博士論文

ステークホルダの行動変容に基づくソフトウェア品質と開発速度の向上

中才 恵太朗

奈良先端科学技術大学院大学
先端科学技術研究科
情報理工学プログラム

主指導教員: 松本 健一
ソフトウェア工学研究室（情報科学領域）

令和 3 年 3 月 17 日提出

本論文は奈良先端科学技術大学院大学先端科学技術研究科に
博士(工学) 授与の要件として提出した学位論文である。

中才　恵太朗

審査委員：
主査　松本 健一 （情報科学領域　教授）
安本 慶一 （情報科学領域　教授）
石尾 隆 （情報科学領域　准教授）
畑 秀明 （情報科学領域　助教）
Raula Gaikovina Kula（情報科学領域　助教）
角田 雅照 （近畿大学　総合理工学研究科　准教授）

ステークホルダの行動変容に基づくソフトウェア品質と開発速度の向上*

中才　恵太朗

内容梗概

本論文では，ソフトウェアの機能や部品化の単位の一つである「モジュール」に着目し，ソフトウェア開発に関わるステークホルダの行動変容を促進することで，モジュールの品質向上を達成しつつ開発速度も向上させる手法の提案と分析を行う．具体的には，プロジェクト管理者の行動変容を促すテスト計画立案法，および，開発者の行動変容を促す Web 検索戦略をそれぞれ提案すると共に，ソフトウェア利用者の行動変容を促す寄付バッジの効果分析を行った．ソフトウェア開発は人手による部分が多く，プロジェクト管理者，ソフトウェア開発者，ソフトウェア利用者といったステークホルダの行動が品質向上に重要な役割を果たすとされている．特に，近年のソフトウェア開発は，大規模，かつ，短納期となっており，その重要性はますます高くなっている．なお，本論文では，モジュールが仕様と異なる動作をする状態を欠陥と定義し，欠陥が多く発生するソフトウェアほど品質が低いとする．

具体的な成果は次の通りである．（１）欠陥モジュールの予測において，多数提案されている予測モデルの中から予測精度が高いモデルを一貫して選択し，テスト計画に反映する手法を，バンディットアルゴリズムを用いて実現した．欠陥モジュール予測において，ある特定の予測モデルが，あらゆるデータセットにおいて高い予測精度を示すとは限らないことが知られている．いわゆる，欠陥モジュール予測における外的妥当性の問題である．バンディットアルゴリズムは，得られる報酬が不明な候補を対象に，期待される報酬の高い候補を順次検索し，得られる総報酬を最大化することができる．提案手法を用いた実験では，３個のデータセットに対して，予測精度が最も高い，もしくは，２番目

に高いモデルが選択されることを確認できた.（２）ソフトウェア開発者がプログラム作成時に行った Web 検索のログを分析するための指標を定義し，Web 検索戦略とプログラミング効率の関係を明らかにした．例えば，開発者が Web 検索エンジンに入力する単語の組（キーフレーズ）を変更しないまま検索結果となる Web ページを多数読み進めるという戦略は，プログラミング効率の面からは避けるべきである，との知見を得た.（３）オープンソースソフトウェアの開発プロジェクトに対する寄付の有無などを示す「寄付バッジ」が，バグレポートへの対応に与える影響を明らかにした．Eclipse プロジェクトを対象に，バグレポート提出者が寄付バッジを有していることが，当該バグレポートへの応答時間（当該バグの解消に係る最初のコメントが付与されるまでの時間）に与える影響を Quantile Difference in Differences (QDID)により分析した．その結果，バグレポート提出者が寄付バッジを有する場合，当該レポートへの応答時間は，中央値で約２時間短縮されることが分かった.

本論文は，ソフトウェア開発に関わるステークホルダの行動変容が，ソフトウェアの品質向上だけでなく，開発速度の向上にも寄与することを具体的に示すものである．欠陥モジュール予測，ソフトウェア開発における Web 検索，そして，オープンソースソフトウェア開発における寄付バッジといった，ソフトウェア開発支援技術の発展にも大きく資する.

キーワード

Web 検索戦略，欠陥予測，バンディットアルゴリズム，寄付バッジ，差分の差分分析

*奈良先端科学技術大学院大学 先端科学技術研究科 博士論文, 令和 3 年 3 月 17 日.

Improvement of Software Quality and Development Speed Based on the Behavior Modification of Stakeholders*

Keitaro Nakasai

Abstract

In this thesis, we propose new methods to promote the behavior modification of stakeholders of software development. The method is expected to improve the development speed and the software quality. More specifically, (1) we propose a test planning method to promote behavioral change in project managers, (2) we propose a web search strategy to promote the behavior modification in developers, and (3) we analyze donation badges' effect to promote the behavior modification in software users. Software development is largely depend on human resources, and therefore stakeholders such as project managers, software developers, and software users play an important role in improving quality. In recent years, managing software development has become important because software size is large and delivery time of software is short. We focus on modules which are one of the units of software functions and components, and define a defect when the behavior of the modules is different from the specification. The more defects are generated, we regard that the lower the quality of the software.

The major results are as follows. (1) We propose a new method which applies the bandit algorithm to defect module prediction. The method dynamically selects a model with high prediction accuracy from among many proposed prediction models during the test phase. In the defect prediction, it is known that there is no prediction model which shows high prediction accuracy on all datasets. This problem is the external validity of defect prediction models. The bandit algorithm can maximize the average accuracy by sequentially searching for candidates whose accuracy is unknown. In the experiment, using the proposed method, the model with the highest or second-highest prediction accuracy was achieved on three datasets. (2) We defined

metrics for analyzing software developers' web search strategy when they make programs. Using the metrics, we clarified the relationship between web search strategies and programming efficiency. For example, we found that reading many web pages without changing search key phrases is inefficient. Therefore, the developer should avoid such behavior when making programs. (3) We analyzed the effect of the donation badge on the response time to bug reports in the Eclipse project. In the analysis, we applied Quantile Difference in Differences (QDID) analysis. For Eclipse projects, a donation badge on the bug report impacts the bug report's response time (the time it takes for the first comment to resolve the bug). The results show that the median of response time to a bug report is reduced by about two hours when the submitter has a donation badge.

This thesis explicitly shows that the behavior modification of stakeholders involved in software development can improve not only software quality but also development speed. This thesis also contributes to the software development support technologies such as defect module prediction, web search in software development, and donation badges in open source software development.

Keywords:

Web search strategy, defect prediction, bandit algorithm, donation badge, difference in differences

.*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 17, 2021.

関連発表論文

査読付き学術論文

1. Keitaro Nakasai, Hideaki Hata, and Kenichi Matsumoto : Are Donation Badges Appealing? A Case Study of Developer Responses to Eclipse Bug Reports, IEEE Software, Special Issue on Managing Software Platforms and Ecosystems, Vol.36, Issue 3, 2019 年 5 月, pp.22-27, 3 章

査読付き学術論文（レター）

1. Teruki Hayakawa, Masateru Tsunoda, Koji Toda, Keitaro Nakasai, Amjed Tahir, Kwabena Ebo Bennin, Akito Monden, and Kenichi Matsumoto : A Novel Approach to Address External Validity Issues in Fault Prediction Using Bandit Algorithms, IEICE Transactions on Information and Systems, Vol.E104-D, No.2, 2021 年 2 月, pp.327-331, 5 章

査読付き国際会議発表

1. Keitaro Nakasai, Hideaki Hata, and Kenichi Matsumoto : Are Donation Badges Appealing? A Case Study of Developer Responses to Eclipse Bug Reports, The 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), Journal First, 2019 年 8 月, 3 章
2. Keitaro Nakasai, Hideaki Hata, Saya Onoue, and Kenichi Matsumoto : Analysis of Donations in the Eclipse Project, Proc. of 8th IEEE International Workshop on Empirical Software Engineering in Practice (IWESEP 2017), 2017 年 3 月, pp.18-22, 2 章
3. Keitaro Nakasai, Masateru Tsunoda, and Hideaki Hata : Web Search Behaviors for Software Development, Proc. of 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2016), 2016 年 5 月, pp.125-128, 4 章

研究会

1. Keitaro Nakasai, Hideaki Hata, and Kenichi Matsumoto : Impact of Donation Badges on Response Time for Eclipse Bug Reports，The 9th International Workshop on Empirical Software Engineering in Practice (IWESEP 2018), Poster Session，2018 年 12 月, 3 章
2. 中才 恵太朗, 畑 秀明, ラウラ・ガイコビナ・クラ, 松本 健一 : Eclipse 寄付バッジによるバグレポート応答時間の影響分析，ソフトウェアエンジニアリングシンポジウム 2018 (SES2018)，2018 年 9 月，pp.208-215, 3 章
3. 中才　恵太朗, 畑　秀明, 角田雅照, 松本　健一 : OSS に寄付をすべきか？統計的因果推論による寄付バッジの効果分析，ソフトウェアエンジニアリングシンポジウム 2017 (SES2017)，2017 年 8 月，pp.278-279, 3 章
4. 中才 恵太朗, 尾上 紗野, 畑 秀明,松本 健一, オープンソースソフトウェアにおける寄付の分析，情報処理学会研究報告，ソフトウェア工学研究会, 2016-SE-194, No.5，2016 年 11 月，pp.1-6, 2 章

その他の発表論文

学術論文

1. 大神 勝也， 中才 恵太朗, 畑 秀明, 松本 健一 : Heijo: 動的なコード実行可視化による Java/Android アプリケーションのリアルタイムプロファイラ，コンピュータソフトウェア，日本ソフトウェア科学会，Vol.36，No.2，2019 年 4 月，pp.93-105
2. 上村 恭平, 中才 恵太朗, 大神 勝也, 畑 秀明, 一ノ瀬 智浩, 松本 健一, 飯田 元 : Codosseum: オープンなソフトウェア開発・分析支援 Web サービス，コンピュータソフトウェア，日本ソフトウェア科学会，Vol.36，No.1，2019 年 1 月，pp.38-47

国際会議論文

1. Keitaro Nakasai, Yoshiharu Ikutani, Daiki Takata, Hideaki Hata and Kenichi Matsumoto : Toward Sustainable Communities with a Community Currency -- A

Study in Car Sharing, Proc. of 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2019), 2019 年 7 月, pp.478-483

2. Keitaro Nakasai, Masateru Tsunoda, Hideaki Hata and Kenichi Matsumoto : Identifying Spoofing Accounts on Twitter Based on Relationships of Accounts, Proc. of 2018 IEEE International Conference on Big Data, Cloud Computing, Data Science and Engineering (BCD 2018), 2018 年 7 月, pp.85-90

研究会

1. 中才 恵太朗, 角田 雅照, 松本 健一 : 瞬目を利用したソフトウェア開発におけるメンタルワークロード定量化にむけての実験提案, ウィンターワークショップ 2020・イン・京都, 2020 年 1 月, pp.11-12
2. 田内 遥夏, 中才 恵太朗, 畑 秀明, 松本 健一 : Waiting Self-Admitted Technical Debt の分析と考察, 情報処理学会研究報告, ソフトウェア工学研究会, 2018-SE-200, No.9, 2018 年 12 月, pp.1-6
3. Natnaree Asavaseri, 中才 恵太朗, 畑 秀明, Raula Gaikovina Kula, 松本 健一, FLOSS 開発者は Microsoft の GitHub 買収をどう考えているか, ソフトウェアエンジニアリングシンポジウム 2018 併設ワークショップ 「ソーシャルコーディングのための実証的ソフトウェア工学」, 2018 年 9 月
4. 上村 恭平, 中才 恵太朗, 大神勝也, 畑 秀明, 一ノ瀬 智浩, 松本 健一, 飯田 元 : Codosseum: OSS プロジェクトモニタリング Web サービス, 日本ソフトウェア科学会第 34 回大会 (2017 年度) 講演論文集, 2017 年 9 月, pp. 97-103
5. 溝口 拓也, 梶原 武紘, 中才 恵太朗, 宮田 明裕, 久保 尋之 : フォグディスプレイを用いた直感的な映像編集システムの提案, 電子情報通信学会 総合大会, D-12-38, 2017 年 3 月, 1page

目次

図目次

表目次

1. はじめに

ソフトウェアは現在社会基盤の一つとなっており，ソフトウェアの品質を高めることは非常に重要である．ソフトウェアの品質には様々な側面があるが[43]，本論文ではモジュールの品質に着目する．すなわち，モジュールが仕様と異なる動作をすることを欠陥と定義し，この欠陥が多く発生するソフトウェアを低品質とする．近年，ソフトウェア開発は大規模，かつ短納期となっており，このような状況下でソフトウェアの品質を維持しつつ短納期を実現することは容易ではない．

ソフトウェア開発は人手による部分が多く，それゆえに開発に関わる人間，すなわちステークホルダの行動が，品質向上に重要な役割を果たす．ソフトウェア開発のステークホルダとは，プロジェクト管理者，ソフトウェア開発者，ソフトウェア利用者である．前述の状況下でソフトウェアの品質を維持しつつ短納期を実現するためには，これまでのステークホルダの行動を大きく変化させる，すなわち行動変容させる必要がある．

欠陥が少なく，短納期の（リリースの早い）ソフトウェアでは，プロジェクト管理者，ソフトウェア開発者，ソフトウェア利用者のそれぞれが適切に行動している．この関係を図 1 に示す．

- プロジェクト管理者: ソフトウェアのテスト時に，ソフトウェアモジュールに対して，欠陥が含まれやすいかどうかを高精度に予測する．これにより，テストの計画やリリース時期を適切に立案することができ，その結果ソフトウェアの欠陥を抑えることができる．
- ソフトウェア開発者: 報告された欠陥や問題点に対し，修正を行うために自身の知識を活用するとともに，Web 検索を有効活用する．これにより適切に，かつ早期に欠陥や問題点を除去することができる．
- ソフトウェア利用者: 自身が利用するソフトウェアに欠陥や問題点を発見した場合，プロジェクト管理者が用意した窓口（バグトラッキングシステム）に報告する．これにより，早期に欠陥や問題点を発見できる．

すなわち，ソフトウェア品質と開発速度を向上（短納期を実現）させるためには,それぞれのステークホルダが上記のような適切な行動を取る必要がある．

ただし，それぞれが以下のような行動を取る場合があり，それがソフトウェアの品質や開発速度向上を阻害している．

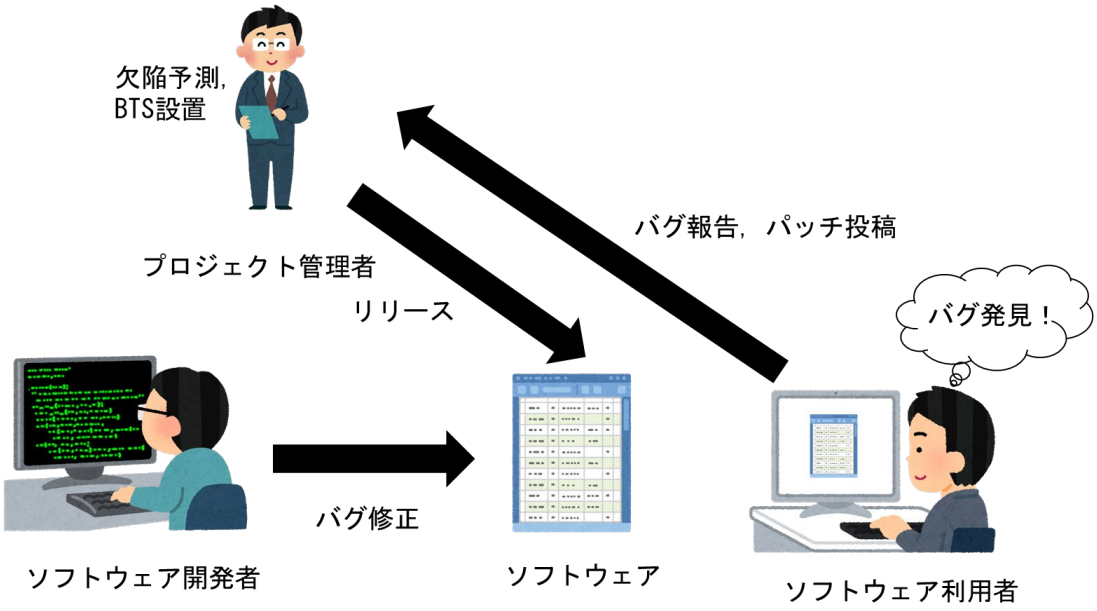


図 1 ソフトウェア品質と開発速度に関わるステークホルダ

- プロジェクト管理者: 欠陥予測の手法が数多く存在するが，常に高い予測精度となるような汎用的な方法は存在していない．そのため，不適切な予測方法を誤って選択してしまい，適切にテスト計画やリリース時期を立案することができない．
- ソフトウェア開発者: ソフトウェアの欠陥や問題点の修正時に行う Web 検索の戦略に，個人差が存在する．また，検索戦略に明確な指針は存在しない．このため，検索戦略の差異が，修正の効率や品質に影響を及ぼす．その結果，開発者によっては欠陥や問題点の除去に時間が掛かる．
- ソフトウェア利用者: 欠陥や問題点を報告しても，ソフトウェア開発者に早期に対応されない．その結果，ソフトウェア品質の低さが利用者の業務などに影響する．さらに，ソフトウェアの品質向上に意欲的なソフトウェア利用者が問題点を報告しても，実際に早期に対応されないため，利用者が問題点を報告しなくなる．

上記の品質や開発速度向上を阻害するような，各ステークホルダの行動を変

容するためには，欠陥予測のために最適な予測手法を提案する，プログラミング時における検索戦略の指針を提供する，意欲的なソフトウェア利用者を判別できるようにするなどが考えられる．このような解決策を提案するためには，ソフトウェア開発データを収集して適切な分析を行うとともに，新たな方法を提案する必要がある．ただし我々の知る限り，これまで上記のような行動変容の促進を支援するような研究は実施されてこなかった．

本論文では，ソフトウェアモジュールの品質向上を達成しつつ，短納期を実現することをゴールとして，ステークホルダの行動変容を促進するというアプローチを取る．より具体的には，プロジェクト管理者の行動変容のための新たなテスト計画立案法の提案，ソフトウェア開発者の行動変容のための新たなWeb 検索戦略の提案，さらには，ソフトウェア利用者の行動変容のための寄付バッジの効果分析を行う.以下はアプローチをより詳細に説明したものである．

- プロジェクト管理者: テスト計画の立案に関して行動変容を促すために，ソフトウェア欠陥予測のための新たな手法を提案するともに，NASA メトリクスデータプログラムのデータを用いて，提案方法の精度について評価する．
- ソフトウェア開発者: コーディング時の Web 検索に関して行動変容を促すために，ソフトウェア開発中の検索行動データを収集し，そのデータの分析を行う．これにより，コーディングにおける効率の高い Web 検索戦略について明らかにする．
- ソフトウェア利用者: OSS への寄付に関する行動変容を促すために，寄付バッジの OSS コミュニティにおける効果を明らかにする．分析では Eclipse プロジェクトのバグレポートデータを用い，寄付バッジを用いたバグレポートがそうでないものに比べ早く対応されることを明らかにする．

以降，2 章において，オープンソースソフトウェアプロジェクトである，Eclipse プロジェクトの寄付データを用いて，効果的な寄付の収集方法を明らかにするため，分析を行った．その結果，(1) 特典が寄付への動機づけとなっていること，(2) 全体的な寄付者のうち開発者の割合は少ないが，開発者の寄付額は開発者でないものより大きかったこと(3)リリース日には寄付が増えるが，バグ数が多いと寄付が落ち込むようにみえた．ことがわかった．

