STUDYING THE EXTRINSIC INCENTIVES IN CROWDSOURCED SOFTWARE ENGINEERING ACTIVITIES

by

JIAYUAN ZHOU

A thesis submitted to the School of Computing

in conformity with the requirements for the

Degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

July 2020

Abstract

MAINTAINING the sustainability of Crowdsourced Software Engineering (SE) is challenging, especially for open source projects and online technical Q&A sites. Maintainers introduce non-monetary/monetary extrinsic incentives to attract crowd workers to participate in Crowdsourced Software Engineering activities.

Prior studies have examined the importance of extrinsic incentives in Crowdsourced SE by analyzing various practices of the non-monetary and monetary extrinsic incentives. For example, the cost-effective of monetary bounties in security bugs detection, the effectiveness of reputation rewards in motivating users to answer SE questions. However, little work has been conducted on how to leverage extrinsic incentives on supporting practitioners in developing (e.g., developing features) and maintaining (e.g., addressing issues) a software system.

In this Ph.D. thesis, we plan to understand the usage (i.e., the benefits and limitations) of monetary extrinsic incentives on the development and maintenance of software systems. Furthermore, since practitioners rely on online technical Q&A websites to get support and feedback from others, we also consider the usage of non-monetary extrinsic incentives on online technical Q&A websites. *[Jiayuan syas:Adjust the order. make it in a higher level.]* In particular, we plan to analyze the donations (i.e., a monetary extrinsic incentive) in GitHub open source projects, the issue bounties (i.e., a monetary extrinsic incentive) in GitHub open source projects, and the reputation bounties (i.e., a non-monetary extrinsic incentive) in online technical Q&A websites (e.g., Stack Overflow).

We believe that empirically understanding the usage of donations and issue bounties across open source projects and the usage of reputation bounties in online technical Q&A websites can provide practical suggestions in how to better use extrinsic incentives with the goal of operating projects, attracting more developers to address project issues and answer practitioners questions.

Our initial results show that the cost of operating open source projects are not just for engineering, but a good portion is used to attract people *[Jiayuan syas:and domain and host and communication]*. The timing of proposing issue bounties is important and the high-value issue bounties work better in projects that never use issue bounties before.

Acknowledgments

Co-authorship

Statement of Originality

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

CROWDSOURCED Software Engineering (SE) is the act of undertaking a software engineering task by an undefined, potentially large group of online workers in an open call format Mao et al. (2017). For example, open source development is one of the most popular forms of Crowdsourced Software Engineering. The use of crowdsourced knowledge on Stack Overflow to support software engineering activities is another form of Crowdsourced Software Engineering since the knowledge was collected from 'crowd workers' LaToza and Van Der Hoek (2015).

Crowdsourced SE has gained great success (e.g., Finifter et al. (2013); Maillart et al. (2017)). However, it is still a challenge to preserve the sustainability (e.g., the quality and the maintainability) of Crowdsourced SE. For example, 64% of well-known and popular open source projects rely on one or two contributors to manage most of their

tasks Avelino et al. (2016), and almost 95% of open source projects are no longer maintained after a year Rich Sands (2012). On Stack Overflow, 47.2% (8,023,388) of questions have yet to receive an appropriate answer.[1]

To keep the sustainability of Crowdsourced SE, extrinsic incentives were introduced to attract crowd workers to participate in Crowdsourced SE activities (e.g., addressing issues and answering questions). There are two types of extrinsic incentives, non-monetary and monetary. The non-monetary extrinsic incentives are related to reputation systems (e.g., reputation rewards and penalties), gamification (e.g., badges and privileges), and career rewards (e.g., delayed career benefits) Katmada et al. (2016), while the monetary extrinsic incentives are related to financial rewards (e.g., money and bitcoins).

Prior studies confirmed the importance of various practices of non-monetary extrinsic incentives (i.e., reputation systems and gamification) in Stack Overflow. However, no efforts have been conducted on supporting practitioners to leverage non-monetary extrinsic incentives on getting questions answered.

Other prior studies investigated various practices of monetary extrinsic incentives (e.g., vulnerability bounties, monetary prizes) in Crowdsourced SE. However, little work has been done on how to leverage monetary extrinsic incentives on maintaining activities of open source projects.

[1] https://data.stackexchange.com/stackoverflow/query/968466

1.1 Thesis Statement

In this Ph.D. thesis, we plan to understand the usage (i.e., the benefits and limitations) of monetary extrinsic incentives on the development and maintenance of software systems. Furthermore, since practitioners rely on online technical Q&A websites to get support and feedback from others, we also consider the usage of non-monetary extrinsic incentives on online technical Q&A websites. In particular, for monetary extrinsic incentive, we plan to analyze the donations in GitHub open source projects and the issue bounties in GitHub open source projects. For non-monetary extrinsic incentive, we plan to analyze the reputation bounties in Stack Overflow. *[Jiayuan sys:Check the order]*

We believe that empirically understanding the usage of donations across open source projects can help the project maintainers better estimate their operating budgets and expenses and attract more contributors. The investigation on the usage of issue bounties across diverse projects can provide practical suggestions for project maintainers on maintaining issue tracking systems, attracting developers to develop new features or fix existing bugs. Through analyzing reputation bounties in Stack Overflow we can provide valuable insights to help practitioners solve their development questions. Therefore, we propose the following thesis statement:

Thesis Statement: In crowdsourced SE, various practices of extrinsic incentives contain valuable information that can help us empirically understand the general benefits and limitations of extrinsic incentives. This yields practical suggestions to better manage a team's budget and attract more participants to address issues and answer development questions.

1.2 Thesis Overview

In this Ph.D. thesis, we first introduce the background of Stack Overflow and GitHub (Chapter 2). Then, we survey the state of the art research on studying the extrinsic incentives in Crowdsourced SE (Chapter 3). We then elaborate the details of each one of three study aspects of the extrinsic incentives in Crowdsourced SE *[Jiayuan syas:ACTIVITIES?]*. *[Jiayuan syas:THINK HOW TO REDUCE ABOVE.]* Lastly, we conclude the thesis (Chapter 7). Below, we briefly summarize the three chapters of our study on the three aspects of the extrinsic incentives on Crowdsourced SE.

1.2.1 Reputation bounties on Stack Overflow (Chapter ??)

1.2.2 Studying issue bounties on GitHub (Chapter ??)

1.2.3 Studying donation on GitHub *[Jiayuan syas:issue reports?]*(Chapter ??)

1.3 Thesis Contribution

In this thesis, we studied the non-monetary and monetary extrinsic incentives in Crowdsourced SE *[Jiayuan syas:ACTIVITIES]*. The results of our empirical studies highlight the value of extrinsic incentives in Crowdsourced SE in offering practical suggestions for developers, maintainers and so on *[Jiayuan syas:BS.... refine it]*. In particular, the thesis contributions are as follows:

1. Our initial results show that the cost of operating open source projects are not all about engineering-related (e.g., development) expenses, 13% of the cost is for

non-engineering-related expenses, for example, making stickers to attract new users.

2. The timing of proposing issue bounties is important and the high-value issue bounties work better in projects that never use issue bounties before.

3. Our results show that the timing of proposing bounties is the most important factor that is related to the issue-addressing likelihood, and high-value bounties work better in projects that first time use bounties.

CHAPTER 2

Background of Stack Overflow and GitHub

In this chapter, we provide a brief background of the Stack Overflow and GitHub.

2.1 Background on Stack Overflow

[Jiayuan sas:Maybe just introduce Stack Overflow in a general way, like introduce the badge, review and so on. And leave the bounty system in that part.]

2.1.1 The Question and Answer Process on Stack Overflow

Stack Overflow is one of the largest software developer communities in the world, with more than 50 million software developers using it every month. Developers ask and

answer questions on Stack Overflow, and they can upvote or downvote answers and questions to reflect their opinions. The score of a post is the sum of its up and down votes. Each question may have many answers, but only one answer can be accepted by the asker as the accepted answer. When a question gets an accepted answer, the question is *solved*.

Stack Overflow uses several gamification features, such as the reputation system, to motivate the members of its community to interact with each other through these questions and answers. For example, a user gains reputation points if the user's posts (i.e., questions or answers) receive upvotes from others. We can approximate how the question asking and answering-skills of a developer are perceived by the Stack Overflow community by looking at their reputation points. There are good reasons for users to have a good reputation on Stack Overflow. For example, Stack Overflow profiles are sometimes used during the recruitment process by software companies (such as Stack Overflow itself) as a measure of the technical knowledge of a developer. In addition, Stack Overflow users get elevated privileges, such as a reduced number of displayed ads on the website, as their reputation grows (Stack Overflow, 2019).

There exist two ways to consume reputation points: (1) by proposing bounties for a question to attract more attention from the community or to reward an existing answer; (2) by downvoting a post. In this study, we focus on the first way, in which users consume reputation points by proposing bounties.

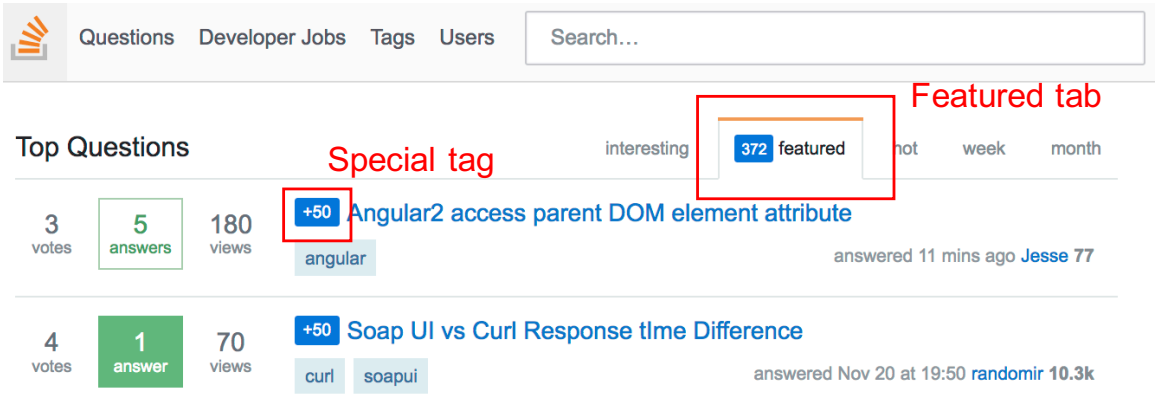


Figure 2.1: A screenshot of Stack Overflow's "featured" tab which highlights bounty questions.

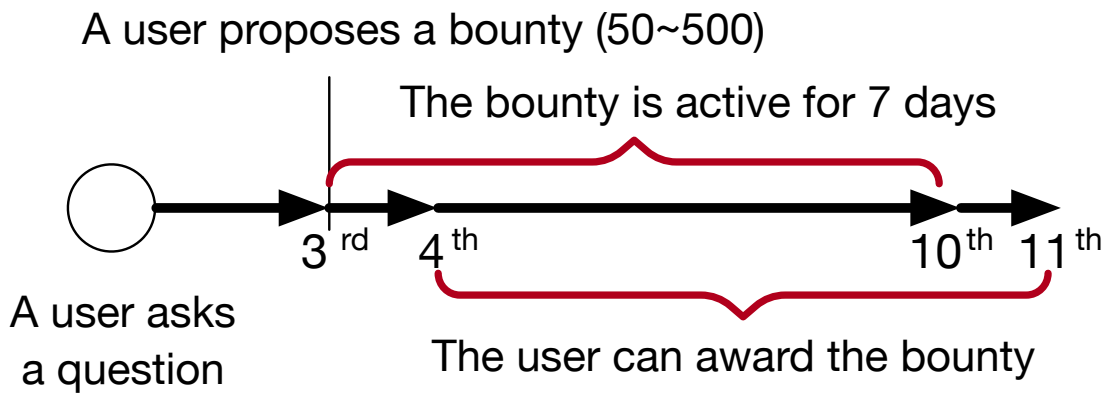


Figure 2.2: The life cycle of a bounty.

2.2 Background on GitHub

2.2.1 Issue tracking system on GitHub

The issue tracking system (i.e., ITS) on GitHub helps developers to manage the issue reports of their project. Users and developers can report bugs or request new features by posting an issue report on the issue tracking system. There are two statuses of an issue report: "open" and "closed". "Open" indicates that the issue report is still active and is

waiting to be addressed. “Closed” indicates that the issue report has been closed. The most common reason for closing an issue report is that the issue has been addressed, but it could also have other reasons (e.g., duplicated issue reports). Users can attach free-text labels to issue reports to indicate the category of an issue report. An issue report contains a title to summarize the issue and a detailed description of the issue. Developers can discuss an issue report by leaving comments, which can include code snippets, links or images to improve the description.

CHAPTER 3

Literature Survey

THIS Ph.D. thesis focuses on studying the benefits and limitations of extrinsic incentives in Crowdsourced SE.*[Jiayuan syas:Need to refine this sentense.]* There are two types of extrinsic incentives, non-monetary and monetary. A non-monetary incentive could be a form of recognition. For example, reputation is a form of non-monetary incentive. The higher the reputation of a worker in a community the more positive feedback that worker received – bringing the worker more benefits in the future (e.g., being noticed by potential employers). A monetary incentive is a monetary compensation. A crowd worker gets paid when a task is completed.

In this chapter, we first introduce our literature selection process, then we discuss the related work along with the aforementioned two types of extrinsic incentives.

3.1 Literature selection

Our literature review focuses on papers that are published in major software engineering journals and conferences. Table 3.1 lists venues from which we started our literature review. We focus our survey on papers that were published in the last 10 years (i.e., from 2010 to 2020). The methodology of our literature selection is based on Mao et al's Mao et al. (2017) survey of Crowdsourced SE. Firstly, we perform an online library search using Google Scholar search engine. We use following search terms: "bounty", "bounties", "incentives", "incentive", "extrinsic", "financial", "monetary", "crowdsourcing", and "crowdsourced". The search is for the papers of which the full-text contains at least one term from the search term list. Then we further read the full-text of the searched papers to select papers that are relevant to the categories of the extrinsic incentives and Crowdsourced SE. Secondly, we perform a reference search (snowballing) based on the selected paper to further identify relevant papers that were not published in the starting venues.

Our literature review is organized along two categories regarding non-monetary and monetary extrinsic incentives. We detail each category of papers below.

3.2 Non-monetary extrinsic incentives in Crowdsourced Software Engineering

Katmada et al. (2016) studied the incentive mechanisms in crowdsourcing and identified several types of incentive mechanisms: reputation systems (e.g., reputation rewards and penalties), gamification (e.g., badges and privileges), social incentive mechanisms (e.g., compliments), career rewards (e.g., delayed career benefits), and financial

Table 3.1: Names of conferences and journals as starting venues of the literature review

Venue Type	Venue Name	Abbreviation
Journal	IEEE Transactions on Software Engineering	TSE
Journal	ACM Transactions on Software Engineering and Methodology	TOSEM
Journal	Empirical Software Engineering	EMSE
Journal	Automated Software Engineering	ASE
Journal	Journals of Systems and Software	JSS
Conference	European Software Engineering Conference / ACM SIGSOFT Symposium on the Foundations of Software Engineering	ESEC/FSE
Conference	International Conference on Software Engineering	ICSE
Conference	International Conference on Automated Software Engineering	ASE
Conference	International Conference on Software Maintenance and Evolution	ICSME
Conference	International Conference on Software Analysis, Evolution, and Reengineering	SANER
Conference	International Conference on Mining Software Repositories	MSR
Conference	Conference on Human Factors in Computing Systems	CHI

rewards (e.g., monetary bounties). The reputation systems, gamification, and career rewards mechanisms utilize non-monetary extrinsic incentives.

Most prior work studied non-monetary extrinsic incentives in Stack Overflow. Stack Overflow combines a reputation system with gamification to motivate users to contribute. The reputation system in Stack Overflow enables users to earn reputations points for their efforts(e.g., providing high quality questions or answers).[1] With more reputation points, users can gain more **privileges** (e.g., the voting feature).[2] Users can also win community recognitions (e.g., **badges**) through making contributions.[3]

The reputation rewards and penalties in Stack Overflow. The rewards and penalties are based on users' contributions, such as answering questions, editing questions or answers. When the contributions of users are voted up by peers, the users will gain

[1] https://stackoverflow.com/help/whats-reputation
[2] https://stackoverflow.com/help/privileges
[3] https://stackoverflow.com/help/badges

reputation points, otherwise the users will lose reputation points. Lotufo et al. (2012) conducted a statistic analysis to investigate the reputation rewards and penalties in Stack Overflow. They observed that the reputation reward mechanism motivates users to edit their peers' questions or answers. They also observed the reputation penalty mechanism improves users' contribution quality effectively.

Reputation bounties in Stack Overflow. Users in Stack Overflow can use their reputation points to offer bounties for the unsolved questions that they are interested in and other users will try to solve the questions and receive the reputation points. Berger et al. (2016) studied the response time of Stack Overflow questions that have reputation bounties. They observed the negligible predictive power of text based features, (e.g., number of images and number of verbs that indicate action) in predicting whether a bounty question will receive an answer within 2.5 days.

Privileges in Stack Overflow. Privileges are designed as an extrinsic incentives in the gamification. In Stack Overflow, some features (e.g., the voting feature) are limited to users and the users have to gain privileges to access such restricted features. Lotufo et al. (2012) conducted an empirical investigation of privileges in Stack Overflow. They analyzed users' contributions frequency before and after being awarded privileges, observing that rewarding privileges are associated with an increased contribution frequency of users.

Badges in Stack Overflow. A badge is a widget used on a website, showing the perceived expertise and community respect of an user. In Stack Overflow, users get badges by completing specific contributions. For example, a "Yearling" badge indicating a user is an active member for a year, earning at least 200 reputation points. Grant and Betts (2013) conducted an initial exploratory study on Stack Overflow user behaviors

through three badges. They observed that badges encourage users to edit questions for a higher quality on both an individual and global level.

Wei et al. (2015) studied the effectiveness of reputation and badges in Stack Exchange, which is a network of online Q&A websites, of which Stack Overflow is a flagship site. They conducted a quantitive study using regression models, observing that reputation-ranking-related badges rather than reputation itself motivate user contributions. Li et al. (2012) and Anderson et al. (2013) analyzed the impact of badges on user engagement in Stack Overflow. They observed a "badge steering" phenomena, where users make significantly more contributions after getting badges. Yanovsky et al. (2019) classified three groups of users that have different levels of reactions towards the phenomena. They proposed a model to predict whether users will move to a group that has lower frequency and less intensity of contribution.

Halavais et al. (2014) studied social influences and badge acquisitions on Stack Overflow. They constructed a user social network using the co-posting behavior of users and analyzed the appearance of the general badges and the topically-constrained (i.e., the tag) badges. They observed a weak relationship between social influence and the badge adoption of a user.

Delayed career benefits in Apache projects and Stack Overflow. The delayed career benefits is a type of career rewards, which involve the rewards from future employers, such as higher compensations or more attractive jobs. In Apache projects, developers have different ranks and the rank is based on their contributions. Hann et al. (2002) conducted a quantitive study on delayed career benefits in Apache projects observing that the higher contribution-based rank of developers is associated with the higher compensations in the future. Xu et al. (2020) studied the delay career benefits in Stack

Overflow. They conducted an empirical analysis on Stack Overflow and Stack Overflow Careers (SOC), a software engineering oriented job matching website. For the same set of Stack Overflow users, they compared the user contributions before and after a job change happens in SOC. They observed a decrease in user contributions after users find new jobs.

Non-monetary extrinsic incentives are critical for the sustainability of Crowdsourced SE, especially for maintaining active, valuable activities in Stack Overflow. However, no efforts have been conducted on how non-monetary extrinsic incentives help developers get questions solved. *[Jiayuan syas:Old: However, no efforts have been conducted on supporting practitioners leveraging on non-monetary extrinsic incentives.]*

3.3 Monetary extrinsic incentives in Crowdsourced Software Engineering

Several studies have examined various practices of monetary extrinsic incentives in Crowdsourced SE.

Vulnerability bounties in crowdsourced software security platforms. In Crowdsourced SE, a 'vulnerability' or a 'bug' bounty are monetary rewards for the discovery of software security flaws. Many crowdsourced software security platforms offer Vulnerability Reward Programs (VRPs), commonly referred to as bug bounty programs (BBPs), for software vendors to propose vulnerability bounties and for bounty hunters to receive compensations for their vulnerability discovery efforts.

Zhao et al. (2015) examined the role of the vulnerability bounties in vulnerability discovery activities by constructing a linear regression model to predict the number of

discovered vulnerabilities monthly. They observed that a significantly strong positive correlation between the value of a bounty and the number of reported vulnerabilities. Finifter et al. (2013) analyzed the VRPs for Chromium and Firefox. They observed that the VRPs of both projects are more cost-effective than the cost of hiring full-time security researchers. Maillart et al. (2017) have a similar observation regarding the cost-effective phenomenon in VRPs.

Zhao et al. (2017) and Maillart et al. (2017) analyzed the effect of different VRP policies from the HackerOne and BugCrowd platforms in an effort to improve VRPs. For example, Maillart et al. suggested software managers dynamically adjust the price of bounties according to the market situation (e.g., increase the monetary value of a bounty when releasing a new version).

Zhao et al. (2014) investigated the characteristics of hunters in Wooyun[4] platform, observing that the diversity of hunters improved the productivity of the vulnerability discovery process. Hata et al. (2017) conducted a quantitative and qualitative user survey to understand the characteristics of vulnerability bounty hunters. They observed that most hunters are not project-specific and that VRP managers should strive to attract non-project-specific security specialists with reasonable bounties.

Prior vulnerability-bounty-related studies investigated vulnerability bounties from economic, price strategies, and characteristics of practitioners perspectives. However, these findings cannot be generalized to other Crowdsourced SE activities due to the different mechanisms of SE tasks. For the security bug detection, the security flaws are unknown until being detected. While for the other SE tasks, the requirements are clear.

Issue bounties in open source projects. Different from vulnerability bounties which are concerned with unknown security flaws, issue bounties are monetary incentives for

[4] http://www.wooyun.org/, last accessed date: 2016-07-17

general software tasks, such as software development tasks, code maintenance tasks, and documentation tasks. Issue bounties are proposed for already-known issues, and once a developer addresses such known issues, the bounty backers can decide to reward the hunters or reject the contributions. Issue bounties are widely used in open source projects. For example, practitioners can propose bounties for issues in GitHub open source projects via the Bountysource[5] or the Bountify[6] platforms.

