Vol.2016-SE-191 No.5
2016/3/14

モジュールごとの活動量を考慮したモジュールオーナー候補者予測：大規模OSSプロジェクトへの適用

山谷　陽亮[1,a)]　大平　雅雄[1,b)]

概要：大量の不具合が日常的に報告される近年の大規模 OSS プロジェクトでは，ボランティア開発者が中心となって不具合修正のためのパッチが作成される．パッチはプロジェクトのコア開発者であるコミッターにより検証され，問題がないことが確認されればリポジトリに取り込まれることで不具合の修正が完了する．しかしながら，大量の不具合を修正するために作成されるパッチも膨大な数に上る．比較的少人数で構成されるコミッターのみではパッチの検証を効率的に行えないため，現在，不具合修正の長期化が問題となっている．コミッターの不足を解消するために，ボランティア開発者の中からコミッターに昇格可能な開発者を予測するための研究が盛んに行われている．先行研究では，既存コミッターの昇格前の活動量（パッチ投稿数など）に基づいて予測モデルを構築し，ボランティア開発者の中からコミッターに昇格し得る開発者を予測する．しかしながら，実際の大規模 OSS プロジェクトの不具合修正プロセスにおいては，各コミッターが担当すべきモジュールが決まっているため，有能なボランティア開発者を全て昇格させれば不具合修正の長期化の問題が解決できるとは限らない．本研究では，担当者が不足しているモジュールに適したコミッタを推薦することが不具合修正の効率化に重要であるという立場を取る．そこで本研究では，特定のモジュールを担当するコミッターをモジュールオーナーと呼び，モジュールオーナーとなるべき開発者をボランティア開発者の中から特定するための予測モデルを構築する．ボランティア開発者の活動をモジュール毎に計測することにより，従来のコミッタ候補者予測モデルよりも細粒度，すなわち，モジュールオーナー候補者予測モデルを実現する．本稿では，5 件の大規模 OSS プロジェクトを対象に行ったケーススタディについて報告する．ケーススタディの結果，既存の予測モデルよりも高い精度でモジュールオーナー候補者を予測できることを示す．

キーワード：大規模 OSS 開発，モジュールオーナー，コミッター予測，ソフトウェアメトリクス

1. はじめに

近年，OSS の社会的普及により，OSS を利用するユーザーが年々増加している．ユーザーの多い大規模な OSS プロジェクトでは，日々大量の不具合が報告されており，報告された不具合は，ボランティアの開発者が中心となり，不具合修正のためのパッチが投稿され修正される．投稿されたパッチはバージョン管理システムに直接変更を加える権限を持つ開発者であるコミッターにより検証され，問題がないことが確認されればリポジトリに取り込まれることで不具合の修正が完了する．コミッターはソフトウェアに対して直接変更を加える権限を持つため，ソフトウェアの品質管理の面で重要な役割を担っている．そのため，プロジェクトに参加する開発者にコミット権限を付与するかどうかの判断は慎重に行われ，既存のコミッターからの推薦や承認によってコミット権限が付与される．このように，開発者にコミット権限を付与するかどうかの判断が慎重に行われているため，コミット権限を持たない開発者に比べコミッターは少人数で構成されている．大規模 OSS プロジェクトでは，大量の不具合を修正するために投稿されるパッチも膨大な数に上り，比較的少人数で構成されるコミッターのみではパッチの検証を効率的に行えなくなっている現状である．コミッターの不足を解消するために，ボランティアの開発者の中からコミッターに昇格可能な開発者を予測するための研究が盛んに行われている．先行研究では，既存のコミッターの昇格前の活動量メトリクス（パッチ投稿数やコメント投稿数など）に基づいて予測モデルを構築し，ボランティアの開発者の中からコミッターに昇格し得る開発者を予測している．実際の大規模 OSS プロジェクトでは，各コミッターが担当すべきモジュール

[1] 和歌山大学
Wakayama Uniersity
[a)] s151049@center.wakayama-u.ac.jp
[b)] masao@sys.wakayama-u.ac.jp

が決まっているため，コミッターの不足を解消するためには，コミッターが不足しているモジュールを担当することのできる開発者を推薦する必要がある．しかし，特定のモジュールを担当するコミッターを推薦するための予測モデルが構築されていない．本研究では，特定のモジュールを担当するコミッターをモジュールオーナーと呼び，モジュールオーナーとなるべき開発者（モジュールオーナー候補者）を特定するための予測モデルを構築する．予測モデル構築には，モジュールごとに計測した活動量メトリクスおよび，モジュールオーナーの適性を示すメトリクスであるパッチ貢献度を用いる．

本研究では，モジュールオーナーとなる開発者を予測する際，本研究で新しく定義したメトリクスであるパッチ貢献度が予測精度向上に寄与するかどうかを確かめることを目的として 5 件の大規模 OSS プロジェクト（Eclipse Platform プロジェクトおよび Mozilla Firefox プロジェクト，GCC プロジェクト，GIMP プロジェクト，WebKit プロジェクト）を対象に，評価実験を行った．評価実験の結果，5 件のプロジェクトのすべてにおいて，パッチ貢献度を用いない予測モデルに比べ，パッチ貢献度を用いた予測モデルは，予測精度が向上することが明らかになった．