3 章において，バグレポート提出後，最初の応答までの時間が寄付バッジの影響を受けるのかを明らかにする．寄付バッジをもたないバグレポートと比較して，寄付バッジをもつバグレポートに対する開発者の応答時間を Quantile Difference in Differences (QDID) という手法で分析した．その結果，寄付バッジが応答時間を中央値で約 2 時間短縮することが明らかとなった．

4 章において， プログラム作成時における検索ログを分析し，プログラミング時における検索戦略の指針について明らかにする．実験では被験者に対し，プログラミングに関する出題を行い，それに対してプログラミングにより回答してもらう．その検索ログに対し，分析メトリクスを設定し分析を行った．その結果，一度使用した検索のキーワードを含むキーフレーズを再び用いることはプログラミング時の検索行動としては効率が高くない可能性があることなどの検索戦略の指針を明らかにした．

5 章では，ソフトウェア欠陥予測において，用いるデータセットが異なる場合，予測モデルの性能が変化してしまう問題に対して，バンディットアルゴリズムを適用することを提案する．また，提案方法の評価を行い，その有用性について明らかにする．実験では，3 種類のデータセットを用い，従来方法として 4 種類の予測方法を用意するとともに，それらとバンディットアルゴリズムによる動的なモデル選択の性能を比較した．さらに majority voting を用いて得られた予測とバンディットアルゴリズムを適用した予測の精度を比較した．その結果，バンディットアルゴリズムのひとつである Epsilon-greedy 法の ε が 0.3 のとき，これらの従来方法を単独で用いた場合と比較して，最も高い予測精度か，2 番目に高い精度を示した．

最後に，6 章にて本論文のまとめを述べる．

2. OSS プロジェクト Eclipse の寄付に関する分析

2.1. まえがき

オープンソースソフトウェア（OSS）とは自由に利用できるソフトウェアで，企業や官公庁，個人でも多くの利用がある．利用している OSS を改善したければ，自身が OSS 開発に参加し，貢献することもできる．OSS 開発に貢献するにはバグを報告する，ドキュメントを書く，コードレビューに参加する，パッチ投稿を行うなどがある．これらの方法で OSS に貢献するためには高い技術が必要である．一方で，技術を要しない OSS 開発への貢献方法として，OSS の運営団体に寄付をするというものがある．

多くの OSS 運営団体では寄付を募っており，LibreOffice で知られる Document Foundation の会計報告によれば 748,029 ユーロ（2015 年度）の寄付が寄せられている．OSS への寄付金はサーバ，インフラストラクチャ，ドメインの維持やワークショップの開催費用などに使われる．さらに，フルタイム開発者を雇うための費用にもなるため OSS 運営団体は寄付を積極的に募る必要がある．多くの寄付を集めるため，OSS の運営団体はいろいろな工夫をしている．例えば，寄付専用の Web ページを作成，様々な寄付手段（クレジットカード，PayPal，ビットコイン，Flattr，銀行口座）の提供，所得控除の申請，さらにノベルティグッズの配布などである．また，2015 年 8 月には JUnit がクラウドファウンディングサイト INDIEGOGO を利用した寄付金募集キャンペーンを行った．目標金額 25,000 ユーロだったが，目標金額の 2 倍以上の 53,937 ユーロ集めることができた．このように，OSS 運営団体では多種多様な手段で寄付を募っている．

現在，寄付の募り方は一様ではなく，方法も確立されていないため効果的な寄付の募り方は明らかになっていない．寄付を行う人や，寄付へのモチベーションを理解できれば効果的な寄付の募り方を提案することが可能となり，より効果的に寄付金を集めることができると考えられる．本章は，効果的な寄付の収集方法を明らかにするため，著名な OSS プロジェクトである Eclipse を対象に調査した．本章では以下のリサーチクエスチョンを設定した．

- RQ1: 特典は寄付額に影響するか？

- RQ2: 開発者からの寄付にはどのような特徴があるか？
- RQ3: リリース日とバグ数は全体の寄付額に影響するか？

2.2. 分析手法

表 1 使用したデータセット

使用データ	入手 URL
寄付者リスト	https://eclipse.org/donate/donorlist.php
Git リポジトリ	https://git.eclipse.org/c/
Bugzilla	https://bugs.eclipse.org/bugs/

図 2 Bugzilla 上のバッジ

2.2.1. リリース日

Eclipse は毎年 6 月末にメインリリースを行い．9 月末と 2 月末から 3 月初めにサービスリリース(SR)を行っている．本章ではメインリリース日とサービスリリース日が寄付の増減に影響を与えていると考え，リリース日前後の寄付の増減を分析した．

2.2.2. データセット

本章で用いたデータセットが公開されている URL を表 1 に示す．データセットは Eclipse Foundation が公開している寄付者リストと Git リポジトリと Bugzilla を使用している．

2.2.2.1. 寄付者リスト

Eclipse Foundation は寄付を開始した 2007 年 12 月 30 日から現在までの寄付者のデータを公開している．寄付者リストから得られるデータは名前，メッセージ，日付，金額である．また，年間 35 ドル以上寄付した人を Friend と呼び，寄付者リストの横にロゴマークがつく．さらに年間 50 ドル以上寄付した人は Best Friend と呼ぶ．Friend と Best Friend は特別なリストが作られており希望する場合はプロフィール画像を載せることもできる．本章では 2007 年 12 月 30 日から 2016 年 6 月 30 日までの期間の 28,349 レコードを使用した．

2.2.2.2. Gitリポジトリ

本章では開発開始から 2016 年 6 月 30 日までの 912 リポジトリの開発履歴を使用した．Krishnamurthy らの報告によれば，開発者は関連のある OSS プラットフォームに寄付を行う[22]．しかし Krishnamurthy らが対象とした Sourceforge.net は OSS 開発プラットフォームであり，利害対象となる関係団体が多く，Eclipse とは性質が異なる．Eclipse のような著名な OSS プロジェクトに寄付する開発者の実態を調査するために Git リポジトリの開発履歴を調べた．

2.2.2.3. Bugzilla

重大なバグの混入は，ユーザの寄付への意欲に影響を与えると考えられる．例えば，ソフトウェアのバージョンアップに期待をしていたユーザは寄付の意欲がなくす可能性がある．一方，ユーザは開発者に対し，ソフトウェアのバグ修正に精力的に取り組んでほしいと願い，寄付を行う可能性がある．そこで，本章では Bugzilla のデータを用いて重大なバグの混入の有無と寄付の増減の関係を調査する. Bugzilla とは Mozilla Foundation が開発した OSS のバグ管理システムである．Eclipse のバグレポートは Bugzilla 上で管理されている．Bugzilla 上ではエンジニアがバグレポートに対し，優先度をつけることができる．優先度が高い順に P1 から P5 まであるが実際には P1，P2，P3 だけが使用されている．そのうち P1，P2 に分けられるバグレポートは少なく，P3 が大半である．よって本章では P1 と P2 を重大なバグであると定義する．本章では分類を Eclipse に限定して，バージョン 3.2 からバージョン 4.6 SR1 までのバグレポー

ト数について調査した（2016 年 10 月 14 日現在）.

2.2.3. 寄付の特典

Eclipse Foundation では寄付を行ったものに特典を用意している. 2016 年 7 月時点では，一年間に 35 ドル以上の寄付者を対象として，通常よりも高速なサーバにアクセスできる権利や Friend of Eclipse のロゴ使用の権利とその他にも書籍が割引価格で買える権利などが与えられる．また，Bugzilla 上でのバッジも用意しており，寄付を行ったユーザは他のユーザに対しアピールすることができる． 図 2 に Bugzilla 上のバッジを示す．図 2 のユーザ A は寄付を行っており，スクリーンネームの横に寄付を行った証であるバッジが表示されている．さらにこのバッジには寄付を促すリンクが貼られてある. 図 2 のユーザ B は通常のユーザであるためスクリーンネームの横にバッジは付いていない．また，50 ドル以上の寄付者は Best Friend の称号が与えられ，T シャツと Eclipse 会議の割引が用意されている. Eclipse の T シャツがもらえる金額は変動するようである．また，2016 年 10 月 17 日現在では会議の割引は用意されていない.

2016 年 7 月時点では自由に寄付額を入力することもできるが，5 ドル，10 ドル，35 ドル，50 ドル，250 ドルの選択肢が用意されていた．またそれぞれ称号が付いており，Donor，Supporter，Friend of Eclipse，Neon Best Friend，Webmaster Idol となっていた.

寄付をする際には名前とメールアドレスとコメントを入力することができるが任意となっており，匿名希望とすることもできる．その際，寄付者リストの名前は Anonymous となる. 寄付をする時には今回限りか毎月か毎年かを選択することもできる.

2.2.4. リサーチクエスチョンの目的と分析手法

本章で行うリサーチクエスチョンの目的と分析方法について説明する.

2.2.4.1. RQ1: 特典は寄付額に影響するか？

OSS の運営団体は一定金額の寄付をした人に特典を与えることが多い．しか

し，その特典がユーザの寄付への意欲に影響を与えているかは明らかにはなっていない．特典が寄付を促すために有効であると明らかになれば，OSS で寄付を募る時には特典を用意することでより多くの寄付が望まれる．特典が寄付に影響を与えていないのであれば，OSS で寄付を募る時には必ずしも特典を用意する必要は無いと考えられる．

具体的な分析手法は全寄付数のうち何%の人が 35 ドル以上寄付をしているのかを調べる．そのうち何%が特典の付与される 35 ドルを過不足なく寄付したのかを調べることによりこのリサーチクエスチョンを検証する．

2.2.4.2. RQ2: 開発者からの寄付にはどのような特徴があるか？

自身が開発に参加したプロジェクトに寄付をする開発者からの寄付の特徴を調べることで，ニーズに合わせた寄付の集め方ができるのではないかと考えられる．仮に開発者の寄付が多いのであれば，Eclipse のイベント参加の割引はニーズに合っている．開発者の寄付が少ないのであれば，利用者主体の特典をつけることで効率よく寄付を集めることができるのではないかと考えられる．

具体的な分析手法は Git リポジトリにコミットしている全コミッターの名前のうち寄付者リスト中の名前と同じものがどれだけあるか調べるとともに，開発者の寄付金額の平均と開発者含む全寄付者の寄付金額の平均を比べて開発者はどれほど寄付をしているのかを調べることによりこのリサーチクエスチョンを検証する．

2.2.4.3. RQ3: リリースとバグ数は全体の寄付額に影響するか？

寄付金が大きく増減した月に着目し，その際何が起きたのかをリリース日のデータを用いて調査する．仮に寄付金が増えた時とリリース日が関係しているのであれば，同じタイミングで寄付を募るキャンペーンを行うと効率よく寄付を集めることができるのではないかと考えられる．また，ソフトウェアをリリースした際に，寄付金が減る原因としては，リリースしたソフトウェアの不具合が多い場合が考えられる．各リリースのバグレポートの数を元に，ソフトウェアの品質を調査し，ソフトウェアの品質が寄付に影響しているのかを調査する．

2.3. 分析結果

表 2　コミッターの寄付額とすべての寄付額

	コミッター	全寄付者
合計	$11,428	$675,601
平均	$45.53	$23.82

表 3　特典がもらえる寄付者とすべての寄付者

	全寄付者	特典がもらえる寄付者
人数	8,836	3,379
金額	$675,601	$533,860

表 4　コミッター数と寄付者数

全コミッター	コミッター中の寄付者	全寄付者
3,549	98	8,836

表 5　寄付者リストの内訳

	全レコード	35 ドル寄付	35 ドル以上寄付
件数	28,358	8,623	11,314

表 6　Eclipse のリリース日とバグレポート数

リリース名	バージョン	メインリリース（バグ数）	SR1（バグ数）	SR2（バグ数）
Neon	4.6	2016 年 6 月 22 日（28）	2016 年 9 月 28 日（0）	2016 年 12 月 21 日予定
Mars	4.5	2015 年 6 月 24 日（38）	2015 年 9 月 22 日（2）	2016 年 2 月 24 日（1）
Luna	4.4	2014 年 6 月 25 日（71）	2015 年 9 月 23 日（2）	2015 年 2 月 25 日（1）
Kepler	4.3	2013 年 6 月 26 日（70）	2013 年 9 月 27 日（3）	2014 年 2 月 28 日（3）
Juno	4.2(3.8)	2012 年 6 月 27 日（107(35)）	2012 年 9 月 28 日（27(2)）	2013 年 3 月 1 日（3(2)）
Indigo	3.7	2011 年 6 月 22 日（77）	2011 年 9 月 23 日（5）	2012 年 2 月 24 日（0）
Helios	3.6	2010 年 6 月 23 日（81）	2010 年 9 月 24 日（6）	2011 年 2 月 25 日（7）
Galileo	3.5	2009 年 6 月 24 日（112）	2009 年 9 月 25 日（2）	2010 年 2 月 26 日（1）
Ganymede	3.4	2008 年 6 月 25 日（132）	2008 年 9 月 24 日（8）	2009 年 2 月 25 日（10）
Europa	3.3	2007 年 6 月 27 日（266）	2007 年 9 月 28 日（9）	2008 年 2 月 29 日（4）
Callisto	3.2	2006 年 6 月 26 日（592）	N/A	N/A

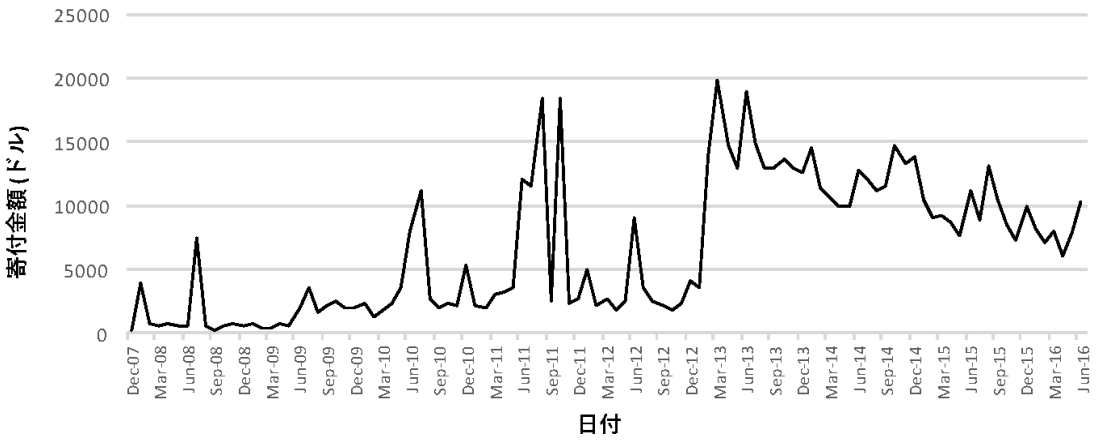


図 3　月ごとの寄付金額

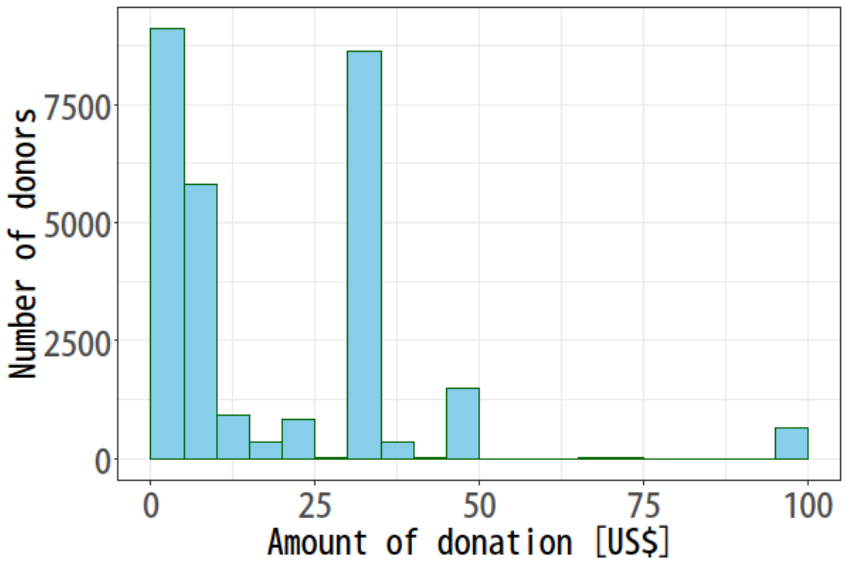


図 4 全体の寄付分布

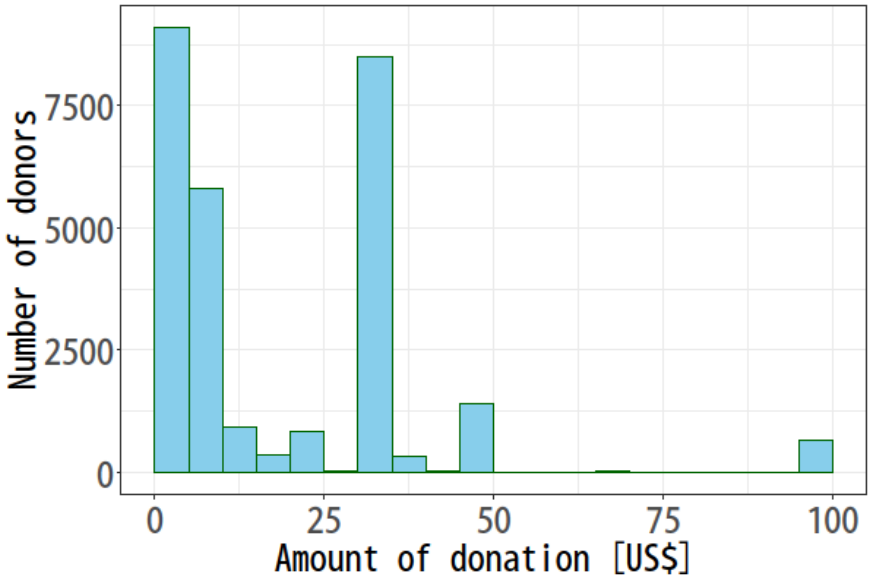


図 5 利用者の寄付分布

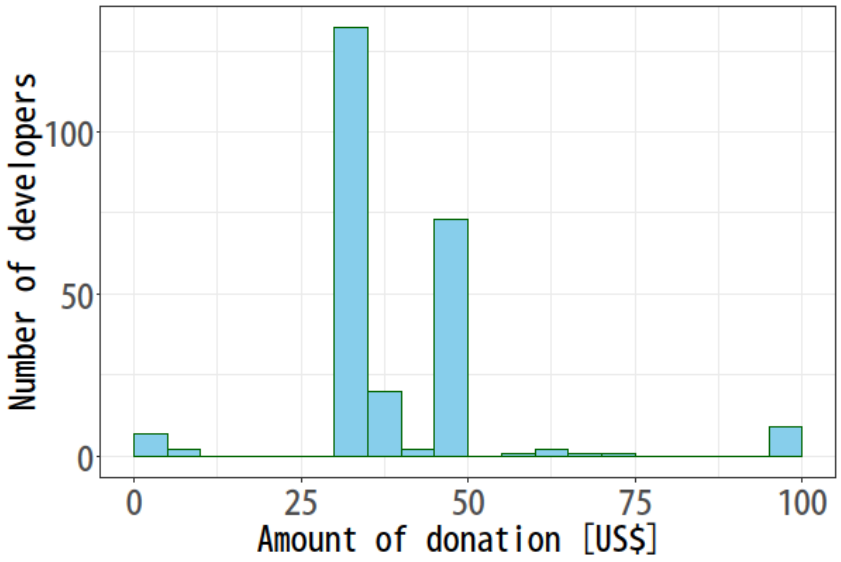


図 6 開発者の寄付分布

本節では 2.2.4 項で述べたリサーチクエスチョンを分析するために行った調査について述べる.