Little work has investigated issue bounties. Kanda et al. (2017) conducted a preliminary study on Bountysource issue bounties in GitHub open source projects. By comparing the closing rate and closing-time between bounty and non-bounty issues, they observed that the closing-rate of bounty issues is lower than that of non-bounty issues, and it takes longer for the bounty issues to get closed than non-bounty issues. However, Kanda et al.'s findings are not generalizable due to the lack of control factors. For example, the popularity of the projects is a control factor that may also potentially affect the closing-rate of issues.

Monetary prizes in commercial crowdsourcing platform. Topcoder[7] is one of the commercial crowdsourcing platforms that is built to support Crowdsourced SE. According to customers' requirements, Topcoder will publish software development challenges (e.g., software implementation) with monetary prizes. Developers can receive monetary prizes once they complete the challenge and their solutions get accepted. Mao et al. (2013) and Alelyani et al. (2017) proposed empirical pricing models to propose the appropriate price for challenges in TopCoder. Wang et al. (2019) conducted an exploratory study on the strategic pricing and worker performance in TopCoder. Wang et al. identified two pricing strategies and developed an algorithm

[5] https://www.bountysource.com/
[6] https://bountify.co/
[7] https://www.topcoder.com/challenges

to analyze the impact of strategies on worker performance. Wang et al. observed that higher-priced tasks can get higher worker performance.

Prior monetary-prize-related studies has yielded important results for predicting prices for TopCoder challenges. However, their findings cannot generalize to the price prediction for SE tasks in open source projects since the challenge in TopCoder is usually related to develop a complete software system, which is different from completing a SE task.

Donations in open source projects. Donations play an important role in the smooth operation of open source projects.[8] Krishnamurthy and Tripathi (2009) investigated the factors that impact donations in an open source software community (e.g., *Sourceforge*)[9] and observed that donations are associated with the community involvement hours and rational commitment of donors. For example, a donor who has a longer association and participation in the community, is more likely to donate more money.

Nakasai et al. (2017, 2018) studied donations in the Eclipse community. They analyzed Eclipse donations in terms of donor's motivations and roles, observing that donation badges and new releases can motivate donors to make donations. They observed that Eclipse developers respond faster to the bug reports which are reported by users that have donation badges.

Software foundations are non-profit organizations, aiming to provide the needed fundings for open and collaborative software development. Software foundations are important sources of donation for crowdsourced software engineering. Izquierdo and Cabot (2018) studied the role (e.g., an advisory or life governance) of software foundations in open source projects. They analyzed the openness and the influence of 18

[8] https://opensource.guide/getting-paid/
[9] https://sourceforge.net/

foundations in the development of open source projects. They observed most of the foundations' missions are providing legal support and leading evangelizing actions.

Yukizawa et al. (2019) promoted new strategies to help open source projects attract donations. They observed that the some phrases that will psychologically encourage donors to make donations. For example, the soft phase "even a single dollar helps" will increases donors' compliance to make donations.

Overney et al. (2020) conducted a mixed-method empirical study on the prevalence, outcomes, and impact of donations in npm (i.e., Node Package Manager) packages and GitHub open source projects. The authors observed that a small fraction of npm packages and GitHub open source projects have donations and these packages and projects are more active, more mature, and more popular than others. The authors conducted a time-series analysis on the effects of donations in terms of project activities and observed. The result showed that there is no strong evidence of the impact of donations on the activity level of a project.

Krishnamurthy (2006) studied the motivations of open source developers in terms of intrinsic-extrinsic incentives, observing empirical evidence of monetary extrinsic incentives that motivate developers to contribute. Krishnamurthy et al. (2014) further studied the acceptance of monetary rewards in open source software development. They observed that intrinsic (e.g., the need for a creative task) and extrinsic (e.g., financial benefits) motivations positively influence the willingness of developers to accept monetary rewards, while community motivation (e.g., contributing to a social community) negatively influences the willingness of developers to accept the monetary rewards.

Prior studies confirm the importance of monetary extrinsic incentives in Crowdsourced SE from the price of monetary rewards, characteristics of participants perspectives. However, little work has been done to provide valuable insights and suggestions on supporting practitioner develop and maintain open source projects.

CHAPTER 4

Studying reputation bounties on Stack Overflow

Technical question and answer (Q&A) websites provide a platform for developers to communicate with each other by asking and answering questions. Stack Overflow is the most prominent of such websites. With the rapidly increasing number of questions on Stack Overflow, it is becoming difficult to get an answer to all questions and as a result, millions of questions on Stack Overflow remain unsolved. In an attempt to improve the visibility of unsolved questions, Stack Overflow introduced a bounty system to motivate users to solve such questions. In this bounty system, users can offer reputation points in an effort to encourage users to answer their question. In this chapter, we study 129,202 bounty questions that were proposed by 61,824 bounty backers. We observe that bounty questions have a higher solving-likelihood than non-bounty questions. This is particularly true for long-standing unsolved questions. For example, questions that were unsolved for 100 days for which a bounty is proposed are more likely to be solved (55%) than those without bounties (1.7%). In addition, we studied the factors that are important for the solving-likelihood and solving-time of a bounty question. We found that: (1) Questions are likely to attract more traffic after receiving a bounty than non-bounty questions. (2) Bounties work particularly well in very large communities with a relatively low question solving-likelihood. (3) High-valued bounties are associated with a higher solving-likelihood, but we did not observe a likelihood for expedited solutions. Our study shows that while bounties are not a silver bullet for getting a question solved, they are associated with a higher solving-likelihood of a question in most cases. As questions that are still unsolved after

two days hardly receive any traffic, we recommend that Stack Overflow users propose a bounty as soon as possible after those two days for the bounty to have the highest impact.

An earlier version of this chapter is published in the Empirical Software Engineering Journal (EMSE) (Zhou et al., 2019).

4.1 Introduction

ONLINE technical Q&A sites have become more and more important for software developers to share knowledge. Developers can post questions on such Q&A sites and receive answers from other developers. Stack Overflow[1] is a prominent example of such a Q&A site. Stack Overflow has more than 16.8 million questions, 25.9 million answers, and 9.7 million users.[2]

Stack Overflow has become an important source on which developers rely to solve various software engineering problems (Ahasanuzzaman et al., 2018; Wu et al., 2018). For example, developers post questions on Stack Overflow about their programming problems, in the hope of receiving helpful responses. However, with the rapid increase of the number of questions on Stack Overflow, solving all the questions has become a challenge for the community. Although many of the questions on Stack Overflow are solved quickly (the median waiting time is less than one hour (Wang et al., 2018a)), 47.2% (8,023,388) of the questions are not solved at all.[3]

Some customer-driven crowd-sourcing marketplaces, such as Fenda (China)[4] and Whale (US)[5], introduced monetary incentives to motivate users to make contributions (Jan et al., 2017). In contrast, Stack Overflow uses gamification in the form of a point-based reputation system to motivate users to make a contribution (e.g., answering questions or revising posts). To motivate users through gamification, Stack Overflow introduced a bounty system. Through this bounty system, *bounty backers* can offer reputation points by proposing a bounty for the user who answers

[1] https://stackoverflow.com/
[2] https://data.stackexchange.com/
[3] https://data.stackexchange.com/stackoverflow/query/968466
[4] https://fd.zaih.com/fenda
[5] https://techcrunch.com/2016/10/31/justin-kan-launches-video-qa-app-whale/

a question. Although bounties have been used since January 2009,[6] the association between bounties and the solving-likelihood and solving-time of a question have never been examined. By understanding this association, we could provide insights on how to better leverage bounties to solve questions.

[Jiayuan syas:Consider to user description command to show the research questions.]

In this paper, we perform a large-scale analysis of the bounty system of Stack Overflow by studying 129,202 bounty questions that were proposed by 61,824 bounty backers. We first conduct a preliminary study in which we uncover that bounty questions have a higher solving-likelihood than non-bounty questions. We show that bounties work particularly well for solving long-standing unsolved questions, and for solving questions in very large communities with a relatively low question solving-likelihood.

In addition, we study in depth which factors are important for the solving-likelihood and solving-time of bounty questions. Finally, we study the impact of bounties on the traffic to questions. The main findings of our study are as follows:

1. Questions are likely to attract more traffic after receiving a bounty than non-bounty questions, particularly when the value of the bounty is high (i.e., 400). In addition, bounty questions have a higher solving-likelihood than non-bounty questions, especially for questions in very large communities with a relatively low question solving-likelihood.

2. Bounty questions with a higher bounty value have a higher solving-likelihood, however, a higher bounty value does not expedite a bounty question getting solved.

[6] https://stackoverflow.blog/2009/01/27/reputation-bounty-for-unanswered-questions/

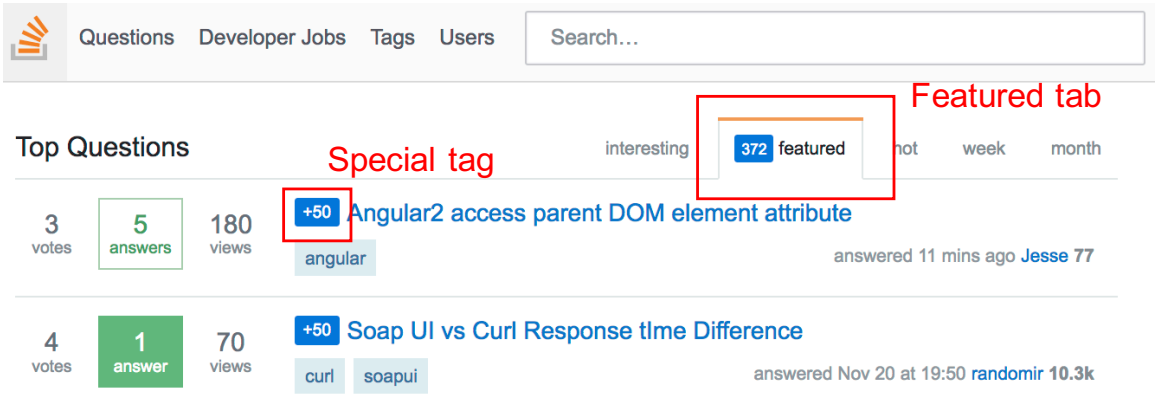


Figure 4.1: A screenshot of Stack Overflow's "featured" tab which highlights bounty questions.

3. Bounty questions tend to have a higher solving-likelihood than non-bounty questions, particularly when focusing on long-standing unsolved questions. For example, questions that were unsolved for 100 days for which a bounty is proposed are more likely to be solved (55%) than those without a bounty (1.7%).

Our study shows that while bounties are not a silver bullet for getting a question solved, they are associated with a higher solving-likelihood in most cases. As questions on Stack Overflow generally are not solved at all if they remain unsolved after two days, we recommend that users post their bounty as soon as possible after these two days.

[Jiayuan syas:Remove the organization.]

4.2 Background

4.2.1 The Bounty System on Stack Overflow

Stack Overflow has a bounty system that allows users to offer reputation points for any user that would produce an accepted answer to a question, in an effort to draw more

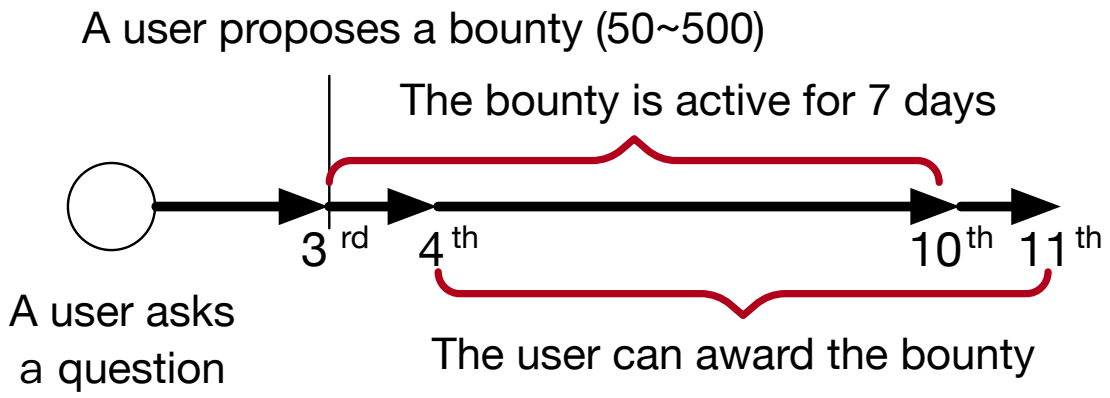


Figure 4.2: The life cycle of a bounty.

attention from users across the site. Figure 4.2 shows the life cycle of a bounty. When a user asks a question, anyone can propose a bounty on that question after two days, thereby becoming a *bounty backer*. A question can only have one active bounty at any time. In other words, one cannot propose another bounty on a question if the question already has an active bounty at that moment. Note that when a bounty is proposed, the reputation points that are offered in the bounty are removed immediately from the bounty backer's reputation (and are never refunded even if the question remains unsolved at the expiry of the bounty).

Users can propose a bounty with a value between 50 and 500 reputation points, in 50-point increments. A bounty can be active for a maximum of seven days. While a bounty is active, the bounty question is labeled with a special tag that highlights its associated bounty value. The question itself is highlighted in the "featured" tab on the Stack Overflow homepage (see Figure 4.1) to help draw attention from the community towards that question.

A bounty can be awarded to an answer by the bounty backer one day after it was proposed. If the bounty backer does not explicitly award the bounty, it will be automatically awarded one day after the expiry date of the bounty. The rules for the automated awarding of bounties are:[7]

1. If the bounty backer is the original question asker, the bounty will be awarded to the answer that was accepted while the bounty was active.

2. An answer that was created after the bounty was offered which has more than one vote (but was not accepted) will be awarded half of the bounty value. If there are multiple answers that meet this criterion, the bounty is awarded to the earliest answer.

3. If no answer meets the above two criteria the offered reputation points are discarded.

When a bounty question gets an answer which is awarded the bounty, we define the bounty question as *solved*. Note that the awarded answer can also be an answer which is not accepted by the question asker.

4.3 Data Collection

StackExchange (Stack Exchange, 2017) provides a Stack Exchange Data Dump,[8] which is composed of a set of XML files that contain data about all questions, answers, tags, votes, and user histories of Stack Overflow. We use the following files from this set:

1. *Posts.xml* contains data about posted questions and answers.

[7] https://stackoverflow.com/help/bounty
[8] https://archive.org/details/stackexchange

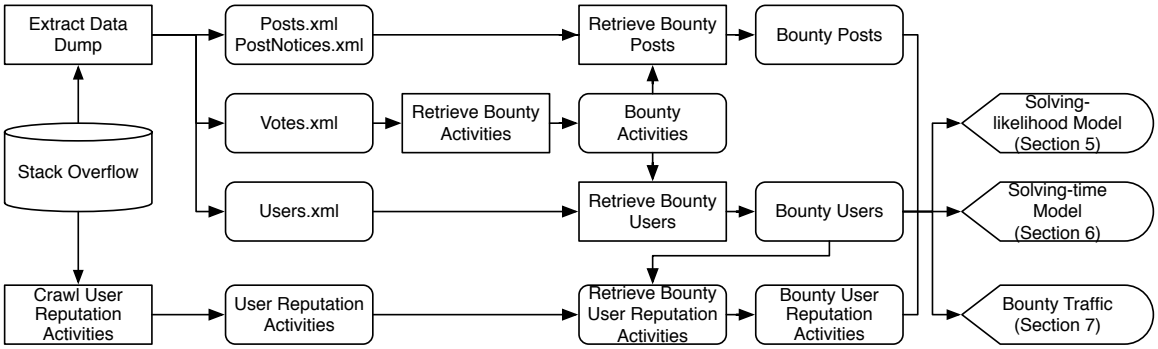


Figure 4.3: An overview of our data collection process.

2. *PostNotices.xml* contains the reasons for offering each bounty.

3. *Votes.xml* contains data about activities, such as the date on which a question was upvoted. *Votes.xml* also contains bounty activity information. For example, the creation and closure date of a bounty, the bounty value, the id of the related user who proposed or won the bounty, and the id of the related question or answer.

4. *Users.xml* contains data about users, such as the user id, the creation date of their accounts, and their reputation at the time of the data archival.

Figure 4.3 gives an overview of our data collection process. We first downloaded the data dump of Aug. 27, 2017. Because the last major change to Stack Overflow's bounty system was made on Sep. 23, 2011,[9] we only study bounties that were proposed between Sep. 23, 2011 and Aug. 27, 2017. Then we collected the data as follows:

1. We first retrieved the bounty activity information from *Votes.xml*. Then we retrieved the posts and users that are associated with the selected bounties accordingly.

[9] https://stackoverflow.blog/2011/09/23/bounty-reasons-and-post-notices/

Table 4.1: Dataset description.

Period	Sep. 23, 2011 to Aug. 27, 2017
Number of bounty questions	129,202
Number of expired bounties	44,635
Number of bounty backers	61,824
Number of non-bounty questions	12,359,663

2. We crawled the history of reputation activities from each user's profile page on Stack Overflow,[10] and we traced back their reputation activities to the moment of proposing a bounty.

We observed that for some bounty questions, the available data about the life cycle of the bounty is incomplete. For example, the question "How to detect which one of the defined font was used in a web page?"[11] only shows when the bounty was rewarded, but not when it was created. After removing the bounty questions with an incomplete bounty life cycle from our data, our dataset contains 129,202 bounty questions, which involved 61,824 bounty backers who proposed bounties, and 12,359,663 non-bounty questions. Table 4.1 gives an overview of our studied dataset.

There exist several 'special' cases of bounties, in which the bounty was used for a purpose other than getting a question solved. To avoid bias in our study, we treat the following cases separately:

1. Bounties that were proposed to reward existing answers. Such bounties can be filtered easily as the bounty was created with the reason "Reward existing answer".

[10] https://stackoverflow.com/users/userid?tab=reputation
[11] https://stackoverflow.com/questions/845/

2. Bounties that were automatically awarded by Stack Overflow. A bounty that was awarded automatically does not reflect that the bounty backer is satisfied with the answer.

3. Bounties that were proposed while the question already had an accepted answer. For example, a bounty was offered with the purpose of drawing attention to a question on April 14, 2012.[12] However, the bounty was eventually awarded to the answer that was already given on April 6, 2012.

We discuss the first case in more detail in Section 4.6. In the second case, we cannot distinguish whether the bounty question was actually solved, as the rewarded answer is not necessarily a solution to the question. The third case is difficult to recognize automatically, as we cannot distinguish between whether the bounty backer wanted to reward the existing answer, or was looking for additional answers. Hence, we remove bounties of these types and their associated questions from our dataset. Note that we keep all unsolved bounty questions for which the bounty expired. After separating the special bounty cases, our dataset contains 79,093 bounty questions.We published our data online.[13]

4.4 Preliminary Study

In this section, we first present basic descriptive statistics about bounties. Then, we discuss the impact of bounties on the solving-likelihood of bounty questions across Stack Overflow tags.

[12] https://stackoverflow.com/posts/10038098/revisions

[13] https://github.com/SAILResearch/supportmaterial-18-jiayuan-SO_bounty/tree/master/data_model

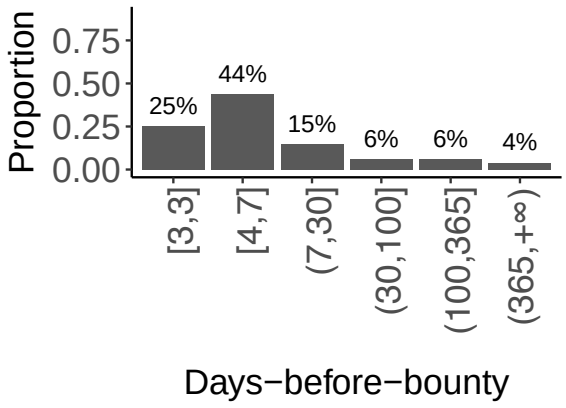


Figure 4.4: The proportion of bounty questions across different values of the days-before-bounty metric.

4.4.1 Basic descriptive statistics of bounties on Stack Overflow

We present basic descriptive statistics about bounties from the following perspectives: (1) the solving-likelihood of a question, (2) the number of days between the creation of a question and the proposal of its first bounty (i.e., the *days-before-bounty* metric), (3) the solving-time of a bounty question after the bounty is proposed, and (4) the bounty value. From these statistics, we get a basic overview of bounties on Stack Overflow.

Results:

In general, bounty questions have a higher solving-likelihood than non-bounty questions. The solving-likelihood of bounty questions is 65.5% which is 30% higher than that of non-bounty questions (i.e., 48.9%). Especially for the questions with more than one bounty, the solving-likelihood is 92.0%.

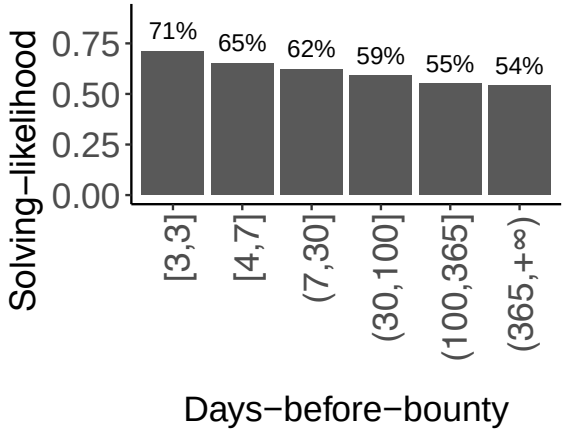


Figure 4.5: The solving-likelihood of bounty questions across different values of the days-before-bounty metric.

Long-standing unsolved questions with bounties are more likely to be solved than those without bounties. Prior work (Anderson et al., 2012) showed that questions either get solved very quickly, or not at all. Figures 4.4 and **??** show the proportion and solving-likelihood of bounty questions for different values of the days-before-bounty metric. 69% of the bounties were proposed within one week while only 10% of the bounties were proposed after 100 days since the creation of a question. However, the solving-likelihood for such"late bounty questions" is around 55% (i.e., 2,605 out of 4,776 questions). In comparison, only 104,831 out of 6,321,124 (1.7%) of the non-bounty questions that were unsolved 100 days after their creation were solved afterwards. Hence, long-standing unsolved questions with bounties are more likely to be solved than those without bounties.. The time after which a bounty is proposed is related the solving-likelihood: 25% of the bounties were proposed three days after the creation of the question, which is the earliest date on which it is allowed to propose a bounty. The solving-likelihood of these bounties is the highest (i.e., 71%).

