コミッター選出のためのプロジェクトメトリクスに関する定量的評価

金城 清史[1]　山谷 陽亮[1]　大平 雅雄[1]

概要：近年，オープンソースソフトウェア (OSS) 開発において重要な役割を担うコミッターに関する研究が盛んに行われている．従来研究は，OSS プロジェクトに参加する開発者個人の活動量を重視した分析を行っているのに対し，本研究では，OSS プロジェクトを取り巻く状況に着目して，コミッターへの昇格のしやすさを分析する．本論文では，OSS プロジェクトを取り巻く状況を「プロジェクトメトリクス」と呼び，その状況を数値化するための具体的な尺度を提案する．また，コミッター昇格のしやすさを「コミッター昇格率」として定義し，プロジェクトメトリクスとの関係を明らかにする．特に本論文では，長期間にわたって運営され，多数のコミッターが在籍している 3 つの大規模 OSS プロジェクト (Eclipse Platform, Mozilla Firefox, GCC) を対象に相関分析を行った結果について報告する．

キーワード：プロジェクトメトリクス，コミッター候補者予測，オープンソースソフトウェア

1. はじめに

大規模オープンソースソフトウェア (OSS) プロジェクトには，日々，大量の不具合報告が報告されている [5]．それら大量の不具合を修正するために，開発者により既存のソースコードを修正するパッチ（コード差分）が作成され，プロジェクトに投稿される．投稿された不具合修正パッチは，コミット権限と呼ぶ特別な権限を与えられた開発者（以降，一般開発者と区別するためにコミッターと呼ぶ）によって検証され，問題がなければプロダクトに反映される．このパッチ検証プロセスは，品質保証の観点から多くの OSS プロジェクトにおいて採用されている．不特定多数の開発者から投稿されるパッチ自体にも不具合を含むことが少なくないため，有能なコミッターによるパッチ検証によってプロダクトに新たな不具合が混入する機会を減らすことが主な目的である．

しかしながら，コミッターは一般的に，プロジェクト参加中に優れた能力を示した開発者の中から選出されるため，多くの OSS プロジェクトでは，全開発者に占めるコミッターの数は極めて少数であることが知られている [7]．特に大規模 OSS プロジェクトでは，少数のコミッターへ過度な負荷がかかるため，結果的に，不具合修正パッチが実際に適用されるまでに多くの時間を要する場合がある [8]．

コミッターの負担を軽減するためには，コミッターになり得る有能な一般開発者にコミット権限を新たに付与し（コミッターとして昇格させる），パッチ検証要員を増やす方法が考えられる．多くの OSS プロジェクトでは，継続的にプロジェクトに貢献した一般開発者の中からコミッターが選出される．選出にあたっては，開発者の過去の活動内容（例えば，不具合修正に対する貢献の量と質，開発に関する議論に積極的に参加してきたか，など）を総合的に判断する．さらに，活動内容を正しく評価する必要があるため，通常は，一年以上の活動期間があることが前提となる．

ただし，コミット権限を持たない一般開発者が OSS プロジェクトに長期間参加するのは稀である [2] ことも知られており，多数の開発者を抱える OSS プロジェクトであっても，コミッターを増やすのは容易なことではない．そのため，出来るだけ早い段階で有能な一般開発者を見つけ出し，コミッター昇格させることを支援する研究（コミッター候補者予測 [16] など）が盛んに行われている．コミッター候補者に関する従来の研究では，OSS プロジェクトに参加する開発者個人の活動量（例えば，パッチ投稿数など）を分析し，コミッター候補者として選定するための基準を構築したり，コミッター候補者を推薦するための予測モデルを提案している [14, 16]．

本研究では，OSS プロジェクトを取り巻く状況に着目する．プロジェクト全体として対処すべき不具合の多い時期や，開発に関する議論が必要な時期は，そうでない時期に比べて，コミッター昇格のしやすさ（プロジェクトにおける新規コミッターに対するニーズ）が異なると考えたた

[1] 和歌山大学 システム工学部 情報通信システム学科
和歌山県和歌山市栄谷 930 番地

めである．従来研究が着目する開発者個人の活動量のみでは，プロジェクトの状況に応じてコミッター候補者を過不足なく推薦することは困難であると考えられるため，プロジェクトを取り巻く状況とコミッター昇格のしやすさとの関係を明らかにすることは，既存研究における推薦精度や基準の改良にとっても意義があることと思われる．

本論文では，OSS プロジェクトを取り巻く状況を「プロジェクトメトリクス」と呼び，その状況を数値化するための具体的な尺度として「コメント人数」「参加期間」「不具合修正率」を用いる．また，コミッター昇格のしやすさを「コミッター昇格率」として定義し，プロジェクトメトリクスとの関係を明らかにする．特に本論文では，長期間にわたって運営され，多数のコミッターが在籍している 3 つの大規模 OSS プロジェクト (Eclipse Platform, Mozilla Firefox, GCC) を対象に，相関分析を行った結果について報告する．

