論　文

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

伊原　彰紀[†a)]　亀井　靖高[††]　大平　雅雄[†]　松本　健一[†]
鵜林　尚靖[††]

A Candidate Committer Prediction Based on Developer Activities in Open Source Software Projects

Akinori IHARA[†a)], Yasutaka KAMEI[††], Masao OHIRA[†], Ken-ichi MATSUMOTO[†], and Naoyasu UBAYASHI[††]

あらまし　本論文では，オープンソースソフトウェア（OSS）プロジェクトに参加する一般開発者の中からコミッターに推薦されるべき有能な開発者（コミッター候補者）を見つけ出すことを目的とする．近年，数百件もの不具合がプロジェクトに日々，報告されている現状から，コミッターへの過度な負担が原因となり不具合修正の長期化を招いている．コミッターの負担を軽減させるためにコミッターを増員するという手段があるが，プロジェクトに参加する一般開発者の中からコミッター候補者を見つけることは容易ではない．本論文では，コミッター候補者を見つけ出すために既存コミッターの過去の活動とその活動量を分析し，コミッター予測モデルを構築した．モデル構築には，コミッター候補者と一般開発者の活動（パッチの投稿，パッチの検証，開発に伴う議論）履歴とプロジェクトでの活動期間を用いた．分析の結果，継続的にパッチの投稿，パッチの検証を行う開発者がコミッターに昇格していることが分かった．また，構築したコミッター予測モデルは，ランダムに予測する場合に比べて予測精度が5〜7倍高いことが分かった．

キーワード　オープンソースソフトウェア開発，コミッター予測，開発者活動量，Eclipseプロジェクト，Mozillaプロジェクト

1. まえがき

オープンソースソフトウェア（OSS）プロジェクトでは，高い技術力や開発者を取りまとめる能力をもっていると認められた開発者（コミッター）にのみ，プロダクトに加えた変更をバージョン管理システムにコミットするための権限（コミット権限）が与えられる[1]．多くのプロジェクトが採用しているこのコミット権利付与方針は，不具合を含むコードがプロダクトに混入する機会を抑制し，たとえ不具合が混入した場合でもその原因の所在を突き止めやすくするという品質保証上のメリットがある．その一方，開発者から投稿される不具合修正パッチの検証作業を，選ばれた少数のコミッターが中心となって負担するという作業負担上のデメリットを生む場合がある．OSSの社会的普及により，プロジェクトに報告される不具合の件数は年々増加しているため，特に大規模プロジェクトではコミッターにかかる負担は極めて大きい[2], [3]．一日に数百件の不具合報告が寄せられるEclipseプロジェクトやMozillaプロジェクトでは，コミッターへの過度な負担が原因となり不具合修正の長期化を招いていることが報告されている[4]〜[6]．

コミット権限付与方針のメリットを維持しつつコミッターの負担を軽減させるためには，コミッターを増員する方法が効果的である．しかし，プロジェクトに参加する一般開発者（大規模プロジェクトでは1万人を超えることも珍しくない）の中からコミッターに

† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, 8916–5 Takayama, Ikoma-shi, 630–0192 Japan
†† 九州大学大学院システム情報科学研究院，福岡市
Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka-shi, 819–0395 Japan
a) E-mail: akinori-i@is.naist.jp

昇格する有能な開発者を見つけることは実際には容易ではない．一般的に，コミッター候補者は，既存コミッターからの推薦と承認を経て昇格する [7]．その際，既存コミッターは，プロジェクトに対するコミッター候補者の過去の活動内容，具体的には機能拡張や不具合修正に関連する活動を総合的に評価している．活動実績評価の必要性から，一般開発者がコミッター候補者として推薦されるには，通常 1 年以上の継続的な活動が求められる．そのため，有能な一般開発者であっても途中で意欲を失いコミッターに推薦される前にプロジェクトを去ってしまうことが少なくない [8]．

この問題を解決するためには，コミッター権限を付与するに足る開発者をできるだけ早い段階で見つけ出しコミッター候補者として推薦する必要がある．本論文では，コミッターに推薦されるべき有能な開発者を見つけ出すために，既存コミッターの過去の活動とその活動量を分析し，コミッター予測モデルを構築する．本論文においてコミッター候補者は実験対象とする期間内にコミッターに昇格した者とし，それ以外の開発者は一般開発者とする．そして，モデル構築には，コミッター候補者と一般開発者の活動（パッチの投稿，パッチの検証，開発に伴う議論）の履歴とプロジェクトでの活動期間を用いる．

モデル構築にあたり本論文では，プロジェクトが長期間にわたって運営され，多くのコミッターが参加している大規模 OSS（Eclipse platform，Mozilla Firefox）プロジェクトを対象に，以下の Research Question に取り組む．

RQ1：コミッター候補者を見つけ出すために有用な活動量は何か？

RQ1 では，コミッター候補者と一般開発者の活動量（パッチ投稿回数，パッチ検証回数，コメント回数），活動期間の違いを分析し，コミッターの予測精度向上に寄与すると考えられる活動を把握する．コミッター候補者と一般開発者の活動量に大きな差がある活動はコミッター候補者を見つけ出すために有用と考えられる．

RQ2：開発者の活動量からコミッター候補者をどの程度の精度で予測できるのか？

RQ2 では，開発者の活動量を用いてコミッター予測モデルを構築し，モデルを評価する．

本論文で実施した実験によって得られる貢献は以下のとおりである．

- コミッター候補者を見つけ出すために有用な活動を明らかにした．
- コミッター候補者を見つけ出すことを容易にした．

以降 **2.** では，関連研究について述べ本論文の立場を明らかにする．**3.** では，パッチがプロダクトに反映されるまでの開発者の活動について詳述し，**4.** では，本論文の目的を達成するための Research Question と，実験手順について述べる．**5.** では，実験に用いるデータセットと，各 Research Question の実験結果について述べる．**6.** で本論文の考察を行い，最後に **7.** で本論文のまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 コミッターの活動支援

OSS が多機能化するにつれて，多くのボランティア開発者が OSS 開発に参加するようになった．その結果，OSS プロジェクトはボランティア開発者から数多くの機能拡張要求や不具合報告などを得られるようになり，より高品質な OSS を実現できるようになった．その一方で，開発者へ修正依頼，パッチ品質の検証，構成管理，ユーザサポート，ドキュメント作成など，コミッターの負担は年々増加している [2]．このような背景から近年，コミッターの活動支援を目的とした研究が数多く行われている．

その一つに，コミッターが関与している機能拡張や不具合修正などの作業進捗を管理・把握することを支援する研究がある．コミッターは通常，多数の機能拡張や不具合修正を担当するため，自身のタスクに関連する他の開発者の作業進捗状況の全てを把握することが難しい．この問題を解決するために，自身のタスク全体の作業進捗や，連携する他の開発者の作業進捗を管理・把握支援するための可視化ツールが提案されている [9]～[11]．