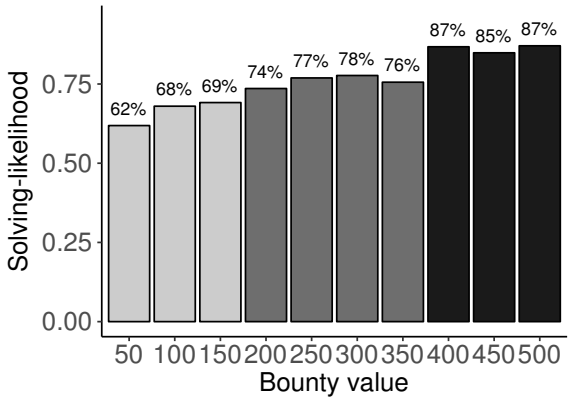


Figure 4.6: The distribution of the solving-likelihood across different bounty values. The bars are marked with different shades to indicate the levels of solving-likelihood that we distinguished.

Bounty questions with higher-valued bounties have a higher solving-likelihood. However, higher bounty values are not associated with expedited solutions. Figure 4.6 shows the solving-likelihood of bounty questions across different bounty values. In general, the solving-likelihood increases as the bounty value increases. In particular, there is a large difference in solving-likelihood (62.9% vs. 88.1%) between the lowest and highest bounty values (50 and 500).

We grouped the bounty values into three groups (showed by different shades in Figure 4.6) that correspond to partitions of 10% of the solving-likelihood (i.e., 60% to 70%, 70% to 80% and 80% to 90%) for our study in Section 4.5.4.

Figure **??** shows the solving-time of bounty questions for different bounty values. Counter-intuitively, we do not observe a clear association between the bounty value and the solving-time. The correlation between the solving-time and value is -0.02 which indicates a weak association.

The solving-likelihood and the solving-time of a bounty question varies across tags. When posting a question, the question asker can assign one or more tags to the

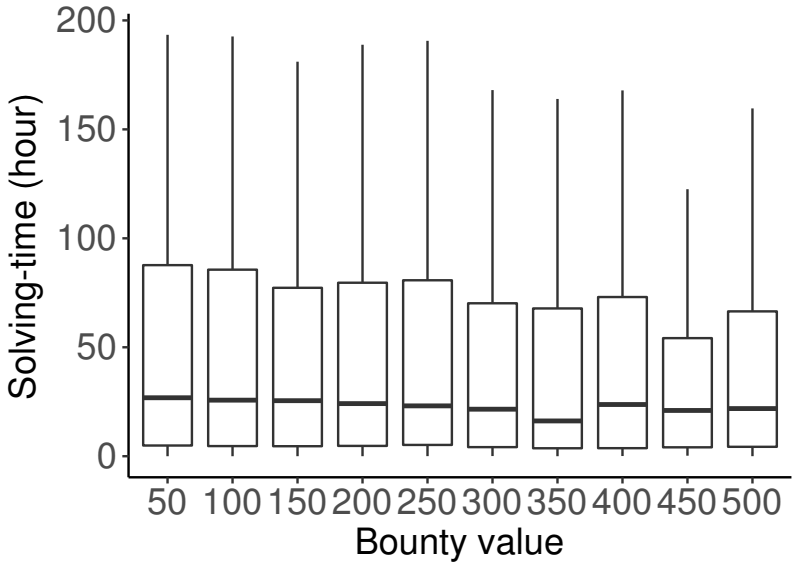


Figure 4.7: The distribution of the solving-time of bounty questions across different bounty values. The bars are marked with different shades to indicate the levels of solving-likelihood that we distinguished.

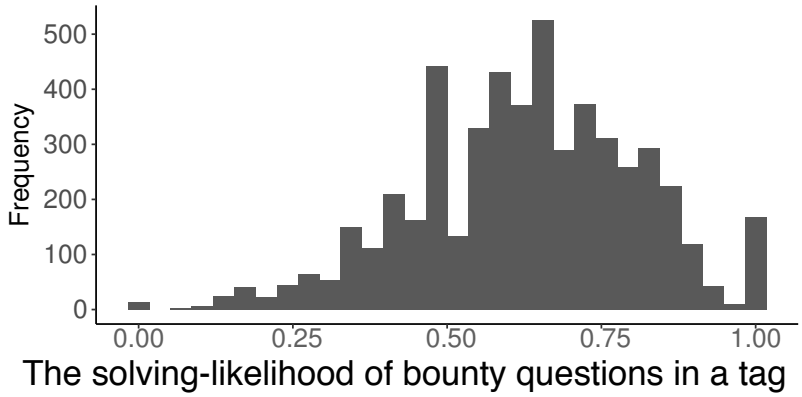


Figure 4.8: Solving-likelihood

Figure 4.9: The distribution of the solving-likelihood of the tags of bounty questions. Each data point in the distribution represents one tag.

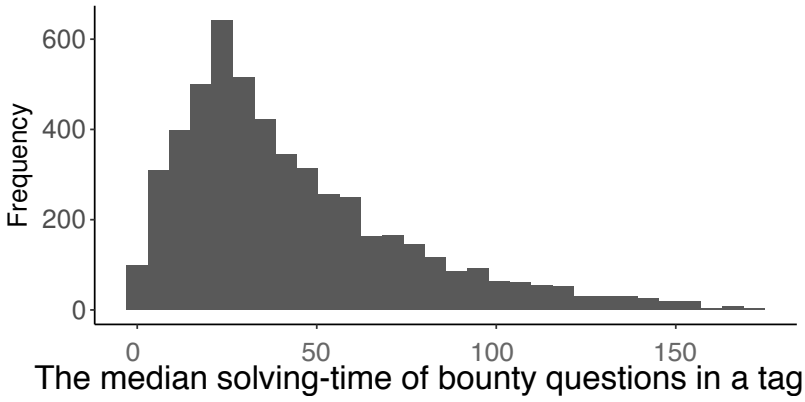


Figure 4.10: The distribution of the solving-time of the tags of bounty questions. Each data point in the distribution represents one tag.

question to attract more targeted traffic. However, some of these tags are more popular than others. To reduce the bias caused by tags which have only a few bounty questions, we only consider the tags which have more than five bounty questions for the following two figures. Figures 4.8 and 4.4 show the frequency of tags in terms of the solving-likelihood and solving-time of bounty questions, respectively. We observed that the solving-likelihood and solving-time of bounties vary across tags. For example, the solving-likelihood of bounty questions with the *applescript-studio* tag is 70%, while the solving-likelihood of questions with the *xcode9-beta* tag is 40%. In the remainder of this section, we look in more detail into the impact of bounties on the solving-likelihood of bounty questions across tags.

Summary: Bounty questions have a higher solving-likelihood than non-bounty questions. Bounties appear to work especially well for long-standing unsolved questions. Bounty questions with a higher bounty value have a higher solving-likelihood. However, there is no association between a bounty's value and its solving-time, which implies that a higher bounty value does not expedite the solving of a bounty question.

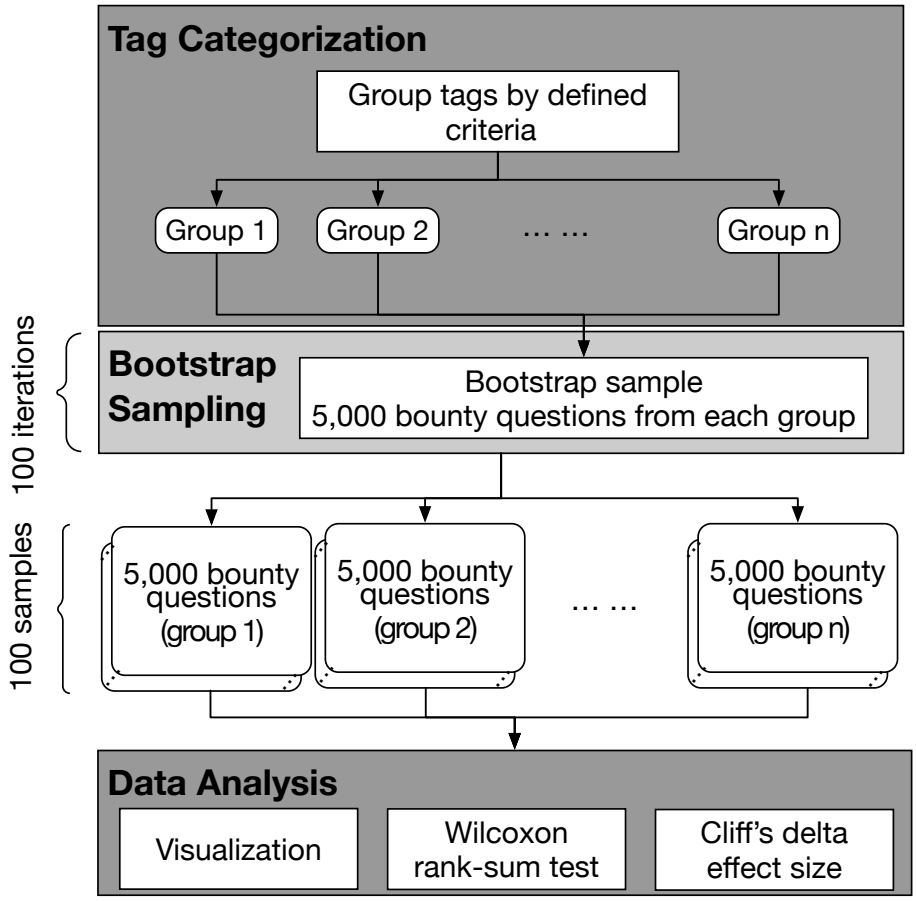


Figure 4.11: Overview of our approach for studying the relation between bounties and the solving-likelihood of bounty questions across tags.

4.4.2 The Association between Bounties and the Solving-likelihood of Bounty Questions across Tags

Figure 4.8 shows that the solving-likelihood of bounty questions differs across tags. In this section, we study how bounties impact the solving-likelihood of bounty questions across answerer communities (i.e., tags) of different sizes and with varying question solving-likelihoods. The population of answerers within a community indicates the size of the community.

Approach:

We first grouped all tags by their size (*size-based*) and question solving-likelihood (*community-quality-based*). Then we used a bootstrap sampling approach to sample tags and questions in each group in order to ensure the statistical stability of our observations. Finally, we studied the solving-likelihood of questions across the size-based and community-quality-based groups. Figure 4.11 gives an overview of our approach. We detail each step below.

Step 1: Tag categorization. Since the answerer population for tags ranges from 1 to 386,885, we grouped the tags into four size-based groups according to the order of magnitude of their answerer population. We created the community-quality-based groups by grouping the tags according to their solving-likelihood for non-bounty questions in intervals of 0.25. To summarize, the tags were grouped based on the following criteria:

Criteria for size-based categorization:

- **Small:** The answerer population of a tag is smaller than 1,000.
- **Moderate:** The answerer population of a tag is between 1,000 and 10,000.
- **Large:** The answerer population of a tag is between 10,000 and 100,000.
- **Very large:** The answerer population of a tag is larger than 100,000.

Criteria for community-quality-based categorization:

- **Micro:** The tag's non-bounty question solving-likelihood is less than 0.25.
- **Small:** The tag's non-bounty question solving-likelihood is between 0.25 and 0.50.

Table 4.2: The distribution of 20,180 bounty-related tags across the size and skill-based groups.

Size-based Categorization			**Community-quality-based Categorization**		
Group Name	**#Tags**	**#Questions**	**Group Name**	**#Tags**	**#Questions**
Small	16,540	47,457	Micro	904	1,416
Moderate	3,182	78,519	Small	10,507	114,567
Large	439	94,320	Medium	8496	166,339
Very Large	19	62,607	High	273	582

- **Medium:** The tag's non-bounty question solving-likelihood is between 0.50 and 0.75.
- **High:** The tag's non-bounty question solving-likelihood is more than 0.75.

Table 4.2 shows the distribution of tags across the size and skill-based groups. To reduce the bias that is caused by the unbalanced number of tags and questions across groups, we employed bootstrap sampling.

Step 2: Bootstrap sampling. We applied a bootstrap sampling approach to sample bounty questions of each size and skill-based group. We first randomly sampled 5000 tags from each group with replacement. Then we randomly sampled one bounty question from each sampled tag, to reduce the bias towards tags with more bounty questions. Hence, we sampled 5,000 bounty questions for each group. To ensure the statistical robustness of our results, we repeated our bootstrap sampling process 100 times with different random seeds. We ended up with 100 samples and for each sample, there are 20,000 bounty questions for the size-based groups and 20,000 for the skill-based groups (5,000 bounty questions for each group).

Step 3: Data analysis. For each sample, we calculated the solving-likelihood across the size and skill-based groups. To compare the differences between two distributions, we

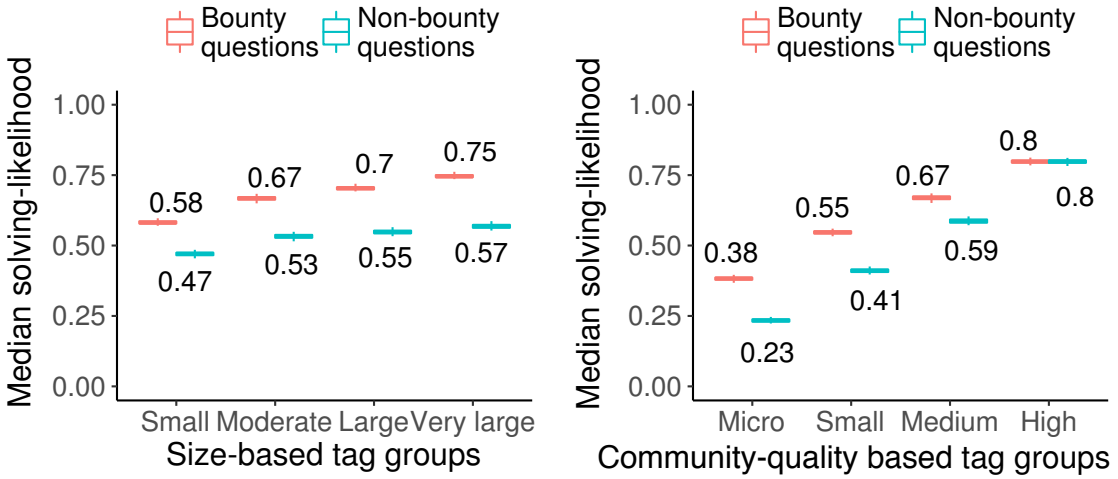


Figure 4.12: The distribution of the median solving-likelihood across the size-based and skill-based tag groups for the 100 studied samples for bounty and non-bounty questions.

used the Wilcoxon rank-sum test (Bauer, 1972), which does not require the distribution to be normally distributed. We also performed the Bonferroni correction (Bonferroni, 1936) to correct the p-values for multiple comparisons. Furthermore, we applied Cliff's delta d effect size (Long et al., 2003) to quantify the magnitude of the differences. We use the following thresholds for d (Romano et al., 2006): $|d| \leq 0.147$ (negligible); $0.147 < |d| \leq 0.33$ (small); $0.33 < |d| \leq 0.474$ (medium); $0.474 < |d| \leq 1$ (large).

Results:

The solving-likelihood of bounty questions is higher than that of non-bounty questions across all size-based tag groups. Figure 4.12 shows the distribution of the solving-likelihood of bounty and non-bounty questions across the size-based tag groups. For all size-based groups, the solving-likelihood of bounty questions is significantly higher than that of non-bounty questions (with a large effect size). The

solving-likelihood of both bounty and non-bounty questions increases as the size of the tag group gets larger. A possible explanation is that a large community has more answerers, so there is a higher chance of someone solving the bounty.

The solving-likelihood of bounty questions is higher than that of non-bounty questions that are asked in communities with a lower question solving-likelihood. Figure 4.12 shows the distribution of the solving-likelihood of bounty and non-bounty questions across the community-quality-based tag groups. The solving-likelihood of bounty questions is higher than that of non-bounty questions that are asked in different community-quality-based groups except the 'High' group. A possible explanation is that as the solving-likelihood in the 'High' group is already very high (80%), it is hard to improve – the unsolved questions may be too hard or unclear to answer. We also observe a few tag outliers in which the solving-likelihood of bounty questions is lower than that of non-bounty questions while still having many bounties. For example, the "*flash-builder*" tag has 50 bounty questions although the solving-likelihood of its bounty questions is 0.26, which is much lower than its non-bounty questions (i.e., 0.53). One possible reason is that the bounty questions under this tag are very specific and require specific knowledge, which not many people possess.

Summary: The solving-likelihood of bounty questions is higher than that of non-bounty questions, especially in very large communities with relatively low question solving-likelihood.

In the next sections, we build logistic regression models to further study the important factors that are associated with the solving-likelihood and solving-time of a bounty question.

4.5 Study Result

In this section, we present the result of our empirical study of reputation bounties on Stack Overflow. We first study the important factors that are associated with the solving-likelihood and solving-time of a bounty questions. Then, we study the association between bounties and the traffic of questions. *[Jiayuan syas:maybe add a conclusion, like, our findings can also do xxxx.] [Jiayuan syas:Summary the rq in a higher level?]*

4.5.1 RQ1: What Are the Important Factors that Are Associated with the Solving-likelihood of a Bounty Question?

In our preliminary study, we observed an association between the solving-likelihood of a bounty question and two bounty-related factors (i.e., the bounty value and the days-before-bounty metric). We also noticed that the solving-likelihood of bounties differs across tags. There may be other factors that impact the solving-likelihood of a bounty question. For example, longer bounty questions with code snippets may have a higher solving-likelihood. In this section, we use a model to study other factors that may have a relation with the solving-likelihood of bounty questions. With a better understanding of this relationship, we can provide insights into how to better leverage bounties to improve the solving-likelihood of questions.

Approach:

We built a logistic regression model to study the relationship between the studied factors and the solving-likelihood of bounty questions. The logistic regression is a robust and highly interpretable technique, which has been applied successfully in several software engineering-related problems (Wang et al., 2018a; McIntosh et al., 2016). The primary goal of our model is not to determine whether a bounty question will be solved, but to better understand the relationship between each factor and the likelihood of a bounty question being solved. We are the first to study the relation between the studied factors and the solving-likelihood of a bounty question, hence we expect future work to expand on our studied factors. In the following subsections, we elaborate on the studied factors, the details of the model construction, and the analysis of our model.

Studied factors: We study factors along the following dimensions:

1. **Question:** Nine factors which reflect the quality of a question and the activities that are related to the question.

2. **User:** Three factors which reflect the reputation of the bounty backer and the question asker.

3. **Bounty:** Six factors which describe the usage (e.g., the value) of bounties of the question and its associated tags.

4. **Tag:** 16 factors which reflect the community of a bounty question in terms of the age, the answerer population and the question-solving skills of the answerer (i.e., the non-bounty question solving-likelihood).

Table 4.3: The description of and rationale for the factors that we used in our logistic regression model for the solving-likelihood of bounty questions. The factors which are marked with '*' are calculated at the time when the bounty is proposed and the factors which are marked with '**' are calculated considering only the data one month before the bounty is proposed.

Factor name	Description	Rationale
The Question Dimension		
Q_url_num	The number of URL links in the content of a question.	These factors reflect the amount of supportive information that a question has. Questions with more supportive information may help potential answerers in solving.
Q_codesnippet_num	The total number of code snippets in the content of a question.	
Q_body_len	The length of the content of a question (in characters).	
Q_code_len	The total length of the code snippets the content of a question (in characters).	
Q_code_proportion	The proportion of code the content of a question (i.e., $\frac{Q\_code\_len}{Q\_body\_len}$).	
Q_answer_score *	The total score of all answers of a question.	These factors reflect the popularity of a question and its answers. Popular questions may attract more attention.
Q_answer_num *	The number of answers that a question received.	
Q_score *	The score of a question.	
Q_favorite_num *	The favourite count of a question.	
The User Dimension		
U_backer_reputation *	The reputation of the backer who proposed the bounty.	A backer with a high reputation may indicate that the backer is a knowledgeable user and the question may be of high quality.
U_asker_answer_num **	The number of prior answers of the question asker.	A question, which is asked by an asker whose prior activity is high, may be of high quality.
U_asker_question_num **	The number of prior questions of the question asker.	
The Bounty Dimension		
B_days_before_bounty	The number of days between the creation of a question and the proposing of a bounty for it.	The timing of proposing a bounty may have a relationship with the solving-likelihood.
B_value	The value of a question's bounty.	A higher bounty may attract more potential answerers.
B_solving_likelihood_min-/max/mean/median *	The min/max/mean/median solving-likelihood of bounty questions for a question's tags.	
The Tag Dimension		
T_age_min/max/mean/sum	The min/max/mean/sum age in days of a question's tags.	Older tags may have a larger community and more potential answerers to solve questions
T_question_num_min/max-/mean/sum **	The min/max/mean/sum number of questions of a question's tags.	These factors reflect the community size of the tags of a question. A larger community size may have more potential answerer to solve questions.
T_answerer_num_min/max-/mean/sum **	The min/max/mean/sum number of answers of a question's tags.	
T_solving_likelihood_normal-_min/max/mean/median *	The min/max/mean/sum age of a question's tags.	These factors reflect the question-solving skill level of answerers of a bounty question's associated communities. Questions that have communities with highly skilled answerers are more likely to be solved.

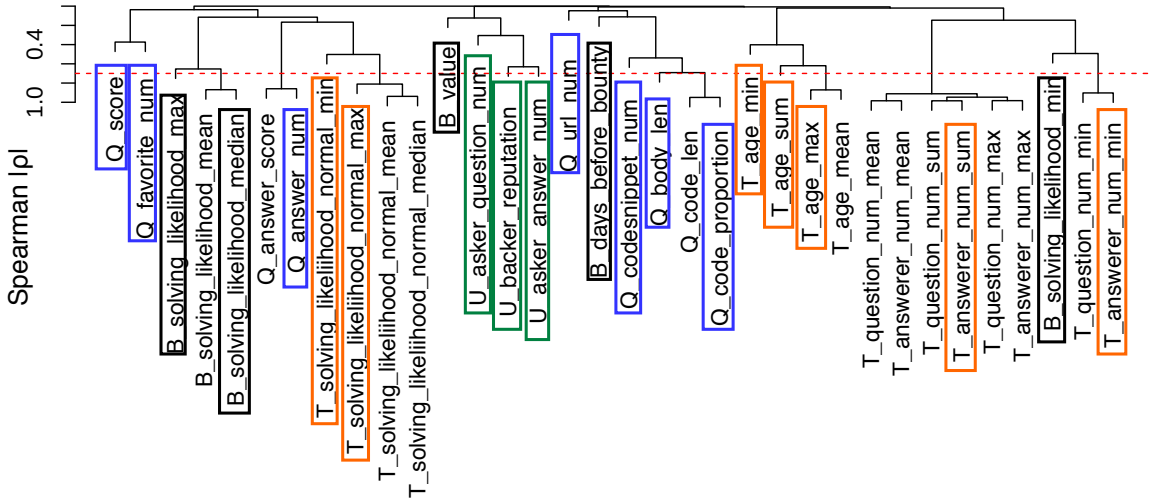


Figure 4.13: The hierarchical clustering plot of factors in our solving-likelihood model. According to the Spearman rank correlation test (using a cut-off value of 0.7), we selected the simplest metrics to compute across each dimension of correlated factors. We ended up with seven factors in the question level dimension (marked in blue), three factors in the user dimension (marked in green), five factors in the bounty dimension (marked in black), and seven in the tags dimension (marked in orange).