本論文の構成は次の通りである．続く 2 章では，コミッター昇格およびコミッター候補者推薦に関連する既存研究を紹介し，本研究との立場の違いを明らかにする．3 章では，本研究で用いる 3 種類のプロジェクトメトリクスについて述べ，それぞれのメトリクスとコミッター昇格率との関係についての仮説を提示する．4 章では，3 つの大規模 OSS プロジェクト (Eclipse Platform, Mozilla Firefox, GCC) を対象に相関分析を行い，本研究の仮説を検証する．5 章では，相関分析によって得られた知見について議論する．最後に 6 章において，まとめと今後の課題について述べ，本論文を結ぶ．

2. 関連研究

本章ではまず，OSS 開発における参加者の役割移動に関する関連研究を紹介する．次に，OSS 開発において中心的役割を担うコア開発者であるコミッターへの昇格と一般開発者の活動期間について分析した研究について述る．最後に，コミッター候補者予測に関する先行研究について述べ，本研究の動機と立場を明らかにする．

2.1 OSS 開発における役割移動

OSS 開発では，コード作成に貢献する開発者のみならず，OSS を利用する中で発見した不具合を報告するユーザも，OSS の品質向上に貢献しているという点において，単なる参加者ではなく共同開発者として見なされることがある [3]．さらに，OSS を利用するのみのユーザであっても，OSS 利用者の存在自体が開発者の動機付けに貢献すると言われている [9]．

Ye ら [11] は，OSS 開発における参加者の役割をいくつかの階層に分類してモデル化（図 1）し，成功している OSS プロジェクトでは，周辺的な参加から徐々に中心的な役割を担う開発者へと役割が移動するプロセスが存在することを明らかにした．このようなプロセスは，正当的周辺参加 (LPP: legitimate peripheral participation) [6] として文化人類学の分野で指摘されている，コミュニティ内での学習プロセスと一致する．OSS 開発を持続的に発展させるためには，知的好奇心を満たすのが主目的 [1] であるコードリーダーや，不具合を積極的に報告するアクティブユーザを LPP のプロセスに上手く取り込み，将来のコア開発者として育成する必要がある．

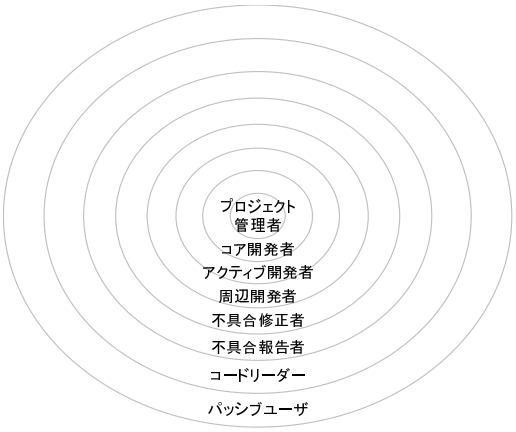

図 1 OSS 開発コミュニティの構造 [11]

本研究では，図 1 におけるコア開発者をコミッター相当の役割を担う開発者と見なす．一方，コミッター候補者は，アクティブ開発者に多く存在するものとして考えている．ただし，本研究における一般開発者は，不具合報告者からアクティブ開発者までを含むこととする．

2.2 コミッター昇格と一般開発者の活動期間

コミッターは，OSS プロジェクトで開発されるプロダクト（ソースコードやドキュメントなど）を管理するためのバージョン管理システム（Subversion[*1] や Git[*2] など）へ修正を加えることの出来る権限（コミット権限）を与えられたコア開発者である．プロダクトの品質に直接影響を与えることができるため，OSS プロジェクトの運営においては信頼のおける人物をコミッターとして採用するのが良いとされる [3]．

Ye らの分析結果を受け，Jensen ら [4] は OSS 開発者らへのインタビューに基づいて，OSS 開発における役割移動をさらに詳細に分析している．特に，一般開発者からコミッターへ昇格する方法には，推薦や投票など，いくつかの方法があることを明らかにしている．多くの OSS プロジェクトでは，プロジェクトに参加してすぐにコミッターに昇格されることは稀である．コミッターなどのコア開発者として選出されるまでには，一定期間プロジェクトへ多大な貢献を行っていることを，既存コミッターから認められる必要がある．

[*1] Apach Subversion: http://subversion.apache.org/
[*2] Git: http://git-scm.com/