2. OSS 開発におけるコミッター

2.1 OSS 開発におけるコミッターの役割と昇格プロセス

大規模 OSS 開発では，日々大量の不具合が報告されており [2]，報告された不具合を管理するために不具合管理システムが用いられている．OSS 開発における不具合修正プロセスについて説明する．ユーザーが不具合を発見した際，まず，発見した不具合の症状や不具合が発生した環境などを不具合管理システムに登録する．登録された不具合を修正するために，不具合に関する議論やパッチファイル（以下，パッチと呼ぶ．）の投稿が行われる．パッチとは，修正元と修正後のソースコードの差分を示すファイルを指す．投稿されたパッチは，プロジェクトのコア開発者であるコミッターによって検証が行われる．コミッターとは，バージョン管理システムに変更を直接加える権限を持つ開発者のことを指す．コミッターがパッチの検証を行った結果，パッチの修正内容が適切であると判断された場合，コミッターはバージョン管理システムにパッチの修正内容をコミットする．このように，コミッターはソフトウェアに直接変更加える権限を持つため，ソフトウェアの品質管理の面で重要な役割を担っている [6]．

コミッターはソフトウェアの品質管理において重要な役割を担う開発者であるため，開発者にコミット権限を付与するかどうかの判断は慎重に行われる．そのため，コミッターに昇格するためには，既存のコミッターやプロジェクトのコアメンバーからの推薦や承認が必要となる [10]．既存のコミッターやプロジェクトのコアメンバーは，コミット権限を持たない開発者にコミット権限を与えるかどうかの判断を開発者のプロジェクトに対するこれまでの活動（パッチ投稿や不具合に関するコメントなど）を基に行う．しかしながら，既存のコミッターやプロジェクトのコアメンバーが推薦や承認を行うための最低限必要となる活動内容や活動量は定義されておらず，既存のコミッターやプロジェクトのコアメンバーが経験的にコミッターを選出している現状である．

2.2 コミッター不足とその問題点

コミット権限を付与するかどうかの判断は慎重に行われるため，コミット権限を持たない開発者に比べ，コミッターは少数である．Mozilla Firefox プロジェクトでは，約 10 年間でパッチの投稿や議論への参加などのプロジェクトに対する貢献を行った開発者は 76,000 人を超えたものの，そのうち，コミッターである開発者は約 550 人であった．投稿されたパッチの検証作業は，少数のコミッターが中心となって行われるため，コミッターに対して負担がかかっている．日々大量に不具合報告が行われている大規模 OSS プロジェクトにおいては，コミッターの過度な負担が原因となり，パッチ検証が効率的に行われず，その結果，不具合の修正時間が長期化していることが報告されている [4]．

コミッターの作業負担を軽減し，不具合の修正時間を短縮させるために，コミッターとして活動する開発者を増やすという手段が効果的であると考えられる．しかしながら，数万人に及ぶ開発者の過去の活動を把握し，コミット権限を付与すべき開発者を特定することは現実的に困難である．また，既存のコミッターやプロジェクトのコアメンバーがコミット権限を持たない開発者にコミット権限を与えるかどうかの判断を行うために，開発者は通常 1 年以上パッチの投稿や議論への参加などといったプロジェクトに対する貢献が求められる．しかし，OSS プロジェクトに参加する開発者は，活動に対する意欲の喪失などの理由から，活動期間が 1 年を超えるとプロジェクトから離脱してしまう確率が上がることが知られている [5]．このような問題を解決するために，開発者の中からコミッターに昇格すべき開発者をできるだけ早い段階で特定する必要がある．

2.3 先行研究

開発者の中からコミッターに昇格すべき開発者を特定するために，既存のコミッターとコミット権限を持たない開発者の活動を学習し，将来コミッターに昇格する開発者を予測する研究が行われている [9,15]．伊原らは，コミッター昇格以前の開発者をコミッター候補者，コミット権限を持たない開発者を一般開発者と定義し，それぞれの開発者において，活動量メトリクス（活動期間，総パッチ投稿数，総コメント投稿数など）を計測している [9,15]．先行研究では，Eclipse Platform プロジェクトおよび，Moziila

Firefox プロジェクトを対象に実験を行い，両プロジェクトともに，再現率（実際にコミッター候補者であるもののうち，コミッター候補者であると予測されたものの割合）が高く，Eclipse Platform プロジェクトでは，総コメント投稿数，Mozilla Firefox プロジェクトでは活動期間が最も予測に寄与するメトリクスであることを明らかにした．