その他にも，機能拡張や不具合修正を担当する開発者の決定や，報告された不具合の内容把握など，コミッターが行うべき活動の一部を自動化する研究がある [12]～[14]．膨大な数の機能拡張要求や不具合報告の内容を一つずつ理解し，かつ，プロジェクトに参加している多数のボランティア開発者の中から各タスクに適任の担当者を割り当てるためには相当な労力が求められる．そこで，開発者が過去に担当した不具合に関するテキスト情報（不具合票に記載されたタイトルや詳細など）をもとにモデルを構築し適当な修正担当者を予測する方法 [12], [13] や，不具合票に記載される

テキスト情報から要約文章を自動作成し，不具合を容易に理解することを支援するシステム [14] が提案されている．

本研究はコミッターの活動を支援するという点で，先行研究と問題意識を共有しているといえる．ただし，先行研究がコミッターという少数のリソースを効率的に活用するアプローチをとるのに対して，本研究は潜在的に高い能力を有するボランティア開発者の推薦によりコミッターのリソース拡大（コミッター 1 人当りの負担を減らす）を目指しており，先行研究とはコミッターの活動支援に対する立場が異なる．

今後も OSS は，大規模化･複雑化が予想されており，積極的に活動するコミッターの増加が不可欠になってくると考えられる．そのため，本論文の提案は重要になってくると考えられる．

2.2 コミッター昇格までのプロセス

既存コミッターがコミッター候補者の能力を理解するために参考にする開発者の活動量はプロジェクトによって異なる．

Jensen ら [7] は，OSS 開発者にインタビューを行い，既存コミッターがコミッター候補者をどのように決定しているのか調査している．Apache プロジェクトでは，開発者の技術的な活動（パッチの投稿など）を参考に，プロジェクトに多く貢献している開発者をコミッター候補者として，既存コミッターがプロジェクト管理委員（PMC Member）に推薦する．その後，PMC Member がコミッター候補者の貢献を認めるとコミッターに昇格することができる．Mozilla プロジェクトも同様に，既存コミッターは開発者が行ってきた技術的な活動に加えて，開発者が行う社会的な活動（開発の指示など）を参考に，コミッター候補者を見つけている．

Bird ら [8] は PostgreSQL プロジェクトにおける開発者の活動期間の分析を通して，約 1 年間の活動実績がある一般開発者はコミッターに昇格する開発者として適切であると結論づけている．開発者の活動期間が 1 年よりも短い場合，コミッターとしてふさわしいかどうかを判断するのは難しく，また，1 年より長い場合，開発者はモチベーションを失ってしまう危険性が高いことを挙げている．各プロジェクトで新たなコミッターを見つけ出すために，開発者の活動量や活動期間を参考にしているが，どの程度の活動を行っている開発者をコミッターとして推薦するのかを示す目安は提示されていない．

現状では，プロジェクト管理者やコミッターは，客観的なデータではなく彼らの経験に基づいて開発者の活動量と活動期間を評価し，コミッター候補者を決定せざるを得ない．そこで，藤田ら [15] は，パッチの投稿・検証回数とパッチの編集・検証時間に関して，コミッター候補者と一般開発者の違いを分析している．その結果，パッチの投稿回数とパッチの検証回数については，コミッター候補者と一般開発者で違いがあることが分かった．しかし，コミッター候補者はパッチの投稿とパッチの検証，共に活躍しているとは限らず，優先的に確認すべき活動については文献 [15] では示されていない．また，これらの活動量を用いてどの程度コミッター候補者を見つけ出すことができるかについても定かではない．本論文では，既存コミッターがコミッター候補者を見つけ出すために参考にする活動の中で，最も注目すべき活動量を提示し，コミッター候補者をどの程度予測することができるかを分析する．

3. 開発者の活動

本論文で実験対象とする Eclipse プロジェクト，Mozilla プロジェクトでは，コミッターへの昇格に関するガイドラインが定められている(注1)．ガイドラインには，プロジェクトに対して貢献が認められた場合にコミット権限を付与すると記載されているが，貢献と認められる活動内容や活動量についての具体的な記述はない．本論文では，既存コミッターがコミッター候補者を推薦する際に参考にしている開発者の技術的・社会的活動（パッチ投稿・検証，開発に伴う議論など）の履歴を用いて，コミッター候補者の活動量と一般開発者の活動量を比較する．開発者が機能拡張や不具合修正を行うプロセスを図 1 に示し，各活動の

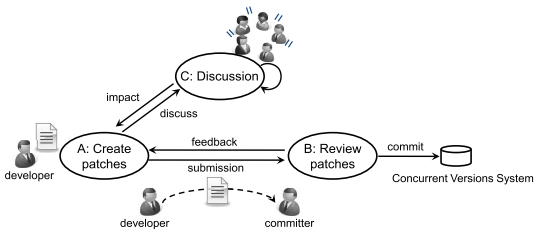


図 1 パッチがプロダクトに反映されるまでのプロセス
Fig. 1 Process for applying patches in the product.

(注1)：Eclipse: http://wiki.eclipse.org/Development_Resources/HOWTO/Nominating_and_Electing_a_New_Committer
Mozilla: http://www.mozilla.org/hacking/committer/

内容について説明する．

A：パッチの投稿：開発者は機能拡張や不具合修正のためにパッチを作成し，プロジェクトに投稿する．パッチは全ての開発者が投稿することができる．

B：パッチの検証：投稿されたパッチに関係する開発者（変更の適用対象となるモジュールの開発者など）がパッチの品質を検証する．そのとき，パッチを検証する開発者の中にバージョン管理システムにコミットする権限をもつコミッターが含まれている．修正内容が不十分であった場合，検証を行った開発者はパッチ投稿者に再修正を依頼する．修正内容が適切であった場合，検証を行ったコミッターがパッチをプロダクトに反映する．パッチの検証は，全ての開発者が行うことが可能である．しかし，パッチをプロダクトに反映できるのはコミッターのみであり，コミッターがパッチを承認しない限りパッチはプロダクトに反映されない．

C：開発に伴う議論：パッチの投稿やパッチの検証と並行して，開発者間で議論を行う．議論の内容は，機能拡張や不具合修正の方針に関する相談や報告，類似不具合に関する情報提供，再修正依頼などが挙げられる．

本論文では，これらの活動についてコミッター候補者の活動量と一般開発者の活動量を比較し，コミッター候補者を予測する．

4. Research Question

本章では，OSS 開発者の活動量を用いてコミッター候補者を予測するために取り組む Research Question と，各 Research Question の実験手順について述べる．

4.1 RQ1：コミッター候補者を見つけ出すために有用な活動量は何か？

動機：OSS の機能拡張や不具合修正のために，開発者らは図 1 に示すようなパッチ投稿，パッチ検証，開発に伴う議論を繰り返す．既存コミッターは，このような過去の活動内容を参照し，積極的に活動している開発者をコミッター候補者に推薦する．それゆえ，コミッター候補者と一般開発者では活動量に違いがあると考えられる．コミッター候補者と一般開発者の活動量に違いが認められれば，1 万人を超える開発者の中からコミッター候補者を容易に見つけ出すことができると考えられる．RQ1 では，コミッター候補者の活動量と一般開発者の活動量を比較し，コミッター候補者を予測するためにはどの活動が有用と考えられるかを分析する．

