チーム開発における貢献度評価の提案

中村 圭助[1]　中山 功一[1]

概要：本研究では，Git と呼ばれる開発プラットフォーム上で行われるソースコードの変更履歴（ログ）に着目し，チーム開発における各メンバーの貢献度を定量的に評価する手法を提案する．チーム開発に取り組むプログラマは，互いの貢献度を把握できているという仮定に立ち，Git のログから評価したプログラマの貢献度を数値化する．この評価手法により，不公平で不当な評価を防ぎ，プロダクトの品質の向上やプログラマの不満を防ぐ．実験結果から，提案手法の評価結果は，被験者となったプログラマの主観的な開発貢献度の相互評価と，高い相関があることが示された．

キーワード：チーム開発，Git，PageRank

1. はじめに

本研究では，ソフトウェアのチーム開発において，多数の開発者の貢献度を，ソースコードの変更履歴（ログ）から適切に評価する手法について提案する．

会社組織として取り組むソフトウェア開発では，発注者の提示する仕様書に基づいて，受注者側の開発責任者が工程を管理し,開発に取り組むのが一般的である.この場合，開発業務に取り組むプログラマは，受注企業の人事考課制度の基準などに基づいて貢献度が評価される場合が多い．しかし，プログラマは，必ずしも評価に対して公平感を得ているとは限らない．自分よりも貢献していないメンバーが高く評価され，自分には働きに見合った給与が支払われていないと感じるプログラマも多い．最近では，プロダクトの開発運用に対する正当な支払いが受けられず，プロダクトに存在するバグの修正がされていないといった問題[1]も表面化している．

また，オープンソースソフトウェア（OSS）の開発では，開発に取り組むプログラマを評価する体制が確立していない場合が多い．このため，OSS 開発では，貢献度に応じて報酬が得られるプロジェクトは少ない．この結果， OSS 管理をしている組織のビジネスチャンスを，大きな組織が不当に奪い，OSS のコミュニティに正当に還元されていないという主張[2]がなされる場合もある．

本研究では，チーム開発の過程で，Git と呼ばれる開発プラットフォーム上で行われるソースコードの変更履歴（ログ）に着目する．互いのソースコードを Git 上で共有するチーム開発において，プロジェクト管理者を含むプログラマは,お互いの貢献度を評価できているという仮定に立ち，Git のログからプログラマの相互評価を数値化するアルゴリズムを提案する．チーム開発において，この評価手法を用いた評価と，チームメンバーの相互評価を比較した実験結果から，提案手法の有効性を示す．この結果，不公平で不当な評価を防ぎ，プロダクトの品質の向上やプログラマの不満を防ぐことを目指す．

2. 貢献度評価アルゴリズム

2.1 前提とする Git の仕様

本研究では，Git のログを基に貢献度を評価する．Git では，共通リポジトリに存在し，開発者全員で共有しているソースコード（以下，共有ソースコードと記す）と，フォークリポジトリに存在し，個々の開発者が個別に管理しているソースコード（以下，個別ソースコードと記す）がある．共通ソースコードに対する変更（上書きや挿入など）の依頼を，プルリクエストと呼ぶ．プルリクエストが，プロジェクト管理者に承認されると，共通ソースコードに，プルリクエストされた個別ソースコードが反映される．この反映作業をマージと呼ぶ.

プルリクエストの承認には，一般的には管理者など他のプログラマによるコードレビューが実施されるため，共通ソースコードにマージされる（プルリクエストが承認される）場合は,そのプルリクエストを依頼したプログラマは，プロジェクトに貢献をしたと仮定する[3].

2.2 ログに基づく貢献度評価手順

提案手法では，以下の手順で貢献度を評価する．

(1) Git ログに基づいた重み付き有向グラフの生成

Git のログから，重み付き有向グラフ[4][5][6][7]を生成する．開発に関わるプログラマの人数を N 人とするとき，プロジェクト全体を意味する 1 個のノード（全体ノード）と，個々のプログラマを意味する N 個のノード（プログラマノード）の，合計 N＋1 個のノードを作成する．初期設定として，全体ノード（P0）から，N 個のプログラマノード（P1～PN）に，重みゼロの N 本のエッジを与える．

プログラマ i のプルリクエストがマージされたとき，P0 から Pi に向けたエッジに，重み 1 を加算する．このアルゴリズムによって，プロジェクトノードから個々のプログラマに向けた重み付き有向グラフが生成される．例えば，プログラマ 1 が 3 回，プログラマ 2 が 5 回，プログラマ 3 が 1 回，プログラマ 4 が 1 回のプルリクエストを反映させた