大規模 OSS 開発では，各コミッターがソフトウェア全体を詳しく理解することは困難であるため，各コミッターが担当すべきモジュールが決まっている．そのため，先行研究のように有能なボランティアの開発者を全て昇格させれば不具合修正の長期化の問題が解決できるとは限らない．本研究では，特定のモジュールを担当するコミッターをモジュールオーナーと呼ぶ．例えば，図 1 に示すように，モジュール A，モジュール B，モジュール C，モジュール D から構成される OSS が存在し，各モジュールのオーナーが存在しているとする．今，モジュール C において，不具合修正のために投稿されたパッチが集中しており，パッチが効率的に検証されないことから不具合修正が長期化している状況である．このような状況において，投稿されたパッチを効率的に検証するためには，モジュール C のモジュールオーナーを増員することが求められる．しかしながら，先行研究の予測モデルでは，各開発者がどのモジュールを担当するのにふさわしい開発者なのかといったより細粒度な予測を行うことができないため，予測された開発者が本来求められるモジュールとは異なるモジュールの担当となる可能性が存在する．そのため，コミッターと予測された開発者が本来求められるモジュールとは異なるモジュールの担当となった場合，不具合修正の長期化の問題を解決することはできない可能性がある．

そこで本研究では，モジュールオーナーの不足による不具合修正の長期化を解決するために，モジュールオーナー候補者を特定するための予測モデルの構築を行う．モジュールオーナー候補者を特定することにより，オーナーが不足しているモジュールに対して，モジュールオーナーを増員することができ，パッチの検証作業が効率的に行われることが期待される．

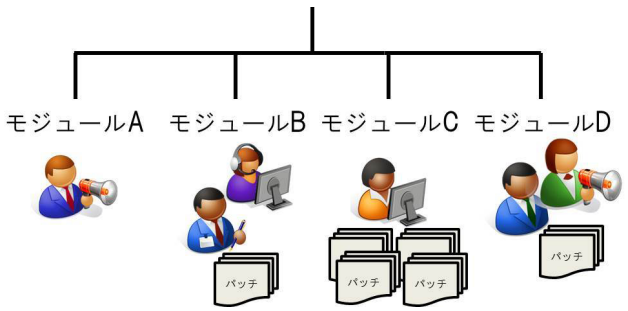

図 1 特定のモジュールにコミッターが必要となる事例

3. モジュールオーナー候補者予測

3.1 概要

本研究では，担当者が不足しているモジュールに適したコミッターを推薦するために，モジュールごとのコミッター候補者予測を行う．モジュールごとのコミッター候補者予測モデルを構築するために，先行研究 [9] で用いられている活動量メトリクスをモジュールごとに計測する．また，モジュールオーナーの適性を表すメトリクスを定義する．モジュールオーナーの適性を表すメトリクスは，開発者が特定のモジュールに対して，どれだけパッチの投稿通じて貢献を行ったかの割合を示すものである．このメトリクスの値が大きいほど，特定のモジュール内での貢献度が大きいことを示しており，モジュールオーナーに昇格する可能性が高くなると想定している．また，予測モデル構築には，先行研究 [9] で用いられているロジスティック回帰分析および，ランダムフォレスト法を用いる．

3.2 予測に用いるメトリクス

モジュールごとのコミッター候補者予測モデルの構築を行うために用いるメトリクスについて説明する．本研究で用いるメトリクスは，モジュールごとの活動量メトリクスおよび，モジュールオーナーの適性を示すメトリクスである．以下にそれぞれのメトリクスについて説明する．

3.2.1 モジュールごとの活動量メトリクス

モジュールごとのコミッター候補者予測モデルを構築するにあたり，まず，既存のコミッター候補者予測モデルで用いられている各開発者の活動量メトリクスを用いることが有効であると考えられる．しかしながら，先行研究のコミッター候補者予測モデルは，開発者がコミッターになるか否かを予測するためのモデルであり，モジュールごとに予測するためのモデルではない．そこで，本研究では，先行研究 [15] で用いられているメトリクスをモジュールごとに計測することとする．以下に，モジュールごとに計測する各開発者の活動量メトリクスについて説明する．

活動期間

活動期間は，各開発者が特定のモジュールに対してパッチ投稿あるいはコメント投稿を行った月数で表わされる．活動期間は，開発者がプロジェクトに参加している期間中に，パッチ投稿あるいはコメント投稿が行われていない月があればカウントされない．

総コメント投稿数

総コメント投稿数は，各開発者が特定のモジュールに対してコメントを投稿した回数で表わされる．

月コメント投稿数

月コメント投稿数は，各開発者が 1 ヵ月に特定のモジュールに対してコメントを投稿した回数の中央値で表される．

表 1　データセットの基本情報一覧

プロジェクト名	対象期間	不具合数	コミット回数	コミッター数	一般開発者数
Eclipse Platform	2001/10-2012/10	96,387	94,990	195	13,297
Mozilla Firefox	1999/07-2013/01	120,468	13,179	530	76,931
GCC	1999/07-2014/03	59,983	106,636	141	10,282
GIMP	2001/02-2013/09	12,030	28,163	87	4,068
WebKit	2000/12-2014/04	121,685	150,392	363	8,017