また，Bird ら [2] の研究では，一般開発者が OSS プロジェクトに参加してから脱退するまでの平均活動期間を分析している．3 つの大規模プロジェクトを分析した結果，いずれのプロジェクトにおいても，一般開発者の平均活動期間は約 1 年程度と短期間であった．実装した新機能が採用されなかったり，作成した修正パッチがすぐに適用されなかったりするなど，コミット権限を持たない一般開発者にとって，プロジェクトに参加する動機が低下してしまう状況が多々発生する．長期にわたってプロジェクトに貢献してもらうには，高い貢献を見込める一般開発者を早い段階で見極め，コミット権限を付与することが望ましい．

一方，Zhou らは，長期にわたってプロジェクトに貢献する開発者の特徴を分析している [12–14]．開発者個人の動機 (attitude) に関する要因 [12] の他，プロジェクト参加時の状況 (macro-climate: プロジェクト全体を取り巻く状況，micro-climate: 開発者個人を取り巻く状況) [13] が長期参加と深く関係していることを明らかにしている [14]．本研究における分析は，コミッターのプロジェクト参加期間の長短を問わないものではあるが，Zhou らの分析は参考にできるものである．Zhou らの分析結果を受けて，本研究では，プロジェクトを取り巻く状況がコミッターの昇格率とも深い関係が存在するのではないかと考えた．Zhou らの研究が開発者視点で分析を行っているのに対し，本研究は，プロジェクト視点でコミッター昇格に関する要因を分析しようとする点に大きな違いがある．

2.3 コミッター候補者予測

コミッターに関する既存研究の分析結果から，近年では，一般開発者の中からコミッター候補者を予測し，既存コアメンバが新規コミッター選出するのを支援する研究が盛んに行われている．

伊原ら [15] らの研究では，32 のプロジェクトを対象にコミッター選出のために参考となり得る指標を明らかにしている．また，[15] での分析結果に基づいて，一般開発者の中からコミッターになり得る候補者を推薦する予測モデルを構築している．しかしながら，これらの研究で用いられるメトリクスは，開発者個人の活動量（開発者毎のパッチ投稿数やコメント数など）に基づいたもののみである．

本研究の最終的な目標は，コミッター候補者予測の精度を向上させることである．そのためには，開発者個人の活動量のみではなく，プロジェクトを取り巻く状況を考慮することが重要であると考えている．本研究では，3 つのプロジェクトメトリクスを用いて，コミッター昇格率との関係を調査する．プロジェクトメトリクスとコミッター昇格率との関係が明らかになれば，プロジェクトの状況に応じて必要な人数のコミッター候補者を見積もることが可能になるため，予測精度の向上のみならず予測結果を既存コアメンバが利用する際の有用性の向上が期待できる．

3. プロジェクトメトリクス

本章では，プロジェクトメトリクスについて述べる．本研究では，プロジェクト全体として対処すべき不具合の多い時期や，開発に関する議論が必要な時期は，そうでない時期に比べて，コミッター昇格のしやすさが異なる，という基本的な仮説に基づいて，3 種類のプロジェクトメトリクスを用いることとした．以降では，それぞれのメトリクスについて説明し，その後，3 種類のメトリクスそれぞれとコミッター昇格率との関係について立てた個別の仮説について述べる．

3.1 プロジェクトメトリクスの分類と定義

伊原らの研究 [15] では，開発者個人の活動量を測るメトリクスとして，参加期間，平均パッチ投稿数，平均レビュー数，平均コメント数，レビュー時間の 5 種類を用いている．これら 5 種類のメトリクスとコミッター昇格率との関係を調査した結果，調査対象となった 32 プロジェクトの多くで，一般開発者がコミッターに昇格するかどうかは，**開発に関する議論への参加度合い**を表す コメント数と**プロジェクトへの参加の長さ**を表す参加期間の 2 種類のメトリクスと関係があることが分かった．また，プロジェクトによっては**不具合修正への関与の度合い**を表す残り 3 種類のメトリクス（平均パッチ投稿数，平均レビュー数，レビュー時間）もコミッター昇格があることが分かった．

そこで本研究では，まず，プロジェクトを取り巻く状況を説明するメトリクスを，**議論に関するメトリクス**，**参加期間に関するメトリクス**，**不具合修正に関するメトリクス**の 3 つのカテゴリに分類する．本論文では，各カテゴリに 1 種類のプロジェクトメトリクスを用いているが，今後各カテゴリに属するメトリクスの種類を増やしていく予定である．

