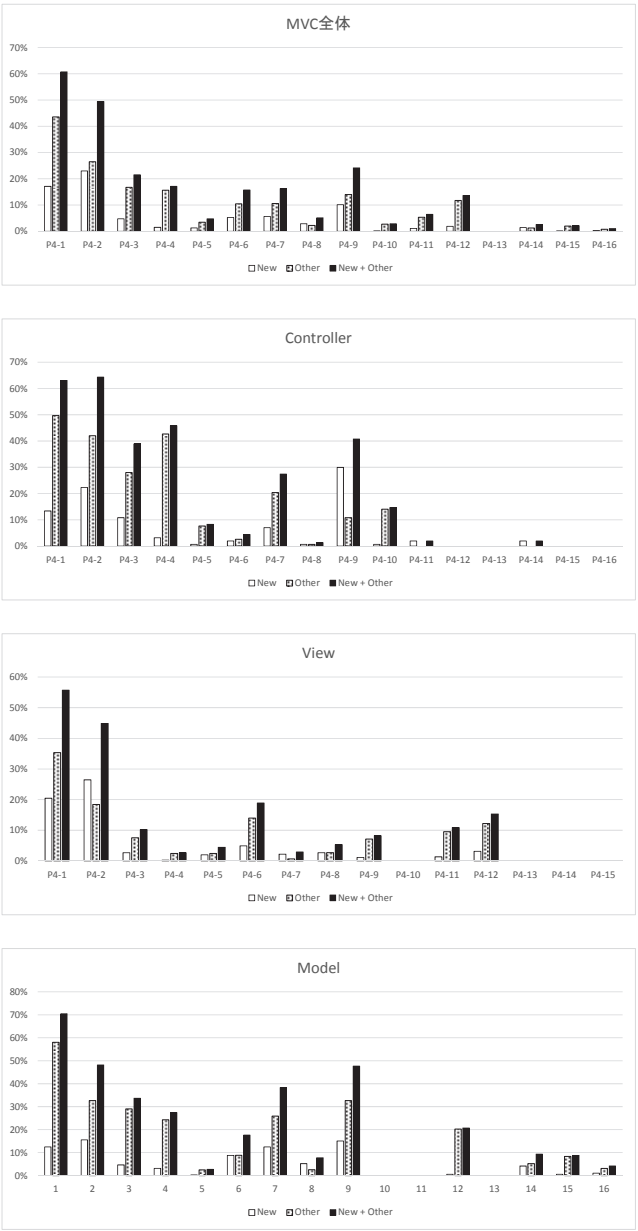

図 12　プロジェクト P_4 の解析結果
Figure 12　Analysis result of project P_4

表 5　開発者の特徴抽出結果（P_1）
Table 5　Result of feature extraction of developers in P_1.

開発者名	(A) 役割	(B2) 量的な貢献	(C1) サポート性	(C2) リーダー性
P1-1	CM	1	false	false
P1-2	-	1	false	false
P1-3	CVM	3	true	true
P1-4	CVM	3	true	false
P1-5	CV	2	false	false
P1-6	CVM	3	true	true

表 6　開発者の特徴抽出結果（P_2）
Table 6　Result of feature extraction of developers in P_2.

開発者名	(A) 役割	(B2) 量的な貢献	(C1) サポート性	(C2) リーダー性
P2-1	CVM	3	true	true
P2-2	-	1	false	false

表 7　開発者の特徴抽出結果（P_3）
Table 7　Result of feature extraction of developers in P_3.

開発者名	(A) 役割	(B2) 量的な貢献	(C1) サポート性	(C2) リーダー性
P3-1	VM	3	true	true
P3-2	V	2	false	false
P3-3	V	1	false	false

表 8　開発者の特徴抽出結果（P_4）
Table 8　Result of feature extraction of developers in P_4.

開発者名	(A) 役割	(B2) 量的な貢献	(C1) サポート性	(C2) リーダー性
P4-1	CVM	3	true	true
P4-2	CVM	3	true	true
P4-3	CVM	3	true	true
P4-4	CVM	3	true	true
P4-5	CV	3	false	false
P4-6	M	1	false	false
P4-7	CVM	3	true	true
P4-8	CVM	3	false	false
P4-9	CVM	3	false	false
P4-10	CV	3	true	true
P4-11	M	2	false	false
P4-12	VM	3	true	true
P4-13	V	3	true	true
P4-14	M	2	false	false
P4-15	M	2	false	false
P4-16	M	1	false	false

表 9　プロジェクト P_1 の解析結果（内訳）
Table 9　Analysis results of P_1 (Detail).

開発者名	作成ファイル数	割合 (A)	修正ファイル数	割合 (B)	(A)+(B)
P1-1	12	5%	13	5%	9%
P1-2	38	14%	11	4%	18%
P1-3	63	24%	34	13%	37%
P1-4	25	9%	48	18%	28%
P1-5	26	10%	8	3%	13%
P1-6	101	38%	47	18%	56%

表 10　P_1 におけるコントロール層の解析結果（内訳）
Table 10　Analysis results of controls layer in P_1 (Deital).

開発者名	作成ファイル数	割合(A)	修正ファイル数	割合(B)	(A)+(B)
P1-1	3	6%	8	16%	22%
P1-2	4	7%	5	10%	18%
P1-3	13	25%	11	22%	47%
P1-4	1	2%	11	22%	24%
P1-5	5	10%	2	4%	14%
P1-6	25	49%	11	22%	71%

表 11　P_1 におけるビュー層の解析結果（内訳）
Table 11　Analysis results of views layer in P_1 (Deital).