手順：RQ1 では，(1) コミッター候補者と一般開発者の活動量をそれぞれ抽出し，(2) コミッター候補者と一般開発者の活動量に差があるか否かを分析する．その後，(3) コミッター候補者と一般開発者の活動量の差がどの程度であるのかを分析し，他の活動と比べて，実質的に活動量の差が大きい活動を見つける．以下で各分析の詳細を説明する．

(1) コミッター候補者と一般開発者の活動量を抽出

本論文で収集する開発者の活動量を表 1 に示す．開発者はパッチ投稿，パッチ検証，開発に伴う議論を年中行っているとは限らない．本論文では，一時期のみ積極的に活動した開発者と継続して活動している開発者を区別するために，本論文では各活動の総回数と各活動の月回数（中央値）を導出する．また，コミッター候補者の活動期間は，パッチ投稿，パッチ検証，開発に伴う議論のいずれかの活動を初めて行った日からコミッターに昇格するまでの期間とする．一般開発者の活動期間は，コミッター候補者と同様に上記のいずれかの活動を行った日から，それらの活動を最後に行った日までの期間とする．

(2) コミッター候補者と一般開発者の活動量に統計的有意差があるかを検定

表 1 に示す活動量に関して，コミッター候補者の活動量と一般開発者の活動量の分布を比較する．分布の比較にはウィルコクソンの順位和検定を用い，有意水準 5%で検定を行う．

(3) コミッター候補者と一般開発者の活動量の差がどの程度であるのかを分析

手順 (2) で行う検定では，活動間の比較は行えないため，コミッター候補者と一般開発者の活動量の差が実質的に大きい活動を効果量（Effect size）を用いて分析する．効果量は，二つのメトリックスの平均値の差を標準化することにより，単位の異なるメトリックス間で，効果の大きさを比較することができる指標である．効果量 (d) は以下の式で求めることができる [16], [17].

$$d = \frac{\tilde{x}_a - \tilde{x}_b}{\sqrt{S_a^2 + S_b^2}}$$

ここで，用いる変数 $\tilde{x}_a$，$\tilde{x}_b$，S_a，S_b はそれぞれ任意の活動における活動量を示しており，$\tilde{x}_a$ はコミッター候補者の活動量の平均値，$\tilde{x}_b$ は一般開発者の活動量の平均値，S_a はコミッター候補者の活動量の分散，S_b

表 1　開発者の活動
Table 1　Developer activities.

活動	メトリックス名	内容
パッチ投稿	総パッチ投稿数	各開発者が全分析期間にパッチ形式のファイルを投稿した総回数.
	月パッチ投稿数	各開発者が活動期間中で 1 か月にパッチを投稿した回数の中央値.
パッチ検証	総パッチ検証数	パッチが投稿された直後に投稿された他の開発者からのコメントをレビューと定義したときの，各開発者ごとの全分析期間におけるレビュー数の総数.
	月パッチ検証数	各開発者が活動期間で 1 か月に検証したパッチ数の中央値.
開発に伴う議論	総コメント投稿数	パッチを含まない不具合票でのメッセージをコメントと定義し，各開発者ごとの全分析期間におけるコメント総数.
	月コメント投稿数	各開発者が活動期間で 1 か月に投稿したコメント数の中央値.
活動期間	活動期間	各開発者の全分析期間において，パッチ投稿，パッチ検証，不具合修正に関する議論のいずれかの活動が実施された月数.

は一般開発者の活動量の分散を示す．効果量の値が大きいほどコミッター候補者と一般開発者の活動量の差が実質的に大きいことを意味する．

4.2　RQ2：開発者の活動量からコミッター候補者をどの程度の精度で予測できるのか？

動機：既存コミッターはプロジェクトに参加する 1 万人を超える開発者の中からコミッター候補者を見つけ出し，推薦しなければならない．そこで，RQ2 では，コミッター候補者を容易に見つけ出すために，表 1 に示す開発者の活動量を用いて，開発者の中からどの程度の精度でコミッター候補者を予測することができるか評価実験を行う．

手順：表 1 に示す全ての活動量を用いてコミッター予測モデルを構築する．本論文では，多くの研究で利用されているロジスティック回帰分析 [18]〜[20] を用いてコミッターの予測モデルを構築する．本論文では，コミッターに昇格するか否かを目的変数とし，予測モデルの出力値（0 から 1 の連続値）が 0.5 以上のときにコミッターに昇格すると判別する．そして，モデルの評価は適合率（Precision），再現率（Recall），F1 値 [21] と正確に予測できたコミッター候補者数を用いる．適合率は，コミッターに昇格すると判断された開発者のうち，実際にコミッターに昇格した開発者の割合を示す．また再現率は，全コミッターの中で，コミッターに昇格すると判別されたコミッターの割合を示す．ただし，適合率と再現率はトレードオフの関係にあるため，両方の評価指標が高いとき，性能の高い予測モデルが構築できたことになる．そこで本論文では，適合率と再現率に加えて，適合率と再現率の調和平均で与えられる F1 値を評価指標の一つとして用いる．F1 値は以下の式で定義され，値が大きいほどモデルの判別精度が高いことを示す．

表 2　実験対象データの概要
Table 2　Summary of the target data.

	Eclipse platform	Mozilla Firefox
実験対象期間	2001/10-2010/12	2004/01-2008/12
全コミッター人数	89	147
コミッター候補者人数	55	51
一般開発者人数	8964	12287

$$F1\text{−}measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

また，本論文では既存コミッターがコミッター予測モデルを使用する場合，本モデルによって予測されるコミッター候補者の中から，実際に昇格させる者を検討するものと想定している．そのため，数多くのコミッター候補者を正確に予測することが予測モデルにとって重要であると考える．そこで，適合率，再現率，F1 値に加え，正確に予測できたコミッター候補者数を評価指標の一つとして用いる．

5.　実　　験

本章では，実際の OSS プロジェクトに参加する開発者の活動量を用いて，コミッター候補者を見つけ出すのに有用な開発者の活動量を分析する．また，開発者がコミッターに昇格するまでの活動量に基いてコミッターの予測モデルを構築し，モデルの評価実験を行う．

5.1　対象データ

本論文では，コミッター候補者と一般開発者の活動がコミッター候補者を見つけ出すために有効であるかどうかを確認するために，大規模な OSS プロジェクトである Eclipse platform プロジェクト，Mozilla Firefox プロジェクトを対象としてケーススタディを行う．本論文で対象とする分析期間，また各プロジェクトにおける開発者数を表 2 に示す．実験対象期間の終了時点までに一度でもバージョン管理システムに