総パッチ投稿数

総パッチ投稿数は，各開発者が特定のモジュールに対してパッチを投稿した回数で表わされる．

月パッチ投稿数

月パッチ投稿数は，各開発者が 1 ヵ月に特定のモジュールに対してパッチを投稿した回数の中央値で表される．

3.2.2　モジュールオーナーの適性

モジュールオーナーとなる開発者を予測するためのモデルを構築するために，モジュールごとに計測される各開発者の活動量メトリクスに加え，モジュールオーナーの適性をメトリクス化する．

モジュールオーナーとなる開発者は，モジュール内において，貢献の割合の高い開発者であると考えられる．そのため，本研究では，モジュールオーナーの適性を表すメトリクスとしてパッチ貢献度を定義する．パッチ貢献度（PCR : Patch Contibution Rate）は式 1 で定義される．

$$PCR = \log e(1 + \frac{\#patchDev}{\#patchModule}) \quad (1)$$

式 1 における $\#patchDev$ は，開発者が特定のモジュールに対して投稿したパッチの回数であり，$\#patchModule$ は，コミッターであれば，コミッターに昇格するまでに特定のモジュールにパッチが投稿された回数を指し，一般開発者であれば，開発者が最後に活動（パッチ投稿あるいはコメント投稿）を行った日付より以前に特定のモジュールにパッチが投稿された回数を指す．つまり，式 1 における $\frac{\#patchDev}{\#patchModule}$ は，特定のモジュールに占める開発者が投稿したパッチの割合を表現している．

また，式 1 における $\frac{\#patchDev}{\#patchModule}$ の値の分布は，べき分布となるため，対数変換を行う．$\frac{\#patchDev}{\#patchModule}$ の値は 0 をとる場合もあるため，$\frac{\#patchDev}{\#patchModule}$ の値に 1 を加え，対数変換を行う．

3.3　予測モデル

本研究で構築する予測モデルは，現在モジュールオーナーでない開発者が将来モジュールオーナーになるかどうかを予測する．そこで，先行研究 [9,15] で用いられているロジスティック回帰分析およびランダムフォレスト法を用いて予測モデルの構築を行う．本研究では，ロジスティック回帰分析を統計解析ソフトウェアである R[*1] の glm 関数を用いて行い，ランダムフォレスト法を R の randomForest 関数を用いて行う．そして，モジュールオーナーに昇格するか否かを目的関数（モジュールオーナーに昇格する場合 1，モジュールオーナーに昇格しない場合 0）とし，予測モデルの出力値が 0.5 以上のときにモジュールオーナーに昇格すると判断する．

4.　実験

4.1　実験の目的

本実験の目的は，モジュールのオーナーにふさわしいコミッターを予測する際，本研究で新たに導入したパッチ貢献度メトリクス（PCR）が予測精度向上に寄与することを確かめることである．5 件の大規模 OSS プロジェクトを対象に，先行研究 [9] で用いられているメトリクスとパッチ貢献度メトリクスのすべての組合せについて予測モデルを構築する．予測精度の評価は，適合率，再現率，F1 値を用いて交差検証によって行い，最も予測精度が高くなるメトリクスの組み合わせを明らかにする．

4.2　データセット

本論文では，大規模 OSS プロジェクトである Eclipse Platform プロジェクト，Mozilla Firefox プロジェクト，GCC プロジェクト，GIMP プロジェクト，WebKit プロジェクトを対象にする．各プロジェクトは，長期に渡って開発・保守が行われているプロジェクトであり，開発に携わる開発者が多いため，本研究の対象プロジェクトとした．各プロジェクトの基本情報を表 1 に示す．

4.2.1　モジュールおよびモジュールオーナーの特定

モジュールごとにパッチ投稿数やコメント投稿数を計測するために，どのモジュールに対して投稿されたパッチあるいはコメントなのかの特定を行う．不具合管理システムにおいて，報告された不具合には，ソフトウェアのどのモジュール（コンポーネント）[*2] の不具合なのかといった情報が記載されている．基本的には，この情報を基に分割を行う．しかしながら，不具合情報に記載されているモジュールの情報では，各モジュールの粒度に大きく差が出てしまう．そのため，ソースコードの構造および，不具合

[*1] 統計解析ソフトウェア R : http://www.r-project.org/index.html
[*2] 本研究では，モジュールとコンポーネントを同じ意味として捉える．

表 2　各プロジェクトでのモジュールオーナーの数

プロジェクト名	モジュール名	モジュールオーナー数
Eclipse Platform	Debug	15
	Releng	4
	SWT	12
	UI	52
	Workspace	16
Mozilla Firefox	App	18
	Bar	66
	Bookmark	54
	DeveloperTools	45
	General	73
	Preferences	66
	Tab	64
GCC	Ada	1
	BackEnd	13
	Build	1
	C	9
	Fortran	3
	Java	1
	MiddleEnd	16
GIMP	General	9
	Plugin	13
	Tools	7
	UserInterface	11
WebKit	API	151
	Build	155
	Inspector	40
	JavaScriptCore	42
	Layout	140
	WebCore	180
	WebKit2	21