We have 34 factors in total. Table 4.3 shows the description of and rationale for the 34 studied factors. These factors are explanatory variables of our model.

Model construction: The presence of correlated and redundant features (i.e., multicollinearity) negatively impact the interpretability of the generated classifiers (Farrar and Glauber, 1967; Tantithamthavorn and Hassan, 2018). Therefore, similar to prior studies (Rajbahadur et al., 2017; Wang et al., 2018a; McIntosh et al., 2016), we first removed correlated and redundant factors to avoid multicollinearity. We used the Spearman rank correlation test to measure the correlation between the studied factors and kept only one of the highly-correlated factors (using a cut-off value of 0.7 (McIntosh et al., 2016; Thongtanunam et al., 2016; Wang et al., 2018a; Tantithamthavorn and Hassan, 2018)). Then we conducted a redundancy analysis to remove redundant factors

to avoid multicollinearity. Finally, we ended up with seven factors in the question dimension, three factors in the user dimension, five factors in the bounty dimension, and seven factors in the tags dimension (Figure 4.13). We built a logistic regression model, which enables us to examine the impact of one or more variables on a response variable while controlling for other variables. Similar to prior work (McIntosh et al., 2016; Wang et al., 2018a), we added non-linear terms in the model to capture the more complex relationship in the data by employing restricted cubic splines (Harrell, 2006). The non-linear factor will be assigned additional degrees of freedom (i.e., *D.F.*). We used the **rms**[14] *R* package to implement our logistic regression model. See our prior work (Wang et al., 2018a) for more details about the non-linear term allocation.

Model analysis: We used the Area Under the ROC Curve (i.e., *AUC*) and a bootstrap-derived approach (Efron, 1986) to evaluate the performance of the logistic regression model following prior studies (McIntosh et al., 2016; Wang et al., 2018a). The AUC ranges from 0 to 1 (with 0.5 being the performance of a random guessing model). A higher AUC indicates that the model has a better ability to capture the relationship between the explanatory variables and the response variable. In the bootstrap-derived approach, we built a model with a bootstrapped sample then we applied the model to the original dataset and the bootstrapped dataset. We used the optimism value, which is the difference of the AUC between the models that are built on the original data and the model that is built on the bootstrapped dataset, to evaluate the amount of overfitting. Small optimism values indicate that the model does not suffer from overfitting. We repeated the bootstrap-derived approach 100 times and used the median optimism value to evaluate the overfitting of our models.

[14] https://cran.r-project.org/web/packages/rms/index.html

To measure the explanatory power of each factor in the constructed model, we used the *anova* function in the *R* package *rms* to compute Wald χ^2 value for each factor. A larger Wald χ^2 value indicates a higher explanatory power of the factor in the constructed model. To test whether a factor contributes a statistically significant amount of explanatory power to the model, we further applied a χ^2-test to the calculated Wald χ^2 values. We also apply a Bonferroni correction (Bonferroni, 1936) to correct the p-values for multiple comparisons. We used the *Predict* function in the *rms R* package to plot the estimated bounty question solving-likelihood against a factor. The analysis allows us to further carefully examine how each factor affects the solving-likelihood. We hold the other factors at their median values when exploring one factor.

Results:

Our model explains our dataset well and has a reliable performance. Table 4.4 shows the result of the performance analysis of our model. Our model obtains a median AUC of 0.708, which indicates that the model explains the relationship between the studied factors and the solving-likelihood well. In addition, the low optimism of the AUC value (i.e., 0.001) suggests that our model does not suffer from overfitting.

A question that received more answers before a bounty was proposed has a higher solving-likelihood, especially when the question has more than 3 answers before the proposal of the bounty. Table 4.4 shows the Wald's χ^2 value of the studied factors. The *Q_answer_num* factor (i.e., the number of answers that a question received before a bounty is proposed) contributes the most explanatory power to the model. Figure 4.14 shows the relationship between the bounty question solving-likelihood and *Q_answer_num*. *Q_answer_num* has a positive relationship with

Table 4.4: The result of our logistic regression model for understanding the relationship between the studied factors and the bounty question solving-likelihood. The factors are ordered by their importance (i.e., overall Wald's χ^2 value) in the model. We also show the non-linear (NL) Wald χ^2 value. We only show factors of significant importance (i.e., the p-value of the χ^2 value is less than 0.002 (i.e., 0.05/22)) to our model. See our supplementary material for the full table (Zhou, 2019).

Factors		**Solving-likelihood Model**	
AUC		0.708	
AUC optimism		0.001	
Factors		**Overall**	**NL**
Q_answer_num	D.F.	1	
	χ^2	1348.604	
B_value	D.F.	9	
	χ^2	597.668	
T_solving_likelihood_normal_min	D.F.	4	3
	χ^2	473.843	7.921
B_days_before_bounty	D.F.	1	
	χ^2	382.611	
T_answerer_num_sum	D.F.	2	3
	χ^2	359.326	108.199
T_solving_likelihood_normal_max	D.F.	3	2
	χ^2	349.808	54.763
B_solved_likelihood_median	D.F.	4	3
	χ^2	164.312	128.110
B_solved_likelihood_min	D.F.	3	2
	χ^2	106.017	104.622
T_age_min	D.F.	1	
	χ^2	64.624	
Q_codesnippet_num	D.F.	1	
	χ^2	50.250	
B_solved_likelihood_max	D.F.	3	2
	χ^2	44.900	41.039
Q_body_len	D.F.	1	
	χ^2	29.932	
T_age_max	D.F.	1	
	χ^2	21.996	
Q_url_num	D.F.	1	
	χ^2	17.373	
U_asker_answer_num	D.F.	1	
	χ^2	12.798	

the solving-likelihood. Once a question has more than three answers, the solving-likelihood of the question is at least 0.9, while the solving-likelihood for questions without an answer is 0.59. One possible explanation is that answerers may benefit from the prior answers of a question. The more prior answers the question has, the more potential solvers are likely to benefit from those answers. For example, the poster of the accepted answer to a question[15] mentioned that "The answer by Yacoby can be extended further." In other words, the accepted answer was based on a prior answer.

The bounty value and the timing of proposing a bounty are important factors that are associated with the solving-likelihood of a bounty question. Table 4.4 shows that *B_value* (i.e., the bounty value) and *B_days_before_bounty* are the second and fourth most important factors in the model. In Figure 4.14, we observed a positive relationship between the bounty value and the bounty question solving-likelihood. One possible explanation is that higher bounties attract more attention to a question, thereby increasing the solving-likelihood.

Figure 4.14 also shows a negative relationship between *B_days_before_bounty* and the solving-likelihood of a bounty question, which indicates that a question for which a bounty is proposed earlier may have a higher likelihood of being solved. We also noticed that after 365 days, the bounty question solving-likelihood drops drastically. We suggest bounty backers to consider proposing bounties earlier. In Section 4.5.4, we further study the interesting relationship between the timing of a bounty and the attention (or *traffic*) that it draws to a question.

[15] https://stackoverflow.com/questions/1809670/how-to-implement-serialization-in-c

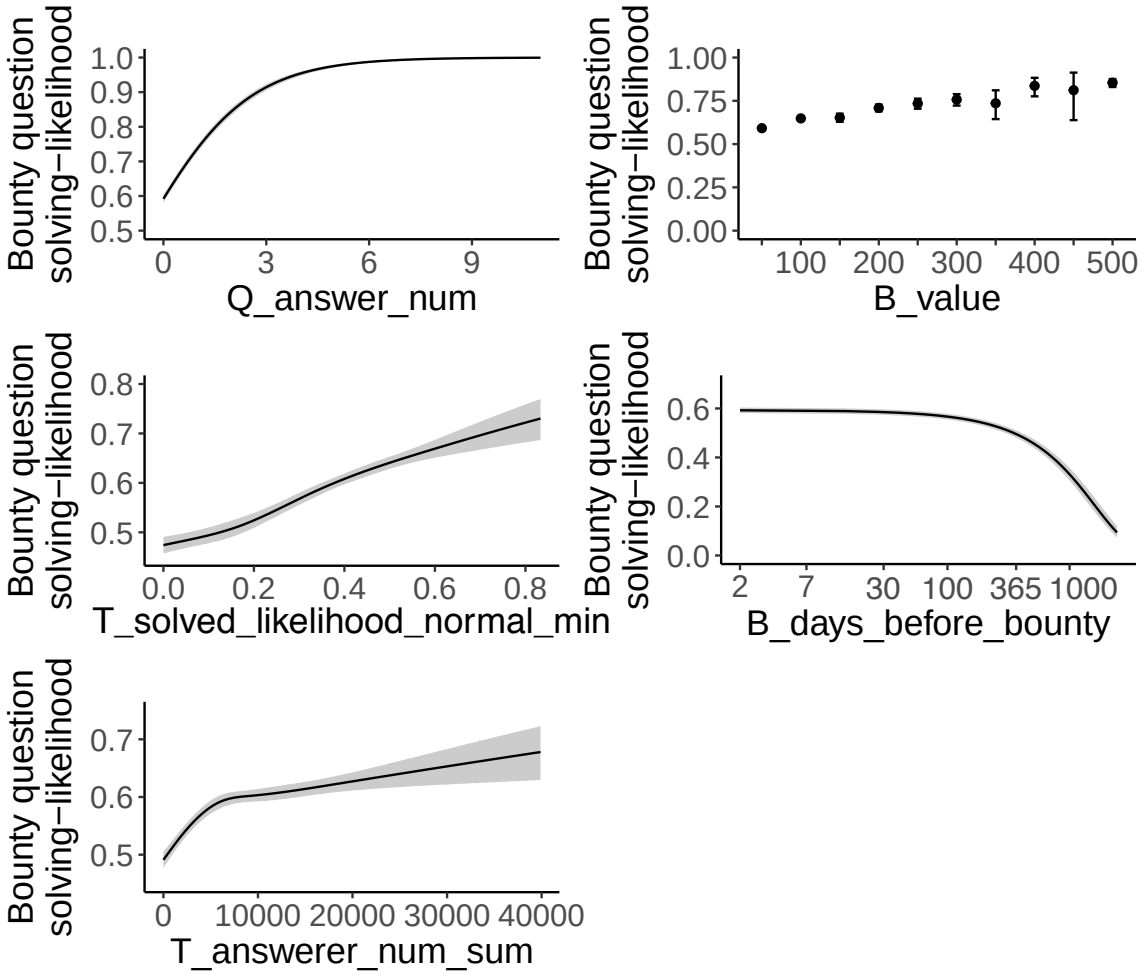


Figure 4.14: The relationship between the five most important factors and the bounty question solving-likelihood in the logistic regression model. For each plot, we set all the factors except the studied factor to their median value in the model while varying the studied factor. The grey area represents the 95% confidence interval. The B_value uses a dot plot instead of a line plot because it is an ordinal variable, as B_value is between 50 and 500 (with an interval of 50), while the other variables are natural numbers.

The associated communities of a bounty question have a significant relationship with its solving-likelihood. The solving likelihood of a tag for non-bounty questions reflects the question-solving skill level of answerers in the community of that tag. Table 4.4 shows that *T_solving_likelihood_normal_min* plays the third most important role in the model which means that the lowest question-solving skill level of answerers in the associated communities of a bounty question has a significant impact on the solving-likelihood of the bounty question. Table 4.4 also shows that *T_answerer_num_sum* (i.e., the total answerer population of the associated communities of a bounty question) plays the fifth most important role in the model. In other words, the number and question solving-likelihood of the answerers in the communities in which a bounty question is asked have an important impact on the solving-likelihood of a question.

In addition, Figure 4.14 shows that *T_solving_likelihood_normal_min* and *T_answerer_num_sum* both have a positive relationship with the solving-likelihood of a bounty question. Hence, bounty questions that are asked in small communities, or communities in which the answerers have a relatively low question solving-likelihood, are less likely to be solved.

Summary: The number of answers before the proposal of a bounty and the value of a bounty are the most important factors that impact the solving-likelihood of a bounty question. In addition, the solving-likelihood of bounty questions is higher in larger communities where the question solving-likelihood of answerers is higher.

4.5.2 RQ2: What Are the Important Factors that Are Associated with the Solving-time of a Bounty Question?

In Section 4.4, we observed that the solving-time of bounty questions varies across tags while the bounty value has no relation with the solving-time. In this section, we study which other factors are related to the solving-time of a bounty question. With a better understanding of this relationship, we can provide insights into how to use a bounty to speed up the process of getting a bounty question solved.

Approach

Similar to Section 4.5.1, we built a logistic regression model to study the relationship between the studied factors and the likelihood of a bounty question being solved fast. Similar to prior studies (Wang et al., 2018a; Tian et al., 2015), we sorted the solved bounty questions by their solving-time (in days) in ascending order and labeled the top 20% questions as fast-solved bounty questions, and the bottom 20% as the slow-solved bounty questions. Table 4.5 shows the 5-number summaries for the solving-times for fast-solved and slow-solved bounty questions. In the following subsections, we explain the additional studied factors compared to solving-likelihood model and the model construction. We analyzed our model for the solving-time in the same way as discussed in Section 4.5.1.

Additional studied factors: We studied 8 factors in the user dimension in addition to the 34 factors that we included in our solving-likelihood model in Section 4.5.1. These eight factors (i.e., Table 4.6) reflect the activity and the question solving-likelihood of answerers whose answers were awarded with bounties. These eight new factors are not included in the model in Section 4.5.1 since they are related to the answer and answerer

Table 4.5: The 5-number summaries for the solving-times of the fast-solved and slow-solved bounty questions.

Question Type	Quantile solving-time (days)				
	Min	1^{st}	Median	3^{rd}	Max
Fast-solved	0.00	0.02	0.04	0.08	0.14
Slow-solved	4.60	5.35	6.07	6.67	8.06

Table 4.6: The description of and rationale for the additional factors that we studied in our logistic regression model for the likelihood of a bounty question being solved fast. The factors marked with '**' are the time-dependent factors which are calculated considering only the activity within a month before the bounty was offered.

Factor name	**Description**	**Rationale**
U_answerer_answer_num **	The number of answers that the answerer posted previously.	A previously active answerer may answer questions faster.
U_answerer_question_num **	The number of questions that the answerer posted previously.	
U_answerer_question_score-_max/median/sum	The max/median/sum scores of the answerer's prior questions.	These factors indicate the question solving and asking-skills of an answerer and may influence the solving-time of a question.
U_answerer_answer_score-_max/median/sum	The max/median/sum scores of the answerer's prior answers.	

of a question, which would not be available for the unsolved bounty questions that we studied in Section 4.5.1. Hence, our model for the solving-time contains 42 factors in total. Also note that we cannot include unsolved questions in our model since such questions would not have a solving-time.

Model construction: We applied the same correlation and redundancy analysis for these 42 factors as discussed in Section 4.5.1 to remove correlated and redundant factors. Finally, we ended up with seven factors in the question dimension, seven factors in the user dimension, five factors in the bounty dimension and seven factors in the tags dimension (Figure 4.15). We also used the same approach as in Section 4.5.1 to add degrees of freedom to non-linear factors.

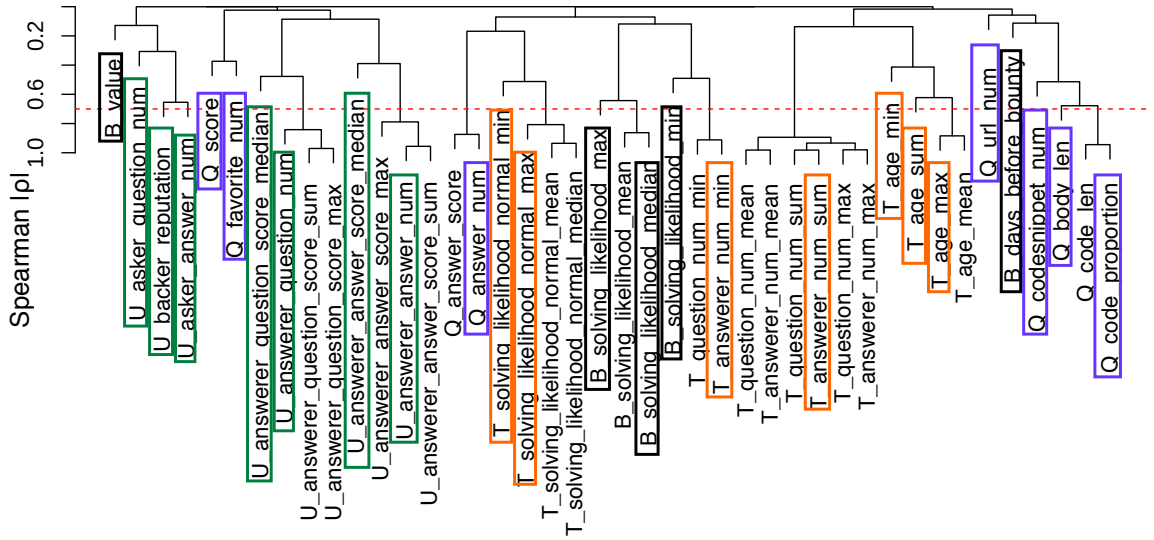


Figure 4.15: The hierarchical clustering plot of factors in our solving-time model. According to the Spearman rank correlation test (using a cut-off value of 0.7), we selected the simplest metrics to compute across each dimension of correlated factors. We ended up with seven factors in the question level dimension (marked in blue), seven factors in the user dimension (marked in green), five factors in the bounty dimension (marked in black), and seven in the tags dimension (marked in orange).

Table 4.7: The result of our logistic regression model for understanding the relationship between the studied factors and the likelihood of a bounty question being solved fast. The factors are ordered by their importance (i.e., overall Wald's χ^2 value) in the model. We also show the non-linear (NL) Wald χ^2 value. We only show factors which are of significant importance (i.e., the p-value of the χ^2 is less than 0.002 (i.e., 0.05/26)) in our model. See our supplementary material (Zhou, 2019) for the full table.

Factors		**Solving-time Model**	
AUC		0.817	
AUC optimism		0.002	
Factors		**Overall**	**NL**
Q_answer_num	D.F.	1	
	χ^2	2032.150	
U_answerer_answer_num	D.F.	3	2
	χ^2	581.880	361.452
T_solving_likelihood_normal_min	D.F.	3	2
	χ^2	391.171	76.639
T_age_max	D.F.	1	
	χ^2	317.732	
T_solving_likelihood_normal_max	D.F.	1	
	χ^2	243.640	
B_days_before_bounty	D.F.	1	
	χ^2	173.308	
Q_code_proportion	D.F.	1	
	χ^2	144.062	
Q_favorite_num	D.F.	1	
	χ^2	74.265	
Q_body_len	D.F.	2	
	χ^2	58.913	
T_age_sum	D.F.	1	
	χ^2	45.573	
T_answerer_num_sum	D.F.	1	
	χ^2	42.458	
Q_codesnippet_num	D.F.	1	
	χ^2	15.294	
B_solving_likelihood_max	D.F.	1	
	χ^2	14.696	

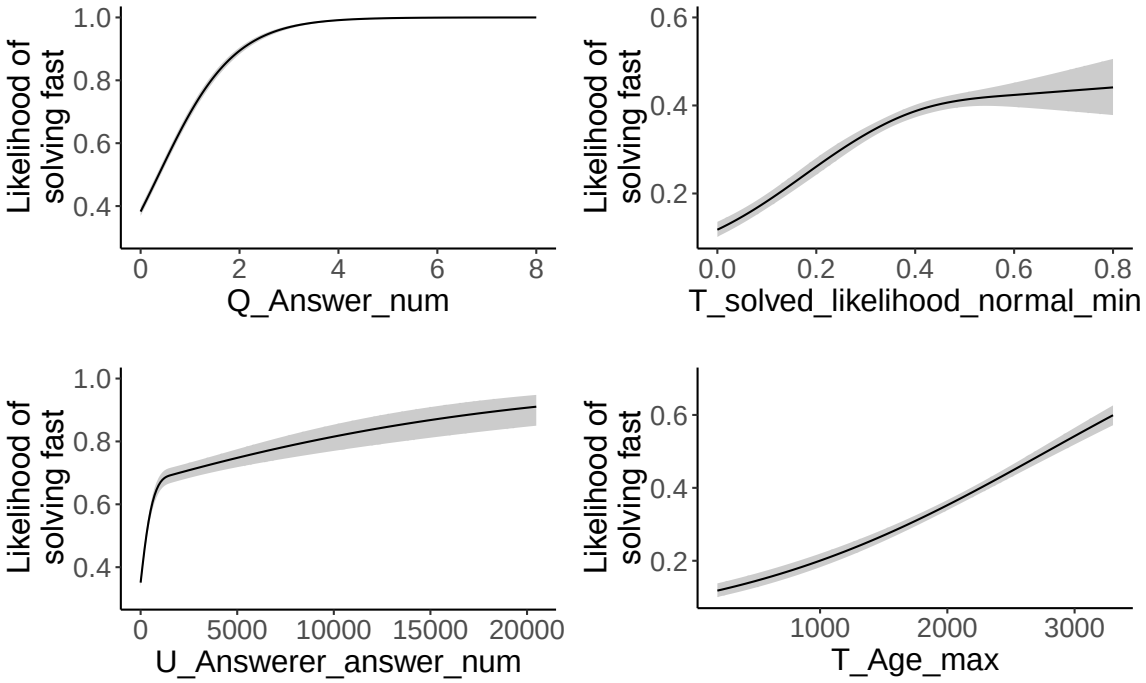


Figure 4.16: The relationship between the studied factors and the likelihood of a bounty question getting solved fast in the logistic regression model. For each plot, we set all the factors except the studied factor to their median value in the model while varying the studied factor. The grey area represents the 95% confidence interval.

4.5.3 Results

Our model explains our dataset well and has a reliable performance. Table 4.7 shows the results of the performance analysis of our model. Our model has a high median AUC of 0.817 which indicates that the model explains the relationship between the studied factors and the likelihood of being solved fast well. In addition, the low optimism of the AUC values (i.e., 0.002) suggests that our model does not overfit the dataset.