1 佐賀大学

場合，図 1 に示す重み付き有向グラフになる．

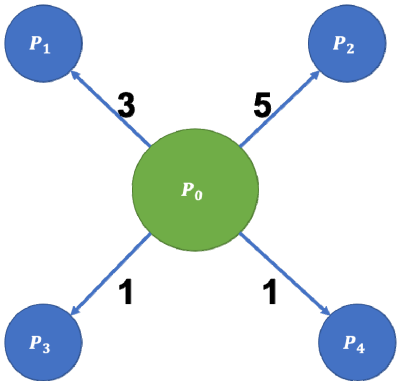

図 1 重み付き有向グラフの例

(2) マージ回数上位のノードへのエッジを追加

(1)では，マージ回数に比例した重み付き有効グラフが生成されるが，マージ回数が多い上位半数のプログラマは，マージの回数の比率以上にプロジェクトへの貢献度が高いと仮定する．そこで，マージ回数が多い上位半数のプログラマノード（Pi）は，マージ回数の少ない下位半数の全てのプログラマノードから，エッジを与えられる．与えられるエッジの重みは，そのプログラマ i のマージ回数を，全プログラマのマージ回数の中央値で割った値とする．すなわち，中央値の M 倍のマージをしたプログラマ i は，毎時回数が中央値以下の全プログラマのノードから，重み M のエッジを得る．図 1 の中央値が 2 の例に対して，上記の処理を実施した場合のグラフを図 2 に示す．

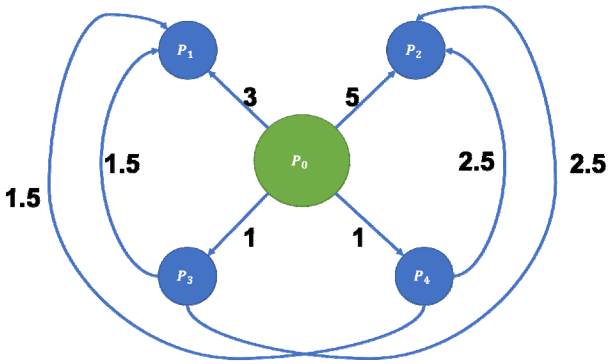

図 2 マージ回数上位のノードへのエッジを追加した例

(3) 重み付き有向グラフの隣接行列表現

(2) までで生成される重み付き有向グラフを隣接行列WG考える．重み付き有向グラフの隣接行列 WG を，式 2 で定義する．

式 2 の行列WGの要素$w_{i,j}$は，Piに対する Pjの貢献の重みである．本研究は，プロジェクトのノードも考慮する必要があったため，プログラマ数をNとした時，行列Gは$N+1$次正方行列となる．

$$WG = \begin{bmatrix} w_{0,0} & \cdots & w_{0,n} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,n} \end{bmatrix}$$

$$0 \leq W_{i,j},\ i \in \{0,1,2,\dots n\}, \boldsymbol{j} \in \{0,1,2,\dots n\}$$

式 2 重み付き有向グラフの隣接行列

図 2 に示したグラフから作成した隣接行列WG'を式 3 に示す．

$$WG' = \begin{bmatrix} 0 & 3 & 5 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 2.5 & 0 & 0 \\ 0 & 1.5 & 2.5 & 0 & 0 \end{bmatrix}$$

式 3　図 2 のグラフから得られた隣接行列

(4) 重み付き有向グラフへの正規化

本研究は，特定プログラマにより貢献度評価が大きく偏ることを防ぐために，各ユーザに対する貢献の重みに対して min-max 正規化をする．min-max 正規化するための，各ノード i から出るエッジの重みの総和 Sum を式 4 に示し，Sum を用いて正規化した行列 NWG を式 5 に示す．

$$\boldsymbol{sum_i} = \sum_{j=0}^{N} WG_{i,j}$$

$$\{\boldsymbol{i} \in \{0,1,\dots,\ N\}\}$$

式 4 各ユーザの枝の重みの合計のベクトル

$$NWG = \begin{bmatrix} \frac{w_{0,0}}{\boldsymbol{sum}_0} & \cdots & \frac{w_{0,N}}{\boldsymbol{sum}_0} \\ \vdots & \ddots & \vdots \\ \frac{w_{N,0}}{\boldsymbol{sum}_N} & \cdots & \frac{w_{N,N}}{\boldsymbol{sum}_N} \end{bmatrix}$$

※ ただし，分母の$\mathrm{Sum}_n = 0$ の場合は 0 とする

式 5 min-max 正規化した重み付き有向グラフ

式 3 を上記の手順で正規化した隣接行列を式 6 に示す．

$$NWG' = \begin{bmatrix} 0.00 & 0.30 & 0.50 & 0.10 & 0.10 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.375 & 0.625 & 0.00 & 0.00 \\ 0.00 & 0.375 & 0.625 & 0.00 & 0.00 \end{bmatrix}$$