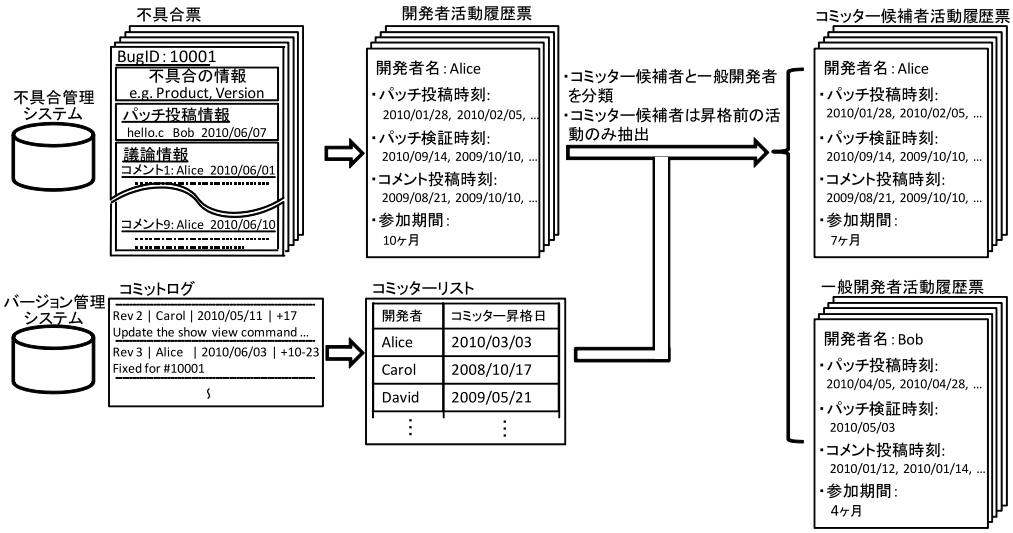


図 2　データ抽出方法
Fig. 2　Method to extract the developer activities.

コミットを行った開発者をコミッターと呼び，そのうち活動を開始してから初めてコミットを行うまでをコミッター候補者と呼ぶ．そして，実験対象期間終了時までにコミッターに昇格していない開発者は（たとえ実験対象期間後にコミッターに昇格している場合でも），一般開発者と呼ぶ．一方で，開発者の中にはプロジェクトに携わり始めたときからコミッターである場合もある．本論文では，実験対象期間以前からコミッターである開発者については，実験対象期間中の活動がその開発者のコミッター昇格に直接関係していないと考えられるため除外している．Eclipse platform プロジェクトでは，89 人のコミッターを確認したが，34 人はパッチの投稿，パッチの検証，開発に伴う議論を行う前にバージョン管理システムにコミットしていたため，プロジェクトに参加した時点で既にコミット権限をもっていたと考えられる．そのため，本論文では残りの 55 人をコミッター候補者として実験を行う．

5.2　実験データの収集と整形

本論文では，パッチ投稿，パッチ検証，開発に伴う議論に関する情報を収集するために不具合管理システム(注2)のデータを用いる．また，コミッター候補者と一般開発者を区別するためにバージョン管理システムのコミットログデータを用いる．不具合管理システム，バージョン管理システムを用いたデータの抽出手順を，図 2 に沿って説明する．

まず，各開発者の活動量を不具合管理システムから抽出する．各 OSS プロジェクトでは，不具合情報や機能拡張を希望する情報などを不具合管理システムにおいて不具合票に記録している．各不具合票には，変更対象のモジュール名，修正を行ったパッチ，修正に伴う議論等が記録されている．本論文では，不具合管理システムから不具合票を収集し，その後，不具合票からパッチ投稿者，パッチ投稿時刻，パッチ検証者，パッチ検証時刻，コメント投稿者，コメント投稿時刻，プロジェクトへの活動期間を抽出する．抽出方法は開発者の活動履歴を用いた従来研究と同様の方法 [15], [22], [23] を用いており，抽出の自動化が可能である．パッチの検証はパッチ投稿直後に投稿されたコメントを当該パッチに対する検証とする．パッチ投稿，パッチ検証，コメント投稿のいずれかの活動が初めて実施された月をプロジェクト参加開始月とし，いずれかの活動が最後に実施された月をプロジェクト参加最終月とする．活動期間はプロジェクト参加開始月からプロジェクト参加最終月までの期間とする．以上の記録を開発者ごとに開発者活動履歴票にまとめる．

次に，開発者がコミッターに昇格した日時をバージョン管理システムから抽出する．バージョン管理システムでは，ソースコードをコミットした開発者名とコミット時刻がソースコード別にコミットログとして記録されている．本論文では，バージョン管理システ

(注2)：Eclippse Bugzilla: https://bugs.eclipse.org/bugs/
Mozilla Bugzilla: https://bugzilla.mozilla.org/

ムからコミットログを収集し，その後，コミットログにコミット経験のある開発者名，各開発者が初めてコミットした時刻を抽出する．

最後にコミッターリストを用いて開発者活動履歴票がコミッターのものか否かを分類する．本論文では，開発者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昇格日時と定義し，コミッターはコミッター昇格日時までの活動のみ抽出する．

コミッター候補者活動履歴票と一般開発者活動履歴票はそれぞれ 2 等分し，一方は RQ1 の分析，及び RQ2 のモデルを作るために用い，もう一方は RQ2 のモデルを評価するために用いる．

5.3 結　　果

5.3.1 RQ1：コミッター候補者を見つけ出すために有用な活動量は何か？

Eclipse platform におけるコミッター候補者と一般開発者の各活動の統計量（中央値，平均値，分散，p 値，効果量）を表 3 に，Mozilla Firefox プロジェクトにおけるコミッター候補者と一般開発者の各活動の統計量を表 4 に示す．Eclipse platform プロジェクトでは，表 1 に示す全ての活動においてコミッター候補者と一般開発者の活動量に統計的有意差があることが分かった．Mozilla Firefox プロジェクトでは，月コメント投稿数を除いて全ての活動量で統計的有意差が見られた．よって，ほとんどの活動は，コミッターに昇格する開発者を見つける際に有用であると考えられる．

次に，Eclipse platform プロジェクトにおける開発者の各活動について効果量の分析を行った結果，全ての活動の中で，コミッター候補者と一般開発者の月パッチ検証数の差が実質的に最も大きいことが分かった．次いで，月パッチ投稿数，月コメント投稿数，総パッチ検証数，総パッチ投稿数，活動期間，総コメント投稿数の順であった．Mozilla Firefox プロジェクトにおいても，コミッター候補者と一般開発者の月パッチ検証数の差が実質的に最も大きいことが分かった．次いで，月パッチ投稿数，活動期間，月コメント投稿数，総パッチ投稿数，総パッチ検証数，総コメント投稿数の順であった．パッチ検証，パッチ投稿，開発に伴う議論のそれぞれのひと月当りの活動回数は同じ活動の合計回数よりも効果量が大きかった．このことから，コミッターに昇格する開発者は一時的に多くの活動量があったのではなく，継続的に活動していたことが分かった．

表 3　開発者の活動の統計量（Eclipse platform）
Table 3　Statistics of developer activities in Eclipse platform project.