The number of answers that a question received before a bounty was proposed (i.e., *Q_answer_num*) is the most important factor to get a bounty question

solved fast. Table 4.7 shows the Wald's χ^2 value of the studied factors. Similar to the solving-likelihood model in Section 4.5.1, *Q_answer_num* contributes the most explanatory power to the solving-time model. Figure **??** shows the relationship between *Q_answer_num* and the likelihood of solving a bounty question fast. The more answers that a question received before a bounty was proposed, the faster it was solved. In addition, the likelihood of a bounty question getting solved fast increases sharply as the number of previously posted answers goes from zero to three answers. After three answers, the likelihood of getting solved fast remains equally high. A possible explanation is similar to the one in Section 4.5.1. More answers may contain useful information that help other answerers to provide an acceptable answer to the question.

The activity level of the answerer has a positive relationship with the likelihood of solving a bounty question fast. *U_answerer_answer_num* is the second most important factor in the model and has a positive relationship with the likelihood of getting a bounty question solved fast (see Figure **??**). This finding is similar to what we observed in our prior work (Wang et al., 2018a), which is that the activity level of answerers is the most important factor that impacts the speed of a question getting solved on Stack Overflow.

Higher bounty values are not associated with faster solving of questions. The value of a bounty (i.e., *B_value*) is of low importance in our model. This might be due to various reasons. For instance, it might take longer to solve high-valued bounty questions due to them being harder or less popular.

The question solving-likelihood of associated communities have a significant impact on the likelihood of solving a bounty question fast. The lowest question-solving skill level of the associated communities (i.e., *T_solving_likelihood_normal_min*) plays the third important role in the model. The *T_solving_likelihood_normal_min* has a positive relationship with the likelihood of a bounty question getting solved fast.

Summary: The number of (unaccepted) answers to a question before a bounty is proposed has the strongest association with the likelihood of a bounty question solving fast. A higher-valued bounty does not help a bounty question to get solved faster. The activity level of potential answerers and the question solving-likelihood of the potential answerer communities have a strong association with the solving-time of a bounty question.

4.5.4 RQ3: Studying the Association between Bounties and the Traffic of Questions

In the previous sections, we found that the popularity of a question (e.g., in terms of the number of existing answers) and the size and question solving-likelihood of the community in which the question is asked (e.g., in terms of the solving-likelihood of questions with a certain tag) are strongly associated with the solving-likelihood and solving-time of a question. These findings suggest that it is beneficial to attract more traffic to a question. In this section, we conduct an empirical study of the association between a bounty and the traffic to the bounty question.

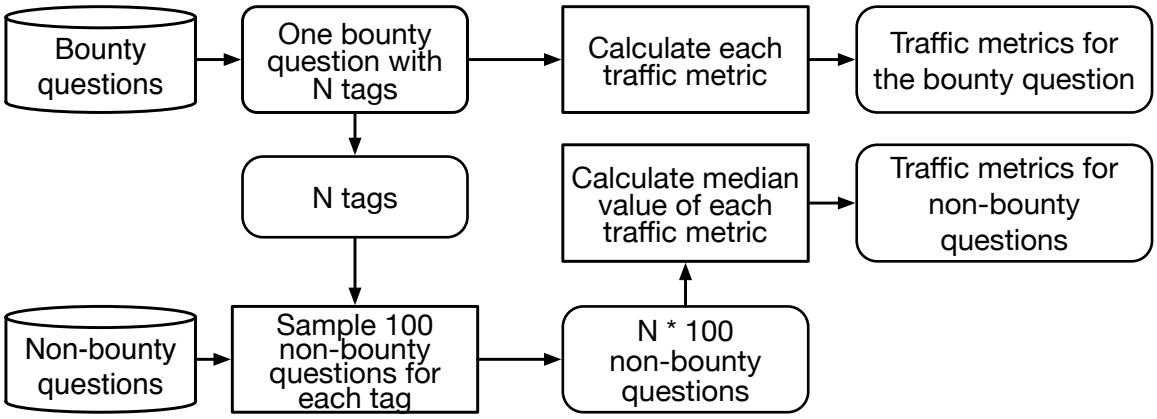


Figure 4.17: An overview of our approach for computing the traffic of bounty and non-bounty questions.

Approach

A bounty in Stack Overflow will be active for a maximum of seven days. Therefore, we only measure the traffic for seven days after the bounty was proposed. We use the following metrics to capture the traffic of a question:

1. **The number of new answers** to a question.

2. **The number of new comments** on a question.

3. **The number of new edits** of answers to a question.

To understand how bounties impact the traffic to a question, we compared the traffic between bounty and non-bounty questions. If the traffic of bounty questions is significantly higher than that of non-bounty questions, it suggests that bounties may help to attract more traffic to a question.

The traffic of a question could be impacted by several conditions under which the question was asked, such as the content of the question, the related tags, and the creation time of the question. Ideally, we can compare bounty and non-bounty questions that share the same conditions. However, due to the richness of the metadata of the questions on Stack Overflow, it is very hard to find a perfect match for the bounty questions automatically. Therefore, we use a sample-based method to identify a set of questions that share similar conditions as a bounty question, and we use the median value of their traffic metrics to represent the traffic of non-bounty questions.

Figure 4.17 gives an overview of our approach for calculating the traffic metrics of bounty and non-bounty questions. The details of each step are explained below.

1. For each bounty question, we calculate its seven-day traffic metrics since the bounty was proposed. In other words, if the bounty was proposed on the m^{th} day after the creation of the question, we calculate the traffic from day m to m+7.

2. We extract all N tags of the bounty question. For each tag, we randomly sample 100 questions from the non-bounty questions that are associated with the tag (without repetition). After this step, we have $N * 100$ sampled non-bounty questions.

3. For each sampled non-bounty question, we calculate the seven-day traffic between m and $m+7$ in the same way as we do for the bounty question.

4. We use the median value of each traffic metric of the sampled non-bounty questions to represent the traffic of the corresponding non-bounty questions.

After calculating the traffic metrics for all bounty questions and their similar non-bounty questions, we categorized them into groups based on the *days-before-bounty*

metric to study the impact of this metric on the traffic as prior studies (Hanrahan et al., 2012; Wang et al., 2018a). We define the *time-based groups* as follows:

1. **[3, 3]:** the bounty is proposed on the third day after the question was created (i.e., the earliest allowed by Stack Overflow – see Section **??**).

2. **[4, 7]:** the bounty is proposed at least four and at most seven days after the question was created.

3. **[8, 30]:** the bounty is proposed at least 8 and at most 30 days after the question was created.

4. **[31, 365]:** the bounty is proposed at least 31 and at most 365 days after the question was created.

5. **[366,∞):** the bounty is proposed at least 365 days after the question was created.

We then compared the traffic between bounty and non-bounty questions as described above across these groups.

To study the impact of the bounty value on traffic, we compared the traffic metrics of bounty questions across different bounty values. We categorized bounty questions into three groups based on their bounty value as identified in Section 4.4. The *bounty-value-based groups* are as follows:

1. **Small (bounty):** the question has a bounty value that ranges from 50 to 150.

2. **Moderate (bounty):** the question has a bounty value that ranges from 200 to 350.

3. **Large (bounty):** the question has a bounty value that ranges from 400 to 500.

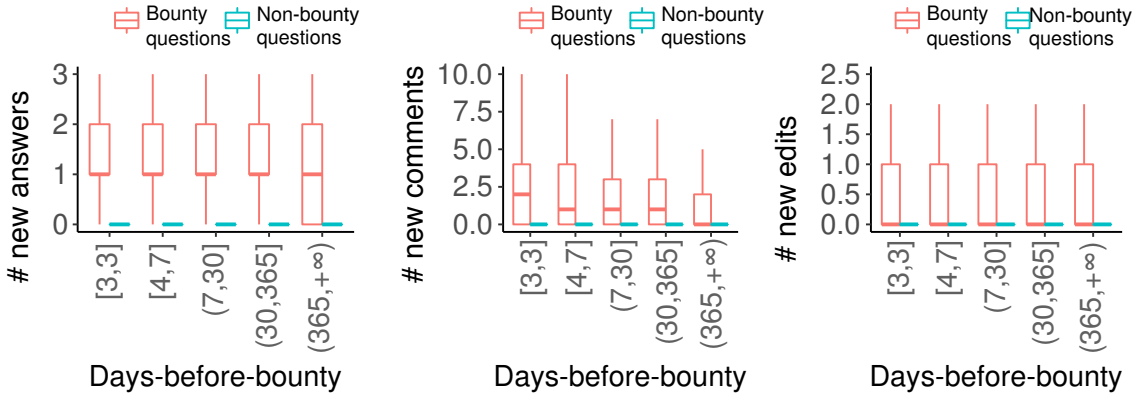


Figure 4.18: The distributions of the traffic metrics (i.e., the number of new answers, new comments and new edits) for bounty and non-bounty questions across different values of the days-before-bounty metric.

To compare the differences of the traffic metrics between bounty and non-bounty questions, we used the Wilcoxon rank-sum test and Cliff's delta effect size as explained in Section 4.4.2.

Results

Questions are likely to attract more traffic than non-bounty questions after they receive a bounty. Figure 4.18 shows the seven-day traffic for bounty and non-bounty questions. For the same time-based group, the traffic of bounty questions is always higher than that of non-bounty questions. For each time-based group, the statistical tests show that the differences in the traffic between bounty and non-bounty questions are significant with large effect sizes. The results indicate that bounty helps attract new traffic. For example, the question[16] was created on Nov 27, 2009, and received five answers before the bounty was proposed on Jan 28, 2015. After proposing the bounty, 12 new answers were created. An interesting additional observation is that non-bounty

[16] https://stackoverflow.com/questions/1809986/

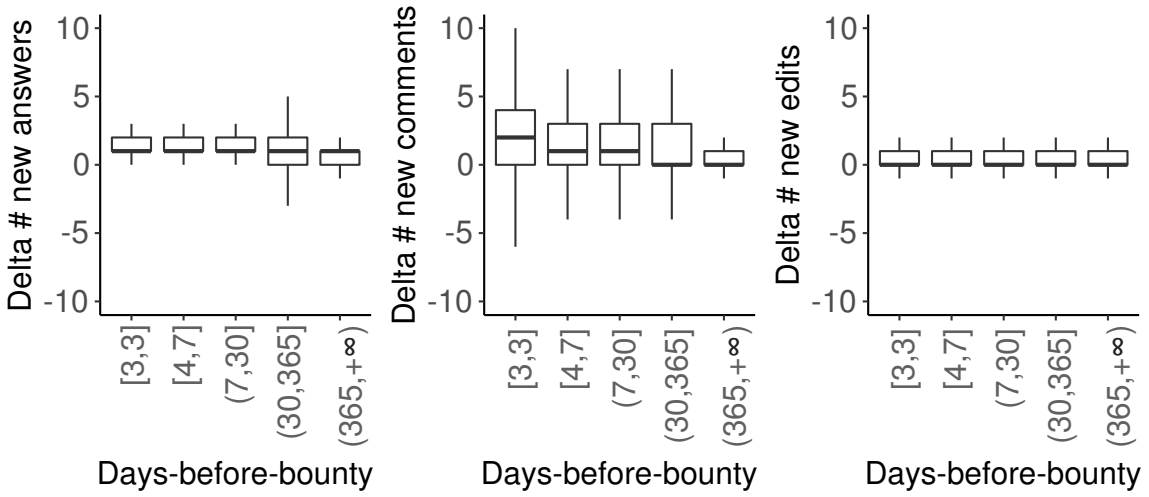


Figure 4.19: The distributions of the (absolute) difference in traffic to a question before and after proposing a bounty. The difference (delta) metrics are calculated by subtracting the value of a traffic metric before the bounty was proposed from the seven-day traffic metric value (i.e., after - before).

questions receive hardly any traffic after 2 days. This observation is a confirmation of the finding in prior work that a question that is not solved fast, is unlikely to be solved at all (Anderson et al., 2012).

To ensure that the above finding is not biased by the popularity of bounty questions (e.g., bounty questions may attract more traffic in general compared to non-bounty questions), we also calculated the (absolute) difference in traffic to a question before and after proposing the bounty. To calculate this difference, we subtracted the value of the traffic metrics before proposing the bounty from the seven-day traffic values after proposing the bounty. Figure 4.19 shows the distributions of these differences. Figure 4.19 shows that the median difference is always at least zero, and in most groups larger than zero for the number of new answers and new comments. These differences indicate that the traffic to a bounty question increased in most cases after the proposal of a bounty.

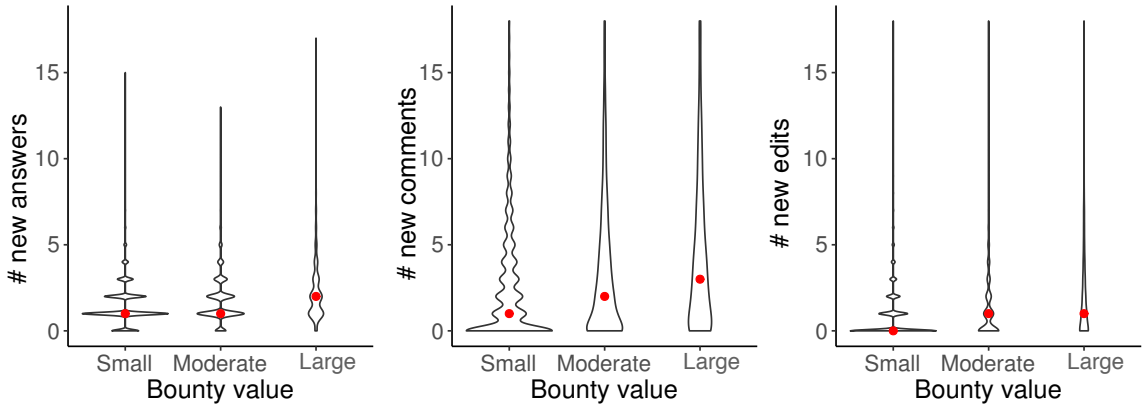


Figure 4.20: The distributions of the traffic metrics (i.e., the number of new answers, new comments and new edits) for bounty and non-bounty questions across different bounty value groups. The red dot is the median value of the corresponding distribution.

Questions with bounties that are proposed early are more likely to have more comments than questions with bounties that are proposed later. Figure 4.18 shows the distribution of traffic metrics across the time-based groups. We can observe that proposing a bounty earlier is not correlated with a higher number of new answers and edits, but it is correlated with a higher number of new comments. We used the Wilcoxon rank-sum test and Cliffs delta d to measure the differences in the traffic metrics between each pair of adjacent time-based groups. We also performed a Bonferroni correction (Bonferroni, 1936) to correct the p-values for multiple comparisons. All pairwise comparisons show that the differences are significant (i.e., the p-value $<$ 0.05/4) with a non-negligible effect size in terms of the number of comments, suggesting that the number of comments is positively correlated with the timing of proposing a bounty.

A higher-valued bounty is more likely to attract more traffic to a question, especially when the bounty value is over 400. Figure 4.20 shows the distributions of

the traffic metrics of bounty questions across the bounty-value-based groups. We observed that a question with a higher-valued bounty is more likely to attract more traffic. More specifically, a question with a large bounty (i.e., with a bounty value of at least 400) attracts more answers than ones with a small bounty (i.e., with a bounty value of 150 or less). Similar trends hold for the number of new comments and edits. Our statistical test results show that the differences between each pair of adjacent bounty-value-based groups are significant. Moreover, the effect size of the differences between the small bounty and large bounty groups are at least small for all the traffic metrics, indicating that a large bounty is more likely to attract additional traffic to a question.

Summary: Questions are likely to attract more traffic after receiving a bounty than non-bounty questions, particularly for questions that receive a bounty with a value of at least 400.

4.6 Discussion

In this section, we discuss bounties for rewarding existing answers. Then, we look into the differences between unsolved and solved bounty questions, the important factors for solving-likelihood of non-bounty and bounty questions. We also discuss the implications of our findings.

4.6.1 Bounties for Rewarding Existing Answers

3% of the bounties were proposed to reward an existing answer. In addition to the main purpose of getting a question solved, we observed (from the user-posted reason for the bounty) that 3,894 out of 129,202 (3%) bounties were proposed to reward an existing answer. We refer to this type of bounty as a *bonus bounty*. The median answer

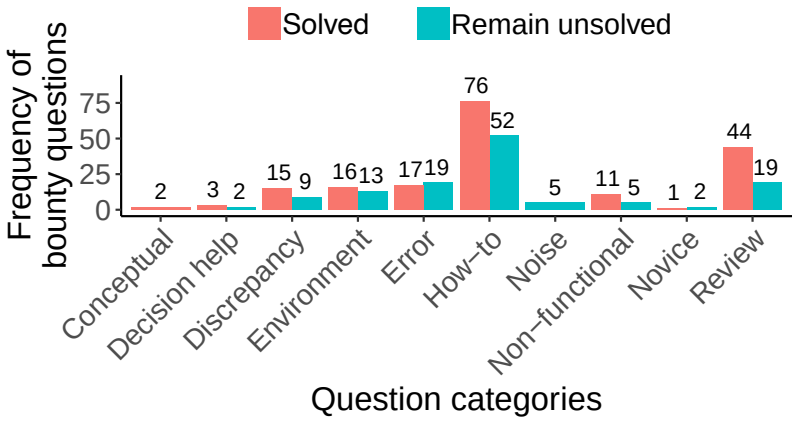


Figure 4.21: The frequency of categories of our samples bounty questions.

score (i.e., the number of upvotes from users) of the answers that were awarded a bonus bounty is 8, while the median score for the other answers, and for accepted answers on Stack Overflow is only 1. In other words, the rewarded existing answers appear to be of a higher quality than the average answer on Stack Overflow.

Bounty backers who proposed bonus bounties are usually users with a high reputation. The median number of reputation points of the bounty backers who proposed a bonus bounty is 4,570, which is six times higher than the reputation points of other bounty backers (i.e., 706). Such backers are usually much "richer" (i.e., have a larger amount of reputation points) than other backers. Moreover, bonus bounties tend to be larger than non-bonus bounties. While the median value is 50 for both types of bounties, the mean value of a bonus bounty is 113 while the mean value of a non-bonus bounty is 82, which indicates that bonus bounties tend to have a higher value. Finally, 55% of the backers of bonus bounties are not the asker of the bounty question (vs. only 15.7% for non-bonus bounties).

Table 4.8: The question categories and examples as defined by (Treude et al., 2011). Note: this table is reprinted from (Treude et al., 2011).

Name	**Definition**	**Example**
How-to	Questions that ask for instructions.	*How to crop image by 160 degrees from center in asp.net?*
Discrepancy	Some unexpected behavior that the person asking the question wants explaining.	*getElementById() returns null even though the element exists?*
Environment	Questions about the environment either during development or after deployment.	*Setting Environment Variables in Rails 3 (Devise + Omniauth)?*
Error	Questions that include a specific error message.	*Getting an ambiguous redirect error.*
Decision help	Asking for an opinion.	*Should I use JSLint or JSHint JavaScript validation?*
Conceptual	Questions that are abstract and do not have a concrete use case.	*Content-Disposition: What are the differences between "inline" and "attachment"?*
Review	Questions that are either implicitly or explicitly asking for a code review.	*Is my file struts.xml is it correct?*
Non-functional	Questions about non-functional requirements such as performance or memory usage.	*Where to store global constants in an iOS application?*
Novice	Often explicitly states that the person asking the question is a novice.	*How to use WPF Background Worker?*
Noise	Questions not related to programming.	*Apple Developer Program.*
Other	Questions that are other than the above categories.	*Where do I find old versions of Android NDK?*

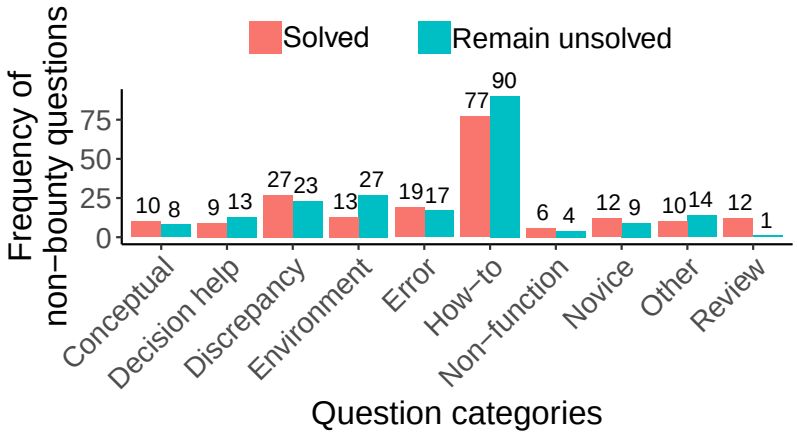


Figure 4.22: The non-bounty questions from a prior study (Treude et al., 2011).

4.6.2 Remain-Unsolved vs. Solved Bounty Questions

There are 44,635 bounty questions of which the bounties expired with no awarded answers. We observed that 64.4% (28,754) of those questions were never solved after that (i.e., *remain-unsolved bounty questions*). To compare the differences between remain-unsolved and solved bounty questions, we sampled 100 (out of 28,754) bounty questions which had no accepted answer at the time of collecting our data, and 100 (out of 79,093) bounty questions which were solved as the two statistically representative samples with a 95% confidence level and a 10% confidence interval. The first two authors manually and independently labeled the categories of these 200 bounty questions into the categories that were defined by (Treude et al., 2011) (see Table 4.8). They discussed conflicts until a consensus was reached. The Cohen's kappa (Gwet et al., 2002) value to measure the inter-rater agreement of this labeling is 0.66 before resolving the conflicts. Note that in our study, some questions have multiple categories. When we calculated the Cohens Kappa, we did not consider partial agreements, instead we consider the labeled categories of a question from two raters in agreement only when their categories

were exactly the same. For example, if a question was assigned categories c1 and c2 by rater 1 and categories c2 and c3 by rater 2, we considered them in disagreement. If we considered partial agreements, the Cohens Kappa would be 0.95.

Figures 4.21 and **??** show the frequency of question categories for our sampled bounty and non-bounty questions which were studied by Treude et al. (2011). We observed that the category "How-to" is the most popular category for both bounty and non-bounty questions. However, the solving-rate of the "How-to" category for bounty questions is 59%, which is higher than that of non-bounty questions (46%). We also observed that bounty questions in the "Review" category are more likely to be solved with a solving-rate of 70% (i.e., 44 out of 63) for bounty questions and 92% (i.e., 12 out of 13) for non-bounty questions. One possible explanation is that review questions may be easier to solve as there is more information about the problematic source code in the question. For example, one question about Biztalk[17] provides a clear description, development environment and code snippet, which makes it easier for answerers to solve the question. Moreover, we observed that "Review" questions are more likely to appear in bounty questions (32%) than non-bounty questions(3%).