表 3 は一回に 35 ドル以上寄付を行い，特典を受け取ることができた寄付者数とすべての寄付者数である．ただし，匿名希望者や名前欄の空白または無記入であった寄付者は含んでいない. 正確には年間に 35 ドル以上寄付をすると特典を受け取ることができる．しかし，今回の調査で使用した寄付者リストから匿名希望者が年間に 35 ドル以上寄付をしたかどうかを調べることは不可能である. したがって一回に 35 ドル以上の寄付者を便宜上特典を受け取ることができた寄付者と定義する.

表 5 は寄付者リストの全レコード数と寄付者のうち 35 ドル寄付をした件数と 35 ドル以上寄付をした件数である. これも匿名希望者が年間寄付した額を調査することは不可能なので一回に寄付した金額について調査した.

表 4 は Git リポジトリにコミットした人の数と寄付者リストの中にあった名前とコミッター名が一致した数と全寄付者数を計測した表である．なお寄付者のうち 17,963 レコードが匿名希望であり 84 レコードが空白または無記入であった．全寄付者のうち空白，無記入，Anonymous はカウントしていない.

表 2 はコミッターのうち寄付を行った人と全寄付者の一回あたりの平均金

額と合計金額である．

表 6 は Eclipse のリリース日と各バージョンにおける Bugzilla で報告された重大なバグ数についてまとめた表である．なお，重大なバグとは 2.2.2.3 項で述べたように開発者がつけた優先度が P1 または P2 である件数である．ソフトウェアのリリース日は寄付に影響を与えているのではないかと推測される．メインリリースは毎年 6 月末である．また，バージョン 4.5 まで，サービスリリース（SR）は 2 回であり，毎年 9 月末と 2 月末から 3 月初めに行われている．SR では主にバグ修正が行われている．バージョン 4.6 以降は SR を年 3 回行う予定である．バージョン 4.2 をリリースした時はバージョン 3.8 を同時リリースしている．バージョン 4.2 は大型アップデートだったため，バージョン 3.7 をバグ修正したバージョン 3.8 を同時リリースした．

図 3 は寄付者の寄付金額を月ごとに集計し折れ線グラフにしたものである．縦軸が月ごとに集計した寄付金額であり，横軸が集計した月である．このグラフは寄付を開始した 2007 年 12 月 30 日から 2016 年 6 月 30 日までのグラフとなっている．なお，2007 年 12 月は 1 日分だけの寄付であるため寄付金額が少なくなっている．

図 4 は全体の寄付分布である．図 5 は，全体の寄付分布から，図 6 の開発者の寄付を抜いた図であり，利用者全体の寄付分布を示している．図 6 は開発者の寄付だけを抽出して，開発者の寄付分布を示している．

2.4. 考察

本節では 2.3 節の結果を用いてリサーチクエスチョンの検証を行う．

2.4.1. RQ1: 特典は寄付額に影響するか？

表 3から全寄付者のうち特典がもらえる寄付者だけで全寄付額のおよそ 8 割を占めていることがわかる．表 5 より全レコード数 28,358 件のうち 35 ドル以上の寄付は 11,314 件であることからおよそ 4 割の寄付が特典を付与されている．35 ドル以上の寄付のうち 35 ドルを過不足なく寄付した件数は 8,623 件であることから 75%以上が特典の付与される下限の金額を寄付したことになる．図 5 の分布からは，利用者においては，35 ドルの寄付より，少額の寄付のほうが

少し多いが，その次に 35 ドルの寄付が多い，また，図 6 の分布をみると特典を最も享受しそうな開発者は特典がもらえる下限である，35 ドルの寄付が最も多い，以上の結果から特典は寄付額に影響すると結論づける．

Eclipse の寄付の特典で最も効果的だと思われるものは O'Reilly.com での割引であると考えられる．Eclipse は統合開発環境であるため使用者のほぼ全てが開発者である．35 ドルの寄付で書籍の割引購入が可能となる特典は寄付への促進に非常に有効であると考えられる．ただし，このような特典をつけることができるのは O'Reilly Media が Eclipse Foundation のスポンサーであるからである．このようにスポンサーをうまく活用して特典をつけるというのは非常に効果的であるといえる．スポンサーがソフトウェア利用者に魅力的なサービスを提供しているのであれば，サービス利用の割引などを提供してもらうことで効果的に寄付を集めることができると考えられる．

2.4.2. RQ2: 開発者からの寄付にはどのような特徴があるか？

表 4 より，全コミッターが 3,549 人，コミッター中の寄付者が 98 人であることから 2.7%の開発者が寄付を行っていることがわかる．全寄付者が 8,836 人なので全寄付者のうち開発者は 1%であることが判明した．この状況から寄付を行う開発者はそれほど多いわけではない．一方で，表 2 より，全寄付者の寄付平均が 23.82 ドルであるのに対し，コミッターの寄付平均は 45.53 ドルであり，開発者は約 2 倍近い額の寄付を行っていることがわかった．図 5 図 6 の分布を見ても，開発者は利用者に比べて，寄付の金額が多い．以上の結果から，我々は自身が開発したプロジェクトに寄付する開発者は多いとは言えないが，一般の寄付者に比べ，多くの額を寄付する特徴があると結論づける．

もちろん，匿名希望者らが全体の 63%存在する中でそれを除外して調査を行っているため，実際には多く，または少ない割合の開発者だけが寄付を行っているのかもしれない．しかしながら，Eclipse の寄付においては開発者が他の人より多くの額を寄付している傾向にある．開発者の寄付の単価は高いので，開発者特化の寄付活動を行うことでより多くの寄付金を集めることができる可能性があると考えられる．したがって，現在 Eclipse Foundation が行っている bugzilla 上で判別できるバッジなどは効果的な特典である可能性が高い．また，

Eclipse会議の割引は常に付与されているわけではないこと(2.2.3項)を考慮すると，50 ドル以上寄付すると手に入る Eclipse 会議の割引は効果的であると考えられる．そのため，OSS の運営団体が寄付金を会議等イベントに使用することは効果的であると考えられる．

2.4.3. RQ3: リリース日とバグ数は全体の寄付額に影響するか？

図 3 から，毎年 6 月に行われているメインリリース時（表 6）には寄付金額が前月より大幅に増えていることがわかる．よってメインリリースは影響が最も大きいのではないかと推測できる．ソフトウェアのリリース時には同時に寄付の呼び掛けを行ったり，寄付の Web ページをリニューアルするなどの工夫をすれば効率よく寄付が集まるのではないかと考えられる．

バージョン 4.2 がリリースされた直後には寄付金額が増えているが大型アップデートを行った時期であるにも関わらず通常のメインアップデートと変わらない増加となっている．これはソフトウェアの品質が関わっているのではないかと推測できる．

実際に，表 6 に示した重大なバグの数を見てみると，バージョン 3.7 まではバージョンを重ねるたびに重大なバグの数は減っている．バージョン 3.2 の時には 592 個あったがバージョン 3.7 の時には 77 個になっている．しかし，バージョン 4.2 がリリースされた時には重大なバグは 107 個でありバージョン 3.7 の時よりも多くの重大なバグが報告されている．

バージョン 4.2 がリリースされた後の SR であるバージョン 4.2 SR2 がリリースされた直後に寄付金額がかなり増加している．これは大型アップデートに伴い重大なバグが多くあったが修正されたため評価が上がったのではないかと考えられる．バージョン 4.2 SR1 のリリース時には寄付金額はそれほど増加はしていない．以前の SR の重大なバグレポート数は 0 個から 1 個程度あったがバージョン 4.2 SR1 の時には重大なバグレポート数は 27 個であった．実際にバージョン 4.2 SR2 の重大なバグレポート数は 3 個であり重大なバグの多くは修正されたといえる．このことからアップデート時のソフトウェアの品質は寄付に影響を与えていると考えられる．

寄付を募り始めた最初（2008 年 1 月）も寄付金額が多い．OSS プロジェクトが最初に寄付を募る時は，多額の寄付が集まる可能性が高いので，専用の Web

ページを作る，SNS 等で大々的に広告を打つことは効果的ではないかと推測される．

これらから全体の寄付額はソフトウェアのメインリリース日や大半のバグフィックスが完了したサービスリリース時，寄付の受付をスタートした時に影響されるということがわかった．

2.5. 関連研究

OSS 開発への各貢献方法についての分析には以下のものがある．Davies らはバグレポートに関する分析を行っており，開発者が望む情報と実際に提供される情報は一致しないことを明らかにした[11]，Parnin らは API ドキュメントに関する分析を行っており，ソーシャルメディアは API ドキュメントとしてカバレッジの高いレベルを提供していることを明らかにした[30]．また，McIntosh らはコードレビューに関する分析を行っており，不十分なコードレビューを行ったコードはソフトウェア品質に悪影響を及ぼすということを明らかにした[25]．

West らは複数の企業が OSS を金銭的支援するそれぞれの動機についてまとめている[44]．OSS 運営団体は資金を得るため商用関与を行うことがある．Zhou らは一つのプロジェクトだけ管理している OSS 運営団体においては商用的な関与が開発者の外部流出につながらないが，複数プロジェクトを管理している OSS 運営団体において商用的な関与は開発者の外部流出につながると主張している[49]．個人からの寄付金は商用的な関与が発生しないため，個人からの寄付を募ることは OSS 運営団体全体の目標である．Franck らは OSS における寄付者と開発者の関係性を仮定し，OSS の社会構造を分析している[13]，Krishnamurthy らは開発者の報酬の受け入れについて研究しており，OSS 開発者に金銭的な報酬を受け入れる意思があると開発者のやる気を起こさせると主張している[22]．

Krishnamurthy らは OSS 開発プラットフォームである SourceForge.net を用いた OSS への寄付活動を分析している[23]．彼らは OSS コミュニティのメンバーは関連する OSS プラットフォームに寄付を行うということ，プラットフォームとの関係の強さが寄付金額に影響を与えていると主張している．本章の研究で

は，比較的大きな OSS を対象としている点，寄付の特典について調査している点，リリース時やソフトウェアの品質に着目している点が異なる．

2.6. まとめ

本章では著名な OSS である Eclipse における寄付についての分析を行った．その結果，寄付への動機付けとして特典は有効であること，自身が開発したプロジェクトに寄付する開発者はおよそ 3%であり，開発者の寄付金額の平均は全員の寄付金額平均の 2 倍近くあること，寄付金の増加にはソフトウェアのメインリリース日や大半のバグフィックスが完了したサービスリリース時であることがわかった．

今後の課題は 2016 年 9 月に行われた寄付キャンペーンによってどれほど寄付が集まったのかを調べることと，Eclipse に寄付を行った開発者にアンケート調査を実施し，なぜ寄付を行ったのかを調査することである．

3. Eclipse 寄付バッジによるバグレポート応答時間の影響分析

3.1. まえがき

寄付はオープンソースソフトウェア（OSS）プロジェクトで重要な役割を果たす．OSS プロジェクトである LibreOffice は，2016 年に 3 年間で 20 万件の寄付を受けたと報告した．そして，OSS コミュニティの多様なエコシステムと個人の寄付者に頼ることにより，大規模な OSS プロジェクトにはただ一つの大規模な企業スポンサーは必要ないことを強調した[41]．このことから、持続可能な OSS プロジェクトを維持するためには効果的な寄付プログラムの管理が重要であることがわかる．

OSS における寄付は重要ではあるが，OSS プロジェクトへの寄付についてはほとんど研究されていない．寄付に影響を及ぼす原因についての研究は，SourceForge のデータを用いた研究[22]があり，寄付の意思決定には，OSS プラットフォームとのコミットメント関係の影響を受けることが報告されている．著者らの以前の研究[27]では，少数ではあるがコミッターは他のグループよりも，一人当たりの寄付額が高いことがわかっている．しかし，これまでの研究では，寄付者への寄付の実用的な効果は，明らかになっていない．

Eclipse は寄付プログラム「Friends of Eclipse」を 2007 年 12 月に開始した．2014 年 11 月より，この寄付の特典として，バグ管理システム（Bugzilla）にバッジが表示され，寄付者がバッジにより区別できるようになっている．現在，35 ドル以上の寄付をした寄付者は，Friend of Eclipse の資格を 1 年間取得することができ，Bugzilla で表示される Friend of Eclipse Badge（以下寄付バッジ）を利用できる．しかし，寄付バッジが寄付者にどのようなメリットをもたらすのかは明らかではない．

本章では，統計的因果推論に基づいて，寄付バッジステータスを持つバグレポート提出者がバグレポートを提出した時，寄付バッジを持つことで応答時間（最初にコメントがつくまでの時間）に影響を与えるかを明らかにする．分析の結果，寄付バッジは，中央値において，約二時間応答時間を短縮する効果が

あることが明らかとなった．さらに，Eclipse コミュニティを対象とする調査では，寄付バッジはバグレポートの対応の優先度に大きな影響を与えないことがわかった．以上の結果から，Eclipse の寄付バッジは，明示的な優先事項ではないが，シグナリング理論のシグナリングとして働き，寄付者に好ましい効用があるということを考察した．本章の分析の一部は IEEE Software に採録されたもの[26]である．本章は IEEE Software の分析を拡張し，さらに Eclipse コミュニティに対して実施したサーベイをくわえ考察した．

3.2. 統計的因果推論

本節では，寄付バッジの効果の分析に使用する，統計的因果推論とその条件について述べる．統計的因果推論は，医療や，政策分野など幅広い分野で利用されている．

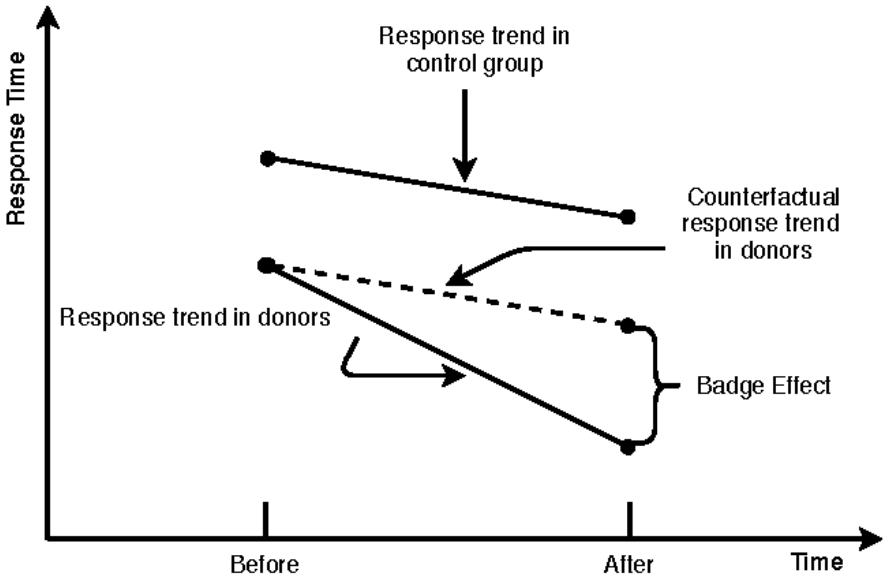


図 7 DID モデルの因果推論例（response time vs. before and after donation badge introduction）

3.2.1. 統計的因果推論

統計的因果推論とは，ある要因が結果に及ぼす影響の度合いを，統計データに基づいて判断するための方法論である[1].

調べたい要因が，ある結果に影響を与えていることを明らかにしたい場合，結果に影響を与える他の要因（交絡因子）を考慮した方法で分析しなければ，因果関係を明らかにすることはできない．調べたい要因のみを完全に取り除く制御された測定が可能であるなら交絡因子を完全に取り除くことができるため因果関係を明らかにすることができるが，人や生物が関わる問題の場合，このような制御された測定を行うことは不可能である．

そのため，統計的に交絡因子を排除し，因果関係を検証する方法として，ランダム化比較試験という方法があり，実験と呼ばれている．対象をランダムに 2 つの群に分け，一方だけに評価しようとしている治療や，政策などの介入を行い，両群を比較することにより介入の因果を推論する．ただし，この手法は実験コストが多くかかり，測定されている既存データ（観察データ）には適用できない．

一方，擬似実験という，観察データから因果関係を導き出す方法がある．擬似実験では，観察データから実験に類似する状況を作成することで，因果関係を示す．本章では，観察データ（過去のバグレポート，寄付者リスト，開発履歴）を使用して，擬似実験を行うことにより寄付バッジの効果を分析する．

3.2.2. 差分の差分分析

本章では，差分の差分分析（Difference in Differences）を観察データに適用することにより寄付バッジの効果を明らかにする．差分の差分分析とは擬似実験の手法の一つであり，トレンドの影響を除外することにより因果推論を行う手法である．図 7 に差分の差分分析のモデルを示す．図 7 の control group は寄付者ではないが，寄付者のバグレポート提出者と類似するバグレポート応答時間のトレンドを持つ群である．一方，donors が寄付者のバグレポート提出者のバグレポート応答時間の群である．y 軸がバグレポート応答時間を示し，x 軸が寄付バッジ導入前と後を示す．実線が各グループの実際のバグレポート応答時間を示す．DID モデル（差分の差分分析で使用するモデル）では，control group

の実際の測定値から，寄付バッジが導入されなかった場合の donors の測定値を仮定する（破線）．寄付バッジが導入されなかった場合の donors のバグレポート応答時間を仮定した値から実際の寄付バッジ導入後の測定値の差（Badge Effect）を求めることで，寄付バッジの効果を推定する．本手法を適用するためにはある条件を満たす必要がある．本稿では 3.2.5 節に条件に関する議論を記述している．

3.2.3. 傾向スコアマッチング

本章では，差分の差分分析を行うための条件を満たすため（3.2.5 項），擬似実験の手法の一つである，傾向スコアマッチングを利用している．傾向スコアマッチングはマッチング法の一つである．マッチング法とは，結果に影響する条件（共変量）が類似するペアを作り，条件が同じグループを作成することで，比較可能な状態にする手法である．傾向スコアマッチングは共変量が多くある場合に利用される手法である．複数の共変量を一つの得点とした傾向スコアを使用し，割り付けられる確率が同じ（傾向スコア）ようなペアを作成する手法である．本手法によって，図 7 のように control group と donors を作成した．

3.2.4. 分位点回帰分析

差分の差分分析を行う際，通常は線形回帰分析を用いて分析を行うが，本分析では線形回帰分析の代わりに分位点回帰分析を行う．分位点回帰分析で差分の差分法を行うことを Quantile Difference in Differences（QDID）とよぶ．線形回帰分析は平均値に着目する場合に用いるのに対し，分位点回帰分析は分布に着目する場合に用いる．ソフトウェア開発データの分析には分布に偏りがあるため，中央値に着目して分析する場合が多い．本分析においても，バグレポートの応答時間は分散が大きいため，分位点回帰分析を行った方が，適切であると考えられる．分位点回帰分析の場合でも線形回帰分析と同じように，目的変数と説明変数を指定し，分析を行う．今回の場合は，目的変数をバグレポート提出時間からコメントが得られるまでの時間とする．表 7 に示す変数を説明変数とする．特に，差分の差分分析では（Donor, Period, Badge）の変数に着目し