		中央値	平均値	分散	p 値	効果量
総パッチ投稿数	コミッター(注3)	11.50	31.39	2634.77	0.00	0.01
	一般開発者	0.00	0.69	66.63		
月パッチ投稿数	コミッター	1.75	4.29	38.58	0.00	0.11
	一般開発者	0.00	0.09	0.34		
総パッチ検証数	コミッター	0.00	2.39	22.25	0.00	0.06
	一般開発者	0.00	0.32	25.83		
月パッチ検証数	コミッター	0.00	0.57	0.72	0.00	0.72
	一般開発者	0.00	0.05	0.06		
総コメント投稿数	コミッター	42.50	77.54	17547.67	0.00	0.00
	一般開発者	2.00	15.03	48197.35		
月コメント投稿数	コミッター	3.00	8.96	228.48	0.00	0.03
	一般開発者	1.00	1.42	6.79		
活動期間（月）	コミッター	6.47	13.94	197.59	0.00	0.01
	一般開発者	1.03	9.51	295.55		

表 4　開発者の活動の統計量（Mozilla Firefox）
Table 4　Statistics of developer activities in Mozilla Firefox project.

		中央値	平均値	分散	p 値	効果量
総パッチ投稿数	コミッター(注3)	1.00	38.83	16719.80	0.00	0.00
	一般開発者	0.00	0.45	44.01		
月パッチ投稿数	コミッター	0.00	1.33	6.04	0.00	0.21
	一般開発者	0.00	0.05	0.23		
総パッチ検証数	コミッター	1.50	31.08	13438.86	0.00	0.00
	一般開発者	0.00	0.27	12.29		
月パッチ検証数	コミッター	0.00	0.75	2.80	0.00	0.25
	一般開発者	0.00	0.04	0.05		
総コメント投稿数	コミッター	17.00	248.67	292007.70	0.00	0.00
	一般開発者	2.00	7.79	8854.90		
月コメント投稿数	コミッター	1.00	8.10	365.67	0.30	0.02
	一般開発者	1.00	1.33	5.10		
活動期間（月）	コミッター	24.12	30.66	584.30	0.00	0.04
	一般開発者	1.00	6.23	157.78		

（注 3）表 3，表 4，表 9，表 10 で，コミッターはコミッター候補者を意味する．

5.3.2 RQ2：開発者の活動量からコミッター候補者をどの程度の精度で予測できるのか？

本論文で（全メトリックスを用いて）構築したモデルにおける適合率，再現率，F1 値，正確に予測できたコミッター候補者数を表 5，表 6 に示す．また，本論文のモデルの有意性を示すために，表 1 で提示する各メトリックスを用いて構築したモデルで予測した場合，及び，ランダムに予測した場合の結果を示す．そして，本論文のモデルを用いて予測する場合と，ランダムに予測した場合と比べて，どの程度精度が向上したかを併記する．具体的なモデルの評価は，適合率や F1 値がランダムで予測した場合より高いモデルの中で，再現率が高い（かつ，正確に予測できたコミッター候補者数が多い）モデルを有意性の高いモデルとして判断

表 5　コミッター候補者の予測結果（Eclipse platform）
Table 5　Committer prediction results in Eclipse platform project.

		適合率	再現率	F1 値	正確に予測できたコミッター候補者数
本実験モデル（全メトリックス）を用いた予測		0.07	0.70	0.13	19
単一メトリックスを用いた予測	総コメント投稿数	0.06	0.70	0.11	19
	月コメント投稿数	0.01	0.33	0.03	9
	総パッチ投稿数	0.05	0.67	0.10	18
	月パッチ投稿数	0.05	0.48	0.09	13
	総パッチ検証数	0.06	0.56	0.10	15
	月パッチ検証数	0.05	0.41	0.10	11
	活動期間	0.00	0.33	0.00	9
ランダム予測		0.01	0.50	0.01	14
向上率(注4)		7.00	1.40	13.00	1.36

表 6　コミッター候補者の予測結果（Mozilla Firefox）
Table 6　Committer prediction results in Mozilla Firefox project.

		適合率	再現率	F1 値	正確に予測できたコミッター候補者数
本実験モデル（全メトリックス）を用いた予測		0.02	0.83	0.05	20
単一メトリックスを用いた予測	総コメント投稿数	0.05	0.67	0.09	16
	月コメント投稿数	0.00	0.54	0.01	13
	総パッチ投稿数	0.05	0.54	0.10	13
	月パッチ投稿数	0.06	0.46	0.11	11
	総パッチ検証数	0.05	0.67	0.10	16
	月パッチ検証数	0.04	0.38	0.07	9
	活動期間	0.02	0.79	0.04	19
ランダム予測		0.00	0.50	0.01	13
向上率(注4)		5.13	1.66	5.00	1.54

（注 4）本論文のモデルを用いて予測する場合がランダムに予測した場合に比べて，どの程度精度が向上したかを意味する．

する．

実験の結果，適合率は両プロジェクトともに 10%以下であるが，再現率は Eclipse platform プロジェクトで約 70%，Mozilla Firefox プロジェクトで約 83%であった．また，ランダムに予測する場合，Eclipse platform プロジェクトでは適合率が約 0.60%（27 人/4483 人），Mozilla Firefox プロジェクトでは，約 0.39%（24 人/6168 人）であるため，本論文で構築した予測モデルはランダムに予測するよりも高い精度で予測可能であることが分かった．

本論文で構築したモデルで予測した場合と単一のメトリックスを用いて構築したモデルを比較すると，Eclpse platform プロジェクトでは，本論文で構築したモデルで予測した場合の適合率，再現率，F1 値が最も高いことが分かった．Mozilla Firefox プロジェクトでは，再現率は本論文で構築したモデルで予測した場合が最も高くなったが，適合率と F1 値は単一メトリックス（例えば，総パッチ投稿数）を用いて構築したモデルで予測した場合が最も高いことが分かった．Mozilla Firefox プロジェクトにおいて，F1 値が本論文で構築した場合よりも単一メトリックスで構築した場合の方が高かった理由は，Eclipse platform プロジェクトよりもコミッター候補者と一般開発者の人数比が偏っているため，適合率が再現率に比べて極端に低く，再現率の変化よりも適合率の小さな変化が F1 値に大きく影響しているからと考えられる．

次に，各モデルで正確に予測できたコミッター候補者数は，Eclipse platform プロジェクトで最大で 19 人（再現率：約 70%），Mozilla Firefox プロジェクトで最大 20 人（再現率：約 83%）であり，両プロジェクト共に本論文で構築したモデルが最も多くのコミッター候補者数を正しく予測することができた．

本論文で構築したモデルと単一メトリックスを用いて構築したモデルはランダムで予測する場合よりも適合率と F1 値が高く，更に，本論文で構築したモデルは再現率が高い（かつ，正確に予測できたコミッター候補者数が多い）ことが分かった．よって，総合的に判断すると，本論文で構築したコミッター予測モデル

表 7　コミッター昇格前後の活動（Eclipse platform）
Table 7　Differences amount of developer activities between the committers and the non-committers in Eclipse platform project.