4.6.3 The Important Factors for the Solving-likelihood of Non-bounty and Bounty questions

To further understand the important factors for solving-likelihood of non-bounty and bounty questions, we built two additional models to explain the important factors of

[17] http://bit.ly/2HsnxbY

the solving-likelihood of non-bounty (*non-bounty-question-model*) and bounty questions (*bounty-question-model*$_{without\_bounty\_factors}$). To be able to compare the factors, we used only non-bounty-related factors in these two models. Table 4.9a and Table 4.9b show the performance and the top five most important factors for the

We found that *T_solving_likelihood_normal_min* and *T_solving_likelihood_normal_max* are the most important factors for both models, which indicates that the question solving-likelihood of a tag is important for both bounty and non-bounty questions. Aside from the factors in the tag dimension, the factors in the question dimension are important for the non-bounty questions (e.g., the length of the question body, the number and the proportion of code snippets in a question). In contrast, for the bounty questions all top five most important factors are tag related.

Table 4.9: The result of our logistic regression models for understanding the relationship between the non-bounty factors and the solving-likelihood of two types of questions (i.e., bounty and non-bounty questions). The factors are ordered by their importance (i.e., overall Wald's χ^2 value) in the model. We only show the top five factors which contribute the most significant importance (i.e., the p-value is less than 0.002) to our models.

(a) *Non-bounty-question-model*

Factors		**Median value**	
AUC		0.668	
AUC optimism		0.001	
Factors		**Overall**	**NL**
T_solving_likelihood-_normal_min	D.F.	4	3
	χ^2	511.513	38.462
T_solving_likelihood-_normal_max	D.F.	3	2
	χ^2	325.337	34.475
Q_body_len	D.F.	1	
	χ^2	148.600	
Q_codesnippet_num	D.F.	1	
	χ^2	126.252	
Q_codesnippet_prop-ortion	D.F.	1	
	χ^2	120.665	

(b) *Bounty-question-model*$_{without\_bounty\_factors}$

Factors		**Median value**	
AUC		0.670	
AUC optimism		0.001	
Factors		**Overall**	**NL**
T_solving_likelihood-_normal_min	D.F.	4	3
	χ^2	979.604	89.328
T_solving_likelihood-_normal_max	D.F.	3	3
	χ^2	864.212	89.213
T_answerer_num_sum	D.F.	4	3
	χ^2	715.245	89.213
T_age_min	D.F.	1	
	χ^2	199.427	
T_age_max	D.F.	1	
	χ^2	136.787	

4.6.4 The Implications of Our Findings

While bounties are not a silver bullet for getting a question solved, bounty questions tend to have a higher solving-likelihood than non-bounty questions, particularly when focusing on long-standing unsolved questions. As we showed in Section 4.5.4, in general bounties attract more traffic to questions. In addition, the solving-likelihood of bounty questions is higher than that of non-bounty questions, particularly for long-standing unsolved questions (see Section 4.4). For example, the solving-likelihood of questions that were unsolved for 100 days increases from 1.7% to 55% after proposing a bounty.

The sweet spot for proposing a bounty is as soon as Stack Overflow allows it. Stack Overflow does not allow the proposal of a bounty within two days after the posting of a question. We observed in Section 4.5.4 that after these two days, the traffic to the vast majority of questions is negligible. Hence, we recommend that in order to maximize the solving-likelihood of a question, the bounty is best proposed as soon as possible after those two days. Section 4.5.1 confirms that the solving-likelihood is the highest for bounties that are proposed after two days.

Stack Overflow should indicate which communities (tags) are more active and have a higher solving-likelihood of bounty questions. We showed in Sections 4.5.1 and 4.5.2 that the number of prior answers (i.e., *Q_answer_num*) is the most important factor for both the solving-likelihood and the solving-time of a bounty question. In addition, in these sections, we observed that the size and question solving-likelihood of a community are important factors when it comes to the solving-likelihood of a bounty question. Stack Overflow should provide guidance to bounty backers about which communities are most likely to benefit from proposing a bounty.

Bounty backers should be aware that a highly-valued bounty increases the solving-likelihood of a question, but does not guarantee a fast answer. Sections 4.5.1 and 4.5.4 show that a higher bounty value attracts more traffic to and increases the likelihood of a question. However, Section 4.5.2 shows that the bounty value contributes little to speed up the solving of a question. We recommend that Stack Overflow provides its users with an estimate of the solving-likelihood and solving-time when proposing a bounty. These estimates can be retrieved from historical data about the success of bounties in a particular community, similar to the analysis that we conducted in this paper.

4.7 Threats to Validity

In this section, we discuss the threats to validity. Threats to **external validity** are related to the generalizability of our findings. We studied only bounty questions on Stack Overflow. Further research should investigate whether our findings are generalizable to other Q&A websites, including non-technical ones (such as the other Stack Exchange websites). In addition, although our models have high explanatory power, there might be additional factors that relate to the solving-likelihood and solving-time of bounty questions. Future studies should explore additional factors.

Threats to **internal validity** relate to the experimenter bias and errors. One threat is that we rely on manual analysis to categorize the questions in Section 4.6, which may introduce a bias due to human factors. To mitigate the threat of bias during the manual analysis, two of the authors conducted the manual analysis. We also measure the inter-rater agreement using Cohen's kappa and the raters discussed their differences until they reached consensus. While this manual analysis is only a small part of our

study, future studies should investigate how questions can be classified automatically to reduce the human classification bias and error.

One threat to the internal validity of our study is our categorization of the fast-solved bounty questions (i.e., the fastest 20%) and slow-solved bounty questions (i.e., the slowest 20%) in Section 4.5.2. We conducted a sensitivity analysis by building the logistic regression model using different thresholds (i.e., 30% and 40%) for slow and fast-solved questions. The built models still had high median AUC and low median AUC optimism values (i.e., 0.78 and 0.002 for the 30% threshold, and 0.74 and 0.002 for the 40% threshold). Moreover, the top four important factors were consistent with the model that was built using the 20% threshold. Therefore, we can conclude that our observations are not particularly sensitive to the threshold that we selected to distinguish slow and fast-solved question.

We selected five as the threshold to filter tags in Section 4.4, which is a threat to the internal validity of our study. Figure 4.23 shows the distribution of the solving-likelihood of different tags of bounty questions without filtering tags. Many of the extreme values (0 and 1) are not meaningful due to the very low number of questions in those tags. We agree that five is an arbitrary threshold, unfortunately, any other threshold will be arbitrary as well but we feel it is necessary to put one to enable a clearer representation of the results.

The way that we selected the non-bounty questions for traffic analysis in Section 4.5.4 is a threat to internal validity. We considered the tags of questions, while there may be other confounding factors that could impact the traffic of questions. Future studies should investigate other techniques for matching bounty questions to non-bounty questions.

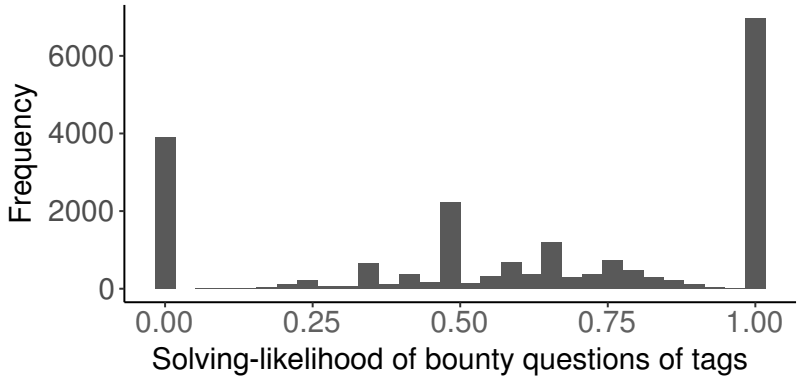


Figure 4.23: The distribution of the solving-likelihood of tags of bounty questions without filtering tags.

One threat to the internal validity of our study is that we measured only the traffic within seven days of posting the bounty. Hence, we did not take any long-term effects of the bounty into account. The main reason is that it is not possible to decide whether these long-term effects were likely caused by the bounty, or by something else. While we cannot claim this causality for the seven-day traffic either, it is more likely that the bounty has a relationship with an increase in traffic while it is active.

A final threat to the internal validity of our study is that while we studied various confounding factors across several dimensions (i.e., the question, user, bounty, tag, answer, and answerer dimensions), there may exist other factors that might potentially have an impact on the solving-likelihood and solving-time of bounty questions. Future studies should investigate the impact of other factors.

4.8 Related Work

In this section, we discuss prior work that is related to our study. We focus on prior work in the research area of improving the question answering process on Stack Overflow.

4.8.1 Improving the Question Answering Process on Stack Overflow

Nowadays, developers rely heavily on Stack Overflow to help solve many software engineering problems. Therefore, it is important to understand the question answering process on Stack Overflow, so that potential improvements can be identified to benefit the users of Stack Overflow. Many prior studies were done in this direction. Wang et al. (2018a) used logistic regression models to study the impact of factors along four dimensions (i.e, answers, questions, askers, answerers) on the speed of a question getting an accepted answer on Stack Overflow and three other famous Q&A Stack Exchange websites. They found that non-frequent answerers are the bottleneck for fast answers and they suggested that Stack Overflow should consider improving their incentive system to motivate non-frequent answerers. Our findings also echo that the answerers of a tag are important for both the solving likelihood and solving time of a bounty question that is associated with that tag. In order to help users to find the right channel to ask questions, several approaches have been developed to help users generate tags automatically when they post a question (Wang et al., 2018c; Xia et al., 2013; Liu et al., 2018; Wang et al., 2014b).

To improve the quality of answers on Stack Overflow, Ponzanelli et al. (2014) proposed an approach to identify low-quality questions. Srba and Bielikova (2016) evaluated how low-quality content on Stack Overflow negatively impacts the community, and proposed ways to solve the problem. Chen et al. (2018) developed a convolutional neural network-based approach to learn editing patterns from historical post edits for predicting the need for editing a post. They also developed an approach that recommends editorial suggestions to improve the quality of a post (Chen et al., 2017). Wang et al. (2018b) analyzed how users revise answers on Stack Overflow under the current

badge system and provided suggestions to improve the revision system. Zhang et al. (2019) investigated how the knowledge in answers becomes obsolete and identified the characteristics of such obsolete answers. Ford et al. (2018) proposed a mentorship program to Stack Overflow in which novice users get assistance with asking a question in an on-site help chat room. They found that the chat room substantially helps to improve the questions that were asked by the novice users.

Different from the prior studies which improve the question answering process by improving the quality of questions and answers, we study the impact of the bounty system on the question answering process in terms of the solving-likelihood and time for bounty questions. We provide users with insights on how to use bounties more effectively.

4.9 Chapter Summary

Stack Overflow introduced their bounty system in 2009 as a way of improving the solving-likelihood of questions. In this system, users can offer reputation points in exchange for an answer to their question.

In this chapter, we studied 129,202 bounty questions (i.e., from Sep. 2011 to Aug. 2017) to study the impact of bounties on the solving-likelihood and solving-time of a question. In addition, we studied the most important factors for the solving-likelihood and solving-time of bounty questions. The main findings of our study are as follows:

1. Questions are likely to attract more traffic after receiving a bounty than non-bounty questions. In addition, bounty questions have a higher solving-likelihood than non-bounty questions, especially in very large communities with a relatively low question solving-likelihood.

2. Bounty questions with a higher bounty value have a higher solving-likelihood, however, a higher bounty value does not expedite the solving of a bounty question.

3. Long-standing unsolved questions with bounties are more likely to be solved than those without bounties. For example, the solving-likelihood of a question that has been unsolved for 100 days increases from 1.7% to 55% after proposing a bounty.

Our study shows that while bounties are not a silver bullet for getting a question solved, they are associated with a higher solving-likelihood of a question in most cases. In particular, when a question is asked in a community (or tag) with a large number of active answerers, the chance of a bounty being successful is relatively high. As questions that are still unsolved after two days hardly receive any traffic, we recommend that Stack Overflow users propose a bounty as soon as possible after those two days for it to be the most successful. In addition, we see an opportunity for Stack Overflow to improve the bounty system by making recommendations to users who are about to propose a bounty about the tag(s) or bounty value that will give the question the highest solving-likelihood.

CHAPTER 5

Studying issue bounties in GitHub open source projects

Due to the voluntary nature of open source software, it can be hard to find a developer to work on a particular task. For example, some issue reports may be too cumbersome and unexciting for someone to volunteer to do them, yet these issue reports may be of high priority to the success of a project. To provide an incentive for implementing such issue reports, one can propose a monetary reward, i.e., a bounty, to the developer who completes that particular task. In this paper, we study bounties in open source projects on GitHub to better understand how bounties can be leveraged to evolve such projects in terms of addressing issue reports. We investigated 5,445 bounties for GitHub projects. These bounties were proposed through the Bountysource platform with a total bounty value of $406,425. We find that 1) in general, the timing of proposing bounties is the most important factor that is associated with the likelihood of an issue being addressed. More specifically, issue reports are more likely to be addressed if they are for projects in which bounties are used more frequently and if they are proposed earlier. 2) The bounty value of an issue report is the most important factor that is associated with the issue-addressing likelihood in the projects in which no bounties were used before. 3) There is a risk of wasting money for backers who invest money on long-standing issue reports.

An earlier version of this chapter is published in the Empirical Software Engineering Journal (EMSE) (Zhou et al., 2020a).

5.1 Introduction

OPEN source software projects often use issue tracking systems (such as BugZilla or GitHub Issues) to store and manage issue reports. For example, developers or users can submit issue reports to report bugs or request new features, and wait for these issues to be addressed. However, some issue reports may never be addressed. For example, developers may avoid addressing issues that they consider too low priority, or difficult to implement. To encourage developers (or *bounty hunters*) to address such issue reports, one or more *backers* can propose a *bounty*.

A bounty is a monetary reward that is often used in the area of software vulnerabilities. Prior studies examined the impact of bounties on vulnerability discovery Zhao et al. (2017); Hata et al. (2017); Finifter et al. (2013). Finifter et al. Finifter et al. (2013) suggested that using bounties as an incentive to motivate developers to find security flaws is more cost-effective than hiring full-time security researchers.

Bounties are now being used to motivate developers to address issue reports, e.g., to fix bugs or to add features. *Bountysource*[1] is a platform for proposing bounties for open source projects across multiple platforms (e.g., GitHub) which currently has more than 46,000 registered developers.[2] Bounty backers can propose several bounties for the same issue report via Bountysource. Although bounties are used in the issue-addressing process, the role that bounties play in this process is not yet understood. For example, it is unclear whether a bounty is associated with improving the issue-addressing likelihood in projects. By understanding this association, we

[1]https://www.bountysource.com

[2]https://blog.canya.com/2017/12/20/canya-acquires-majority-stake-in-bountysource-adds-over-46000-users/

could provide insights on how to better leverage bounties to evolve open source projects, and on how to improve the usability and effectivity of bounty platforms.

[Jiayuan syas:Show research questions.] In this paper, we study 3,509 issue reports with 5,445 bounties that were proposed on Bountysource from 1,203 GitHub projects, with a total bounty value of $406,425. We used a logistic regression model to study the association between 26 factors (including the timing of proposing a bounty and the bounty-usage frequency of a project) along 4 dimensions (i.e., the project, issue, bounty, and backer dimensions) and the issue-addressing likelihood. We found that:

1. The timing of proposing bounties is the most important factor that is associated with the issue-addressing likelihood.

2. Bounty issue reports are more likely to be addressed in projects which are using bounties more frequently.

3. Issue reports are more likely to be addressed if bounties are proposed earlier. Additionally, there is a risk of wasting money for backers who invest money on long-standing issue reports.

4. The total bounty value of an issue report is the most important factor that is associated with the issue-addressing likelihood in the first-timer projects.

We also manually identified the reasons why developers ignored bounties (i.e., the cases in which bounty issue reports were addressed while the bounty remained unclaimed) that are worth more than $100. We found that some developers addressed an issue cooperatively, making it difficult to choose a single developer that would be awarded the bounty. In addition, some developers are not driven by money to address issues.

Based on our findings, we have several suggestions for backers and the Bountysource platform. For example, backers should be cautious when proposing small (i.e., < $100) bounties on long-standing issue reports since the risk of losing the bounty exists. Bounty platforms should consider allowing for splittable multi-hunter bounties.

[Jiayuan syas:Removed organization.]

5.2 Background

In this section, we briefly introduce the open source bounty platform Bountysource.

5.2.1 Bountysource

Bountysource is a platform on which users can pledge a monetary incentive (a *bounty*) to address an issue report of an open source project. There exist two roles on Bountysource: the bounty backer and the bounty hunter roles.

Bounty backers, which may be anonymous, are users or developers who propose bounties for issue reports. A backer can set an expiration period for their bounty that has a value of more than $100. When the bounty expires, the money is refunded to the backer; otherwise, the bounty stays with the issue report until someone claims it. Note that bounties that are smaller than $100 are not refunded if they remain unclaimed. An issue report can have multiple bounties from one or more backers and a bounty can only be proposed for one issue report.

Bounty hunters are developers who address issue reports that have bounties. If a hunter works on an issue report, the hunter can attach certain information (i.e., the

estimated time of addressing, the code URL, or some comments) on Bountysource to indicate the progress. However, a bounty hunter could also work on the issue report without notifying Bountysource. Once a developer claims to have addressed an issue report, its bounty backer(s) can choose to accept (no response will be taken as an acceptance) or reject the claim. In this situation, backers have two weeks to make the decision (accept or reject). If no backer explicitly rejects the claim, the bounties will be paid to the developer automatically. Multiple bounty hunters can work on an issue report at the same time, but the bounties of an issue can only be rewarded to one bounty hunter. In particular, this is the bounty hunter who first claims the bounties while no backer explicitly rejects the claim.

When an issue report is submitted by an issue reporter, one or more bounty backers can propose bounty(ies) on the issue report. One or more developers of the issue report can choose to become bounty hunters to address the issue report but only one bounty hunter can get the bounty(ies).

Developers and users from more than 12 platforms (e.g., GitHub) propose bounties for issue reports through Bountysource. In this study, we focus on GitHub issue reports, since the majority of the bounties (see Section 5.3 for more details) that are proposed on Bountysource are for GitHub issue reports. Figure 5.1 shows the workflow of the bounty processes between GitHub and Bountysource. The workflow of a bounty starts with a bounty backer offering a bounty on Bountysource for a GitHub issue report. The bounty backers pledge money to Bountysource (the money is held by Bountysource) and they can choose to add bounty information to the GitHub issue report. For example, tagging the issue report on GitHub with a bounty label (see the

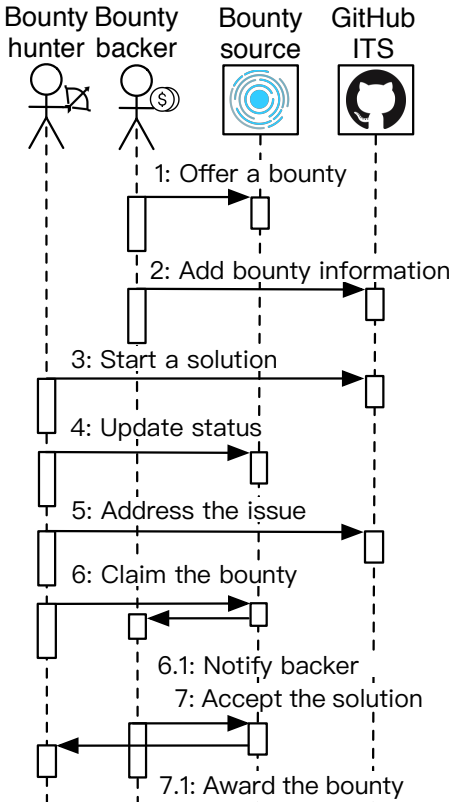


Figure 5.1: The workflow of the bounty between GitHub and Bountysource.

example[3] for details) to "advertise" the bounty, appending the bounty value to the title of the issue report or mentioning the bounty in the discussion of the issue report in GitHub. When a bounty hunter starts working on an issue, they can update their working status on Bountysource. After the issue report is addressed, the bounty hunter can submit a claim for the bounty on Bountysource and the backer will be notified by Bountysource. Once the bounty backer accepts the solution, the bounty hunter receives the money from Bountysource.

Based on the status of an issue report and whether a bounty is paid out, a bounty issue report has the following three statuses:

[3]https://github.com/austinpray/asset-builder/issues?q=label%3Abounty

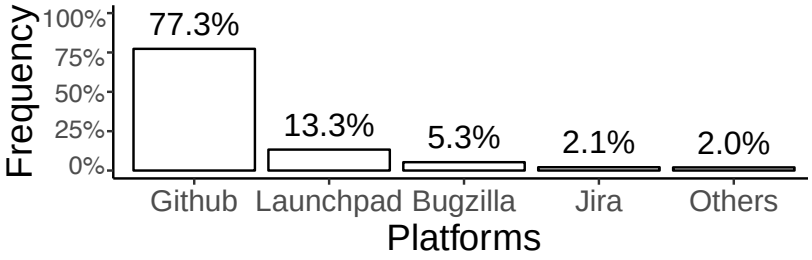


Figure 5.2: The distribution of Bountysource bounties across the supported ITSs.

Closed-paid: the issue report is closed and the bounty has been successfully rewarded to a bounty hunter. We defined such issue reports as *successful* bounty issue reports.

Open-unpaid: the issue report is open and the bounty is active. We defined such issue reports as *failed* bounty issue reports.

Closed-unpaid: the issue report is closed but the bounty remains unclaimed. We defined such issue reports as *ignored* bounty issue reports.

5.3 Data Collection

In our study, we focus on the bounties that are proposed through the Bountysource platform since it is one of the most popular platforms for open source projects. As explained in Section 5.2, Bountysource supports issue reports from several ITSs (e.g., GitHub and Bugzilla). Figure 5.2 shows the distribution of Bountysource bounties across its supported ITSs. The majority of the issue reports come from GitHub (77.3%), hence we focus our study on the bounties that were proposed for GitHub issue reports.

All information about the bounties is stored on Bountysource and all details about issue reports and their corresponding projects are stored on GitHub. Hence, we collected data for our study along three dimensions: the bounty, the issue report, and the project.

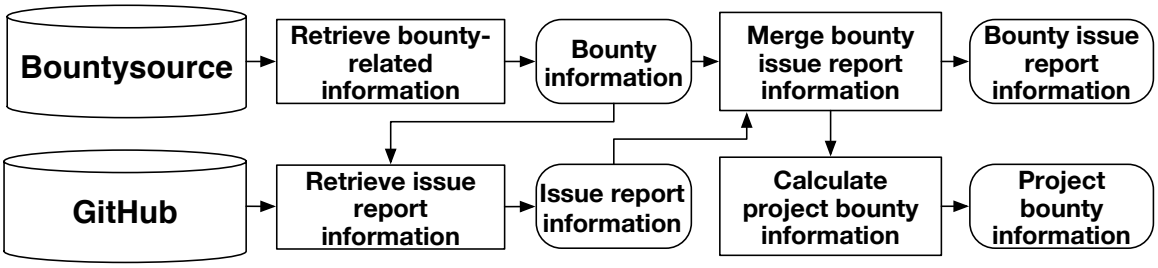


Figure 5.3: An overview of our data collection process.