て分析を行う．偏回帰係数 Badge が寄付バッジの効果量を示す．残りの説明変数はバグレポートの応答時間を予測する際に，調整のために用いた変数である．

表 7　QDID で使用する説明変数

Metric	説明
Donor	寄付者であるか
Period	提出時間が寄付バッジの導入後か
Badge	寄付バッジを持つか
Enhancement	バグ深刻度（Severity）が Enhancement か
Windows	報告対象が Windows OS か
Linux	報告対象が Linux OS か
MacOS	報告対象が Mac OS か
Component	属する component の応答日数の中央値
Community	属する component の応答日数の中央値
Time	寄付バッジ導入日を基準とした，提出時期
Relationship	最初の応答者とバグレポータのペアの出現回数

3.2.5.　因果推論を行う上での分析条件

SUTVA 条件: 因果効果を推定する場合，（a）個体を特徴付ける潜在的な結果が他の個体の受ける処置に依存しないことかつ，（b）処置内容を明確にすること以上２条件の仮定が置かれている．これを SUTVA（stable unit treatment value）条件と呼ぶ．本分析においては，（a）ある寄付バッジ持つバグレポータの提出したバグレポートに対する応答時間は他のバグレポータが寄付バッジを持つことにより，干渉を受けるかどうかと（b）が論点となる．

寄付バッジを持つバグレポータの割合が多い場合，バグレポータは他のバグレポータの干渉を受けると考えられる．ただし，分析期間においては，Eclipse のバグ貢献者は 4,872 人に対し，寄付バッジを持つバグレポータは 55 人（2016 年 12 月）[27]である．寄付バッジを持つバグレポータは少ないため，個人の寄付バッジを持つバグレポータへの干渉はごくわずかであると考えられる．その

ため，（a）の条件は満たしている．（b）についてはバグレポート提出時以前，一年間において 35 ドル以上を寄付しているかどうかを条件としており，処置内容は明確である．

差分の差分分析の条件: 寄付バッジの因果効果を推定するため，差分の差分分析を行う．差分の差分分析によって，因果効果を正しく推定するためには，平行トレンド仮定と共通ショック仮定を満たす必要がある．平行トレンド仮定を満たすには，寄付バッジをもつバグレポータが提出したバグレポートに対する応答時間群と寄付バッジを持たないバグレポータが提出したバグレポートの応答時間群において，もし，Bugzilla 上で寄付バッジを表示するという実装がされなかった場合に両群は平行したトレンドとなる必要がある．共通ショック仮定を満たすには，寄付バッジの実装前の測定と寄付バッジ導入後の測定との間に結果に影響を与える可能性のある別の要素が起きていない，もしくは起きている場合でも両群に対し，同じく作用している必要がある．

本分析では，平行トレンド仮定を満たすため２件の工夫をしている．1 件目は，仮想的に寄付バッジの有無以外は同じ経験を持つバグレポータを傾向スコアマッチングにより抽出してそのバグレポートを比較している．2 件目は，バグレポートを比較する際，応答時間に影響があると思われる要素を分位点回帰モデルに追加して，分析を行っている．以上の工夫により，両群では寄付バッジの有無以外には他の要素は同じである状態であると考えられるため，統計的に平行トレンド仮定を満たしていると考えられる．

また，共通ショック仮定においては，著者の知る限りでは寄付バッジの有無以外に結果を与える要素は存在しない．また，両群共，同じ集計期間，同じ Eclipse のバグレポートを計測対象としている．そのため，仮に結果に影響を与えるよう要素が存在していたとしても，両群共に同じく作用していると考えられるため，共通ショック仮定においても満たしていると考えられる．

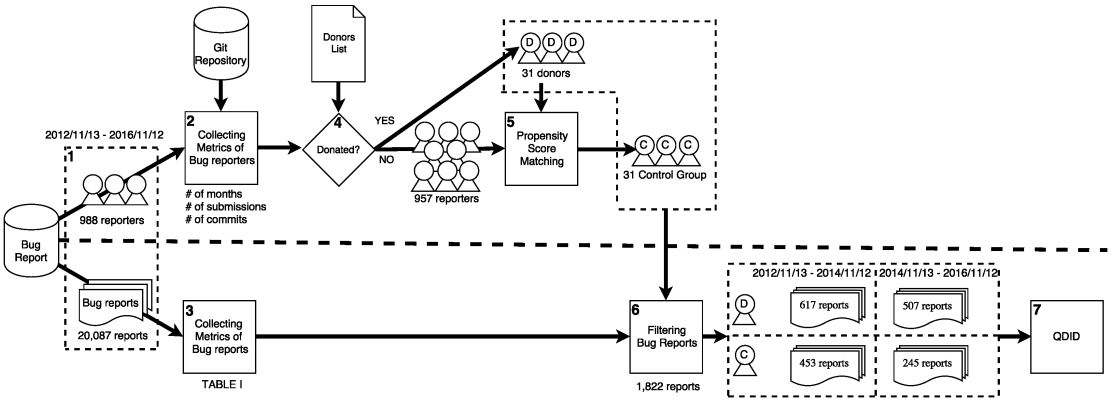


図 8　データ収集と統計的因果推論の分析プロセス

3.3.　分析

3.3.1.　Friend of Eclipse と寄付バッジ

Eclipse は，2017 年 12 月に「Friend of Eclipse」という寄付プログラムを開始した．Friend of Eclipse は Linux の寄付プログラム[6]に触発されて，寄付者に多くの特典を提供した．特に，2014 年 10 月のピーク時には，T シャツや，技術書の割引などを提供していた．しかし，寄付者は特典に関心がないという Eclipse が実施したアンケート調査[8]の結果から，現在は，寄付の特典は寄付バッジと Friend of Eclipse のロゴが使用できる権利だけが提供されている．

Eclipse では，年間 35 ドル以上寄付をすることで寄付バッジが寄付者に提供される．寄付バッジとは，Eclipse のバグ管理システム（Bugzilla）上のバグレポートに氏名と共に表示されるものであり，バグレポート利用者は，バグレポート貢献者から寄付者を認識することができる．寄付バッジの提供に関する議論は Bugzilla 上で行われ，寄付バッジは 2016 年 11 月 13 日から提供が始まった[7]．

3.3.2.　分析手法

分析概要について図 8 に示す．図 8 に示すように，本分析は control group と donors のマッチング（図上部）とバグレポートの差分の差分分析（図下部）

で構成されている．以下，各工程の説明を行う．

Step 1: バグレポートの収集．寄付バッジが最初に導入された日は 2014 年 11 月 13 日[7]である．差分の差分分析を行うためには，寄付バッジ導入前後のデータが必要である．本分析では，2012 年 11 月 13 日から 2016 年 11 月 12 日までの 4 年間のバグレポートを分析期間とした．前後で同じ集団で比較を行うため，寄付バッジ導入の前後の両期間で少なくとも 1 回以上のバグレポートを提出したバグレポータのバグレポートのみを対象とした．バグレポートの応答時間を適切に測るため，バグレポータとバグレポートに割り当てられた人物（assignee）が同じ場合はそのバグレポートを除外した．また，見落とされていた，または意図的にレスポンスを延期したバグレポート（数週間から 1 年以上経ってからレスポンスがあるバグレポートが存在する）を除外するために最初のレスポンスが 3 日以内に来なかったバグレポートも除外した．以上の操作からバグレポートは全体の 60%となった．

Step 2: バグレポータのメトリクス．バグレポータのメトリクスは，傾向スコアマッチング（Step 5）で使用する．バグレポート報告経験月数，導入前と後それぞれのバグレポート投稿回数，コミット回数をバグレポータの計測メトリクスとした．

Step 3: バグレポートのメトリクス．バグレポートのメトリクス（表 7 に示す）は QDID で使用される．

差分の差分分析で必要な（Donor, Period, Badge）に加えて，開発者の応答時間に影響を及ぼす可能性のある要因について検討した．以下に検討したメトリクスについて説明する．バグ深刻度（Severity）の高いバグは速く修正され[47]，バグが発見されたオペレーティングシステム（OS）はバグ修正時間に影響を与える[46]と報告されている．そのため，7 レベルの Severity と OS に関連するメトリクスを検討した．Component は，バグレポートのコンポーネントの応答日数の中央値を測定した．パイロットスタディによって，開発者の応答時間はコンポーネントによって異なることがわかった．いくつかのコンポーネントでは応答時間は中央値で半日以内であるが，応答時間が中央値で 10 日を超えるものも存在する．Community は，コンポーネントのコミュニティ内の貢献者のサイズを示す．Time[48]は，寄付バッジの導入日を基準とした，提出時期を示す．時間の影響を考慮するためこのメトリクスを追加した．Relationship は，レポー

タと応答した開発者との間の社会的および個人的な関係を考慮するため，用意した． Ortu らは，感情的なコメントがバグ修正に影響を与える可能性があることを報告した[29]．ただし，既存のセンチメント分析ツールはソフトウェア工学のドメインには必ずしも適用されないと報告されているため，感情的な要素は追加しなかった[19].上記のメトリクスから，Akaike 情報基準（AIC）に基づいてモデルに使用するメトリクスを選択した．表 8 に，最小 AIC 値のメトリクスを示す．応答時間の計測には，バグレポートを提出してから，バグレポータ以外の開発者からの最初のコメントが得られるまでの間の時間を使用している．

Step 4: 寄付者の分類．Eclipse の寄付サイトには，寄付者の名前，寄付をした日，寄付額がリストとして公開されている．この情報から，各寄付者のバグレポートにおける寄付バッジの出現期間を特定することができる．ただし，バグレポータの名前が同じかつ異なる電子メールアドレスのバグレポートが存在した時,バグレポータと寄付者の名前を一意に関連付けることができないため,そのバグレポートは除外した．その結果，31 人の寄付者が特定された．

Step 5: 傾向スコアマッチング．傾向スコアマッチングでは，よく知られている最近傍探索アルゴリズムを使用した．寄付をしていない 957 人のバグレポータから,31 人の donors とマッチングされた 31 人のバグレポータが control group として選択された．

Step 6: バグレポートのフィルタリング．図 8 上部で選択されたバグレポータの提出したバグレポートのみを抽出する．図 8 に示すように、これらのバグレポートは，2 つの期間と 2 つのグループで構成される．

Step 7: Quantile Difference in Differences（QDID）．本章では，比較的早く対応されたバグレポートに焦点を当てるため，時間単位で影響を分析した．表 7 に示す収集されたバグレポートのメトリクスを使用して、QDID を行う．QDID では, 0.1 分位点刻みの Badge の偏回帰係数の結果と 0.5 分位点に着目して結果を報告する．

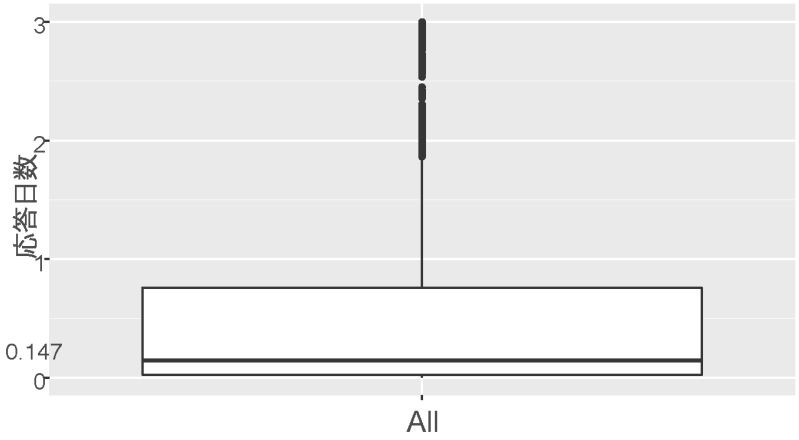


図 9　分析対象のバグレポートの応答日数

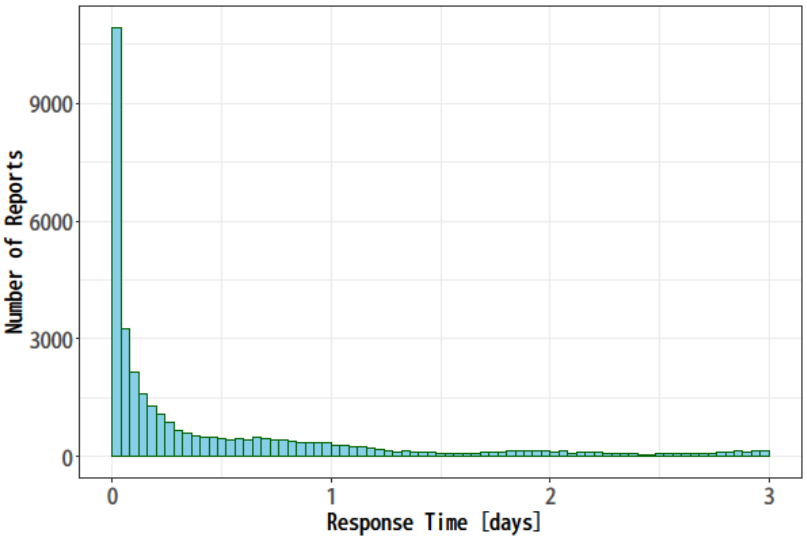


図 10　バグレポート全体の応答日数

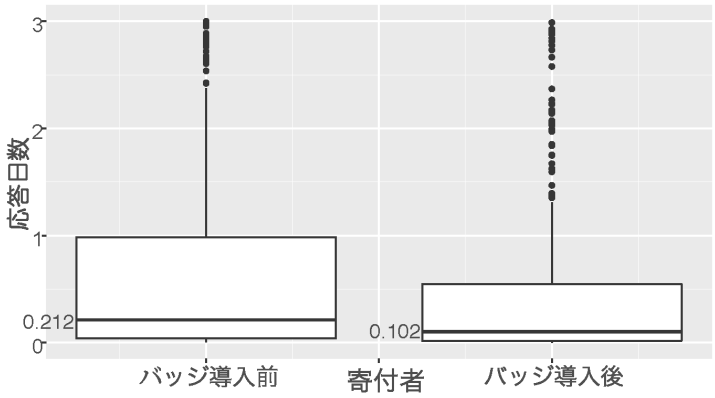


図 11 donors のバグレポートの応答日数

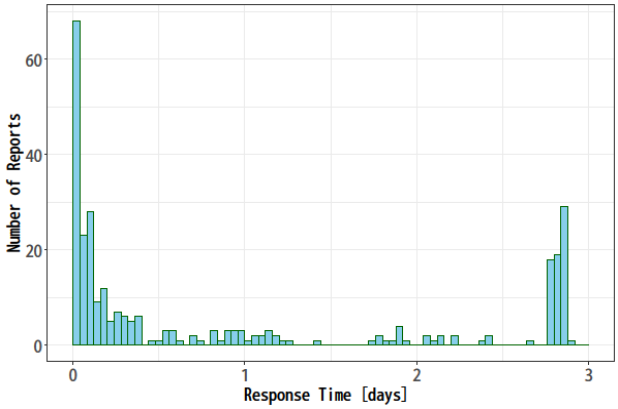


図 12 donors のバグレポートの応答日数（バッジ導入前）

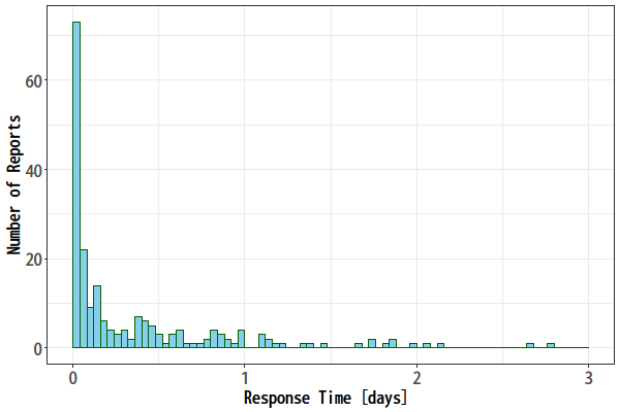


図 13 donors のバグレポートの応答日数（バッジ導入後）

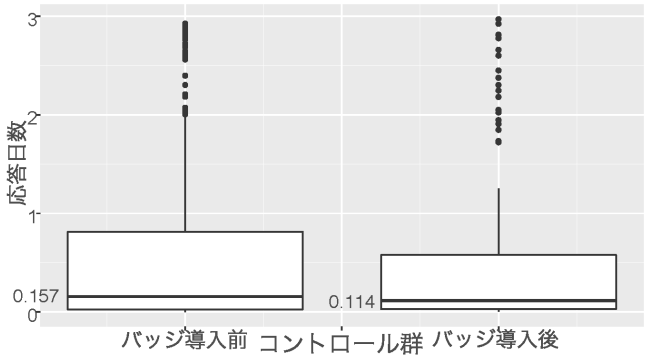


図　14　control group の応答日数

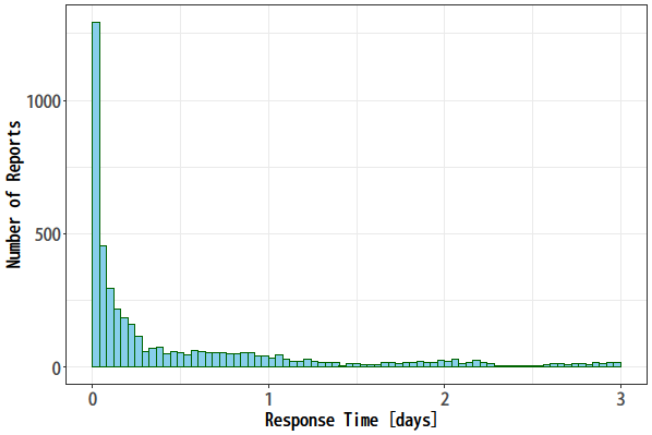


図　15　donors 以外のバグレポートの応答日数（バッジ導入前）

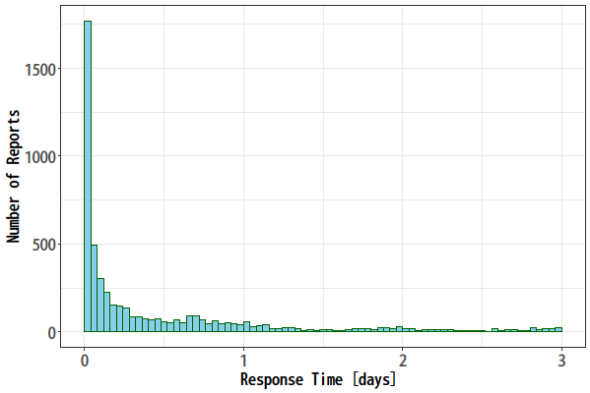


図　16　donors 以外のバグレポートの応答日数（バッジ導入後）

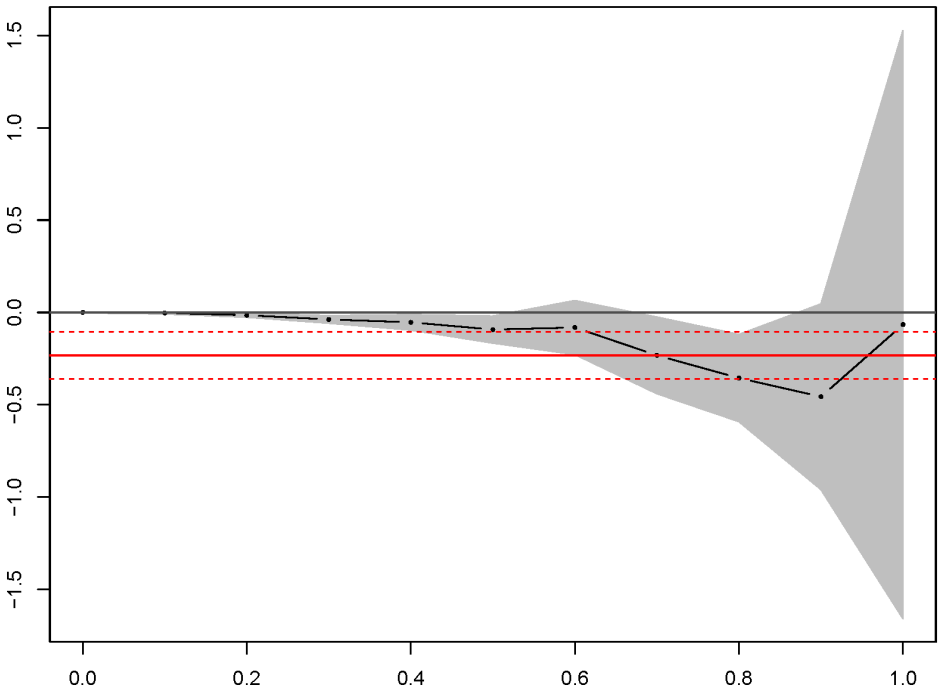


図 17　偏回帰係数 Badge の各分位点の結果

表 8　QDID の 0.5 分位点の結果

Metric	Coeffs (Errors)	t value	Pr (>|t|)
(Intercept)	2.237 (1.120)	1.998	0.046
Donor	2.259 (0.834)	2.708	0.007
Period	-1.598 (1.227)	-1.302	0.193
Badge	-2.219 (1.061)	-2.092	0.037
Enhancement	0.668 (0.815)	0.820	0.412
Windows	1.119 (0.692)	1.617	0.106
Linux	0.675 (0.949)	0.711	0.477
MacOS	0.793 (0.907)	0.875	0.382
Component	0.317 (0.198)	1.603	0.109
Community	0.000 (0.001)	0.105	0.917
Time	0.152 (0.117)	1.303	0.193
Relationship	0.005 (0.005)	0.894	0.371