	総コメント投稿数		月コメント投稿数		総パッチ投稿数		月パッチ投稿数		総パッチ検証数		月パッチ検証数	
	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後
中央値	50.00	442.00	2.00	5.00	12.00	20.00	1.50	1.00	1.00	20.00	0.50	1.00
平均値	78.11	1236.84	8.05	16.97	31.93	87.96	4.49	2.05	17.27	73.02	2.49	1.48
分散	12434.32	4064627.92	193.27	759.85	2437.66	19424.67	43.59	10.29	1540.46	17268.13	23.07	4.57

表 8　コミッター昇格前後の活動（Mozilla Firefox）
Table 8　Differences amount of developer activities between the committers and the non-committers in Mozilla Firefox project.

	総コメント投稿数		月コメント投稿数		総パッチ投稿数		月パッチ投稿数		総パッチ検証数		月パッチ検証数	
	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後	昇格前	昇格後
中央値	11.00	13.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00
平均値	141.63	193.94	4.41	3.10	25.80	17.57	1.03	0.32	19.25	33.39	0.69	0.57
分散	156099.81	441047.58	184.38	52.04	8493.65	1724.27	3.58	0.27	6563.62	8789.65	1.65	1.27

が最も有意性が高いと考えられる．

本実験では適合率が低い値となったが，その原因は，コミッター候補者と一般開発者の人数比が大きく偏っている点にある．Menzies ら [24] は予測対象のサンプル数（本論文ではコミッター候補者の人数）が極端に少ない場合，高い適合率を得ることは難しいと指摘している．また，再現率が高ければ，ランダムに予測するより適合率が高い場合に限り，そのモデルは有用であると述べている．

本実験では，ロジスティック回帰分析の出力値が 0.5 以上の場合にコミッター候補者として判定した．コミッターとしてふさわしい開発者を正確に抽出したい場合はしきい値の値を大きく，多くのコミッター候補者を挙げたい場合はしきい値を小さくすることが可能である．ただし，しきい値を大きくしすぎると，コミッター候補者と判断される開発者数が減少するため適合率は向上する一方で再現率は低下する．反対にしきい値を小さくしすぎると，コミッター候補者と判断される開発者数が増加するため再現率が向上する一方で適合率は低下する点に注意する必要がある．

6. 考　　察

6.1 コミッター昇格後の活動

本論文で対象とした，Eclipse platform プロジェクト，Mozilla Firefox プロジェクトでは共に約 50 人の開発者がコミッターに昇格していた．我々はコミッター昇格前の活動を分析したが，コミッターに昇格した後も継続してプロジェクトに貢献しているかどうかを確認していない．既存コミッターは開発者の中から，昇格後も積極的に活動し続ける開発者をコミッター候補者として推薦する必要がある．そこで，本論文で対象としたコミッター候補者が，昇格後も継続して積極的に活動しているか否かを確認した．Eclipse platform プロジェクトにおけるコミッター候補者の昇格前後の活動量を表 7 に，Mozilla Firefox プロジェクトにおけるコミッター候補者の昇格前後の活動量を表 8 に示す．両プロジェクト共に，ほとんどの活動は，コミッター昇格後に活動量が減少していなかった．唯一パッチ投稿数が減少していたが，二つの理由が考えられる．一つ目は，自身でパッチをコミットすることができるため，他の開発者から検証を受ける義務がなくなった．二つ目は，パッチ検証，開発の指示，プロジェクト運営のための活動など，機能拡張や不具合修正以外の活動が増加するため，パッチ投稿数が減少した．このことはパッチ検証数やコメント投稿数が増加していることからも見て取れる．

コミッター昇格後，多くの開発者が積極的に活動する一方で，コミッター昇格の後に活動を停止する開発者も存在していた．しかし，コミッター昇格後もいずれかの活動を 1 年以上継続する開発者は Eclipse platform プロジェクトで 55 人中 44 人（約 80%），Mozilla Firefox プロジェクトで 51 人中 30 人（約 59%）存在していた．多くの開発者がコミッター昇格後も継続して活動しているプロジェクトを対象として実験した本論文の妥当性は高いと考えられる．今後は昇格後の開発者の活動も見据えてコミッター候補者を推薦する方法を検討する．

6.2 コミッターに昇格する開発者の活動量

本論文では，実験対象期間中にコメント投稿，パッチ投稿，パッチ検証を行った開発者を対象とした．しかしながら，一般開発者の中にはパッチ投稿やパッチ検証を一度も行っていない者は Eclipse plarform プロジェクトで 4481 人中 4022 人（約 90%），Mozilla Firefox で 6119 人中 5772 人（約 94%）も存在していた．このような一般開発者がコミッターに昇格することは一般的に考えにくい．そこで，パッチ投稿とパッチ検証を一度も行っていない開発者を除いて RQ1 と RQ2 の追加実験を行い，RQ1 の結果を表 9 と表 10 に，RQ2 の結果を表 11 に示す（なお，RQ2 では，学習データに対してのみ当該開発者を除いており，テストデータに対しては行っていない）．

Eclipse platform プロジェクトでは，パッチ投稿やパッチ検証の経験をもつ開発者のみを対象としても，コミッター候補者と一般開発者のパッチ投稿数やコメント投稿数に有意な差（有意水準 5%）があった．よって，Eclipse platform プロジェクトの既存コミッターはパッチ投稿やコメント投稿をより多く行う開発者を調査することで，コミッター候補者を見つけ出すことができると考えられる．その一方で，Mozilla Firefox プロジェクトでは，コミッター候補者のパッチ投稿数やコメント投稿数の中央値は一般開発者に比べて多いものの，有意な差はなかった．つまり，Mozilla Firefox プロジェクトでは，パッチ投稿やパッチ検証を行っているか否かでフィルタリングするのは，コミッター候補者にならない開発者を選定する上では有効であるが，より多く行っているからといってコミッター候補者であるとは限らないことが分かった．

また，パッチ投稿数とパッチ検証数が 0 件の開発者を削除して構築したモデルでコミッター候補者を予測した結果，本論文で構築したモデルよりも再現率が低い（正確に予測できたコミッター候補者数が少ない）ことが分かった．その理由は，パッチ投稿やパッチ検証の経験はないがコメント投稿数の多いコミッター候補者が学習データに含まれていないため，モデルの表現力が下がり，主にコメント投稿を行っているコミッ

表 9 パッチ投稿やパッチ検証の経験をもつ開発者の活動の統計量（Eclipse platform）
Table 9 Statistics of the activities of developers who submitted and reviewed patches in Eclipse platform project.