Figure 5.3 presents an overview of our data collection process, which is broken down as follows:

Step 1: We retrieved the bounty and issue information from Bountysource using its official web API automatically.[4] The bounty information includes the backers who proposed the bounty, the proposed bounty value and the hunter who addressed the issue report. In addition, we collected basic information about the GitHub issue reports such as their id and URL.

Step 2: We retrieved the details of the issue reports, which are linked to Bountysource bounties by using the URL and id that we retrieved in step 1, from GitHub using its official web API automatically.[5] For example, we collected the description of the issue report, the creation date of the issue report, the comments that developers left under the report, and the labels of the issue report.

Step 3: We calculated the corresponding project's bounty information for each collected bounty issue report, such as the number of total bounty issue reports of a project.

In total, we collected 5,445 bounties with a total value of $406,425, together with their corresponding issue reports which were reported between Oct 19, 2012, and Oct

[4]https://bountysource.github.io/
[5]https://developer.github.com/v3/

Table 5.1: Dataset description.

Total number of bounties	5,445
Total number of claimed bounties	2,402
Total bounty value	$406,425
Total number of bounty hunters	882
Total number of bounty backers	2,534
Total number of issue reports	3,509
Total number of issue reports with multiple bounties	795
Total number of projects	1,203

5, 2017. Since some bounty issue reports were just created when we collected the data, we updated the status of the collected bounty issue reports after 200 days (i.e., Apr. 22, 2018) to have a more reliable status for these issue reports. We published our dataset online.[6] Table 5.1 describes our dataset.

We observed that **62.7% of the bounty issue reports are closed, while the bounties in almost one-third of these closed issue reports remain unpaid with a value of $41,856 in total.** Figure 5.4 shows the distribution of bounty issue reports across the three statuses. 37.3% of the bounty issue reports are failed (i.e., open-unpaid). Although 62.7% of the bounty issue reports were closed, almost one-third of their bounties were ignored (i.e., closed-unpaid). The total value of the ignored bounties ($41,856) is "frozen" in the Bountysource platform unless someone claims the bounty.

In the rest of the paper, when we discuss the issue-addressing likelihood, we only refer to the bounty issue reports that are successful (i.e., closed and paid out) or failed (i.e., still open). We do not take the issue reports which were ignored into consideration because the hunters might not be driven by the bounty in such issue reports. We

[6]https://github.com/SAILResearch/wip-18-jiayuan-bountysource-SupportMaterials

conducted a qualitative study of these closed-unpaid bounty issue reports to better understand them in Section 5.6.1. When a bounty issue report is closed and the bounty is paid out, we define this bounty issue report as addressed.

5.4 Preliminary Study

We aim to understand the association between the issue-addressing likelihood of an issue report and the factors that are related to the bounties of the issue report (e.g., the total value of bounties being proposed for an issue report) in different projects. Therefore, in this section, we present basic descriptive views of such bounty-related factors. From these statistics, we can get a basic view of the characteristics of bounties, and of how bounties are used across projects.

Approach:

We first present the following basic descriptive statistics: (1) the distribution of the number of days between the reporting of an issue and its first bounty being proposed (*I_B_days_before_bounty*); (2) the distribution of the total bounty value of an issue report (*I_B_total_value*); (3) the distribution of the number of bounties that a bounty issue report has (*I_B_cnt*); (4) the distribution of bounty issue reports that have a bounty label (*I_B_has_label*). We also investigate how bounties are used across projects. We present the distribution of the total number of bounties used in projects (the bounty-usage frequency).

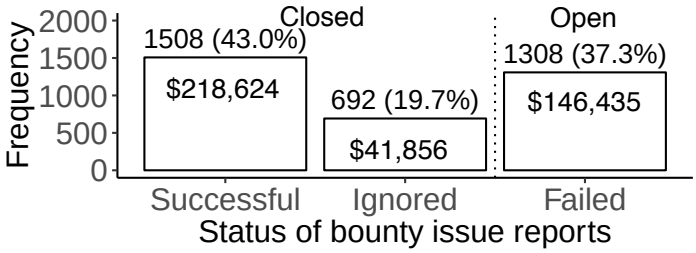


Figure 5.4: The distribution of the possible statuses of bounty issue reports and their corresponding cumulative bounty value.

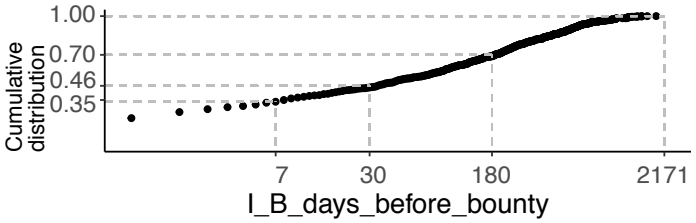


Figure 5.5: The empirical cumulative distribution of *I_B_days_before_bounty*.

Results:

35% of the bounties were proposed within 7 days from the creation of an issue report, while 30% of the bounties were proposed after more than 180 days. Figure 5.5 shows the empirical cumulative distribution function of *I_B_days_before_bounty*. We observe that in 35% of the issue reports their first bounty was proposed within seven days after their creation. Only 11% of the bounties were proposed between 7 and 30 days after the creation of an issue report. 24% of the bounties were proposed between 30 and 180 days and the remaining 30% of the bounties were proposed after 180 days. The frequency with which bounties are proposed is lower in the first seven days than later on. One possible explanation is that bounty backers may wait and see if issues are addressed without a bounty. After waiting for a period of time without getting their issue addressed, bounty backers start to propose bounties.

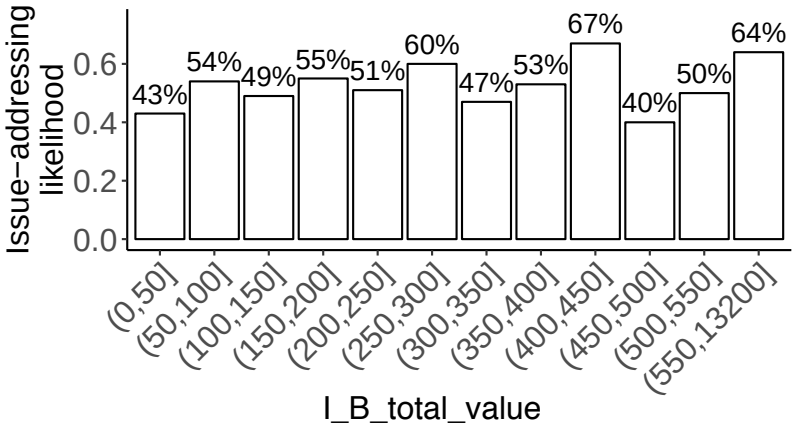


Figure 5.6: The issue-addressing likelihood of the proposed bounty value ranges.

The distribution of *I_B_total_value* is skewed and the correlation between *I_B_total_value* and the issue-addressing likelihood is weak. We observe that the skewness and kurtosis values of the distribution of *I_B_total_value* are 13 and 236, respectively. The first, second, and third quartile values are $15, $30 and $100. Figure 5.6 presents the issue-addressing likelihood of an issue report against the bounty value of the issue report. We do not observe an obvious pattern between them. The correlation between the bounty value and the issue-addressing likelihood is surprisingly weak (0.14).

90% (i.e., 2,541) of the studied bounty issue reports only have one or two bounties. We observe that 75% of the bounty issue reports only have one bounty and 15% of the bounty issue reports have two bounties.

We also observe that 56% (i.e., 1,568) of the bounty issue reports are explicitly labeled as such.

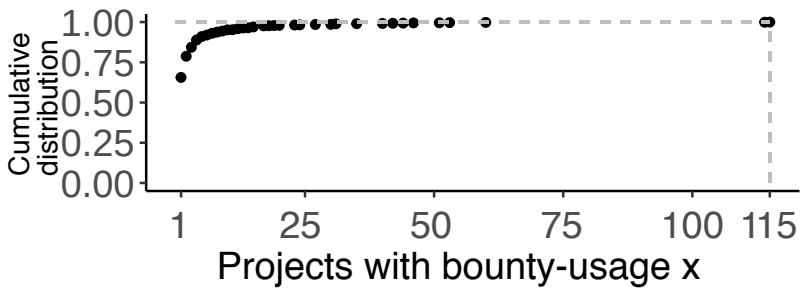


Figure 5.7: The empirical cumulative distribution of the bounty-usage frequency of projects. The bounty-usage frequency is the total number of used bounties in a project.

More than half of the projects only used a bounty once, while two projects used bounties very frequently (more than 100 times). Figure 5.7 shows the empirical cumulative distribution of the bounty-usage frequency across projects. As shown in Figure 5.7, the distribution is skewed (with a variance of 57.02). 612 (66%) projects used a bounty only once. 52 (6%) projects used bounties at least 10 times and only 9 projects used a bounty more than 50 times. In order to better study the research questions, we propose a bootstrap-derived data preprocessing method to reduce bias caused by different bounty-usage frequency across projects in Section 5.5.

5.5 Study Result

In this section, we present the result of our empirical study on issue bounties in GitHub open source projects. We first investigate which studied factors are associated with the issue-addressing likelihood. Then, we investigate how the association between the studied factors and the issue-addressing likelihood changes in projects with a different bounty usage frequency. By understanding this association, we can provide insights for the backers into how to best leverage bounties to improve the issue addressing process.

We can also provide suggestions for the Bountysource platform to improve its system. *[Jiayuan syas:Repeat with RQ2]*

5.5.1 RQ1: Are the studied factors associated with the issue-addressing likelihood of bounty issue reports in GitHub projects?

[Jiayuan syas:Connect to preliminary study.] Prior studies showed that bounty-related factors (e.g., the value of bounties) have an association with various software development tasks, such as developing new features (Krishnamurthy and Tripathi, 2006) and addressing security issues (Maillart et al., 2017). However, little is known about how these factors are related to the issue-addressing likelihood of bounty issue reports. In addition, factors that are related to a bounty issue report itself and its backers may have associations with the issue-addressing likelihood of the bounty issue report. For example, an issue report that attracts more attention (e.g., comments and participants) from the community may have a higher likelihood of being addressed. Therefore, in this section, when examining the association between bounty-related factors and the issue-addressing likelihood of bounty issue reports, we also take the factors that are related to issue reports and backers into consideration.

Approach

We construct logistic regression models to study the relationship between the studied factors and the issue-addressing likelihood. Note that our goal of constructing models is not for prediction but for interpretation. The logistic regression model is a robust and

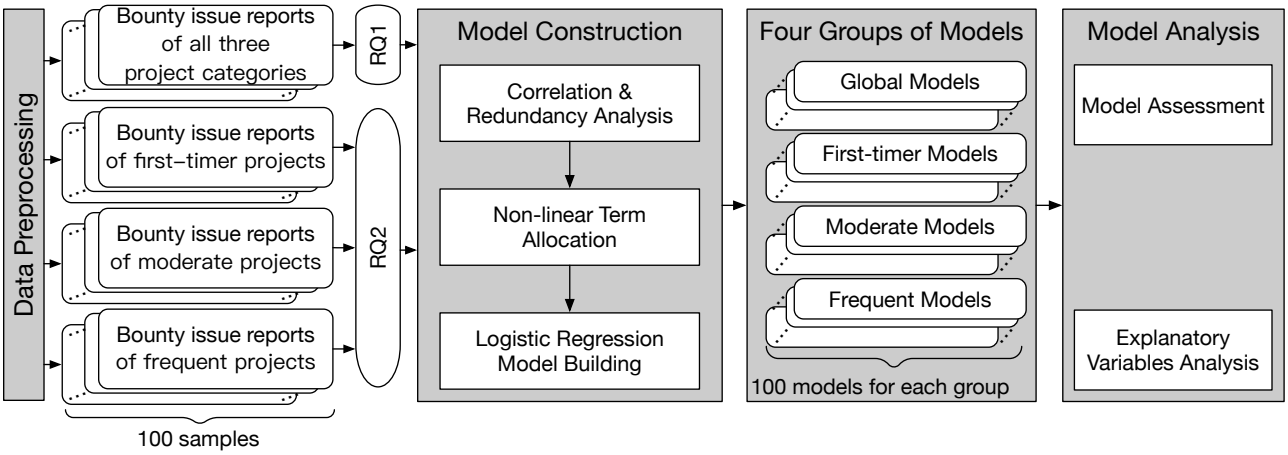


Figure 5.8: An overview of the data preprocessing, model construction, and analysis steps of our approach.

highly interpretable technique, which has been applied successfully in software engineering studies, e.g., to predict the closure rate of GitHub issues (Jarczyk et al., 2018), predict bugs (McIntosh et al., 2016; Palomba et al., 2017), and classify the information that is discussed in GitHub issues (Arya et al., 2019).

Figure 5.8 shows the flow of our approach. Below, we elaborate on the studied factor, the processes of the data preprocessing, the model construction, and the analysis of our models.

Studied factors: Through the process that is described in Section 5.3, we extracted 26 factors along the following 4 dimensions:

1. **Issue report basic:** Eight factors which estimate the length and the popularity of an issue report.

2. **Issue report bounty:** Four factors which describe the bounty usage within a bounty issue report.

3. **Project bounty:** Six factors which reflect the bounty usage within a project.

Table 5.2: The description and rationale for the factors in the *Issue report basic*, the *Issue report bounty*, *Project bounty* and the *Backer experience* dimensions. The factors which are marked with '*' are time-dependent factors which are calculated at the time when the bounty is proposed

Factor name	Description	Rationale
Issue report basic		
I_content_len*	The length of an issue report and its comments (in characters).	These factors reflect the amount of supportive information that an issue report has. Issue reports with more supportive information may help developers to address them.
I_code_len*	The total length of the code snippets in an issue report and its comments (in characters).	
I_code_proportion*	The proportion of code in an issue report and comments (i.e., $\frac{I\_code\_len}{I\_content\_len}$).	
I_link_cnt*	The number of links in an issue report and its comments.	The discussion activities reflect the popularity of an issue report, which may have a relationship with the issue-addressing likelihood.
I_img_cnt*	The number of images in an issue report and its comments.	
I_cmnt_cnt	The number of comments that an issue report received.	
I_participant_cnt*	The number of participants in the discussion of an issue.	
I_cmnt_per_day_mean*	The mean number of comments per day for an issue report.	
Issue report bounty		
I_B_days_before_bounty*	The number of days between the creation of an issue report and its first bounty.	The timing of proposing bounties may have a relationship with the issue-addressing likelihood.
I_B_total_value	The total bounty value of the issue report.	A higher bounty may attract more developers.
I_B_cnt	The number of bounties that a bounty issue report has.	A higher number indicates that more backers are interested in getting this issue addressed.
I_B_has_label	Whether a bounty issue report is tagged with a bounty label.	A bounty label could help draw attention from the community (i.e., because the label acts as an advertisement), which may have an association with the issue-addressing likelihood.
Project bounty		
P_B_I_cnt*	The total number of issue reports with at least one bounty of a project.	These five factors reflect the bounty activity of the project. A different level of activity may have a different association with the issue-addressing likelihood in the project.
P_B_paid_cnt*	The total number of paid bounty issue reports of a project.	
P_B_open_cnt*	The number of open bounty issue reports of a project.	
P_B_paid_proportion*	The proportion of paid bounty issue reports of a project.	
P_B_total_value*	The total value of the bounties of a project.	
P_B_usage_group	The group of projects.	Different groups of projects may have different issue-addressing likelihoods (see Section **??**).
Backer experience		
Backer_exp_B_median-/sum/max_value*	The median/sum/max value of bounties which the backers of this bounty have ever proposed in the past.	Bounties from a backer who has proposed bounties often, or proposed high-value bounties in the past may attract more attention from developers.
Backer_exp_B_median-/sum/max_cnt*	The median/sum/max number of bounties which the backers of this bounty have ever proposed in the past.	
Backer_role_any_insider*	Whether any of the backers has ever contributed to the project.	A backer who has ever interacted with the project before may help the bounty attract more attention from the community.
Backer_role_have_reporter*	Whether the issue reporter is one of the backers for that issue report.	

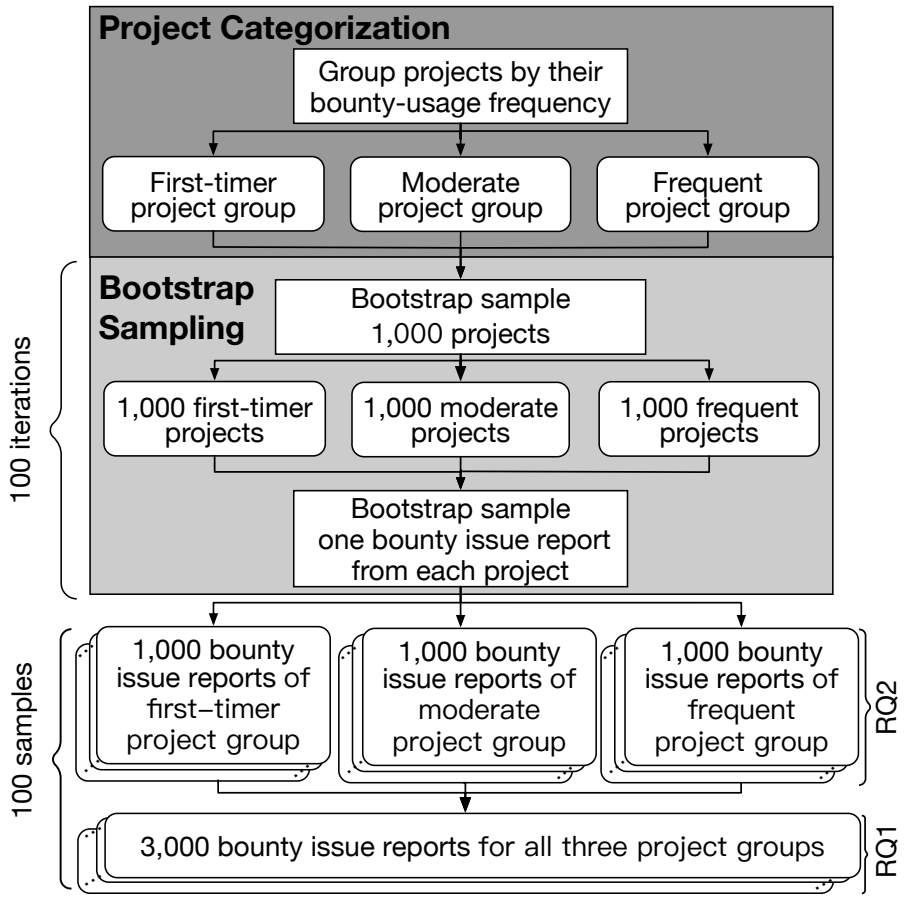


Figure 5.9: An overview of our data preprocessing approach.

4. **Backer experience:** Eight factors which capture the bounty experience of the backers of a bounty issue report.

Table 5.2 summarizes the descriptions of and rationales behind the studied factors. The factors which are marked with '*' are time-dependent factors which are calculated at the time when the bounty is proposed. For example, the *I_content_len** factor is the length (in characters) of an issue report and its comments when the first bounty of the issue report was proposed.

Note that the factors in the project bounty, issue report basic, and backer experience dimensions cannot be changed by a backer who wants to propose a bounty and we consider these factors as the confounding factors for which we want to control. The bounty backers can control the factors in the issue report bounty dimension. For example, a bounty backer can choose the timing of proposing a bounty on an issue report (i.e., *I_B_days_before_bounty*), the bounty value (i.e., *I_B_total_value*), and whether to add a bounty label to the issue report (i.e., *I_B_has_label*).

Data preprocessing: Figure 5.9 gives an overview of our data preprocessing approach. We elaborate on each step below.

Project categorization: Given the variance of the bounty-usage frequency across different projects, it is not advisable to study all the issue reports as one group when we study the bounties at the issue report level. Therefore we categorize the projects into the following three groups:

1. **First-timer project:** Projects which have only one bounty issue report.

2. **Moderate project:** Projects which have 2 to 50 bounty issue reports.

3. **Frequent project:** Projects which have more than 50 bounty issue reports.

It is important to study the bounties in the first-timer projects, since users of such projects may not have former bounty experience. We grouped the projects that have more than 50 bounty issue reports as well since we assume that in such projects the community is more familiar with the use of bounties. Note that we set the threshold 50 for moderate and frequent projects empirically. We performed a sensitivity analysis on different thresholds (i.e., 40 and 60) and the results show that our findings still hold (see Appendix C in our supplementary material (Zhou, 2018) for more details).

After grouping the projects into the above mentioned three groups, we have 550 (59%) first-timer projects with 550 bounty issue reports, 374 (40%) moderate projects with 1,717 bounty issue reports, and 9 (1%) frequent projects with 549 bounty issue reports.

Bootstrap sampling: After grouping the projects into the three groups, we used a bootstrap sampling approach to sample issue reports across projects in order to balance the data. We used bootstrap sampling to reduce the bias that is caused by the unbalanced number of projects across the three groups. We first randomly sampled 1,000 projects from each group with replacement. Then we randomly sampled one bounty issue report from each sampled project, to avoid a bias towards projects with more issue reports than other projects in the same group. Hence, we sampled 1,000 bounty issue reports from each of the 3 project groups. To make our results more reliable, we repeated the sampling process 100 times with different random seeds. We ended up with 100 samples with 3,000 issue reports each (1,000 issue reports for each group). On average, 54.3% of the bounty issue reports were sampled during one iteration of the bootstrap sampling process.

Data construction: Figure 5.8 shows an overview of our model construction approach. The presence of correlated and redundant features greatly impacts the interpretability of the generated models (i.e., multicollinearity) (Farrar and Glauber, 1967). Hence, we performed correlation and redundancy analysis (see Section 4.5.1) to removed correlated and redundant factors (see Appendix A in our supplementary material Zhou (2018) for more details and the factors that were included in the models). We ended up with three factors in the project bounty dimension, six factors in the issue report basic dimension, four factors in the issue report bounty dimension, and three factors in

the backer experience dimension. We added non-linear terms in the model to capture more complex relationships in the data by employing restricted cubic splines (Harrell, 2006). Finally, we built logistic regression models based on 100 samples (3,000 issue reports with 1,000 issue reports for each group) and ended up with 100 models. We refer to these 100 models which are constructed to understand the global relationship as the **global model**. See our supplementary material Zhou (2018) for more details about our model construction.