AIC = 4,025, Pseudo R^2 = 0.011

3.4.　結果

3.4.1.　寄付バッジの効果

分析対象のバグレポートの応答日数の分布を図 9 に示す．分析対象全体の応答時間の中央値は 3 時間 32 分であった．また，バグレポート全体のヒストグラムを図 10 に示す．donors のバグレポートの応答日数の分布を図 11 に示す．donors の寄付バッジ導入前の応答時間の中央値は 5 時間 5 分，導入後の応答時間の中央値は 2 時間 26 分であった．また，寄付バッジ導入前のヒストグラムを図 12 に，寄付バッジ導入後のヒストグラムを図 13 に示す．分布を見ても，傾向として，バッジ導入後の応答日数が小さくなっている．control group の応答日数の分布を図 14 に示す．寄付バッジ導入前のヒストグラムを図 15 に示す．また，寄付バッジ導入後のヒストグラムを図 16 に示す．ヒストグラムのほうは，マッチング前のデータの分布を示している．control group の寄付バッ

ジ導入前の応答時間の中央値は 3 時間 46 分，導入後の応答時間の中央値は 2 時間 44 分であった．図から，傾向として，バッジ導入後の応答時間は短くなっている．以上の結果から，寄付者も control group も寄付バッジ導入後の応答時間のトレンドは下降傾向にあることがわかった．また，donors のバッジ導入前の応答時間は他のグループと比べ長いことがわかった．QDID の結果を図 17 と表 8 に示す．図 17 は，0.1 分位点ごとに分位点回帰分析の Badge の結果をプロットしたものである．灰色の範囲は誤差範囲を示している．全ての分位点において，Badge の偏回帰係数はマイナスである．ただし，分位点が大きくなるほど誤差範囲が大きくなる．特に，0.8 分位点が一番 Badge の効果が大きい．表 8 は 0.5 分位点における分位点回帰分析の結果である．Badge の偏回帰係数の値は-2.219（時間）であり，統計的に有意な結果となった．以上の結果から，全体のバグレポートの中央値 3 時間 32 分に対し，寄付バッジが導入されたことにより，寄付バッジを持ったバグレポータの提出したバグレポートは中央の分布で約二時間最初の応答時間が短縮されるということがわかった．

3.5. 考察

寄付バッジの分析結果の理由を探り，フォローアップするため，Eclipse コミュニティに対し，サーベイを行なった．以降，サーベイ内容を記述する．

3.5.1. サーベイ

どのようなバグレポートを優先的に扱うのかについて，Eclipse コミュニティに対し，アンケート調査を行った．

調査方法: Google フォームで，調査フォームを作成し，Eclipse の Marketing Specialist に調査フォームの広報を依頼した．2018 年の 4 月 12 日と 6 月 7 日の 2 回に渡り，Eclipse の保有する公式 Web サイト，ソーシャルメディア（Twitter, Facebook, Linkedin）上で，調査フォームのリンクをポストしてもらった．さらに, 7 月 3 日に Eclipse が毎月発行する Eclipse Newsletter のイントロダクションで取り上げてもらった．集計期間は 2018 年 4 月 12 日から 7 月 6 日である．回答者は 27 人．うち，有効回答 23 人であった．

調査項目: 調査項目は以下のとおりである．

- 回答者の属性

- バグレポートを読むことに影響する要因
- バグレポートにコメントをすることに影響する要因
- バグを優先的に直すことに影響する要因（バグを修正したことがある人のみ回答）
- 寄付バッジの認知度
- 寄付バッジに対する意識

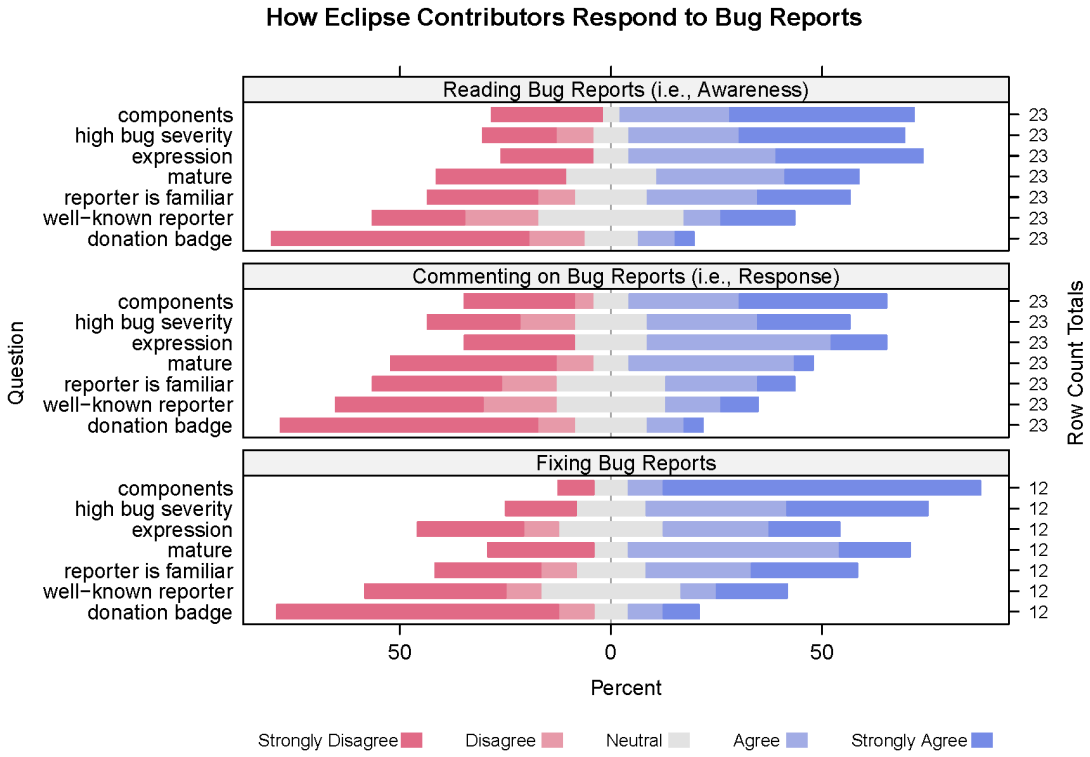


図 18　バグレポートの扱いの優先順位

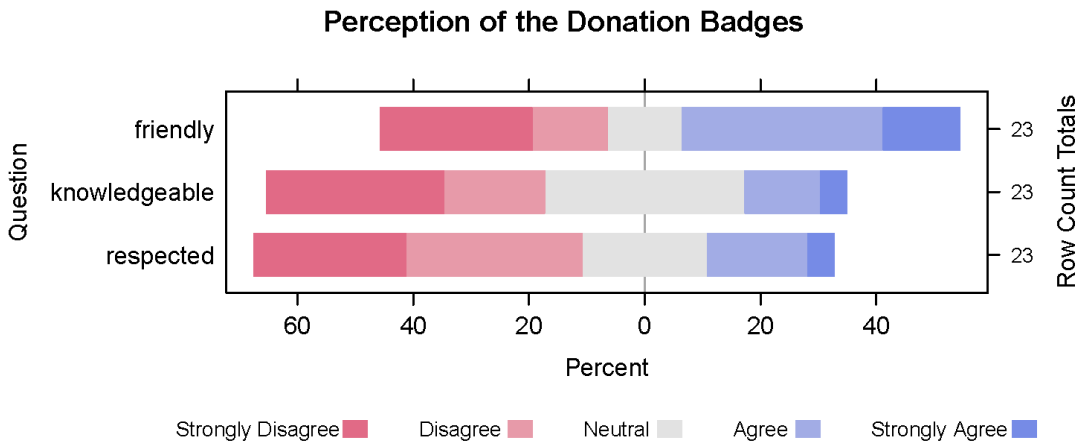


図 19　寄付バッジの感じ方

3.5.2.　サーベイの結果

寄付者の属性: 回答者は 23 人であり，そのうち 12 人がバグを直した経験があると答えた．また，End-user が一番多く，次に多い属性はバグレポート貢献者であり，両方ともその割合は 60%以上であった（複数回答）．寄付をした経験がある回答者は 6 人で，現在，寄付バッジを利用する権利を持っている人は 2 人であった．

バグレポートの扱いの優先順位: 図 18 にバグレポートに応答することに影響をする要因についての結果を示す．バグレポートを読むこと，コメントをすること，直すことに影響する要因について質問した．どの質問の場合も，よく知られたバグレポータによって提出されたバグレポート（well-known reporter）と寄付バッジがついたバグレポート（donation badge）は肯定意見より，否定意見の方が多かった．自分がよく知っているコンポーネントに関するバグレポート（components）とバグ深刻度が高いバグレポート（high bug severity）はどの質問においても肯定意見の方が多く，他の要因と比べても肯定の結果が上位になっている．肯定的な感情を伴う簡潔で丁寧で完璧な方法で書かれたバグレポート（expression）はどの質問においても肯定意見の方が多いが，バグを直すことに関しては肯定意見の割合は小さくなっている．成熟して，追加のフォローアップが必要のないバグレポート（mature）と自分がよく知っているバグレポータによって提出されたバグレポート（reporter is familiar）は，読むこと，バ

グを直すことに関しては肯定意見が多かったが，コメントをする場合には否定意見が多い．

寄付バッジの認識: 寄付バッジについて知っていた人は 23 人中 10 人で，半数もいなかった．寄付バッジについて情報を提示した上で，寄付バッジを知らなかった人の中で，寄付バッジに興味を持ったのは 3 人で，興味を持たなかったのは 13 人であった．

寄付バッジの感じ方: 寄付バッジを見たときに感じることに関する質問の結果を図 19 に示す．寄付バッジに対して，friendly メンバーであると認識し，コミュニティに暖かく歓迎されるべき（friendly）という質問に対しては，肯定意見のほうが否定意見よりも多かったがほぼ同じであった．また，寄付バッジに対して，knowledgeable メンバーであると認識し，初心者ではなく，バグレポートの品質が良い（knowledgeable）と respected メンバーであると認識し，高く優先すべきである（respected）という質問に関しては否定意見のほうが多い結果となった．

まとめ: 寄付バッジの効果の分析では，寄付バッジを持つことによって，応答時間を短くなるという結果が得られた．しかし，サーベイの結果では，Eclipse のコミュニティにおいて，寄付バッジの要素ではバグレポートを読む，コメントする，バグを直すことに関して優先的には扱おうとは思っていないことが明らかとなった．他の人に関する項目のよく知っているバグレポータや，よく知られたバグレポータに関しても同様に優先度の割合が低い結果となっている．一方，バグの内容に関する項目であるよく知ったコンポーネントや，バグ深刻度が高いバグレポートに関しては優先的に扱おうと思われていることがわかった．

これらの結果から，寄付バッジを優先的に扱おうとは思ってはいないが，無意識に寄付バッジを持つバグレポートに対し，少しは優先的に応答しているということがわかった．この無意識に働いている効果はシグナリング効果であると考えられる．シグナリングとは，私的情報を開示する行動である．Tsay らはプロジェクトマネージャは Pull request の受け入れの際に開発者のシグナリング情報を利用していると報告している[40]．この寄付バッジを持つというシグナルは，長期的にプロジェクトに関わり貢献したいという意思を示唆するものであると考えられる．

3.6. まとめ

本章は QDID によって応答時間が寄付バッジの影響を受けるのかを分析した．その結果バグレポート全体の応答時間の中央値 3.5 時間に対し，寄付バッジを持つことで，中央値で約二時間短くなることがわかった．さらに，どのようなバグレポートを優先的に扱うのかについて，Eclipse のコミュニティに対し，アンケート調査を行なった．その結果，寄付バッジを持つバグレポートは，優先的には扱おうとは思ってはいないことがわかった．優先的にバグレポートを扱っていないのに関わらず，実際のバグレポートの分析で寄付バッジを持つことで，バグレポートの応答時間を短くなっているという結果がでたのは，シグナリング効果で説明できると考えられる．寄付バッジをもつというシグナルは，長期的にプロジェクトに関わり貢献したいという意思を示唆するものと考えられる．

4. ソフトウェア開発者の Web 検索戦略の分析

4.1. まえがき

World Wide Web 上には，プログラミングに有用な情報が多数存在する．例えば，プログラミング言語の公式リファレンスの多くは Web 上に公開されており，プログラミングに関する Q&A サイトも存在する．ソフトウェア開発者は多くの場合，これら開発効率を高める有用な情報を発見するために，Google に代表される Web 検索エンジンを活用している[5].

ただし，（「検索フレーズを頻繁に変更する」などの）Web 検索戦略は，開発者によって異なりうる．検索戦略が不適切であれば，有用な情報が含まれる Web ページの発見に時間が掛かり，その結果プログラミングの作業効率が低くなる可能性がある．本章ではこのような効率低下を防ぐため，開発者の検索戦略を分析し，適切な Web 検索の指針を示すことを試みる．

これまで，プログラミングにおける Web 検索行動を分析した研究は多く存在する[3][17][33][35][34][45]．ただし，これらの研究のほとんどは，Web 検索がどのように使われているか（頻度，検索対象など）を分析している．例えば Sadowski ら[35]は，開発者がどのような頻度で，何を検索対象として検索しているかを分析している．一方，これらの研究では，どのような Web 検索戦略を取るべきかの指針は示していない．

4.2. Web検索メトリクス

本章では，ソフトウェア開発者のプログラミングにおける Web 検索戦略を定量的に分析するために，5 つの Web 検索メトリクスを定義した．具体的には，Web 検索戦略で重要と思われる，検索キーワードの選択（検索エンジンへの入力）と検索結果の理解（検索エンジンの出力）に着目し，これらに基づいてメトリクスを定義した．

RPV (Result pages Per Viewed pages)

$$RPV = r / v \qquad (1)$$

Web 検索エンジンにより，検索キーワードに関連する Web サイトのリストからなる Web ページ（検索結果 Web ページ）が作成される．r はプログラミング

（タスク）中にこのページを表示（アクセス）した回数を表す．v は，プログラミング中に表示した Web ページ数（検索結果 Web ページを含む）を表す．

Google を検索エンジンとした場合，初期設定では 1 つの検索結果 Web ページに 10 個の Web ページの URL が含まれる．開発者がプログラミング中に 1 つの検索結果 Web ページを表示し，9 個の Web ページ（検索結果に含まれる Web ページ全て）にアクセスした場合，RPV は 0.1 となる．

検索エンジンの出力の利用方法は 2 種類に分けることができる．一つは，検索結果に含まれる Web ページのタイトルと要約に基づいて，各ページが有用かどうかを判断する方法である．もう一つは検索結果ページを読むだけではなく，検索結果に含まれる Web ページにアクセスして内容を読むことにより，有用かどうかを判断する方法である．RPV が高い場合，開発者には前者の傾向が，低い場合は後者の傾向があるといえる．

PPR (unique key Phrases Per web Result pages)

$$PPR = p / r \qquad (2)$$

開発者が Web 検索エンジンに入力した単語の組をキーフレーズと定義する．例えば「クラス インタフェース」や「null pointer exception」がキーフレーズとなる．p は，プログラミング（タスク）中に入力された全てのキーフレーズから重複を取り除いた数を表す．例えば入力されたキーフレーズが「クラス インタフェース」，「null pointer exception」，「クラス インタフェース」であるとき，p は 2 となる．

PPR が大きい時，開発者はキーフレーズを頻繁に変更しているといえる．逆に小さい時は，開発者は検索結果 Web ページを読み進めていることになる．

WPP (the number of key Words Per the number of unique key Phrases)

$$WPP = w / p \qquad (3)$$

ここで w は，（重複を除いた）全てのキーフレーズに含まれるキーワードの数を表す．例えばキーフレーズが「クラス インタフェース」の場合，w は 2 となる．すなわち，WPP は各キーフレーズにおける平均キーワード数を表す．

Google による検索ガイドラインでは，キーフレーズに含まれるキーワードは少ないほうが好ましいとされる．このガイドラインがプログラミングにおいて有効かを確かめるため，WPP を定義した．WPP が大きい場合，開発者はキーフレーズに多くのキーワードを含めていることになる．

RPT (the number of web Result pages Per Task time)

RPT = r / t (4)

t は開発者がプログラミング（タスク）終了までに掛けた時間である．例えば開発者が 10 分間のタスクにおいて，検索結果 Web ページを 10 ページ閲覧した場合，RPT は 1 となる．RPT が高い場合，プログラミングや検索結果以外のページ閲覧より，Web 検索（または検索結果ページの読み進め）が多いことを示す．

UPW (the number of Unique keywords Per the number of key Words)

UPW = u / w (5)

u は，プログラミング（タスク）中に入力された全てのキーワードから重複を取り除いた数を指す．例えば,「クラス interface」,「クラス abstract」の場合，u は 3，w は 4 となる．UPW が高い場合，新たなキーフレーズを作成する際，開発者は以前に使用した同じキーワードを使用する頻度が低いことを示す．

4.3. 実験

4.3.1. 概要

実験では，プログラミングに関する問題を複数出題し，被験者はそれらの回答に取り組んだ．

正解の扱い：実験では，被験者が現在の問題を正しく回答した場合，次の問題が表示されるようにした．すなわち，ある問題への回答完了時には，被験者はその問題に正解していることになる．

不正解の扱い: 実験では回答時間の制限を設けなかったが，20 分を超える場合は問題をスキップできることとした．この場合は不正解として扱った．問題のスキップ後は，その問題を修正できないこととした（例えば問題 2 の回答中に，スキップした問題 1 を修正できない）．

被験者は情報科学を専攻する 20 歳代の大学生 10 人である．1 人は修士課程の学生，残りは学部生である．被験者は，Web ブラウザは Google Chrome，検索エンジンは Google，エディターは Eclipse を用いた．被験者はソフトウェア開発の実務者ではないが，文献[36]では学生と実務者の違いはあまり大きくなく，学生を実務者の代替とできることが示されている．よって，実務者を被験者としても分析結果は大きく変わらないと考えられるが，実務者を被験者とす

ることは今後の課題である．

分析ではプログラムのコーディング時間は考慮していない．これは，適切なライブラリを発見できれば，プログラミング（問題を解くこと）自体は容易であるような問題を出題したためである．

4.3.2. プログラミングに関する問題

プログラミングに関する問題は， Java を題材とした．これは被験者全員が大学の講義を通じて Java を習得していたためである．問題では修正すべきソースコードをあらかじめ与えており，被験者は Web 検索を活用することにより修正すべき点を特定し，ソースコードを修正した．実験後には被験者にアンケートを行い，あらかじめ解法を知っていたかを確認した．

問題 1：Java では，小数点の計算において誤差が生じる場合がある．正確な計算を行うためのライブラリを用いるように修正する問題である[4].