表 1 は，本研究で用いるプロジェクトメトリクスの一覧である．以下では，それぞれのメトリクスについて説明する．

3.1.1 コメント人数

議論に関するメトリクスとしては，開発に関する議論に参加した人数を用いる．開発に関する議論は，メーリングリストや不具合管理システムの掲示板を用いて行われることが一般的である．本論文では，OSS 開発において広く普及している不具合管理システムである Bugzilla[*3] における議論データを用いることとする．メトリクス計測期間（3 ヶ月）毎に Bugzilla 内の（不具合毎に利用可能な）掲示板で不具合修正に関するコメントをおこなった開発者の人数を**コメント人数**とする．コメント人数が多いほど，プロジェクト内で不具合修正に関する議論が活発になされてい

[*3] Bugzilla: http://www.bugzilla.org/

表 1　本研究で用いるプロジェクトメトリクス

カテゴリ	メトリクス名	定義
議論に関するメトリクス	コメント人数	開発に関する議論に参加した開発者の人数
参加期間に関するメトリクス	平均参加期間	コミッターの平均参加期間
不具合修正に関するメトリクス	不具合修正率	報告された不具合が修正 (Fixed) された割合

本研究では，3 ヶ月毎にメトリクスを集計する．したがって，上記のメトリクスの値は，メトリクス計測毎に得られる値となる．

ることを意味する．

3.1.2　参加期間に関するメトリクス

参加期間に関するメトリクスとしては，メトリクス計測期間に活動してるコミッターの平均プロジェクト参加期間（**平均参加期間**）を用いる．コミッターの識別および平均参加期間の算出には，ソースコードやドキュメントなどのプロダクトを管理するためのバージョン管理システムに記録されているコミットログを用いる．コミットログは，コミッターがプロダクトに変更を加える（コミットする）際に記録され，「いつどのコミッターがどのファイルのどの部分に変更を加えたか」などの情報を含む．メトリクス計測期間に存在するコミット履歴から，その時期に存在するコミッターを特定することができる．コミッターを特定できれば，そのコミッターが初めてバージョン管理システムにコミットした日時をコミッターに昇格した日時として，コミッターの参加期間を算出することができる．メトリクス計測期間に存在するすべてのコミッターの参加期間をコミッターの人数で除算することで平均参加期間が得られる．平均参加期間が長いということは，古参のコミッターが多く存在することを表し，平均参加期間が短いということは新規のコミッターが多い（コミッターの構成が流動的）であることを意味する．

3.1.3　不具合修正に関するメトリクス

不具合に関するメトリクスには，**不具合修正率**を用いる．不具合修正率には Bugzilla に記録された不具合に関する情報を用いる．

不具合修正率は，メトリクス計測期間中に Bugzilla で管理されているすべての不具合に対して修正が完了した不具合を割合として表したものである．Bugzilla で管理される不具合は，不具合が解決 (Resolution) した際に，解決方法の種類によって複数のタイプに分類される．不具合の主な解決方法には，FIXED（修正された），INVALID（無効），DUPLICATE（重複している），WONTFIX（修正しない），WORKSFORME（確認できない），MOVED（移動）などがあり，報告された不具合が実際に解決されたことを示すのは実質的に FIXED のみである．本研究では，修正が完了した不具合の数を計測する際には，解決方法が FIXED となった不具合のみを対象としている．不具合修正率が高ければプロジェクトに報告される不具合が特に問題なく修正できていることを意味し，不具合修正率が低ければ，既存コミッターが不具合修正あるいは修正パッチの検証などの作業に手間取っている可能性がある．

3.2　仮説

以下では，各プロジェクトメトリクスとコミッター昇格率との関係について，本研究の仮説を述べる．コミッター昇格率は，メトリクス計測期間内に存在する既存コミッターに対する新規コミッターの割合である．新規コミッターは，メトリクス計測期間内に昇格したコミッター（初めてコミットした開発者）をコミットログから特定することで集計する．

3.2.1　コメント人数に関する仮説

不具合に関する議論に参加した（コメントした）参加者が多いプロジェクトでは，多くの参加者に発言の機会が平等に与えられているプロジェクトであると考えられる．プロジェクトの初期段階ではオープンな文化があっても，長期にわたって運営されてきたプロジェクトでは，次第に固定したメンバーのみで議論がおこなわれるようになることも珍しくない [10]．コメント人数が少ないプロジェクトでは新規コミッターのニーズがあまり無い可能性が高く，コメント人数とコミッター昇格率の間には関係があると考えた．

3.2.2　参加期間に関するメトリクスの仮説

コミッターの平均参加期間が短いプロジェクトでは，古くから参加しているコミッターが少なく，流動性の高いプロジェクトであると考えられる．流動性の高いプロジェクトでは，古参のコミッターが相対的に少ない代わりに，新規のコミッターが多く昇格すると考えられるので，平均参加期間が短いプロジェクトは，コミッターの昇格率が高くなると考えた．