式 6　正規化した隣接行列の例

(5) **貢献度の計算**

ユーザ n の貢献度C_nを式 7 のように定義する．

$$C_n = \frac{\sum_{j=0}^{N} NWG_{n,j}}{\sum_{i=0}^{N}\sum_{j=0}^{N} NWG_{i,j}}$$

式 7 各ユーザの貢献度の計算式

式 7 に従い個々のプログラマ 1～4 の貢献度を計算すると，$C_1 = 35.0$，$C_2 = 58.3$，$C_3 = 3.3$，$C_4 = 3.3$となる．

3. 実験

3.1 実験の概要

本研究の評価アルゴリズムの正当性を確かめるため，大学生 8 人で行ったチーム開発の貢献度を評価した．開発内容は，web 上で動作するリバーシの作成である．8 人がそれぞれ仕様の作成，設計，実装までを担当した．被験者のスキルにおいて全員，簡単な制御構文，変数，配列といった基礎的なプログラムが不自由なく書けた．また，実験前から Git の使い方を知っている被験者が 2 名程度いた．被験者の中に，ソフトウェア設計の手法やボードゲームを作る際に利用できる高度なアルゴリズムなどを深く知っているものはいなかった．

本研究の評価アルゴリズムと，被験者の主観的な相互評価との一致度を確認するため，開発終了後に，それぞれの被験者に対して，自分を含めた全被験者の貢献度の順位付けをしてもらった．被験者nの全被験者からの平均順位をRとする時，その被験者nの順位スコアを$n - R$とした．この順位スコアと提案手法の評価値の相関係数を確認した．

3.2 実験の結果

実験の結果，順位スコアと評価値を表 1 に，その散布図を図 3 に示す．順位スコアと提案手法の評価について，相関係数$r = 0.867$となり，被験者の評価と本研究の評価アルゴリズムによる評価の間に高い相関が見られた．

表 1 順位スコアと評価値

	A	B	C	D	E	F	G	H
順位スコア	3.5	3.5	6.8	3.8	6.4	1.3	7.3	1
評価値	1.4	1.4	40	1.4	27	1.4	27	1.4

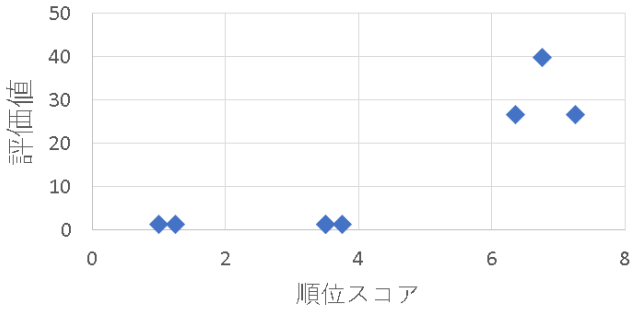

図 3 順位スコアと評価値の散布図

開発メンバーの評価値と本研究の評価アルゴリズムが計算した評価値は高い相関を示したことから,開発メンバーの納得のいくような評価が概ねできていると考えられる．しかし，被験者 C と被験者 G のような順位スコアと評価値に差異が見られたため，今後，このような結果になった原因を追求し，より精度の高い評価値を出せるようなアルゴリズムの改良が必要である．

4. まとめ

本研究では，Git 上で行われるソースコードの変更履歴（ログ）から，チーム開発におけるプログラマの相互評価を数値化するアルゴリズムを提案した．実際に 8 名のチーム開発で実験しところ，提案手法と被験者の相互評価に高い相関があったことから，提案手法の有効性を示した．提案手法により，プロジェクトにおける開発者の貢献度を多くの開発者が納得いくような評価が可能となる可能性を示唆された．今後は，評価値に基づいたプログラマへの報酬還元方法などに取り組む予定である．

参考文献

[1] 日本経済新聞：「COCOA」不具合　スマホ OS 更新、アプリ修正後手に
https://www.nikkei.com/article/DGXZQODZ0512V0V00C21A2000000/
[2] MongoDB now released under the Server Side Public License
https://www.mongodb.com/blog/post/mongodb-now-released-under-the-server-side-public-license
[3] 貢献度推定装置
https://www.j-platpat.inpit.go.jp/p0200
[4] The PageRank Citation Ranking.
http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
[5] 検索エンジンの検索アルゴリズム
https://www.jstage.jst.go.jp/article/jkg/54/2/54_KJ00000980270/_pdf/-char/ja
[6] 情報伝達に基づいた 有向重み付き複雑ネットワーク解析
http://tsuzuki.ise.ibaraki.ac.jp/MyPaper/Paper/IPSJ-TOM0201008.pdf
[7] ページランク・アルゴリズムの可視化
https://mas.kke.co.jp/wp-content/uploads/2020/01/ページランク・アルゴリズムの可視化.pdf