		中央値	平均値	分散	p 値	効果量
総パッチ投稿数	コミッター(注3)	25.00	43.95	3163.21	0.00	0.01
	一般開発者	1.00	6.73	611.01		
月パッチ投稿数	コミッター	4.25	6.00	44.00	0.00	0.12
	一般開発者	0.00	0.91	2.59		
総パッチ検証数	コミッター	1.50	3.35	28.24	0.04	0.00
	一般開発者	1.00	3.16	243.68		
月パッチ検証数	コミッター	0.75	0.80	0.83	0.11	0.34
	一般開発者	0.00	0.48	0.41		
総コメント投稿数	コミッター	62.00	97.45	23187.94	0.00	0.00
	一般開発者	7.00	104.71	460840.25		
月コメント投稿数	コミッター	3.25	7.98	97.93	0.00	0.06
	一般開発者	0.00	1.37	43.17		
活動期間（月）	コミッター	12.55	17.33	235.76	0.39	-0.01
	一般開発者	19.70	27.65	684.23		

表 10 パッチ投稿やパッチ検証の経験をもつ開発者の活動の統計量（Mozilla Firefox）
Table 10 Statistics of the activities of developers who submitted and reviewed patches in Mozilla Firefox project.

		中央値	平均値	分散	p 値	効果量
総パッチ投稿数	コミッター(注3)	3.00	15.78	988.07	0.05	0.01
	一般開発者	1.00	7.02	640.87		
月パッチ投稿数	コミッター	1.00	0.97	1.19	0.17	0.06
	一般開発者	0.00	0.79	2.99		
総パッチ検証数	コミッター	1.50	8.72	172.68	0.06	0.02
	一般開発者	1.00	4.16	175.67		
月パッチ検証数	コミッター	1.00	0.78	0.54	0.22	0.27
	一般開発者	1.00	0.58	0.49		
総コメント投稿数	コミッター	12.00	64.44	24207.56	0.62	0.00
	一般開発者	7.00	75.66	130965.30		
月コメント投稿数	コミッター	0.00	0.94	1.85	0.58	-0.01
	一般開発者	0.00	1.13	22.70		
活動期間（月）	コミッター	18.54	21.15	301.81	0.81	-0.01
	一般開発者	18.67	25.27	497.15		

表 11 パッチ投稿やパッチ検証の経験をもつコミッター候補者の予測結果
Table 11 Committer prediction results based on the model using the activities of developers who submitted and reviewed patches.

		適合率	再現率	F1 値	正確に予測できたコミッター候補者数
Eclipse platform	本実験モデルを用いた予測	0.07	0.70	0.13	19
	パッチ投稿やパッチ検証の経験をもつ開発者を用いた予測	0.1	0.56	0.20	15
Mozilla Firefox	本実験モデルを用いた予測	0.02	0.83	0.05	20
	パッチ投稿やパッチ検証の経験をもつ開発者を用いた予測	0.00	0.63	0.01	15

ター候補者を予測できなかったと考えられる．高い技術力がなくても，プロジェクトを運営する力をもつ開発者がコミッターに昇格することがあり [1]，このことから，パッチ投稿やパッチ検証が 0 件の開発者をデータセットに含めることは，主にコメント投稿を行っている開発者を予測するために有効であると考えられる．

RQ1 の分析結果，及び，本節で行ったパッチ投稿とパッチ検証を一度も行っていない開発者を削除した実験の効果量に関する結果から，各活動量でコミッター候補者と一般開発者の活動量の実質的な差は，総回数よりも 1 か月当りの回数の方が大きいことが示された．実際に Eclipse platform プロジェクトの月パッチ投稿数や月パッチ検証数の上位者を分析すると，月パッチ投稿数が上位 10 名（月パッチ投稿数 15 件以上）の開発者のうち 6 名はコミッター候補者，月パッチ検証数が上位 10 名（月パッチ検証数 3 件以上）の開発者のうち 3 名はコミッター候補者であった．月パッチ投稿数，月パッチ検証数の上位者の一般開発者の中には，今後コミッターに昇格する開発者が存在する可能性が高い．今後，これらの活動量がコミッターを推薦するための目安になることが期待できる．ただし，一般開発者の中にはコミッター候補者と同等以上に活発に活動している開発者が存在していることが分かった．今後，コミッターにふさわしい開発者を推薦するために，開発者の活動内容（コメント内容やパッチの品質など）を分析することも重要である．

6.3 本論文の制約

本論文では，コミッター候補者の活動として，パッチ投稿からプロダクト反映までのプロセス中の活動に着目した．モデル構築に用いた開発者の活動量には，コメント投稿，パッチ投稿，パッチ検証以外に，パッチの行数などが挙げられる．従来研究 [22] で，投稿される大多数のパッチは 10 行以下であること（パッチの変更行数に差がないこと）が示されており，開発者が投稿したパッチの総量は投稿回数と非常に高い相関が出る（パッチ投稿回数の多いコミッター候補者はパッチの総量が多く，パッチ投稿回数の少ない一般開発者はパッチの総量が少ない）ことが分かっていたため，本論文ではパッチの総量を用いなかった．一方で，前節で述べたコメントの内容やパッチの品質はコミッター候補者と一般開発者で技術的な違いがあると考えられる．しかしながら，コミッターによる建設的なコメントや，承認されるパッチを作った開発者を正確に自動抽出することは現時点で難しい [25] ため，これらの活動量をモデル構築に用いることは今後の課題とする．

また，コミッターに昇格するための貢献は，本論文で挙げた不具合管理システムに記録される活動だけでなく，メーリングリストなどでの活動も含まれる可能性がある．コミッターに昇格する開発者の予測精度向上に向けて，プロジェクトにおけるその他の活動量について調査を続けていく．

コミッターの中にはコミッター昇格前の活動が少ない（一般開発者とほぼ同等の活動量）開発者が存在する．このような開発者はプロジェクト参加当初からコミッターである可能性が高い．本論文では，コミッター昇格前（初めてリポジトリにコミットする以前）に活動していない開発者は除外した．しかし，初めてリポジトリにコミットするまでにパッチ投稿，パッチ検証，コメント投稿を行えば，当該開発者は一般開発者として活動したことになる．実際に OSS プロジェクトの既存コミッターが本手法を適用する場合，データセットの準備時にコミッター候補者か否かを正確にラベル付けできるため，再現率の向上が見込まれる．

7. むすび

本論文では，コミッター選出を支援することを目的として，コミッター候補者に推薦されるべき有能な開発者を見つけ出すために有用な活動量を分析し，開発者の活動量からコミッターに昇格する開発者をどの程度の精度で予測することができるかを実験的に評価した．Eclipse platform プロジェクト，Mozilla Firefox プロジェクトを対象に実験を行った結果，Eclipse platform プロジェクトでは再現率が約 70%，Mozilla Firefox プロジェクトでは約 80%の精度でコミッターに昇格する開発者を予測できることが分かった．また，継続的にパッチの投稿，パッチの検証を行う開発者がコミッターに昇格していることが分かった．今後はコミッターの予測精度向上に向けた実験，並びに昇格後のコミッターの活動を見据えて新たなコミッターを見つけ出す方法を検討する．

謝辞 本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．

また，本研究の一部は，文部科学省科学研究補助費（若手 B：課題番号 22700033）による助成を受けた．

文　献

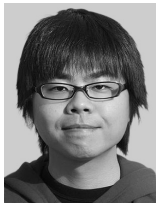
[1] K. Fogel, Producing open source software: How to run sucessful free software project, Sebastopol, CA, O'Reilly Media, 2005.