3.2.3　不具合修正に関するメトリクスの仮説

不具合修正率が低い，すなわち，未解決の不具合が多い場合には，不具合報告が多く既存コミッターには様々な負荷がかかっている状態と考えることができる．多数の不具合修正パッチの作成と検証を少数のコミッターでおこなう必要がある場合には，検証作業が終わるまでは不具合管理システムにおいて不具合修正が完了されたとは見なされない．したがって，不具合修正率が低い時期には，新規コミッターへのニーズが高い状態であると考えられるため，コミッター昇格率も高くなると考えた．

表 2　データソースとプロジェクトメトリクスの関係

データソース	メトリクス
コミットログ（バージョン管理システム）	コミッター昇格率および平均参加期間
不具合報告（不具合管理システム）	コメント人数，不具合修正率

表 3　データセット

対象プロジェクト	分析期間	不具合報告数	コミット数	コミッター数
Eclipse Platform	2002/10-2012/10	83,342	119,198	334
Mozilla Firefox	2007/04-2013/01	10,339	170,130	2,464
GCC	2004/01-2014/01	44,219	100,027	410

4. 分析

本章では，前章での仮説を検証するためにおこなった分析について述べる．

4.1 概要

本分析の目的は，3 種類のプロジェクトメトリクスとコミッター昇格率との関係を調べ，仮説が正しいかどうかをを確かめることである．本分析では，大規模 OSS プロジェクトである Eclipse Platform, Mozila Firefox, GCC プロジェクトを対象に，相関分析を用いてプロジェクトメトリクスとコミッター昇格率との関係を調査する．

4.2 データセット

本論文では，バージョン管理システムから抽出したコミットログと，不具合管理システムから抽出した不具合報告データを用いてデータセットを作成する．それぞれのシステムから計測できるプロジェクトメトリクスとコミッター昇格率との関係を表 2 に示す．また，分析で用いるデータセットの基本統計量を表 3 に示す．

4.3 分析手順

分析手順を以下に示す．

(1) **各プロジェクトに対して不具合データとコミットログを 3 ヶ月単位で区切る．** より小さな期間，例えば，1 ヶ月単位でデータを区切った場合，新規に昇格したコミッターが存在しない区間が出来てしまい，コミッター昇格率などのメトリクスを計算できない場合がある．そのため，本分析では 3 ヶ月単位でデータを区切ることとした．

(2) **3 ヶ月単位で区切ったコミットログからコミッター昇格率を求める．** 先に述べたように，コミッター昇格率は，区切られたデータ期間中に存在した既存コミッターと新規コミッターの比で求める．

(3) **全期間をコミッター昇格率の高い期間と低い期間に 2 分割する．** 本研究では，プロジェクトを取り巻く状況に依存してコミッター昇格率が変化するという基本的な仮説を前提としている．したがって，データセットの全期間の各プロジェクトメトリクスの値とコミッター昇格率の値を用いて相関分析をおこなうのは適切ではなく，コミッター昇格率が高い時期と低い時期それぞれにおいて各プロジェクトメトリクスと相関関係が存在するかどうかを確かめる必要がある．そこでコミッター昇格率の高い時期と低い時期を区別するために，全期間のデータからコミッター昇格率の中央値を用いる．中央値以上のコミッター昇格率示す期間を「コミッター昇格率の高い期間」，中央値以下の期間を「コミッター昇格率の低い期間」とする．

(4) **全期間，高い期間，低い期間の 3 つそれぞれについてスピアマンの順位相関を用いて相関係数を求める．** 本研究で用いるメトリクスの値は正規性を期待できないため，ノンパラメトリックな指標であるスピアマンの順位相関を用いる．

(5) **全期間，高い期間，低い期間の 3 つそれぞれについてスピアマンの順位相関に関する無相関検定をおこなう．** Step (5) で得られる相関係数が有意なものかどうかを検証するために，無相関検定をおこなう．

4.4 分析結果

表 4 に各プロジェクトに対して，スピアマンの順位相関係数を求めた結果を示す．

本論文では，相関係数の絶対値が 0.4 以上で，有意差がある (有意水準 5 %で) メトリクスについて相関関係があると判断する．

4.4.1 Eclipse Platform

Eclipse Platform プロジェクトにおけるコミッター昇格率の高い期間では，3 つのプロジェクトメトリクスすべてに対して，中程度からやや強い有意な相関がみられた．しかし，不具合修正率以外のプロジェクトメトリクス（コメント人数，参加期間）は，仮説とは逆の結果を示す相関が確認された．次章では，仮説と逆の結果を示した，メトリクスについて考察する．なお，全期間およびコミッター昇格率の低い期間では，3 つのプロジェクトメトリクスすべてに対してコミッター昇格率との有意相関は見られなかった．