開発者名	作成ファイル数	割合(A)	修正ファイル数	割合(B)	(A)+(B)
P1-1	1	1%	2	1%	2%
P1-2	33	19%	6	3%	22%
P1-3	45	25%	19	11%	36%
P1-4	17	10%	30	17%	26%
P1-5	20	11%	6	3%	15%
P1-6	62	35%	30	17%	52%

表 12　P_1 におけるモデル層解析結果（内訳）
Table 12　Analysis results of models layer in P_1 (Deital).

開発者名	作成ファイル数	割合(A)	修正ファイル数	割合(B)	(A)+(B)
P1-1	8	22%	3	8%	31%
P1-2	1	3%	0	0%	3%
P1-3	5	14%	4	11%	25%
P1-4	7	19%	7	19%	39%
P1-5	1	3%	0	0%	3%
P1-6	14	39%	6	17%	56%

表 13　P_1 におけるシステム環境構築のためのソース・プログラムのコミット履歴
Table 13　Commit history of source programs for the construction of system environment in P_1.

管理ツール名	ファイル名	開発者名
bower (js/css manager tools)	bower.json	P1-6
composer (php)	composer.json	P1-6
gulp (nodejs)	gulpfile.js	P1-6
composer (php)	modules/Agent/composer.json	P1-1
composer (php)	modules/Area/composer.json	P1-1
composer (php)	modules/Common/composer.json	P1-3
composer (php)	modules/Master/composer.json	P1-3
composer (php)	modules/Shop/composer.json	P1-1
composer (php)	modules/Trade/composer.json	P1-6
composer (php)	modules/User/composer.json	P1-3
npm (nodejs package manager tools)	package.json	P1-6
bower (js/css manager tools)	public/master-static/bower.json	P1-3

表 9～表 13 に，P_1 の解析結果の詳細を示す．例えば，表 13 の結果において，開発者 P1-6 は作成・修正したソース・ファイル数の割合が 50%を超えており，量的な貢献度が非常に高い．このため，表 5 の結果において，量的な貢献度がレベル 3 と抽出されていることがわかる．また，P1-6 はサポート性があり，表 13 にあるようにシステム環境構築を行っているので，リーダー性も抽出されている．また，開発者 P1-4 は，量的な貢献度も高く，サポート性もあるが，システム環境構築を行っておらず，リーダー的な役割は担っていないと算出された．さらに，開発者 P1-2 は，量的な貢献度も低く，CVM の担当も見られないので，プロジェクトがある程度進行した後に，新規参入した開発者と推定される．

プロジェクト P_1 は，著者ら自身の Git プロジェクトであり，表 5 に示した各開発者の特徴の抽出結果は，実際の状況に適合していることを確認している．

5. おわりに

本稿では，分散ソースコード管理システムの共同開発履歴に基づいた開発者の特徴抽出手法について提案した．実際の Git プロジェクトを対象とした実験により，提案方式の実現可能性を検証した．

今後の課題として，今回の実験では行わなかった質的な貢献度の評価を，プロジェクト中の重要なモジュール群の開発量に応じて算出することにより実施する予定である．また，複数のプロジェクトにまたがって開発を行っている開発者を対象としたグローバルな評価を行うことで，より実際の開発状況に適った開発者の特徴抽出手法を開発していく予定である．さらに，提案方式は，いわゆる「正解データ」が存在しないため評価が困難である．提案方式を改善するとともに，有効性を評価するための方法論を検討していくことも課題として挙げたい．

参考文献

1) GitHub API v3
https://developer.github.com/v3/
2) Exploring Expressions of Emotions in GitHub Commit Messages
http://geeksta.net/geeklog/exploring-expressions-emotions-github-commit-messages/
3) Ruby on Rails
http://rubyonrails.org/
4) Laravel
http://laravel.com/
5) 尾上紗野，畑秀明，松本健一: GitHub 上の活動履歴分析による開発者分類，情報処理学会論文誌，Vol.56，No.2，pp.715-719 (2015).
6) 檀上未来，乃村能成，谷口秀夫: ソーシャルコーディングにおける有益な提案の抽出について，情報処理学会研究報告．マルチメディア通信と分散処理研究会報告 2013-DPS-157(4), pp.1-8 (2013).
7) 坂口英司，伊原彰紀，尾上紗野，畑秀明，松本健一: 複数のオ

ープンソースプロジェクトに参加する開発者による貢献の分析, 情報処理学会研究報告. グループウェアとネットワークサービス研究会報告 2014-GN-92(15), pp.1-4 (2104).
8) 矢萩寛人: ソフトウエアの共同開発における編集衝突低減手法, 情報処理学会 全国大会講演論文集 2013(1), pp.449-451 (2013).
9) 吉岡 弘隆: OSS に見る IT の最新動向:1.OSS の進化 -コミュニティ開発のもたらすもの-, Addison-Wesley, Reading, Massachusetts, 2nd edition (1990) 情報処理 56(3), pp226-232 (2015).
10) Novielli, N., Calefato, F., and Lanubile, F.:The challenges of sentiment detection in the social programmer ecosystem, Proceedings of the 7th International Workshop on Social Software Engineering, pp.33-40 (2015).
11) Guzman, E., Azócar, D., and Li, Y.: Sentiment analysis of commit comments in GitHub: an empirical study, Proceedings of the 11th Working Conference on Mining Software Repositories, pp.352-355 (2014).

正誤表

p.8 の記載の一部に誤りがありましたので，下記のように訂正します.

（誤）

9) 吉岡 弘隆: OSS に見る IT の最新動向:1.OSS の進化 -コミュニティ開発のもたらすもの -, Addison-Wesley, Reading, Massachusetts, 2nd edition (1990) 情報処理 56(3), pp226-232 (2015).

（正）

9) 吉岡 弘隆: OSS に見る IT の最新動向:1.OSS の進化 -コミュニティ開発のもたらすもの -, 情報処理 56(3), pp.226-232 (2015).