[2] G. Canfora and L. Cerulo, "Supporting change request assignment in open source development," Proc. Symposium on Applied Computing (SAC'06), pp.1767–1772, 2006.
[3] B. Shibuya and T. Tamai, "Understanding the process of participating in open source communities," Proc. ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS'09), pp.1–6, 2009.
[4] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful... really?," Proc. International Conference on Software Maintenance (ICSM'08), pp.337–345, 2008.
[5] M. Nurolahzade, S.M. Nasehi, S.H. Khandkar, and S. Rawal, "The role of patch review in software evolution: an analysis of the mozilla firefox," Proc. Joint International and annual ERCIM Workshops on Principles of Software Evolution and Software Evolution Workshops (IWPSE-Evol'09), pp.9–18, 2009.
[6] A. Ihara, M. Ohira, and K. Matsumoto, "An analysis method for improving a bug modification process in open source software development," Proc. Joint International and annual ERCIM Workshops on Principles of Software Evolution and Software Evolution Workshops (IWPSE-Evol'09), pp.135–144, 2009.
[7] C. Jensen and W. Scacchi, "Role migration and advancement processes in ossd projects: A comparative case study," Proc. International Conference on Software Engineering (ICSE'07), pp.364–374, 2007.
[8] P.D.A.S. Christian Bird, Alex Gourley, and G. Hsu, "Open borders? immigration in open source projects," Proc. International Workshop on Mining Software Repositories (MSR'07), p.6, 2007.
[9] C.R. deSouza, S. Quirk, E. Trainer, and D.F. Redmiles, "Supporting collaborative software development through the visualization of socio-technical dependencies," Proc. 2007 International ACM Conference on Supporting Group Work (GROUP'07), pp.147–156, 2007.
[10] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," Proc. International Conference on Software Engineering (ICSE'09), pp.23–33, 2009.
[11] M. Lanza and M. Pinzger, "A bug's life": Visualizing a bug database," Proc. International Workshop on Visualizing Software for Analysis and Understanding (VISSOFT'07), pp.113–120, 2007.
[12] J. Anvik, L. Hiew, and G.C. Murphy, "Who should fix this bug?," Proc. International Conference on Software Engineering (ICSE'06), pp.361–370, 2006.
[13] D. Cubranic, "Automatic bug triage using text categorization," Proc. International Conference on Software Engineering & Knowledge Engineering (SEKE'04), pp.92–97, 2004.
[14] S. Rastkar, G.C. Murphy, and G. Murray, "Summarizing software artifacts: A case study of bug reports," Proc. International Conference on Software Engineering (ICSE'10), pp.505–514, 2010.
[15] S. Fujita, A. Ihara, M. Ohira, and K. Matsumoto, "An analysis of committers toward improving the patch review process in oss development," Supplementary Proceedings of the International Symposium on Software Reliability Engineering (ISSRE'10), pp.369–374, 2010.
[16] V.B. Kampenes, T. Dybå, J.E. Hannay, and D.I.K. Sjøberg, "Systematic review: A systematic review of effect size in software engineering experiments," Information and Software Technology, vol.49, pp.1073–1086, 2007.
[17] M.W. Lipsey and D.B. Wilson, Practical meta-analysis, Applied social research methods series, Thousand Oaks, 2001.
[18] V.R. Basili, L.C. Briand, and W.L. Melo, "A validation of object-oriented design metrics as quality indicators," IEEE Trans. Softw. Eng., vol.22, pp.751–761, 1996.
[19] Y. Takagi, O. Mizuno, and T. Kikuno, "An empirical approach to characterizing risky software projects based on logistic regression analysis," Empirical Software Engineering, vol.10, pp.495–515, 2005.
[20] G. Liang, L. Wu, Q. Wu, Q. Wang, T. Xie, and H. Mei, "Automatic construction of an effective training set for prioritizing static analysis warnings," Proc. International Conference on Automated Software Engineering (ASE'10), pp.93–102, 2010.
[21] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Trans. Inf. Syst., vol.22, pp.5–53, 2004.
[22] P. Weißgerber, D. Neu, and S. Diehl, "Small patches get in!," Proc. International Working Conference on Mining Software Repositories (MSR'08), pp.67–76, 2008.
[23] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," Proc. Conference on Computer Supported Cooperative Work (CSCW'10), pp.301–310, 2010.
[24] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," Automated Software Engineering, vol.17, pp.375–407, 2010.
[25] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links: Bugs and bug-fix commits," Proc. International Symposium on Foundations of Software Engineering (FSE'10), pp.97–106, 2010.

（平成 23 年 7 月 20 日受付，10 月 18 日再受付）

伊原　彰紀　（学生員）

平 19 龍谷大・理工卒．平 21 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了．現在，同大学博士後期課程在籍．修士（工学）．ソフトウェア工学，特にオープンソースソフトウェア開発支援の研究に従事．情報処理学会，IEEE 各会員．



亀井　靖高　（正員）

平 17 関西大・総合情報卒．平 19 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了．平 21 同大学院博士後期課程了．同年日本学術振興会・特別研究員（PD）．平 22 カナダ・Queen's 大学・博士研究員．平 23 九州大学大学院システム情報科学研究院・助教．博士（工学）．ソフトウェアメトリクス，マイニングソフトウェアリポジトリ等の研究に従事．情報処理学会，IEEE 各会員．

大平　雅雄　（正員）

平 10 京都工繊大・工芸・電子情報工学卒．平 15 奈良先端科学技術大学院大学情報科学研究科博士課程了．同年同大学産学官連携研究員．平 16 同大学情報科学研究科助手（平 19 より助教）．オープンソースソフトウェア工学の研究に従事．コラボレーション支援環境の開発に興味をもつ．ACM 会員．

松本　健一　（正員）

昭 60 阪大・基礎工・情報工学卒．平元同大大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平 5 奈良先端科学技術大学院大学助教授．平 13 同大学教授．博士（工学）．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集/利用支援の研究に従事．情報処理学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．

鵜林　尚靖　（正員）

昭 57 広島大・理・数学卒．平 11 東京大学大学院総合文化研究科広域科学専攻広域システム科学系博士課程了．博士（学術）．昭 57〜平 15（株）東芝に勤務．平 14〜15 芝浦工業大学システム工学部非常勤講師．平 15 九州工業大学情報工学部助教授，平 22 九州大学大学院システム情報科学研究院教授，現在に至る．平 15 年度情報処理学会山下記念研究賞受賞．ソフトウェア工学，プログラミング言語モデルに興味をもつ．日本ソフトウェア科学会，情報処理学会，ACM，IEEE-CS 各会員．