問題 2：ArrayList を使ったデータ構造で実装されたプログラムを，問題で示された仕様を満たすために，連想配列を使ったデータ構造に書き換える問題である．ただし，問題では連想配列を使うことを指示しておらず，仕様を満たすためには連想配列の必要性を使う必要があることに気づく必要がある．

問題 3：ある URL からデータを取得した際，文字が表示されずに文字コードが表示されるプログラムに対し，正しく文字が表示されるよう修正する問題である．

問題 4：正しいエラーメッセージを表示しないプログラムに対し，Java の例外クラスの構造を理解して，正しいメッセージを表示するよう修正する問題である．

問題 1, 2 は比較的難易度が低く，問題 3, 4 は比較的高い．これらの問題は，適切な Web ページを発見できれば容易に解くことができる．検索戦略については，被験者に事前に指示していない．実験前にブラウザの検索履歴は削除した．

4.4. 分析

対象データ：被験者 10 人が 4 つの問題に回答したため，合計で 40 件のデータが得られた．うち，用いるべきライブラリ（解法）を事前に知っていたケースは 7 件であった．正解者の人数は，問題 1 で 7 人，問題 2 で 8 人，問題 3，4

で 6 人であった．全問正解は 3 人，全問不正解は 1 人であった．

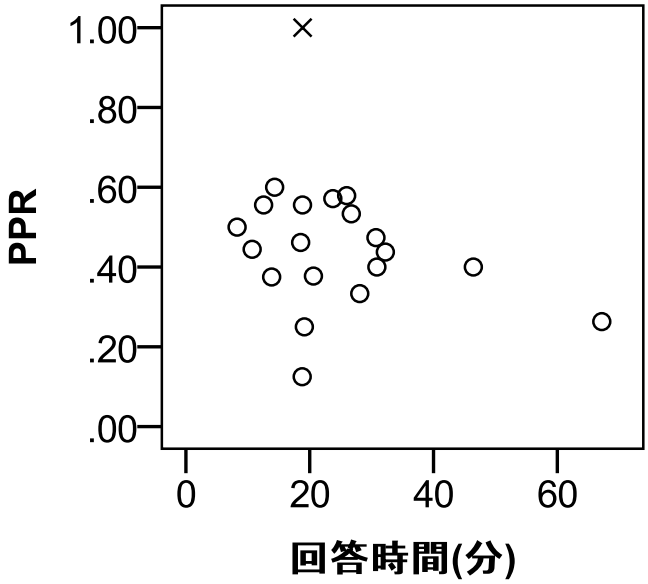
解法を知っていたケース 7 件を除外すると，33 件のデータとなる．ここから問題をスキップした不正解のケース 13 件を除外すると，分析対象は 20 件となる．データ件数がやや少ないため，参考として，不正解も含めた 33 件のデータについても分析した．

回答時間との関連：回答時間が短ければ効率が高いとみなし，回答時間と Web 検索メトリクスとの関連について分析した．関連の分析では，外れ値の影響を避けるために，スピアマンの順位相関係数を用いた．心理学では相関係数の絶対値が 0.2 を超える場合，弱い相関があるとみなされる．被験者実験は人間が分析対象であり，心理学と近い状況のため，絶対値が 0.2 を超えた場合の関連に着目した．

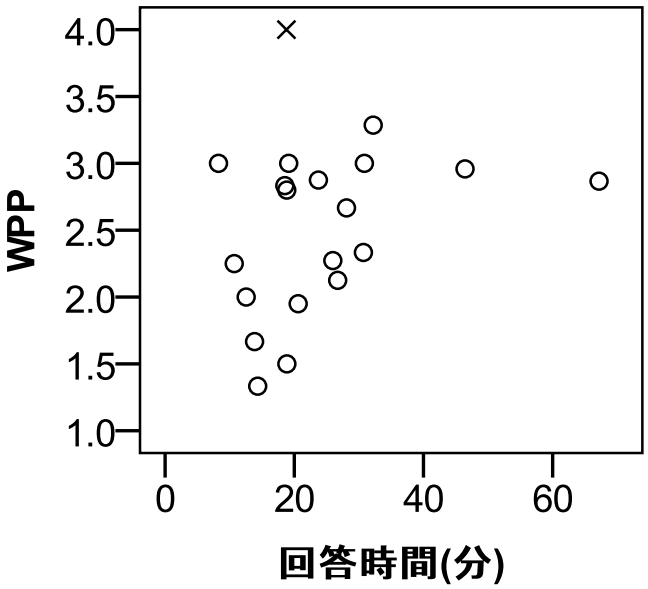
回答時間と Web 検索メトリクスとの相関係数を表 9 に示す．正解データ 20 件では，全メトリクスの相関係数の絶対値が 0.2 を上回っていた．不正解を含めた 33 件のデータでは，RPV のみ絶対値が 0.2 を下回っていた．

表 9　回答時間と Web 検索メトリクスとの相関係数

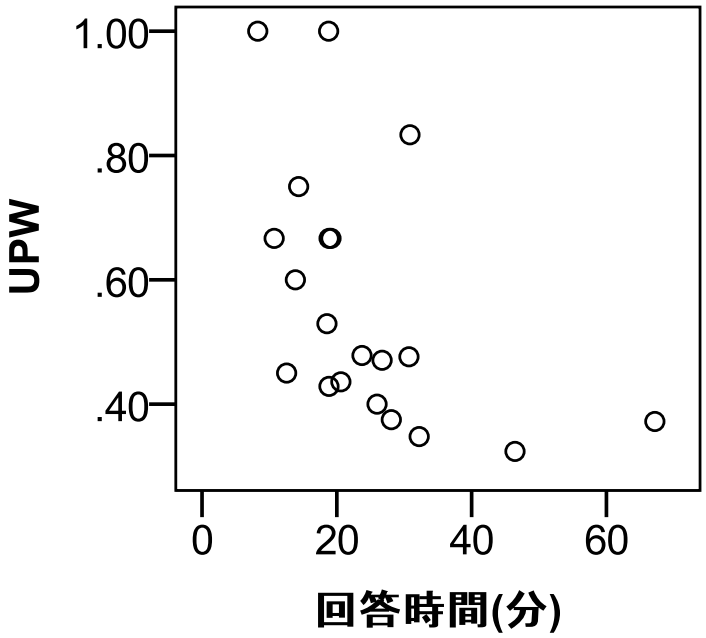
		RPV	PPR	WPP	RPT	UPW
含不正解	ρ	0.038	-0.291	0.245	0.266	-0.557
33 件	p 値	0.833	0.101	0.169	0.135	0.001
正解のみ	ρ	0.341	-0.255	0.325	0.305	-0.621
22 件	p 値	0.142	0.278	0.162	0.191	0.003


1.00
.80
.60
.40
.20
.00
PPR
0
20
40
60
回答時間(分)

(a) PPR


4.0
3.5
3.0
2.5
2.0
1.5
1.0
WPP
0
20
40
60
回答時間(分)

(b) WPP



(c) UPW

図 20　Web 検索メトリクスと回答時間との関係

これらの結果より，PPR が小さい場合，すなわちキーフレーズをあまり変更せずに検索結果 Web ページを読み進める傾向がある場合，回答時間が長くなる傾向があるといえる．よって，検索の効率を高めるためには「キーフレーズを変更せずに，検索結果ページを多数読み進めること」は避けたほうがよいといえる．

WPP: 正解データ 20 件における回答時間との散布図を図 20 の(b)に示す．図では「X」で表したケースの WPP が他のケースと大きく異なる．このケースを除外した場合の相関係数は 0.415（p 値は 0.077）であり，除外しない場合よりも相関が強くなった．図では，回答時間が 10 分前後の WPP の値域（最小値と最大値)は，回答時間が 30 分前後の WPP の値域よりも低いことが読み取れる．

これらの結果より，WPP が大きい場合，すなわち開発者がキーフレーズに多くのキーワードを含めていると，回答時間が長くなる傾向があるといえる．よって，検索効率を高めるため「キーフレーズに多数のキーワードを含めること」は避けるべきといえる．

RPT: 回答時間と正の相関であることから，回答時間が長いほど，回答時間あたりの Web 検索が多いことを示す．これは単に作業の行き詰まりを示している（適切な Web ページが見つからず，検索回数が増えた）可能性が高い．

UPW: p 値が 0.05 を下回っていたため，5%の有意水準で統計的に有意な相関であるといえる．正解データ 20 件における回答時間との散布図を図 20 の(c)に示す．図からも UPW が小さいと回答時間が長い傾向が見られる．

よって，UPW が高い場合，すなわち，新たなキーフレーズに以前使用したキーワードをあまり含めないと，回答時間が短くなる傾向があるといえる．すなわち，検索効率を高めるため「新たなキーフレーズを作成する場合，以前用いたキーワードを多く含めること」は避けたほうがよいといえる．

4.5. まとめ

本章では，プログラミングにおける開発者の Web 検索戦略を分析した．戦略を分析するため Web 検索メトリクスを定義するとともに，実験を行い，プログラミング（Java ライブラリ）に関する問題を被験者に出題した．被験者が問題を解いた際の回答時間と Web 検索メトリクスとの関連を分析した結果，下記を検索戦略とすると，時間を短縮できる可能性が示された．

- キーフレーズを変更せずに，検索結果ページを多数読み進めることは避けたほうがよい．
- キーフレーズには多くのキーワードを含めないほうがよい．
- 新たなキーフレーズを作成する場合，以前用いたキーワードはあまり含めないほうがよい．

実際のソフトウェア開発において，開発者が Web 検索を行う理由は様々であるが，上記の検索戦略は，特に開発者がライブラリを検索する際に有効であると考えられる．

5. バンディットアルゴリズムに基づくソフトウェア欠陥予測

5.1. まえがき

ソフトウェアモジュールに含まれる欠陥の有無を予測することは，ソフトウェア開発プロジェクトの計画立案および品質管理において非常に重要である．ソフトウェア欠陥予測を行うために，これまでの研究では，さまざまな予測方法，例えば，変数選択の手法[21]が提案，適用されてきた．多くの研究では，複数のデータセットを用いて，様々な手法により構築されたモデルの予測精度を比較，評価してきた[9][15]．ただし多くの場合，用いるデータセットが異なると，モデルの予測精度も異なるという問題があり[9][21]，実際のソフトウェアプロジェクトへの適用時に，ある予測方法が必ずしも高い予測精度が得られるかは不明である．これは，ソフトウェア開発のデータセット個別性が高いためであり，平均的に予測精度の高い手法を選択しても，用いるデータセットによっては予測精度が低下するリスクが存在する．D'Ambros らは，欠陥予測モデルには外的妥当性の問題があると指摘しており[9]，この問題は現在も解決していない．プロジェクトマネージャは特定の欠陥予測手法をあらゆるソフトウェアプロジェクトに適用しがちであるが，それらのプロジェクトのデータが同一の特徴を持っているとは限らないため,外的妥当性の問題が発生しうる.また，新たな欠陥予測方法が複数存在し，それらを直接比較した研究が存在しない場合，どちらの予測方法を選択すべきなのかが明らかでない．

様々なデータセットにおいて，一貫して予測精度の高い予測モデルを選択することを支援するため，本章では，モデルを選択する過程においてバンディットアルゴリズムを適用することを提案する．バンディットアルゴリズム[37]は比較的古典的なアルゴリズムであるが，近年では，例えば Web ページの最適化など，様々な分野において適用されつつある．次節ではバンディットアルゴリズムの詳細について説明する．

5.2. 提案手法

5.2.1. バンディットアルゴリズム

バンディットアルゴリズムは，しはしばスロットマシーンを比喩として用いて説明される．プレイヤーは 100 枚のコインを複数存在するスロットマシーンのうちのどれかに賭け,プレイヤーはその報酬を最大にすることを前提とする．多くの場合，プレイヤーはひとつのスロットマシーンを選択し，100 枚全てのコインをそのスロットマシーンに賭ける．それに対し，バンディットアルゴリズムでは，プレイヤーは最も報酬の大きいマシーンを探索するため，それぞれのスロットマシーンにコイン 1 枚だけを賭けることを繰り替える．これは，多腕バンディット問題と呼ばれ，バンディットアルゴリズムでは，総報酬を最大化するために，期待される報酬が不明な複数の候補から，最も期待される報酬の高い候補を順次探索していく.それぞれの候補はアームと呼ばれる.以下に，最も単純なバンディットアルゴリズムを示す.

- プレイヤーはひとつのアームに 1 コインを賭ける.
- **探索フェーズ**：現在選択されているアームの平均報酬が他のアームより少ない場合，プレイヤーは他の平均報酬が高いアームを選択する．この行動は最も報酬の高いアームを見つけることを目的としており，探索（explore）と呼ばれている.
- **活用フェーズ**：現在のアームの平均報酬が高い，もしくは他のアームと同じであるとき，プレイヤーは同じスロットマシーン（アーム）をプレイし続ける．この行動は高い報酬のアームを活用（exploitation）することが目的である.

表 10 にバンディットアルゴリズムの例を示す．最初に平均報酬は，それぞれのアームで 0 としている.

1. アーム A がランダムに選択される．このとき，アーム A の報酬は-1 であるため，アーム A の平均報酬は-1 となる.
2. アーム B の平均報酬はアーム A より大きいため,アーム B が選択される．アーム B の報酬は 1 であるため，アーム B の平均報酬は 1 となる.
3. アーム B の報酬が腕 A より大きいため，アーム B が再び選択される.

表 10 バンディットアルゴリズムの例

# of trial	Selected arm	Reward	Avg. reward A	Avg. reward B
1	A	-1	-1	0
2	B	1	-1	1
3	B	1	-1	1

最初の試行は活用フェーズとみなせる．また，二回目と三回目の試行は探索フェーズとみなせる．

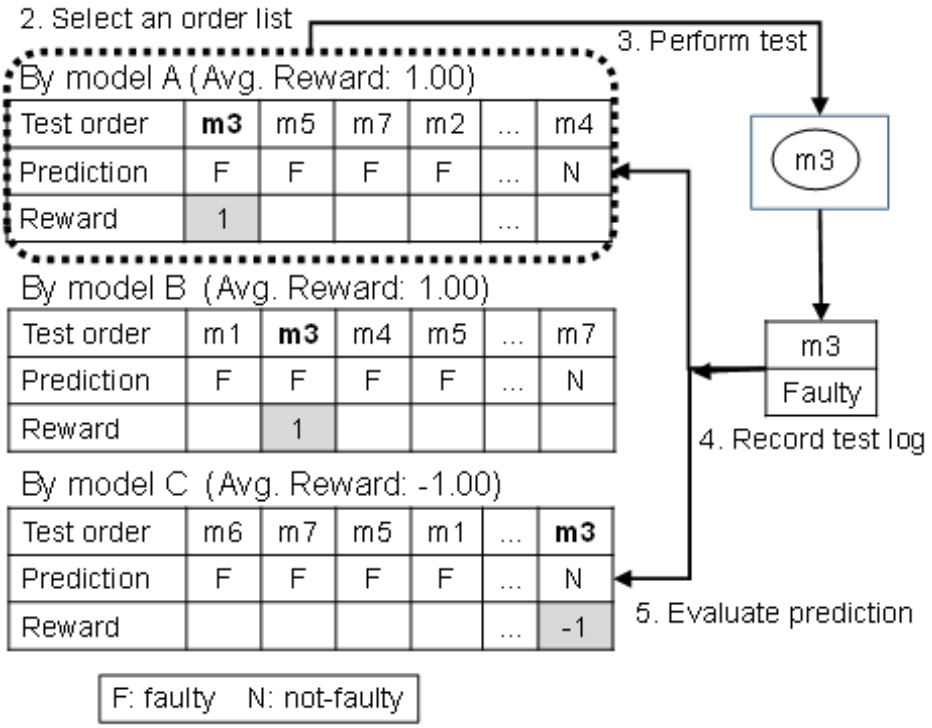


図 21 提案手法における最初のループ

epsilon greedy 法は ε の確率でランダムにアームを選択する方法である．アームは以下のアルゴリズムにより選択される．

- それぞれのアームの平均報酬に基づく活用を前提として，最も平均報酬

の高いアームが以下の確率で選択される．

1 - ε (0 ≤ ε ≤ 1)

- 探索を前提として，平均報酬を無視してアームのうちひとつが確率 ε でランダムに選択される．

確率 ε が 0 であるとき，アームは常にそれぞれのアームの平均報酬に基づき選択される．逆に確率 ε が 1 であるならば，アームは常にランダムに選択される．本章ではバンディットアルゴリズムとして epsilon greedy 法を用い，実験においては，確率 ε として 0, 0.2, 0.3 を設定した．

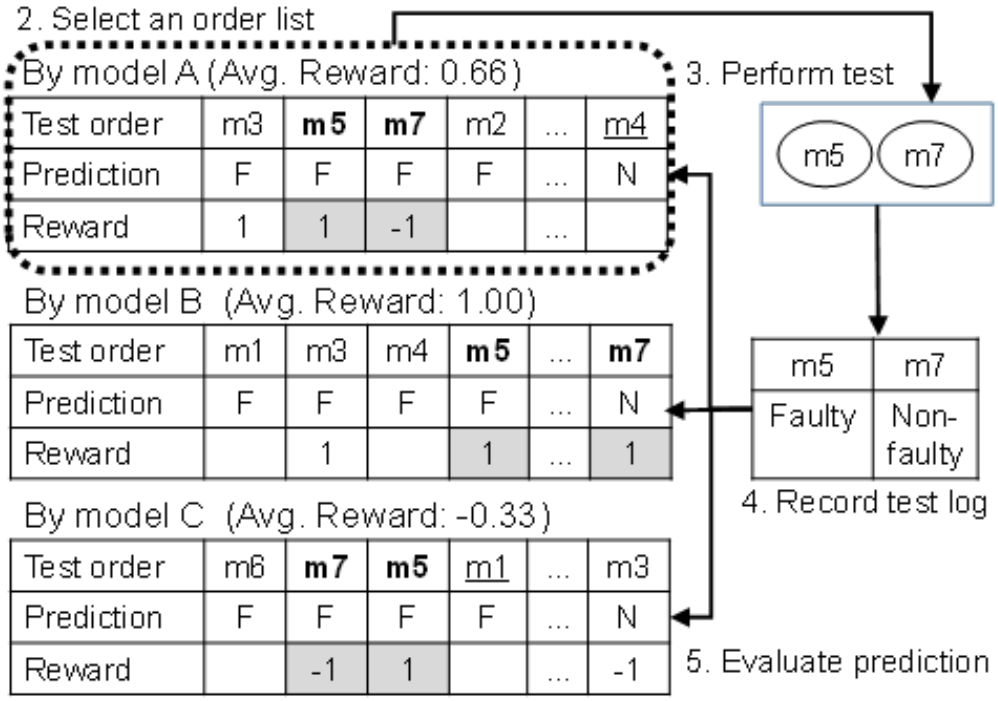


図 22　提案手法における 2 回目のループ

(a) Faults found on/after integration test	F				N	
(b) Faults found on integration test	F		**N**		N	
(c) Prediction	F	N	**F**	**N**	F	N
(e) Reward ((b) = (c)?)	1	-1	**-1**	**1**	-1	1
			100 – *p*%	*p*%		