表 4　分析結果

		Eclipse Platform		Mozilla Firefox		GCC	
		相関係数	p 値	相関係数	p 値	相関係数	p 値
全期間	コメント人数	-0.25	0.11	-0.14	0.51	0.31	0.04*
	平均参加期間	0.07	0.67	-0.19	0.36	-0.30	0.05
	不具合修正率	-0.02	0.90	0.18	0.39	0.13	0.39
高い期間	コメント人数	-0.59	0.00**	-0.03	0.90	-0.02	0.90
	平均参加期間	0.50	0.02*	-0.79	0.00**	-0.10	0.67
	不具合修正率	-0.71	0.00**	-0.75	0.00**	-0.01	0.93
低い期間	コメント人数	0.20	0.39	0.00	0.98	0.07	0.76
	平均参加期間	-0.21	0.38	0.40	0.22	-0.10	0.67
	不具合修正率	-0.17	0.45	0.59	0.06	-0.05	0.82

4.4.2　Mozilla Firefox

Mozilla Firefox プロジェクトにおけるコミッター昇格率の高い期間では，平均参加期間と不具合修正率の 2 つのプロジェクトメトリクスに対して，やや強い有意な相関が見られた．Eclipse Platform プロジェクト同様，全期間およびコミッター昇格率の低い期間では，3 つのプロジェクトメトリクスすべてに対してコミッター昇格率との有意な相関は見られなかった．

4.4.3　GCC (GNU Compiler Collection)

GCC では，分析をおこなった 3 つの期間 (全期間，コミッター昇格率の高い期間，コミッター昇格率の低い期間) において，本論文で提案した 3 つのプロジェクトメトリクスとコミッター昇格率の間に相関関係は見られなかった．次章では，GCC プロジェクトが他の 2 つのプロジェクト（Eclipse Platfrom プロジェクト，Mozilla Firefox プロジェクト）と異なった傾向を示した理由について考察する．

5.　考察

本章では，4 章での分析結果を踏まえ考察をおこなう．また，本論文の制約について述べる．

5.1　分析結果に関する考察

5.1.1　Eclipse Platrom

Eclipse Platform プロジェクトでは，3 つのプロジェクトメトリクス（「コメント人数」,「参加期間」,「不具合修正率」）全てで有意な差がみられたが，2 つのプロジェクトメトリクス（「コメント人数」,「参加期間」）で仮説と異なる結果が得られた．以下では，仮説と異なる結果に至ったメトリクスについて考察する．

仮説と異なる結果に至ったメトリクスを考察するために，図 2 に各プロジェクトにおけるコミッター昇格率を示す．

図 2 より，コミッター昇格率の特に高い期間である，2011 年 7 月から 2012 年 6 月までの 1 年間を調査した．調査の結果，Eclipse プロジェクトは 2004 年 6 月から 1 年周期で新たなバージョンをリリースしている．また，2011 年 6 月 22 日にバージョン 3.7 をリリースした後，2012 年 6 月 27 日にバージョン 4.2 をリリースしており，2011 年 6 月 22 日から 2012 年 6 月 27 日までの期間は，メジャーバージョンアップの準備期間と考えられる．つまり，2011 年 6 月 22 日から 2012 年 6 月 27 日までのコミッターの昇格は，メジャーバージョンアップが原因の 1 つと考えられる．

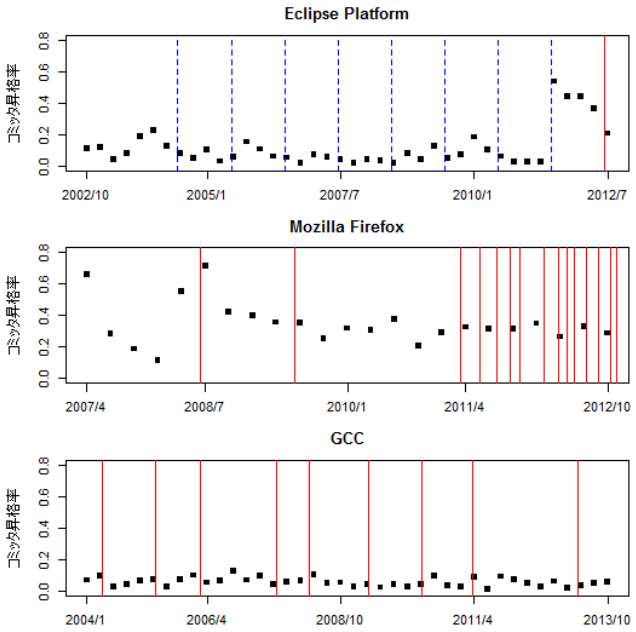

図 2　コミッター昇格率

点線はマイナーバージョンアップ，実線はメジャーバージョンアップを示す．

本論文では，バージョンが大きく変化した場合をメジャーバージョンアップとする．（例　バージョン 3.2 からバージョン 4.0 などをメジャーバージョンアップとする．）