管理システムのコンポーネント情報，プロジェクトの Wiki ページを用いて総合的にモジュールの特定を行う．また，本研究では，モジュールオーナーの特定方法として，特定のモジュールに対して投稿されたパッチをバージョン管理システムに反映しているかどうかを調べる．具体的には，特定のコミッターがコミットを行ったコミットログの説明文から不具合 ID を取得し，不具合 ID から不具合管理システムのコンポーネント情報を参照し，モジュールを特定する．以上の方法で特定した各プロジェクトのモジュールオーナーの数を表 2 に示す．2 から，GCC プロジェクトでの Ada，Build，Java モジュールは，モジュールオーナーの数が 1 人であるため，予測することができない．そのため，本研究で対象とするモジュールから除外した．

4.3　実験方法

4.3.1　評価指標

予測精度の評価指標として適合率（Precision），再現率（Recall），F1 値を用いる．適合率とは，モジュールオーナーであると予測した結果のうち，実際にモジュールオーナーであった割合を指し，再現率とは，実際にモジュールオーナーである開発者のうち，モジュールオーナーと予測された割合を指している．

また，適合率と再現率はトレードオフの関係にあるため，適合率と再現率の調和平均を取った値である F1 値を用いて予測精度の評価を行う．F1 値は式 2 によって定義される．

$$F1 - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2}$$

4.3.2　比較評価

モジュールオーナーを特定するための有用な指標を明らかにするために，モジュールごとに計測した活動量メトリクスおよび，パッチ貢献度のすべての組み合わせについて予測モデルを構築する．そして，本研究で新たに定義したメトリクスであるパッチ貢献度が予測精度向上に寄与するかどうかを確かめるために，パッチ貢献度を含むモデルとパッチ貢献度を含まないモデルでの比較評価を行う．

4.3.3　実験手順

本研究で行う評価実験の手順を以下に示す．

手順 1：データセットの分割　モジュールオーナーと非モジュールオーナーのそれぞれのデータをランダムに 2 分割し，半分を学習データ，もう半分をテストデータとする．例えば，モジュールオーナーのデータが 100 件，非モジュールオーナーのデータが 10,000 件あったとする．このとき，ランダムに抽出したモジュールオーナーのデータを 50 件，同じくランダムに抽出した非モジュールオーナーのデータを 5,000 件取得し，学習データとする．そして抽出されなかった残りのデータをテストデータとする．

手順 2：学習データのバランス調整　非モジュールオーナーに比べモジュールオーナーの割合は著しく低いため，正しい予測ができない恐れがある [8, 12]．そのため，学習データのバランスの調整を行うために，非モジュールオーナーの学習データをモジュールオーナーの学習データと同じ数に合わせる．例えば，モジュールオーナーの学習データが 50 件，非モジュールオーナーの学習データが 5,000 件あった場合，非モジュールオーナーの学習データから 50 件ランダムに抽出し，ランダムに抽出した 50 件のデータを学習データとする．

手順 3：学習　手順 2 で作成した学習データの学習を行う．本研究では，学習アルゴリズムとしてロジスティック回帰および，ランダムフォレスト法を用いる．モジュールオーナーである開発者を 1，非モジュールオーナーである開発者を 0 として学習する．

手順 4：予測　手順 3 で学習した結果に基づき，テストデータを用いて予測精度を算出する．予測精度の評価指標には，適合率，再現率，F1 値を用いる．予測の結果，目的関数が 0.5 以上であった場合，モジュールオーナーであると判断する．

手順 5：交差検証　学習データとテストデータはランダムに作成されるため，予測結果が毎回変化する．そのた

表 3　実験結果

	ロジスティック回帰分析					ランダムフォレスト法			
プロジェクト	PCR 有無	予測モデル	適合率	再現率	F1 値	予測モデル	適合率	再現率	F1 値
Eclipse	PCR 有	PCR	**0.146**	**0.967**	**0.248**	PCR	**0.124**	**0.967**	**0.218**
Platform	PCR 無	パッチ数	0.107	0.856	0.183	活動期間+パッチ数	0.116	0.909	0.196
Mozilla	PCR 有	PCR	**0.278**	**0.992**	**0.412**	PCR	0.267	1.000	0.404
Firefox	PCR 無	パッチ数	0.276	0.912	0.409	活動期間+パッチ数	0.267	1.000	0.404
GCC	PCR 有	PCR	**0.074**	**1.000**	**0.129**	PCR	**0.074**	**1.000**	**0.129**
	PCR 無	パッチ数	0.031	0.917	0.044	先行研究のモデル	0.017	0.938	0.033
GIMP	PCR 有	PCR	**0.124**	**1.000**	**0.219**	PCR+活動期間	**0.129**	**0.938**	**0.225**
	PCR 無	パッチ数	0.115	0.813	0.201	パッチ数	0.123	0.875	0.213
WebKit	PCR 有	PCR	**0.213**	0.880	**0.332**	PCR+活動期間	**0.168**	0.987	**0.284**
	PCR 無	パッチ数	0.189	**0.905**	0.308	活動期間+パッチ数	0.165	**0.989**	0.280