図 23　実際の欠陥と予測された欠陥および報酬との関係

5.2.2. 提案手法

バンディットアルゴリズムの適用に必要な作業：ソフトウェアテストにおけるモジュールの欠陥予測に対し，バンディットアルゴリズムを適用するためには，以下の作業を実施する必要がある．

1. 複数の予測モデルを構築する．
2. テスト結果を記録する．
3. 予測結果と実際のテスト結果を比較する．
4. 平均報酬に基づいてモデルを選択する．

作業 1 は一度だけ実施される．作業 2 はバンディットアルゴリズムを適用しない通常のテストにおいても実施される．作業 3 と 4 はテストの履歴を活用することにより，自動的に実施することができる．通常のテストよりも追加の作業が必要となるが，バンディットアルゴリズムを適用することにより，精度が低い予測モデルを用いるリスクを抑えることができる．

提案手法の詳細：ソフトウェアの結合テストでは，ビックバンテストを除いて，モジュールはひとつずつ順にテストされる．そのため，ソフトウェアモジュールのテストの順序（テストの優先度）は欠陥予測モデルの結果に基づいて決められる．欠陥を修正する時間を考慮すると，欠陥はできるだけ早期に発見し，除去することが推奨される．「欠陥あり」と予測されたモジュールからテストするほうが，欠陥を早期に発見できる可能性が高くなる．これはリスクベースドテスト[12]と呼ばれ，以下の手順により実施される．

1. 様々な予測モデルを用いて，テスト順序のリストを作成する．
2. epsilon greedy 法を用いて，テスト順序のリストのうちのひとつを選択する．

3. モジュールはリストの順序に基づいてテストされる．
4. テストケースのログを記録する．
5. 予測とテストログが同じである場合，予測の報酬を 1 に設定する．予測とテストログが異なる場合，予測の報酬を-1 に設定する．
6. 手順 2 に戻る．

手順 1 においては，各モジュールのソースコードメトリクス（例えば，LOC（Lines of Code）やコードの複雑度など）に基づいて，それぞれが「欠陥あり」，「欠陥なし」のどちらかに予測される．次に，モジュールは予測結果によりソートされ，テスト順序のリストが作成される．図 21 は手順 1 において作られたリストの例を示している．図にはモデル A，モデル B，モデル C により作成された 3 種類のリストが含まれている．ここでは，どのリストが最も高い精度となるかどうかは不明である．提案方法では，手順 2 から 6 において，これらのリストからひとつを選択し，動的に新しい順序を作成する．図 21 と図 22 では，m3, m5, m7（図中の太字を参照）と m1，m4（図中の下線部を参照）が含まれた新しいテストの順序が作成されている．モジュール m3, m5, m7 の順序はモデル A によって作成されたものであり，m1，m4 はモデル B によって作成されたものである．

図 21 は最初のループにおける手順 2 から手順 5 を示している．この図では，モデル A によって作られたリストがステップ 2 でランダムに選択されている．選択されたリストにおいて「欠陥あり」と予測されていることから，モジュール m3 が最初にテストされる．手順 3 では，モジュール m3 がテストされ，モジュール m3 内に欠陥が見つかっている．手順 4 ではテストログが記録される．手順 5 では，m3 のテストログと A による予測結果が同じ結果である（すなわち m3 が「欠陥あり」である）ことを確認している．従って，報酬は 1 と設定される．同様に，B と C によって作成されたリストに含まれる予測結果を評価する．その後，手順 6 において手順 2 に戻る．

図 22 では，2 回目と 3 回目のループを示している．手順 2 は報酬の平均に基づき，モデル A によって作成されたリストの順序が再び選択される．図 22 では選択されたリスト A において，モジュール m5 と m7 は「欠陥あり」と予測されているため，m5 と m7 がテストされる．手順 5 では，テストログと比較

され，リストに含まれている m5 の予測結果は正しくなっている．ただし，m7 の予測結果は誤っている．従って，選択されたリスト A の報酬は 1（m5）と-1（m7）にそれぞれ設定される．同様に，B と C によって作成されたリストに含まれた予測結果を評価する．モデル A，B，C によって作られた順番リストの平均報酬を比較すると，モデル B が最も平均報酬が高くなる．

4 回目のループでは，手順 2 において，報酬の平均値に基づきモデル B によって作られた順序が選択される．モジュール m1（図 22 の下線部）は選択されたリスト B の最も先頭にあるため，最初にテストされる．モジュール m1（すなわち 5 回目のループにおいて）がテストされた後，モジュール m4（図 22 下線部）がテストされる．ただし，選択されたリスト B において，モジュール m3 が m1 の後に出現した場合，m3 はすでにテストを完了しているため，モジュール m4 が 5 回目のループにおいてテストされる．

一般的なソフトウェアテストでは，手順 1 から 4 は一度だけ行われるのに対し，提案方法では手順 2 から 6 がテスト実施中に繰り返し行われる．ステップ 2 におけるテスト順序のリスト（すなわち，予測モデル）の候補は，予測精度が高くなると過去の研究で示されているものを中心に選択する. 例えば, Ghotra らの研究では，ロジスティック回帰とそれに適した変数選択方法による予測精度が高いこと示されており[15][16], それらをモデルの候補として用いるとよい.

典型的なバンディットアルゴリズムとの違い：提案手法と典型的なバンディットアルゴリズムでは 2 つの違いがある．ひとつは，典型的なバンディットアルゴリズムでは，それぞれの試行で 1 つのアームを評価するのに対し，提案方法では，各ループ（試行）において，全ての順序のリスト（予測結果）を評価することができることである．前者は partial information game と呼ばれるのに対し，後者は full information game と呼ばれることがある．

もうひとつの違いは，提案方法では報酬を常に正しく設定することができないことである．全ての欠陥は結合テスト中に検出されるわけではない．すなわち, 結合テスト中に欠陥が検出されない場合でも, 後工程（システムテスト中，またはソフトウェアリリース後）において欠陥が検出される可能性がある．したがって，図 23 に示すように，報酬が誤る（統合テスト中に欠陥が検出されなかったが，後工程で見つかった場合．太字部分）確率を p%に設定する．本章の実験では p を 20 に設定し，この報酬誤りの影響を考慮して実験した．その

他に，テストの結果が非決定的（ある結果が欠陥であるかどうかを判断することができない）可能性も考慮する必要がある．この場合，予測結果に対する報酬を決定できないが，本章では，その場合については除外している．

アンサンブル法との違い：バンディットアルゴリズムとアンサンブル法[31]は，どちらも複数の予測モデルを利用するという点では同じである．ただし，アンサンブル法は予測時に探索を行わない．提案方法では，アンサンブル法の予測結果を，アームのひとつとみなすこともできる．従って，バンディットアルゴリズムとアンサンブル法は対立するものではなく，同時に使用できるものである．

5.3. 実験

実験では，NASA Metrics Data Program に含まれる 3 つのデータセットを用いた．それぞれのデータセットの詳細は以下の通りである．

- KC4: 125 モジュール, 61 (48.8%) faulty モジュール
- MW1: 403 モジュール, 31 (7.7%) faulty モジュール
- PC4: 1458 モジュール. 178 (12.2%) faulty モジュール

これらのデータセットは，各データセットのサイズと，欠陥モジュールの比率が異なったものとなるように選択した．これらのデータセットは比較的古いものであるが, 利用が容易なものであり, 最近の研究でも用いられている[15].

欠陥モジュールを予測するため，決定木（DTree）ニューラルネットワーク（NN)，ロジスティック回帰（Logistic)，線形判断分析（LDA）を用いて異なる予測モデルを構築した．これらは欠陥予測の研究で広く用いられてきたものであり（[15][16]など)，これらの単独使用を従来方法とした．機械学習による予測方法として，他にランダムフォレストやサポートベクターマシンなども用いられることもあるが，これらを用いることは今後の研究の課題である．

さらに，その他の既存アプローチと提案手法との性能を比較するため，上記の 4 つのモデルを用いた majority voting（Voting）[50]を適用した．Voting は欠陥予測の研究においても使用されている（[20]など）

バンディットアルゴリズムを適用する場合，予測モデルに基づくテスト順序は，モジュールサイズも考慮してソートされる（例えば,「欠陥あり」かつ規模の大きなモジュールが最初にテストされる). これはリスクベースドテスティン

グ[12]を前提としている．なお，予測結果の分析においては，モジュールサイズの間接的な影響を考慮するべきである[39].

前述したように，結合テストにおいて全ての欠陥が見つかるわけではない．大規模な企業横断的データによると，結合テスト後に欠陥の 17%が検出されている[18]．すなわち，モジュールが実際に欠陥を含んでいる場合，バンディットアルゴリズムによる予測結果の評価は 17%の確率で正しくないということになる．欠陥を見落とす可能性を考慮して，モジュールに欠陥が含まれる場合，バンディットアルゴリズムによる報酬の正負を 20%の確率でランダムに反転させた（図 23 で示す確率 p を 20 に設定した）.

予測精度の評価方法として，ホールドアウト法を適用した．データセットは学習データとテストデータにランダムに分割し，それらの比率を 3:1 とした．予測性能を評価するために 10 回分割を繰り返し, 繰り返しから得られた平均値を評価基準の値とした.

それぞれの予測モデルを評価するため，Area Under the Curve(AUC)を用いた．AUC は判別予測の閾値を特定の値に設定せずに, 判別予測モデルの性能を評価することができる．バンディットアルゴリズムの AUC を計算するため，予測結果として得られたものを用いた．例えば図 21 と図 22 において，m3，m5，m7，m1, m4 が「欠陥あり」として予測されているため，それに基づき AUC を計算した.

ほとんどの予測モデルにおいて，予測値は実数となる．バンディットアルゴリズムを適用時には, カットオフ値は 0.5 に設定して予測値を二値（「欠陥あり」，「欠陥なし」）に変換した．なお，予測値が二値しか取らない場合，AUC は新陽性率と真陰性率の平均と同一となる．比較条件を同一にするため，従来の予測モデルによる予測値についても，二値に変換して AUC を算出した.

表 11 Epsilon-greedy の精度

	AUC			Rank			Avg. AUC	Avg. Rank
	PC4	KC4	MW1	PC4	KC4	MW1		
ε=0	0.73	0.73	0.56	4	4	4	0.67	4.0
ε=0.1	0.77	0.74	0.57	3	1	3	0.69	2.3
ε=0.2	0.78	0.73	0.58	2	3	2	0.69	2.3
ε=0.3	0.78	0.73	0.60	1	2	1	0.70	1.3

表 12 従来モデルの精度

	AUC			Rank			Avg. AUC	Avg. Rank
	PC4	KC4	MW1	PC4	KC4	MW1		
DTree	0.70	0.74	0.58	3	2	3	0.67	2.7
LDA	0.69	0.68	0.60	4	6	1	0.66	3.7
Logistic	0.72	0.69	0.57	2	5	4	0.66	3.7
NN	0.61	0.76	0.52	6	1	6	0.63	4.3
Voting	0.65	0.73	0.54	5	4	5	0.64	4.7
ε=0.3	0.78	0.73	0.60	1	3	2	0.70	2.0

5.4. 結果

最初に,バンディットアルゴリズムによる予測精度の比較を行った(表 11). 表 11 は各データセットにおける 4 つのパラメータの AUC の順位と値を示している．また，表 11 には AUC の平均値と 3 つのデータセットにおける平均値も示している．Epsilon-greedy (ε =0.3) が平均 AUC と平均の AUC のランクが 4 つのアルゴリズムで最も高かった．よって，Epsilon-greedy (ε =0.3)は 4 つのパラメータで最も精度が高いといえる．

次に，Epsilon-greedy (ε =0.3)の予測精度と従来法の比較を行った．表 12 にこれらの予測精度を示す．MW1 データセットの場合，最も精度の高いモデル

LDA と Epsilon-greedy の AUC おおむね同じであった．KC4 データセットの場合，最も精度の高いモデル（NN）と Epsilon-greedy との差は 0.03 であった．PC4 データセットの場合，最も精度の高いモデルは Epsilon-greedy (ε =0.3)であった．すなわち，Epsilon-greedy の予測精度は，最悪のケースにおいて最も精度の高いモデルよりも 0.03 低かった．majority voting の平均 AUC は 2 番目に低く，平均順位は予測モデルの中で最も低かった．3 つのデータセットにおいて Epsilon-greedy の AUC の平均と順位は，4 つの予測方法と majority voting よりも高かった．従って，Epsilon-greedy (ε =0.3)を用いることにより，どのモデルが最適かを考慮することなく，高い欠陥予測精度を得られるといえる．

なお，PC4 データセットにおいて，4 つの予測モデルの中でバンディットアルゴリズムの AUC が最も高かった．これは，モジュールサイズによって予測結果がソートされていることが影響していた．モジュールサイズが大きい場合，バンディットアルゴリズムは LDA による予測結果を選択し，モジュールサイズが小さい時，NN による予測結果を選択していた．これが，予測精度を改善する効果を与えていた．

5.5. CPDPへの適用

欠陥を予測するモデルを構築するためには，あらかじめデータを収集しておく必要がある．モデル構築に用いられるラーニングデータは，通常，予測対象のソフトウェアのバージョンよりも前のバージョン（例えば予測対象のバージョンが 1.2 なら 1.1 など）で収集されたデータをラーニングデータとする．これは WPDP (within project defect prediction)と呼ばれる．

ただし新規開発ソフトウェアの場合，前バージョンのデータが存在しない．このため，新規開発のソフトウェアにおいて欠陥予測を行う場合，別のソフトウェアから収集したデータをラーニングデータとし，モデル構築する必要がある．これは cross project defect prediction (CPDP)と呼ばれる．

一般に，CPDP では，WPDP よりも予測精度が低下することが多い．これは，ソフトウェアが異なると，説明変数（ソフトウェアの複雑度など）と目的変数（欠陥の有無）の関係が大きく異なりうるためであると考えられる．CPDP の予測精度を高めるために，ラーニングデータとの変形や，ラーニングデータとテストデータの特徴比較なども行われているが，精度を大きく改善できないた

め，大きな課題となっている．

そこで CPDP にバンディットアルゴリズム（BA）を適用する．CPDP に BA を適用すると，例えば 5 個のソフトウェアのデータから構築した予測モデルが，利用する予測モデルの候補となる．それらの中から，各モジュールのテスト時点で最も精度の高いモデルを動的に選択することを繰り返す．ここでは BA として ε -greedy（ε =0, 0.1, 0.2, 0.3），Thompson sampling (TS)，UCB を用いる．

実験: 実験では， NASA データセットから，C 言語で開発された 7 個のプロジェクトデータを用いた．（WPDP 以外で）最も精度の高いデータセットが選択された最善の場合の精度，ランダムにデータセットを選択した時の平均的な精度，提案方法の精度の 3 つの予測精度を比較した．予測精度の評価指標には AUC を用いた．交差検証法として fold-out 法を用い，データの分割を 10 回繰り返した．

まず，BA アルゴリズム間の精度を比較した結果を表 13 に示す．TS の精度が最も高かったため，以降では TS と従来方法について比較する．従来方法との比較結果を表 14 に示す．表は提案方法と従来方法の予測精度の差分を示しており，値が正の場合は提案方法の予測精度が高かったこと，負の場合はその逆を表す．最善のものが選べた場合と比べると AUC が低下しているが，その差は平均で 0.02 ほどであり大きくなく，そもそも最善のものを選ぶことは困難である．ランダムな場合と比較すると，AUC が平均で 0.07 改善しており，比較的改善の度合いが大きいといえる．

表 13 BA アルゴリズム間の精度比較

	CM1	MC2	MW1	PC1	PC2	PC3	PC4	平均
ε=0	0.69	0.60	0.67	0.67	0.65	0.65	0.57	0.64
ε=0.1	0.68	0.57	0.68	0.67	0.65	0.66	0.55	0.64
ε=0.2	0.68	0.64	0.67	0.67	0.63	0.67	0.60	0.65
ε=0.3	0.68	0.59	0.63	0.66	0.68	0.66	0.59	0.64
TS	0.74	0.62	0.67	0.70	0.73	0.74	0.69	0.70
UCB	0.69	0.59	0.65	0.68	0.61	0.65	0.56	0.63

表 14 提案方法と従来方法の予測精度の差分

	CM1	MC2	MW1	PC1	PC2	PC3	PC4	平均
最善	-0.01	-0.06	-0.05	-0.05	0.00	0.03	0.01	-0.02
ランダム	0.05	0.02	0.04	0.06	0.10	0.10	0.10	0.07

5.6. 関連研究

オンライン学習：バンディットアルゴリズムはオンライン学習の手法とみなすことができる[14]．これまでの研究において，オンライン学習の手法を欠陥予測に適用したものがある（[38][42]など）ただし，これまでの研究では，欠陥予測モデルは予測対象となるソフトウェアが時間の経過とともに変化していくために，予測モデルも継続的に再構築すべきと仮定している点が異なる．この仮定に基づき，ソフトウェア開発中に順次得られた新しいデータを用いて，予測モデルを継続的に（オンラインで）構築することを提案している．ただし，これらの研究では，予測最適化のため予測モデルの精度を比較することは行っていない．本章では，予測モデルが時間の経過とともに変化しないことを前提としており，予測モデルの精度はこの前提に基づいて評価される．

Wang ら[42]は，欠陥予測モデル構築に用いられるデータセットの分布が不均一であることに対処するために，オンラインオーバーサンプリングとアンダーサンプリングの手法をソフトウェアの欠陥予測に適用している．ただしこの研究では，モデルを予測精度に基づいて最適化することは考慮していない．

Tabassum ら[38]は，Just-In-Time ソフトウェア欠陥予測において，3 つの異なる予測手法に対してオンライン学習に適用している．この研究では majority voting[24]（多数決）を使いつつ，順次得られるデータをデータセットに追加して複数の予測モデルを継続的に再構築するアプローチをとっている．ただしこの研究でも，モデル構築時に予測結果と実測値を比較していない．この研究では，予測モデルをテスト工程で用いるだけでなく，ソフトウェア開発プロセス全体でも使うことを前提としている．実験ではあるソフトウェア開発において 9～10 ヶ月間に収集されたデータと，オープンソースプロジェクトにおいて 6～14 年間で得られたデータを使用している．この場合，予測モデルの性能評価（とラーニングデータの性質）は時間の経過とともに変化する（一定でない）という前提のほうが適している．