メジャーバージョンアップがコミッター昇格に影響を与えた可能性を考慮し，コミッター昇格率が高い期間を対象に，2011 年 7 月から 2012 年 6 月までの期間を除き，追加分析をおこなった．表 5 に追加分析によって求めた，スピアマンの順位相関係数の結果を示す．

追加分析の結果を表 5 に示す．追加分析の結果，Eclipse Platform では 3 つのプロジェクトメトリクス全てで，有

表 5　追加分析の結果

コメント人数		平均参加期間		不具合修正率	
相関係数	p 値	相関係数	p 値	相関係数	p 値
0.22	0.39	0.02	0.94	-0.47	0.06

Eclipse Platform における 2011 年 7 月から 20126 月までを除いたコミッター昇格の高い期間.

意な差がみられなかった．つまり，Eclipse Platform プロジェクトではメジャーバージョンアップがコミッターの昇格に影響を与える場合があると考えられる．本論文の Eclipse Platform プロジェクトにおける分析期間は，表 3 に示すように，2002 年 10 月から 2012 年 10 月までの 10 年間である．この期間でのメジャーバージョンアップは，2012 年 6 月 27 日にリリースされたバージョン 4.2 のみである．今後は分析期間中に 1 度しか起きないようなイベント（Eclipse Platform におけるメジャーバージョンアップなど）に影響しないような分析を考える必要がある．

5.1.2　Mozilla Firefox

表 4 より，Mozilla Firefox プロジェクトは他の 2 つのプロジェクトと比べて，コミッターの人数が多い．これは古くから参加しているコミッターが少ないと考えられ，仮説の通り「平均参加期間」が短い期間では，コミッターが増加すると考えられる．また「不具合修正率」においてもコミッター昇格率の高い期間で相関がみられ，不具合完了率が減少するとコミッターが増加すると考えられる．

Mozilla Firefox プロジェクトでは，Eclipse Platform プロジェクトと比べて，メジャーバージョンアップが頻繁におこなわれている．（2006 年 10 月，2008 年 6 月，2009 年 6 月，2011 年 6 月，2011 年 8 月，以降約 1 ヶ月単位で）

また，Eclipse Platform と違いバージョンアップに期限が決められているわけではない．（Eclipse Platform は 1 年に 1 回バージョンアップする．）よってバージョンアップによるコミッター昇格への影響はないと考えられる．

5.1.3　GCC (GNU Compiler Collection)

GCC プロジェクトでは，分析をおこなった 3 つの期間 (全期間，コミッター昇格率の高い期間，コミッター昇格率の低い期間) で本論文で述べたプロジェクトメトリクスとの相関は見られなかった．GCC プロジェクトにおいても定期的にメジャーバージョンアップがおこなわれており，メジャーバージョンアップによるコミッター昇格への影響はないと考えられる．

また，図 2 より GCC プロジェクトではコミッター昇格率が Eclipse Platform,Mozilla Firefox プロジェクトと比べ比較的一定であり，特定の期間にコミッターを増やすのではなく，定期的に一般開発者がコミッターに昇格していることが考えられる．つまり，プロジェクトの状態がコミッターの昇格に影響を与えないと考えられる．

5.2　今後のシナリオ

伊原らは開発者の活動量（例えば，パッチ投稿数やコメント数など）を用いてコミッター候補者予測モデルを構築している．[16] 今回得られた知見は，開発者の活動量ではなく，プロジェクトに関するメトリクスであり，伊原らが提案したメトリクスに加えることで，コミッター候補者予測モデルに精度を向上できると考えられる．得られた知見と今後の活用方法について以下に示す．Eclipse Platform プロジェクトはメジャーバージョンアップが大きく影響している可能性があり，今回提案した 3 つのプロジェクトメトリクスはコミッター候補者予測モデルに組み込むべきではない．Mozilla Firefox プロジェクトは，流動性も高く，「平均参加期間」,「不具合修正率」をコミッター候補者予測モデルに組み込むことで精度が向上すると考えられる．GCC プロジェクトは，コミッターの昇格が比較的一定であり，プロジェクトが取り巻く状況がコミッターの昇格に影響を与えない場合があり，コミッター候補者予測モデルに組み込むべきではない．