め，手順 1 から手順 4 を 100 回繰り返し，100 回行った結果の中央値を最終的な予測精度とする．

手順 6：全モデルでの実行　手順 1 から手順 5 をメトリクスのすべての組み合わせについて行う．

手順 7：全モジュールに対しての実行　手順 1 から手順 6 は，特定のモジュールについて行われる．そのため，手順 1 から手順 6 を全モジュールに対して行う．

4.4　結果

評価実験を行った結果について述べる．表 3 には，ロジスティック回帰およびランダムフォレスト法を用いて行った各プロジェクトの F1 値が最も高くなったモデルの予測結果を示している．各評価指標の値は，各モジュールでの結果の平均となっている．ロジスティック回帰分析を用いて行った結果，すべてのプロジェクトについて，適合率および F1 値が最も高くなったモデルはパッチ貢献度のみを用いたモデルであった．パッチ貢献度を含まないモデルと比較すると，F1 値が最大 2.9 倍向上した．

また，ランダムフォレスト法を用いて行った結果では，Eclipse Platform プロジェクト，Mozillla Firefox プロジェクト，GCC プロジェクトでは，パッチ貢献度のみを用いたモデルの精度が高く，GIMP プロジェクトおよび WebKit プロジェクトでは，パッチ貢献度と活動期間を用いたモデルが最も精度が高くなった．パッチ貢献度を含まないモデルと比較すると，Mozila Firefox プロジェクトを除くすべてのプロジェクトで精度が向上し，F1 が最大 3.9 倍向上した．これらの結果から，パッチ貢献度は，モジュールオーナーとなる開発者を予測する際，予測精度向上に寄与することが明らかとなった．

5.　考察

5.1　パッチ貢献度とコメント貢献度の比較

本研究では，モジュールオーナーの適性を定量化するために，パッチ貢献度メトリクスを定義した．しかし，先行研究 [9] にもあるように，OSS に参加する開発者の活動には，パッチ投稿の他にもコメント投稿がある．そこで，本節では，パッチ貢献度と同様に，コメント貢献度を各開発者で計測し，コメント貢献度がモジュールオーナーを特定する際に有力な指標となり得るのかを確かめる．

表 4　パッチ貢献度とコメント貢献度の比較

プロジェクト名	モデル名	適合率	再現率	F1 値
Eclipse Platform	PCR	**0.146**	**0.967**	**0.248**
	CCR	0.039	0.878	0.074
	PCR+CCR	0.100	0.902	0.175
Mozilla Firefox	PCR	**0.278**	**0.992**	**0.412**
	CCR	0.023	0.897	0.044
	PCR+CCR	0.163	0.967	0.268
GCC	PCR	**0.074**	**1.000**	**0.129**
	CCR	0.041	1.000	0.076
	PCR+CCR	0.073	0.969	0.128
GNOME	PCR	**0.124**	**1.000**	**0.219**
	CCR	0.018	0.896	0.036
	PCR+CCR	0.093	0.659	0.163
WebKit	PCR	**0.213**	0.880	**0.332**
	CCR	0.173	0.83	0.269
	PCR+CCR	0.196	**0.888**	0.312

コメント貢献度（CCR：Comment Contribution Rate）は，パッチ貢献度と同様に式 3 で定義される．

$$CCR = \log e(1 + \frac{\#commentDev}{\#commentModule}) \quad (3)$$

ここで $\#commentDev$ は，開発者が特定のモジュールに対して投稿したコメントの回数であり，$\#commentModule$ は，コミッターであれば，コミッターに昇格するまでに特定のモジュールにコメントが投稿された回数を指し，一般開発者であれば，開発者が最後に活動（パッチ投稿あるいはコメント投稿）を行った日付より以前に特定のモジュールにコメントが投稿された回数を指す．

コメント貢献度がモジュールオーナーを特定する際に有力な指標となり得るのかを確かめるために，評価実験で対象とした 5 件のプロジェクトに対して追加実験を行う．実験の方法は，評価実験でモジュールオーナーを特定する際に有力な指標であると判断されたパッチ貢献度との比較を