動的モデル選択：いくつかの研究では[28][32]，予測モデルが異なるデータセ

ットにおいて必ずしも高い性能を発揮しない問題について着目している．これらの研究では，利用可能なモデルのセットから予測モデルを動的に選択するアプローチを採っている．Di Nucci ら[28]は予測対象のモジュールに対し，動的に予測モデルを選択している．Rathore[32]らは欠陥数を予測するために，同様に動的に予測モデルを選択している．ただしモデルの選択は．予測対象モジュールのコードメトリクスなどの特徴に基づいており，予測モデルの精度を考慮してモデルの最適化を行っていない．従って，これらの手法は予測モデルの外的妥当性の問題に必ずしも適用できるとは限らない．

5.7. まとめ

本章では精度の高い欠陥予測モデルを動的に選択するため，バンディットアルゴリズムを適用した．バンディットアルゴリズムは，最適化の問題に適用されることが多いアルゴリズムである．これまでの研究では，予測モデルの精度は，テスト工程前に評価すること（オフライン）が前提であった．このため，欠陥予測の外的妥当性が重要な問題となっていた．これに対し，提案するアプローチでは，モデルの精度はテスト工程において（オンラインで）に評価可能であることを実験的に示した．我々の知る限り，バンディットアルゴリズム（もしくはオンライン学習）を欠陥予測に適用し，様々な予測手法の精度と比較した研究は存在しない．

実験では，ε（探索の確率）をパラメータとする Epsilon-greedy 法を適用した．3 つのデータセットを用いて，4 つの予測モデルと majority voting を従来法として評価した．その結果，Epsilon-greedy （$\varepsilon = 0.3$）を用いたとき，各データセットにおいて，予測精度が最も高いか 2 番目に高くなった．この結果より，バンディットアルゴリズムは，従来方法や majority voting よりも高い予測性能が得られているといえ，精度の低い欠陥予測モデルを避けることができる可能性を示唆している．

今後の課題は，より多様なデータセットに適用するとともに，様々な手法を用いて提案方法を適用して性能を評価することである．

6. おわりに

本論文では，ソフトウェアモジュールの品質向上を達成しつつ，短納期を実現することをゴールとして，ステークホルダの行動変容を促進するアプローチを取った．

まず，オープンソースソフトウェアプロジェクトである，Eclipse プロジェクトの寄付データを用いて，効果的な寄付の収集方法を明らかにするため，分析を行った．その結果，(1) 特典が寄付への動機づけとなっていること，(2) 全体的な寄付者のうち開発者の割合は少ないが，開発者の寄付額は開発者でないものより大きかったこと (3) リリース日には寄付が増えるが，バグ数が多いと寄付が落ち込むようにみえたことがわかった．

次に，バグレポート提出後，最初の応答までの時間が寄付バッジの影響を受けるのかを明らかにするため，寄付バッジをもたないバグレポートと比較して，寄付バッジをもつバグレポートに対する開発者の応答時間を Quantile Difference in Differences (QDID) という手法で分析した．その結果，寄付バッジが応答時間を中央値で約 2 時間短縮することが明らかとなった．この結果より，ソフトウェア利用者に対しては，OSS へ寄付するという行動変容が期待される．これにより欠陥の早期除去が期待され，結果として品質と開発速度の向上にもつながる．さらには寄付による OSS の発展も期待される．

次に，プログラム作成時における検索ログを分析し，プログラミング時における検索戦略の指針について明らかにするため，実験では被験者に対し，プログラミングに関する出題を行い，それに対してプログラミングにより回答してもらう．その検索ログに対し，分析メトリクスを設定し分析を行った．その結果，一度使用した検索のキーワードを含むキーフレーズを再び用いることはプログラミング時の検索行動としては効率が高くない可能性があることなどの検索戦略の指針を明らかにした．この結果より，ソフトウェア開発者は本論文で示した Web 検索戦略に基づいて行動変容することが推奨され，これにより開発速度の向上が期待される．

最後に，ソフトウェア欠陥予測において，用いるデータセットが異なる場合，予測モデルの性能が変化してしまう問題に対して，バンディットアルゴリズム

を適用することを提案した．また，提案方法の評価を行い，その有用性について明らかにした．実験では，3 種類のデータセットを用い，従来方法として 4 種類の予測方法を用意するとともに，それらとバンディットアルゴリズムによる動的なモデル選択の性能を比較した．さらに majority voting を用いて得られた予測とバンディットアルゴリズムを適用した予測の精度を比較した．その結果, バンディットアルゴリズムのひとつである Epsilon-greedy 法の ε が 0.3 のとき，これらの従来方法を単独で用いた場合と比較して，最も高い予測精度か，2 番目に高い精度を示した．この結果より，プロジェクト管理者はテスト計画立案時に予測モデルを 1 つだけ選択せず，テスト中にモデルを動的に選択するように行動変容することが推奨される．これにより予測精度，すなわち計画の正確性が向上し，より多くの欠陥がテスト中に除去されるようになる．

今後は，バンディットアルゴリズムをより多様なデータセットに適用するとともに，様々な手法を用いて適用して性能を評価する予定である．また，開発者に有効な支援のためには，単なるソフトウェア開発データだけでなく，開発者自身の生体情報を分析して，その開発者の精神状況のモニタリングを行うことで，より有効な支援ができると考えている．そのため，サーモグラフィーを用いた，開発者の生体情報の分析を行い，開発者支援に役立てることを予定している．

謝辞

本研究を進めるにあたり，多くの方々からご指導，ご協力，ご助言をいただきました．この研究にかかわってくださった全ての方に対して，心から感謝いたします．誠にありがとうございました．

奈良先端科学技術大学院大学 先端科学技術研究科 松本 健一 教授には，本論文の主指導教員をご担当いただきました．先生からは，工学研究は事実を解明するだけではなく，研究にはどういう背景があり，どう役に立つかが重要であるといった，研究の根幹的なことを始め，発表資料や論文における詳細な部分まで，丁寧なご指導，ご助言をいただきました．また，日々の研究だけでなく，就職活動や，生活面など多方面に渡り，多大なご支援をいただきました．心からの深い感謝を申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 安本 慶一 教授には，本論文の副指導教員をご担当いただきました．先生からは，博士前期課程から副指導教員を担当していただき，学内の発表において，多数のご質問，ご指導をいただきました．先生からいただいた研究の展望や，改善点などのご助言は研究を客観的に見つめなおすうえで重要なものとなりました．ここに深く感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 石尾 隆 准教授には，本論文の副指導教員をご担当いただきました．先生の的確なご指摘やアドバイスは，研究を進める上で大きな助けとなりました．また，生活面においても多大なご支援をいただきました．深く感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 畑 秀明 助教には，本論文の副指導教員をご担当いただきました．先生には，博士前期課程に入学してから，研究や学生生活において，多大なサポートをいただきました．本研究は先生のご指導，ご助言なしには進めることはできなかったと思います．先生の研究に対する姿勢や考え方のご指導は今後の研究活動を進める上で大変参考になりました．心より感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 Raula Gaikovina Kula 助教には，本論文の副指導教員をご担当いただきました．先生には，論文の回

答書作成，英語の添削など，多くの丁寧なご指導，ご助言をしていただきました．また，生活面でも支えになってくださりました．心より感謝申し上げます．

近畿大学 総合理工学研究科 角田雅照 准教授には，本論文の副指導教員をご担当いただきました．先生には，奈良先端科学技術大学院大学に入学する前から長きに渡り，研究の指導や，生活面，就職活動など多くのご指導，サポートをいただきました．先生のサポートなしには，学生生活を続けることができなかったと思います．心より感謝申し上げます．

本論文の大部分は，Eclipse コミュニティのデータを用いて分析しており，その中で実施したアンケートにおいては，Eclipse コミュニティの方々に直接協力をいただいて，実施しております．Eclipse コミュニティの方々に心より感謝申し上げます．

和歌山大学 伊原 彰紀 講師には，博士前期課程において，研究内容，研究生活，研究者としての姿勢について多くのご助言をいただきました．心より感謝申し上げます．

ソフトウェア工学研究室 秘書の高岸詔子さんには，学生生活や出張手続等において，様々な面でサポートしていただきました．高岸さんのサポートなしには，学生生活を円滑に過ごすことができませんでした．心より感謝申し上げます．

パナソニック株式会社 尾上 紗野さんには，本研究について多くのご助言をいただきました．先輩からの丁寧なご助言は研究を進める上で大変参考になりました．心より感謝申し上げます．

株式会社富士通研究所 中川 尊雄さんには，博士後期課程の学生生活を送るにあたって多くのご助言をいただきました．また，先輩が修了されてからも学生生活を続ける上で多大なサポートをいただきました．ここに，深く感謝申し上げます．

ユニファ株式会社 坂口英司さんには，丁寧なご助言や，英語の添削など多くのご支援をいただきました．また，就職活動にあたってもご助言いただきました．心より感謝申し上げます．

株式会社東芝 上村 恭平さんには，IEEE 学生支部の運営や，共同研究において，多数の協力をいただきました．また，学生生活を送るうえで精神的な支えとなっていただきました．心より感謝申し上げます．

freee 株式会社 池田 祥平さんには，博士前期課程の時から，発表練習や，添削にお付き合いしていただきました，また，就職活動にあたっても多くのご協力いただきました．心より感謝申し上げます．

株式会社ゲームフリーク 大神勝也さんには，共同研究を進めるにあたって，多くの協力をしていただきました．また，多くの技術的な面白い話を聞かせていただき，研究生活を進めるにあたって励みとなりました．心より感謝申し上げます．

奈良先端科学技術大学院大学ソフトウェア工学研究室の学生の皆様には，発表練習や，論文添削にお付き合いいただくなど，多大なるご支援，ご協力をいただきました．皆様のおかげで，非常に充実した５年間を過ごすことができました．心より感謝申し上げます．特に，幾谷 吉晴さん，上田 裕己さんには研究のご助言だけでなく，精神的な支えとなってくださりました．本当にありがとうございました．

最後に，研究を進めるにあたって，金銭的，精神的にも支えてくださった．家族，祖父母に感謝いたします．

参考文献

[1] J. D. Angrist and J. S. Pischke, Mostly harmless econometrics: An empiricist's companion, Princeton university press, 2008.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund and R. Schapire, “Gambling in a rigged casino: The adversarial multi-armed bandit problem,” Proc. of Annual Foundations of Computer Science, pp. 322-331, 1995.

[3] S. Bajracharya, and C. Lopes, “Analyzing and mining a code search engine usage log,” Empirical Software Engineering, vol.17, no.4, pp.424-466, 2012.

[4] J. Bloch, and N. Gafter, “Java Puzzlers: Traps, Pitfalls, and Corner Cases,” Addison-Wesley Professional. 2005.

[5] S. Bubeck and N. Cesa-Bianchi “Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems,” Foundations and Trends in Machine Learning, vol.5, no.1, pp.1-122, 2012.

[6] Bug282088: Expand Friends of Eclipse Program, 2009.

[7] Bug434249: Add decorator for Friends of Eclipse, 2014.

[8] Bug514954: Make "Friends of Eclipse" more appealing, 2017.

[9] M. D’Ambros, M., Lanza, and R. Robbes, “Evaluating defect prediction approaches: a benchmark and an extensive comparison,” Empirical Software Engineering, vol.17, no.4-5, pp.531-577, 2012.

[10] V. Dani and T. Hayes, “Robbing the bandit: less regret in online geometric optimization against an adaptive adversary,” In Proc. of annual ACM-SIAM symposium on Discrete algorithm (SODA), pp.937–943, 2006.

[11] S. Davies and M. Roper, “What's in a Bug Report?”, Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, New York, NY, USA, ACM, no. 26, pp.1-10, 2014.

[12] M. Felderer and R. Ramler, “Integrating risk-based testing in industrial

test processes," Software Quality Journal, vol.22, no.3, pp.543-575, 2014.

[13] E. Franck, and C. Jungwirth, Reconciling rent-seekers and donators - The governance structure of open source, Journal of Management and Governance, Vol. 7, No. 4, pp. 401-421, 2003.

[14] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol.55, no.1, pp.119-139, 1997.

[15] B. Ghotra, S. McIntosh, and A. Hassan, "A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models," Proc. of International Conference on Mining Software Repositories (MSR), pp.146-157, 2017.

[16] B. Ghotra, S. McIntosh, and A. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," Proc. of International Conference on Software Engineering (ICSE), pp.789-800, 2015.

[17] R. Gallardo-Valencia, and S. Sim, "Information used and perceived usefulness in evaluating web source code search results," Proc. of CHI '11 Extended Abstracts on Human Factors in Computing Systems, pp.2323-2328, 2011.

[18] Information-technology Promotion Agency (IPA), Japan, The 2018-2019 White Paper on Software Development Projects, IPA, 2018 (in Japanese).

[19] R. Jongeling, P. Sarkar, S. Datta and A. Serebrenik, On Negative Results when Using Sentiment Analysis Tools for Software Engineering Research, Empirical Softw. Engg., Vol. 22, No. 5, pp. 2543-2584, 2017.

[20] T. Khoshgoftaar, P. Rebours, and N. Seliya, "Software quality analysis by combining multiple projects and learners," Software Quality Journal, vol.17, pp.25-49, 2009.

[21] M. Kondo, C. Bezemer, Y. Kamei, A. Hassan, and O. Mizuno, "The impact of feature reduction techniques on defect prediction models," Empirical Software Engineering, vol.24, no.4, pp.1925–1963, 2019.

[22] S. Krishnamurthy, and A. K. Tripathi, Monetary donations to an open source software platform, Research Policy, Vol. 38, No. 2, pp. 404-414, 2009.

[23] S. Krishnamurthy, Ou, S. and A. K. Tripathi, Acceptance of monetary rewards in open source software development, Research Policy, Vol. 43, No. 4, pp. 632-644, 2014.

[24] N. Littlestone, and M. Warmuth, “The Weighted Majority Algorithm,” Information and Computation, vol.108, no.2, pp.212-261, 1994.

[25] S. McIntosh, Y. Kamei, B. Adams and A. E. Hassan, “The Impact of Code Review Coverage and Code Review Participation on Software Quality: A Case Study of the Qt, VTK, and ITK Projects”, Proc. of the 11th Working Conference on Mining Software Repositories, MSR 2014, New York, NY, USA, ACM, pp. 192-201, 2014.

[26] K. Nakasai, H. Hata and K. Matsumoto, Are Donation Badges Appealing?: A Case Study of Developer Responses to Eclipse Bug Reports, IEEE Software, vol.36, no.03, pp. 22-27, 2019.

[27] K. Nakasai, H. Hata, S. Onoue, and K. Matsumoto, “Analysis of Donations in the Eclipse Project”, 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP), pp. 18-22, 2017.

[28] D. Di Nucci, F. Palomba, R. Oliveto and A. De Lucia, “Dynamic Selection of Classifiers in Bug Prediction: An Adaptive Method,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol.1, no.3, pp.202-212, 2017.

[29] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, “Are Bullies More Productive?: Empirical Study of Affectiveness vs. Issue Fixing Time”, Proc. of the 12th Working Conference on Mining Software Repositories, MSR '15, Piscataway, NJ, USA, IEEE Press, pp. 303-313, 2015.

[30] C. Parnin, and C. Treude, “Measuring API Documentation on the Web”, Proc. of the 2nd International Workshop on Web 2.0 for Software Engineering, Web2SE '11, New York, NY, USA, ACM, pp. 25-30, 2011.

[31] J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, “Building an Ensemble for Software Defect Prediction Based on Diversity Selection,” Proc. of International Symposium on Empirical Software Engineering and Measurement (ESEM), no.46, p.1–10, 2016.

[32] S. Rathore and S. Kumar, “An Approach for the Prediction of Number of Software Faults Based on the Dynamic Selection of Learning Techniques,” in IEEE Transactions on Reliability, vol.68, no.1, pp.216-236, 2019.

[33] M. Rahman, J. Barson, S. Paul, J. Kayani, F. Lois, S. Quezada, C. Parnin, K. Stolee, and B. Ray, “Evaluating how developers use general-purpose web-search for code retrieval,” Proc. of International Conference on Mining Software Repositories (MSR), pp.465-475, 2018.

[34] S. Sim, M. Umarji, S. Ratanotayanon, and C. Lopes, “How Well Do Search Engines Support Code Retrieval on the Web?” ACM Transactions on software Engineering and Methodology, vol.21, no.1, article 4, 2011.

[35] C. Sadowski, K. Stolee, and S. Elbaum, “How developers search for code: a case study,” Proc. of Joint Meeting on Foundations of Software Engineering (ESEC/FSE), pp.191-201, 2015.

[36] I. Salman, A. Misirli, and N. Juristo, “Are students representatives of professionals in software engineering experiments?” Proc. of International Conf. on Software Engineering (ICSE), pp.666-676, 2015.

[37] R. Sutton, and A. Barto, Reinforcement Learning: An Introduction, A Bradford Book, 1998.

[38] S. Tabassum, L. Minku, D. Feng, G. Cabral, and L. Song, “An Investigation of Cross-Project Learning in Online Just-In-Time Software Defect Prediction,” Proc of International Conference on Software Engineering (ICSE), 2020.

[39] A. Tahir, K. Bennin, S. MacDonell, and S. Marsland, “Revisiting the size effect in software fault prediction models,” Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM), article 23, p.1-10, 2018.

[40] J. Tsay, L. Dabbish, and J. Herbsleb, “Influence of social and technical factors for evaluating contribution in GitHub”, Proc. of the 36th international conference on Software engineering, ACM, pp. 356-366, 2014.

[41] I. Vignoli, 200,000 thanks, 2016.

[42] S. Wang, L. Minku, and X. Yao, “Online Class Imbalance Learning and Its Applications in Fault Detection,” International Journal of Computational Intelligence and Applications, vol.12, no.4, 2013.

[43] G. M. Weinberg, Quality Software Management, Volume 1: Systems Thinking, Dorset House Publishing Co., Inc., 1992.

[44] J. West, and S. Gallagher, Challenges of open innovation: the paradox of firm investment in open-source software, R&D Management, Vol. 36, No. 3, pp. 319-331, 2006.

[45] X. Xia, L. Bao, D. Lo, P. Kochhar, A. Hassan, and Z. Xing, “What do developers search for on the web?” Empirical Software Engineering, vol.22, no.6, pp.3149-3185, 2018.

[46] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan, “An Empirical Study on Factors Impacting Bug Fixing Time”, Proc. of the 2012 19th Working Conference on Reverse Engineering, WCRE '12, Washington, DC, USA, IEEE Computer Society, pp. 225-234, 2012.

[47] H. Zhang, L. Gong, and S. Versteeg, “Predicting BugFixing Time: An Empirical Study of Commercial Software Projects”, Proc. of the 2013 International Conference on Software Engineering, ICSE '13, Piscataway, NJ, USA, IEEE Press, pp. 1042-1051, 2013.

[48] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, V. and B. Vasilescu, “The Impact of Continuous Integration on Other Software Development Practices: A Large-scale Empirical Study”, Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Piscataway, NJ, USA, IEEE Press, pp. 60-71, 2017.

[49] M. Zhou, A. Mockus, X. Ma, L. Zhang and H. Mei, In flow and Retention in OSS Communities with Commercial Involvement: A Case Study of

Three Hybrid Projects, ACM Trans. Softw. Eng. Methodol., Vol. 25, No. 2, pp.13:1-13:29, 2016.

[50] Z. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2012.