5.3　本論文の制約

本論文では，不具合管理システムとして Bugzilla，構成管理ツールとして Subversion を利用しているプロジェクトのみを対象とした．近年では分散型リポジトリ GIT を利用しているプロジェクトが増加している．分散型リポジトリの特徴の一つは，全ての開発者がそれぞれローカルの計算機でリポジトリを構成しており，誰でもコミットをおこなうことができる点である．ただし，最終的にマージをおこなうことができるのは権限をもった一部の開発者のみである．集中型リポジトリと分散型リポジトリでの開発者の役割を単純に比較することは不可能であるため，本研究では分析対象外とし今後の課題とする．また一部のプロジェクトでは，ボランティアの開発者だけではなく，企業の開発者も参加している場合があり，企業の開発者は優先的にコミッターとして選出されている可能性が考えられる．また，各プロジェクトにおける分析期間のはじめから既にコミッターとして活動を行っている開発者も多く存在しており，分析結果に影響している可能性がある．

6.　おわりに

本論文では，OSS プロジェクトにおけるコミッター昇格とプロジェクトメトリクスとの間に関係があるかを分析した．Eclipse Platfrom,Mozilla Firefox,GCC を対象にケー

ススタディをおこなった結果，以下のような知見が得られた．

- Eclipse Platform プロジェクトでは，メジャーバージョンアップする時にコミッターの昇格が多く，メジャーバージョンアップが大きく影響している場合がある．
- Mozilla Firefox プロジェクトでは，平均参加期間が短い時期や，不具合修正率が低い時期にコミッター昇格が起こりやすい傾向があることがわかった．
- GCC プロジェクトでは，コミッター昇格が定期的におこなわれており，プロジェクトを取り巻く状況がコミッターの昇格に影響を与えない場合があることがわかった．

本研究の分析を通して，コミッター昇格率に影響を与えるプロジェクトメトリクスが確認された．伊原らは開発者の活動量（例えば，パッチ投稿数やコメント数など）を用いてコミッター候補者予測モデルを構築している．[16] 今後は伊原らのコミッター候補者予測モデルに今回得られた知見を組み合わせることで予測モデルの精度を向上させることができると考えられ，プロジェクト管理者がコミッター選出を適切に出来るようになると考えられる．

謝辞 本研究の一部は，文部科学省科学研究補助金（基盤 (C): 24500041）による助成を受けた．

参考文献

[1] : The Free/Libre/Open Source Software Survey for 2003, http://www.stanford.edu/group/floss-us/.

[2] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR '07)*, p. 6 (2007).

[3] Fogel, K.: *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media (2005).

[4] 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).

[5] Jeong, G., Kim, S. and Zimmermann, T.: Improving Bug Triage with Bug Tossing Graphs, *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE '09)*, pp. 111–120 (2009).

[6] Lave, J. and Wenger, E.: *Situated Learning: Legitimate Periperal Participation*, Cambridge University Press, Cambridge, UK (1991).

[7] Mockus, A., Fielding, R. T. and Herbsleb, J. D.: Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Trans. Softw. Eng. Methodol.*, Vol. 11, No. 3, pp. 309–346 (2002).

[8] Ohira, M., Osawa, N., Hassan, A. and Matsumoto, K.: The impact of bug management patterns on bug fixing: A case study of Eclipse projects, *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM)*, ICSM '12, Washington, DC, USA, IEEE Computer Society, pp. 264–273 (online), DOI: 10.1109/ICSM.2012.6405281 (2012).

[9] Raymond, E. S.: *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly and Associates (1999).

[10] Shihab, E., Bettenburg, N., Adams, B. and Hassan, A. E.: On the Central Role of Mailing Lists in Open Source Projects: An Exploratory Study, *Proceedings of The 3rd International Workshop on Knowledge Collaboration in Software Development (KCSD2009)*, pp. 91–103 (2009).

[11] Ye, Y. and Kishida, K.: Toward an Understanding of the Motivation Open Source Software Developers, *In Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pp. 419–429 (2003).

[12] Zhou, M. and Mockus, A.: Developer Fluency: Achieving True Mastery in Software Projects, *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 137–146 (2010).

[13] Zhou, M. and Mockus, A.: Does the Initial Environment Impact the Future of Developers?, *Proceedings of the 33rd International Conference on Software Engineering*, pp. 271–280 (2011).

[14] Zhou, M. and Mockus, A.: What Make Long Term Contributors: Willingness and Opportunity in OSS Community, *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*, pp. 518–528 (2012).

[15] 伊原彰紀，藤田将司，大平雅雄，松本健一：OSS 開発におけるコミッター選出のための開発者の活動量に関する実証的分析，ソフトウェア工学の基礎 XVIII, 日本ソフトウェア科学会 FOSE2011，pp. 81–90 (2011).

[16] 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. J95-D, No. 2, pp. 237–249 (2012).