行う．具体的には，パッチ貢献度のみを用いた予測モデルとコメント貢献度のみを用いた予測モデル，パッチ貢献度およびコメント貢献度を用いた予測モデルの 3 つの予測モデルでの比較を行う．パッチ貢献度のみを用いたモデルに比べ，コメント貢献度のみを用いた予測モデルあるいは，パッチ貢献度およびコメント貢献度を用いた予測モデルの方が予測精度が高ければ，コメント貢献度メトリクスがモジュールオーナーを特定する際に，有力な指標であるというになる．評価指標としては，適合率，再現率，F1 値を用いる．また，学習アルゴリズムとして，ロジスティック回帰分析を用いる．表 4 に追加実験を行った結果を示す．表 4 に示すように，WebKit プロジェクトの再現率を除くすべての評価指標において，パッチ貢献度のみを用いたモデルが最も高い値となった．このことから，コメント貢献度は，モジュールオーナーを特定する際の有力な指標にはならないことが明らかとなり，モジュールオーナーの適性として，パッチ貢献度が適切であることが結論付けられた．

5.2 制約

5.2.1 モジュール分割方法

本研究で行った評価実験では，各プロジェクトをモジュール単位で分割する際，ソースコードの構造や不具合管理システムのコンポーネント情報，プロジェクトの Wiki ページを参照し，これらの情報を基に，著者が分割を行った．そのため，モジュール分割の際の明確な基準は存在しないため，本研究のモジュール分割方法では，著者の主観による分割となっている．そのため，分割を行う人によって，モジュール分割の結果および，モジュールの粒度が異なる可能性がある．モジュール分割の結果が異なることにより，予測結果が異なる可能性があることに留意されたい．

5.2.2 不具合情報とコミット情報のひも付け方法

本研究では，モジュールオーナーを特定のモジュールに投稿されたパッチをバージョン管理システムにコミットしている開発者とした．そのため，不具合の情報と不具合修正のために行ったコミットの情報とのひも付けを行う必要があった．不具合情報とコミット情報のひも付け方法として，コミットの説明文に含まれるテキストから，不具合 ID を抽出するアプローチをとった．しかしながら，不具合修正のためのコミットであるのにも関わらず，コミットの説明文に不具合 ID を記載しないコミットが存在する．そのようなコミットが存在することにより，データセットに偏りが発生してしまい，本来の結果とは異なる結果になってしまう可能性があることが報告されている [3]．

本研究においても，この不具合情報とコミット情報のひも付けミスによるデータセットの偏りが生じる可能性があるものの，今回行った実験では，総コミット回数が 15 万回以上あるプロジェクトを対象にしていることから，各コミットが不具合修正のためのコミットなのかどうかを目視で判断することは現実的に不可能である．そのため，今回では，コミットの説明文に含まれるテキストから，不具合 ID を抽出するアプローチをとり，機械的に不具合情報とコミット情報のひも付けを行った．

6. 関連研究

6.1 コミッター昇格に関する研究

本研究は，OSS 開発において重要な役割を担うコミッターに注目した．コミッターに関する研究が盛んに行われており，本節では，OSS 開発における，開発者のコミッター昇格プロセスに関する研究について述べる．

Jensen らは，OSS プロジェクト内での開発者の役割が変化するプロセスを明らかにするために，開発者にインタビューを行うことや，公開されているドキュメントなどの分析を行っている．分析の結果，プロジェクトのコア開発者からの推薦や承認を経て，コミッターへの昇格が行われていることを明らかにしている [10]．

Zhou らは，活動期間の長い開発者を Long-Term Contributor（LTC）とし，開発者がプロジェクトに参加した初期の時点で将来開発者が LTC となるかどうかの予測を行っている [14]．ケーススタディの結果，開発者が LTC となるかどうかは，プロジェクトを取り巻く環境および，開発者の意欲，開発者の能力によってモデル化できることを明らかにした．

Shinha らは，コミッターに昇格するなどといったプロジェクトのコア開発者になる要因を，ソースコードの変更履歴や，報告された不具合の修正履歴などの情報を用いて分析している [13]．Shinha らは，コア開発者になる開発者は，OSS にとってより重要な部分のコードに対しての貢献があることや，プロジェクトのコアメンバーである開発者と協調作業を行っていることを明らかにした．

Gharehyazie らは，コア開発者の昇進にはパッチ投稿などの技術的な貢献だけでなく，議論に参加するなどの社会的な貢献も考慮されると考え，開発者間のコミュニケーションによるソーシャルネットワークメトリクスを用いて，将来の開発者（コミッターなど）を予測している [7]．

これらの研究では，OSS に参加している開発者がプロジェクトのコア開発者になるまでのプロセスのモデル化を行っているが，本研究では，各開発者のモジュールごとの適性を考慮している点で異なっている．

6.2 開発者の適性に関する研究

本研究では，モジュールのオーナーにふさわしい開発者を推薦するために，モジュールオーナーの適性を定義した．本研究と同様に，各開発者の適性を把握するための研究が行われている．以下に，各開発者の適性を把握するための研究について述べる．

Mockus らは，開発者の適性を可視化するために，Ex-

pertise Browser というツールを開発した [11]．Expertise Browser は，開発者のソースコードの変更履歴を収集することにより，各開発者がどういった言語を得意としているか，といったことや，特定のファイルはどの開発者によって主に変更されているかなどといった情報をユーザーに提示している．

Alonso らは，各コミッターがどの技術分野に特に詳しいのかを明らかにするために，各コミッターが得意とする技術分野を可視化するツールを開発した [1]．Alonso らが開発したツールでは，ソースコードの構造および，コミット時の説明文から，各コミッターがよく変更しているファイルに含まれるキーワードや，コミット時の説明文によく記述しているキーワードを抽出する．そして，それらのキーワードをタグクラウドによって表示している．

このように，各開発者の適性を把握するための研究が盛んに行われており，本研究でも，各開発者のモジュールごとの適性を考慮していることから，これらの研究と共通する部分も少なくない．しかしながら，本研究は，モジュールオーナーとなる開発者を予測している点でこれらの研究とは異なっている．

7. おわりに

本研究では，特定のモジュールを担当するコミッターをモジュールオーナーと呼び，モジュールオーナーとなるべき開発者をボランティアの開発者の中から特定するための予測モデルを構築した．予測モデル構築のために，先行研究 [9, 15] で用いらている活動量メトリクス（活動期間，パッチ投稿数，コメント投稿数など）をモジュールごとに計測した．また，モジュールオーナーの適性を表現するメトリクスであるパッチ貢献度を新たに定義した．そして，モジュールオーナーとなる開発者を予測する際，パッチ貢献度が予測精度向上に寄与するかどうかの評価実験を行った．評価実験は，5 件の大規模プロジェクト（Eclipse Platform プロジェクト，Mozilla Firefox プロジェクト，GCC プロジェクト，GIMP プロジェクト，WebKit プロジェクト）を対象に行った．評価実験の結果，5 件の OSS プロジェクトのすべてにおいて，パッチ貢献度を用いない予測モデルに比べ，パッチ貢献度を用いた予測モデルは，予測精度が向上することが明らかになった．今後は，適合率の向上を目指し，新たなメトリクスを定義する．

謝辞 本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参考文献

[1] Alonso, O., Devanbu, P. T. and Gertz, M.: Expertise Identification and Visualization from CVS, *Proceedings of the 5th International Workshop on Mining Software Repositories (MSR'08)*, pp. 125–128 (2008).

[2] Anvik, J., Hiew, L. and Murphy, G. C.: Who should fix this bug?, *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, pp. 361–370 (2006).

[3] Bachmann, A. and Bernstein, A.: Software Process Data Quality and Characteristics A Historical View on Open and Closed Source Projects, *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops (IWPSE-EVOL'09)*, pp. 119–128 (2009).

[4] Bettenburg, N., Premraj, R., Zimmermann, T. and Kim, S.: Duplicate Bug Reports Considered Harmful . . . Really?, *Proceedings of the 24th International Conference on Software Maintenance (ICSM'08)*, pp. 337–345 (2008).

[5] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07)*, p. No.6 (2007).

[6] Dabbish, L., Stuart, C., Tsay, J. and Herbsleb, J. D.: Leveraging Transparency, *IEEE Software*, Vol. 30, No. 1, pp. 37–43 (2013).

[7] Gharehyazie, M., Posnett, D. and Filkov, V.: Social Activities Rival Patch Submission For Prediction of Developer Initiation in OSS Projects, *Proceedings of the 29th International Conference on Software Maintenance (ICSM'13)*, pp. 340–349 (2013).

[8] He, H. and Garcia, E.: Learning from Imbalanced Data, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 9, pp. 1263–1284 (2009).

[9] Ihara, A., Kamei, Y., Ohira, M., Hassan, A. E., Ubayashi, N. and Matsumoto, K.: Early Identification of Future Committers in Open Source Software Projects, *Proceedings of the 14th International Conference on Quality Software (QSIC'14)*, pp. 47–56 (2014).

[10] Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, pp. 364–374 (2007).

[11] Mockus, A. and Herbsleb, J. D.: Expertise Browser: A Quantitative Approach to Identifying Expertise, *Proceedings of the 24th International Conference on Software Engineering (ICSE'02)*, pp. 503–512 (2002).

[12] Owen, A. B. and Lin, Y.: Infinitely Imbalanced Logistic Regression, *Machine Learning*, Vol. 8, pp. 761–773 (2007).

[13] Sinha, V. S., Mani, S. and Sinha, S.: Entering the Circle of Trust: Developer Initiation as Committers in Open-Source Project, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, pp. 133–142 (2011).

[14] Zhou, M. and Mockus, A.: Developer fluency: achieving true mastery in software projects, *Proceedings of the 18th ACM SIGSOFT international symposium on Foundations of software engineering (FSE'10)*, pp. 137–146 (2010).

[15] 伊原彰紀，亀井靖高，大平雅雄，松本健一，鵜林尚靖：OSS プロジェクトにおける開発者の活動量を用いたコミッター候補者予測，電子情報通信学会論文誌，Vol. 95, No. 2, pp. 237–249 (2012).