Model analysis: For each logistic regression model, we used the Area Under the Receiver Operating Characteristic Curve (i.e., *AUC*) (see Section 4.5.1) to evaluate the performance. A higher AUC meaning that the model has a higher ability to capture the relationships between the explanatory factors and the response factor. To check whether the models are not overfitted, we calculate their *optimism* values using a bootstrap-derived approach (see Section 4.5.1) *[Jiayuan syas:check the appendix.]* (see Appendix B in our supplementary material Zhou (2018) for the calculation of the optimism value). A small optimism value suggests that a model does not suffer from overfitting, while an optimism of 1 indicates that the model is 100% overfitting the dataset

To measure the explanatory power of each factor in the model, we computed the Wald χ^2 value (see Section 4.5.1). A larger Wald χ^2 value indicates a higher explanatory power of the factor in the model. We further applied a χ^2-test (see Section 4.5.1) to the calculated Wald χ^2 values to test whether a factor contributes a statistically significant amount of explanatory power to the model. In this study, we consider factors of which the χ^2-test has a p-value of less than 0.001 as significantly important.

Table 5.3: The 5-number summary of AUC and optimism values of our models.

Model Types		**Quantile**				
		Min	1^{st}	Median	3^{rd}	Max
Global	AUC:	0.72	0.73	0.74	0.74	0.75
	optimism:	0.00	0.00	0.01	0.01	0.02
First-timer	AUC:	0.70	0.73	0.74	0.76	0.80
	optimism:	0.00	0.00	0.01	0.02	0.04
Moderate	AUC:	0.66	0.68	0.70	0.71	0.74
	optimism:	0.00	0.01	0.01	0.02	0.05
Frequent	AUC:	0.79	0.81	0.82	0.83	0.86
	optimism:	0.00	0.01	0.01	0.02	0.04

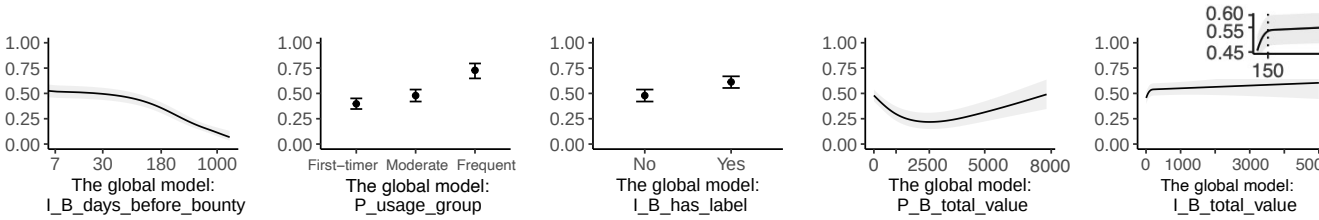


Figure 5.10: The plots show the relationship between the studied factors and the issue-addressing likelihood for the global models. For each plot, we adjusted all factors except the studied factor to their median value in the model and recomputed the issue-addressing likelihood. The grey area represents the 95% confidence interval.

In addition, to further understand how a factor influences the value of the response variables, we plotted the estimated issue-addressing likelihood against a factor. Since all models across 100 samples showed similar patterns of influence for the factors, we randomly selected a sample as an example to build models and visualize the results (see Figures 5.10 and 5.12). The analysis allows us to further understand how a factor affects the issue-addressing likelihood. We used the *R rms* package during the construction and analysis of our models.

Table 5.4: The results of the model analysis for four groups of models. The **NL** indicates the non-linear term and the **D.F.** indicates the degree of freedom.

		Global Model		**First-timer Model**		**Moderate Model**		**Frequent Model**	
Factors		Overall	NL	Overall	NL	Overall	NL	Overall	NL
I_B_days_before_bounty	D.F.	4	3	4	3	4	3	4	3
	χ^2	172.30***	35.71***	26.83***	4.79	43.59***	7.44	51.67***	10.47
P_B_usage_group	D.F.	2		-		-		-	
	χ^2	35.08***		-		-		-	
I_B_total_value	D.F.	2	1	2	1	2	1	2	1
	χ^2	31.19***	29.80***	34.00***	0.28	3.07	2.97	13.67	9.99
I_code_proportion	D.F.	1		1		1		1	
	χ^2	11.62		0.74		14.84***		0.21	
I_B_has_label	D.F.	1		1		1		1	
	χ^2	33.08***		4.16		6.99		0.022	
Backer_exp_B_max_value	D.F.	3	2	3	2	3	2	3	2
	χ^2	16.21	16.14	1.95	0.80	1.36	1.32	22.86***	21.87***
P_B_paid_proportion	D.F.	3	2	-		2	3	2	2
	χ^2	14.56	2.51	-		7.77	0.01	29.39***	29.39***
P_B_total_value	D.F.	2	1	-		2	1	2	1
	χ^2	15.75***	12.18***	-		1.44	0.34	13.86	10.99
I_img_cnt	D.F.	1		1		1		1	
	χ^2	1.58		0.68		1.55		0.69	
I_link_cnt	D.F.	1		1		1		1	
	χ^2	0.09		3.09		0.02		2.60	
I_content_len	D.F.	1		1		1		1	
	χ^2	4.32		0.02		2.8		1.72	
I_cmnt_perday_mean	D.F.	2	1	2	1	2	1	2	1
	χ^2	4.37	0.76	3.83	0.01	0.73	0.34	0.56	0.07
I_B_cnt	D.F.	1		1		1		1	
	χ^2	0.433		2.54		1.48		8.39	
I_cmnt_cnt	D.F.	1		1		1		1	
	χ^2	0.49		0.00		1.55		7.34	
Backer_role_any_insider	D.F.	1		1		1		1	
	χ^2	1.96		7.92		1.71		7.60	
Backer_role_have_reporter	D.F.	1		1		1		1	
	χ^2	0.34		7.92		1.78		0.03	

P-value of the χ^2 test: '***' < 0.001

Results

Our models capture the relationship between the explanatory variables and the response variable well, and have a reliable performance. The median AUC of our global models is 0.74 (see Table 5.3), which indicates that our models have a good ability to capture the relationship between the explanatory variables and the response variable, and the low median optimism values (0.01) indicate that our models do not overfit the dataset.

In the global view, the timing of proposing the bounties is the most important factor that has a significant relation with the issue-addressing likelihood. Table 5.4 shows that the timing of proposing the bounties, the bounty-usage frequency of projects, the bounty label of issue reports, the total value of the bounties of a project, and the total bounty value of the issue report contribute a significant amount of explanatory power to our models. The timing of proposing the bounties contributes the most explanatory power by far, based on the Wald χ^2 value.

Projects that use bounties more frequently have a higher bounty issue-addressing likelihood. We observe a positive association between the issue-addressing likelihood and *P_B_usage_group* in the global models. One possible explanation is that projects with a higher bounty-usage frequency are more likely to maintain documents to introduce how bounties work in such projects, so that backers can gain more experience and background about proposing bounties (e.g., at the proper time with a proper value) and the hunters react to bounties more actively than in projects with a lower bounty-usage frequency. For example, the *eslint* project

maintains a document on how bounties work.[7] The *eslint* project has 43 successful (i.e., closed-paid) and only one failed (i.e., open-unpaid) bounty issue report.

To further test our assumption, we performed a qualitative study to investigate whether projects that use bounties more frequently are more likely to have a bounty document. We calculated the representative sample sizes (Cochran, 2007) and randomly sampled 80 first-timer projects and 77 moderate projects as statistically representative samples with a 95% confidence level and a 10% confidence interval. We selected all nine frequent projects. The first two authors manually examined the GitHub pages of each sampled project and checked whether the project has a document that explains the bounty process. The Cohen's Kappa is 0.83, which indicates a high level of agreement. The proportions of projects that have bounty documents are 5% (4/80), 31% (24/77), and 89% (8 out of 9) in the first-timer, moderate, and frequent projects, which suggests that projects that use bounties more frequently are more likely to have a bounty document.

In general, issue reports for which bounties were proposed earlier have a higher likelihood of being addressed. We observe a negative trend of the issue-addressing likelihood as the time to propose a bounty increases, especially for the issue reports in which bounties were proposed after 180 days. One possible explanation is that as time progresses, the risk of a report becoming obsolete exists, leaving the issue report unaddressed even after a bounty is proposed. For example, an issue report[8] that was created on Feb 4, 2016 in the *uappexplorer* project requested a new feature for an Ubuntu Phone Application. The owner of the application and another developer both showed great interest in this issue. Because of the lack of time, the feature was never added. A

[7] https://eslint.org/docs/developer-guide/contributing/working-on-issues
[8] https://github.com/bhdouglass/uappexplorer/issues/69

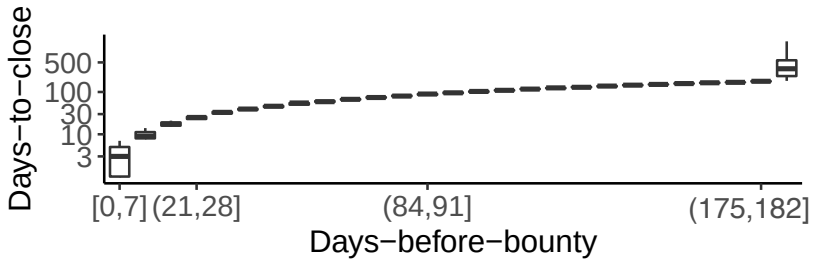


Figure 5.11: The distribution of the number of days to close issue reports since bounties were proposed across different time ranges.

bounty of $5 was proposed[9] after almost one year, on Jan 12, 2017. However, the issue report was closed because Ubuntu Phone was no longer used making the issue report obsolete. In addition, **backers carry the risk of wasting their money by proposing small bounties on such long-standing issue reports as such small amounts are not refunded to the backer in case the bounty fails.**

Another assumption for the lower issue-addressing likelihood of the issue reports for which bounties were proposed later is that such issue reports are difficult to address. To test our assumption, we studied the relationship between the issue-addressing speed and *I_B_days_before_bounty*. Figure 5.11 shows the boxplot of the number of days that were taken to close issues (i.e., *days-to-close*) against different *days-before-bounty*. We observe that the issue reports in which bounties were proposed later took longer to be addressed.

Issue reports with a bounty label have a higher likelihood of being addressed than bounty issues without a bounty label. Whether a bounty issue has a bounty label (i.e., *I_B_has_label*) is the third most important factor in the global model. Figure 5.10 shows that bounty issue reports with a bounty label have a higher likelihood of being addressed. It is intuitive that a better exposure of the bounty can help attract more attention from the community. Tagging an issue report with a bounty label is the most

[9]http://bit.ly/2Q3BIns

direct way of advertising a bounty because the label will be shown in the ITS. In addition, developers can search for bounty issue reports easily using the bounty label.

Finally, *I_B_total_value* contributes significant explanatory power to the global model and we suggest one to propose bounties with a value of $150. Figure 5.10 shows that the issue-addressing likelihood increases from 0.45 to 0.54 as the bounty value increases from $5 to $150 and stays almost stable after $150. In other words, the bounty value does not improve the issue-addressing likelihood further once the bounty value is equal to $150. The *P_B_total_value* is a significantly important factor in the global model, which indicates that the total amount of bounties that a project has is also of significance. Figure 5.10 shows that the issue-addressing likelihood and *P_B_total_value* has a negative relationship when the *P_B_total_value* is no more than $2,500. After $2,500, the higher *P_B_total_value*, the higher the issue-addressing likelihood.

Summary: The timing of proposing bounties is the most important factor that has a significant relation with the issue-addressing likelihood. Issue reports are more likely to be addressed if they are for projects in which bounties are used more frequently and if they are proposed earlier. In addition, it is important to advertise bounties for bounty issue reports by tagging bounty labels. The total value of bounties of a project and an issue also have a significant relation with the issue-addressing likelihood.

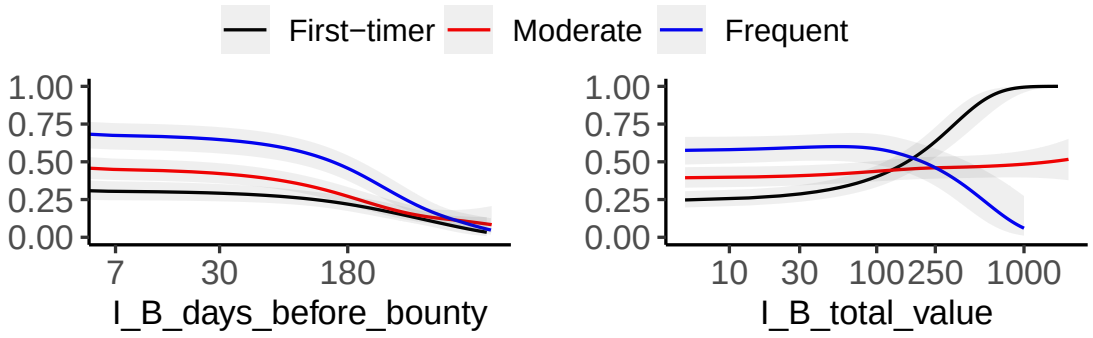


Figure 5.12: The plots show the relationship between the studied factors and the issue-addressing likelihood for the first-timer models, the moderate models and the frequent models in the selected sample. For each plot, we adjusted all factors except the studied factor to their median value in the model and recomputed the issue-addressing likelihood. The grey area represents the 95% confidence interval.

5.5.2 RQ2: How does the association between the studied factors and the issue-addressing likelihood change in projects with different bounty usage frequencies?

In 5.5.1, we investigate which studied factors are associated with the issue-addressing likelihood. In addition, prior work shows that the impact of bounties on the addressing of software security issues varies across projects (Maillart et al., 2017). Similarly, in this section, we further investigate how the association between the studied factors and the issue-addressing likelihood changes in projects with a different bounty usage frequency.

Approach

To understand how the association between bounties and the issue-addressing likelihood changes in projects with a different frequency of using bounties, we follow the

same model construction and analysis approach as introduced in Section 5.5.1. Instead of building models on the entire set of issue reports, we build logistic regression models on the bounty issue reports of each project group separately (i.e., the first-timer projects, the moderate projects, and the frequent projects). To condense our writing, we refer to the models for the first-timer, moderate, and frequent projects as the **first-timer**, **moderate**, and **frequent models**, respectively.

Results

Our models capture the relationship between the explanatory variables and the response variable well, and have a reliable performance. The median AUCs for the first-timer, moderate, and frequent models are 0.74, 0.70, and 0.82, respectively (see Table 5.3), which indicates that our models have a good ability to capture the relationship between the explanatory variables and the response variable. The low median optimism values (i.e., 0.01 for all models) indicate that our models do not overfit the dataset.

The timing of proposing bounties still plays a significantly important role in all three categories of projects. Table 5.4 shows that *I_B_days_before_bounty* is the most important factor (i.e., it contributes the highest explanatory power) in the moderate and the frequent models. In the first-timer model, *I_B_days_before_bounty* is the second important factor. Figure 5.12 presents the relationship between the issue-addressing likelihood and *I_B_days_before_bounty* for the first-timer, moderate, and frequent models. We observe that *I_B_days_before_bounty* has the same negative relationship with the issue-addressing likelihood in all three models. We also observe that the frequent model has the highest issue-addressing likelihood compared with

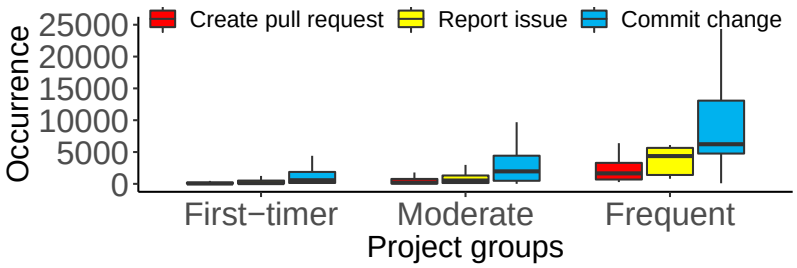


Figure 5.13: The distributions of the occurrences of three activities (i.e., the create pull request, the report issue and the commit change) in each project group.

the first-timer model and the moderate model when receiving bounties in the same number of *days-before-bounty*, which indicates that proposing bounties earlier will achieve the highest issue-addressing likelihood in projects which use bounties frequently.

The total bounty value of an issue report is the most important factor that has an association with the issue-addressing likelihood in the first-timer projects, while it is less important in the projects where bounties are used more frequently. From Table 5.4, we can see that *I_B_total_value* (i.e., the total bounty value of a bounty issue report) is the most important factor in the first-timer model with a positive association with the issue-addressing likelihood, while it is not a significantly important factor (the p-value is larger than 0.001) in the moderate and frequent models. When comparing the ratio of the bounty value between successful and failed issue reports among the first-timer, moderate, and frequent projects, we can see that the first-timer projects have a larger ratio (2.5) than the moderate (2) and frequent projects (1.4). This explains why the value of bounty is more important in the first-timer projects than that in the moderate and frequent projects. The highest ratio in the first-timer projects also indicates that developers may expect a better payout when addressing issues in first-timer projects than in other projects.

Why do the first-timer projects have a larger ratio than moderate and frequent projects? One possible assumption is that the first-timer projects may not be as active as moderate and frequent projects, therefore backers would be required to propose bounties with higher values to attract enough attention from the community for addressing issues. To investigate this assumption, we examined the frequency of various activities of the projects, in terms of the number of pull requests, issue reports, and commits. Figure 5.13 shows the distributions of the occurrences of these three activities in each project group. **Projects with fewer bounty issue reports are usually less active (in terms of the number of pull requests, issue reports, and commits) than projects with more bounty issue reports.** Another possible explanation is that backers in first-timer projects have no experience in proposing bounties and sometimes overestimate the value of addressing an issue report. In this situation, the overestimated bounty issue reports may be more likely to attract more attention from the community and get addressed.

For the frequent model, we observe a negative relationship between the issue-addressing likelihood and the total bounty value. One possible explanation is that in the frequent projects, where communities have more experience in using bounties, backers are more likely to propose bounties with a well-estimated value. Therefore, issue reports with bounties of higher value are more likely difficult to resolve and have a lower issue-addressing likelihood. For the moderate model, we observe a weak positive relationship.

Except for the bounty-related factors that we discussed above, we observed other factors from the project bounty and the backer experience dimensions which are also significantly important (i.e., the p-value of the χ^2-test is less than 0.001) in frequent

models. In the backer experience dimension, the max value of bounties which the backers of this bounty have ever proposed in the past (i.e., *Backer_exp_B_max_value*) is significantly important in the frequent models, while it is not significantly important in the other two models. In other words, the experience of backers is more important in projects that use bounties frequently than in those that use bounties less frequently. We also observed that the proportion of paid bounty issue reports (i.e., *P_B_paid_proportion*) plays a significant role in the frequent models, while its role is not significant in the other two models. In addition, it has a positive association with the issue-addressing likelihood of bounty issue reports. In short, the project bounty and the backer experience dimensions are more important in frequent models than in another two models.

In the issue report basic dimension, the proportion of code in an issue report (i.e., *I_code_proportion*) is important in moderate models.

Summary: In general, the timing of proposing bounties is the most important factor that has a relation with the issue-addressing likelihood in moderate and frequent projects. The total bounty value that an issue report has is the most important factor that has a relation with the issue-addressing likelihood in the first-timer projects, while it is not as important for projects in which bounties are more frequently used.

5.6 Discussion

In this section, we first study the ignored bounty issue reports. Then we highlight the implications of our findings.

5.6.1 Studying ignored bounty issue reports

In Section 5.4, we observed that in 19.7% of the bounty issue reports the bounties were ignored (i.e., closed-unpaid). In these cases, the issue reports were closed but the bounties remained unclaimed. It seems that money was not the driver that motivated developers to address these issues. To understand the possible reasons behind this phenomenon, we manually studied all 692 ignored bounty issue reports (with a total bounty value of $41,856). Because the "closed" status of an issue report does not necessarily mean that the issue was addressed (e.g., a report may have been a duplicate of another issue report), it is difficult to automatically identify whether an issue in the closed issue report was addressed. Therefore, we need to manually examine the closed-unpaid bounty issues reports to filter out the reports that were closed for another reason than the issue being addressed.

21.8% (479 out of 2,200) of the addressed bounty issue reports were not paid out. We identified that 479 out of the studied 692 bounty issue reports were closed because the issues were addressed. Such cases are interesting since the developers could have claimed the bounty but they did not. We manually examined the discussion for these 479 issue reports. We identified 19 cases in which developers gave an explanation for not claiming the bounty. We grouped the explanations as follows:

The developer is not driven by money. In 7 out of 19 cases a developer refused to claim the bounty because they were not motivated by money to address the issue. For example, one developer was against the bounty because they felt that the issue-addressing process should be driven by the interests of the community rather than money. A contributor of the *Brython* project, refused the bounty because he wanted to keep *Brython* free from monetary motivations: "*What is this 'bounty' thing? Needless to say, I refuse*

that anybody (me included, of course) gets paid for anything related to Brython."[10] In addition, he also asked backers to remove all bounties within the *Brython* project although he respected prior paid bounties. There were five bounty issue reports in the *Brython* project and four bounty issue reports that were addressed without claiming the bounty.

The developer is afraid of sending the wrong message. Krishnamurthy and Tripathi (2006) pointed out that financial incentives may cause confusion in the community because the financial incentives may drive a project's own product development cycle away from what is in place. We observed that developers expressed similar concerns. A developer of the *Facebook/HHVM* project, explained that: "*That's very generous of you, but I can't accept a bounty for doing my job. :-P It would be a conflict of interest, and I worry it sends the wrong message about how we prioritize issues from the community.*"[11]

The issue report was addressed by more than one developer. We found nine cases where bounties ended up unclaimed because an issue report was addressed by multiple developers cooperatively and they felt inappropriate to claim the bounty by one developer. For example, the issue[12] was addressed by two developers and because a bounty cannot be split into two parts, no one claimed it.

5.6.2 The implications of our findings

Backers should consider proposing a bounty as early as possible and be cautious when proposing small bounties on long-standing issue reports. The timing of

[10] http://bit.ly/2OTYx0x
[11] http://bit.ly/2OZw1uw
[12] http://bit.ly/2PrMiHV

proposing a bounty is an important factor that is related to the issue-addressing likelihood. In Sections 5.5.1 and 5.5.2, we showed that issue reports for which bounties were proposed earlier are more likely to be addressed. Additionally, we observed that issue reports for which bounties were proposed earlier are more likely to be addressed faster. Backers benefit from the higher issue-addressing likelihood and faster issue-addressing speed by proposing bounties earlier.

In Section 5.5.1, we also noticed a drop (i.e., from 53.2% to 30.1%) of the issue-addressing likelihood when backers proposed bounties for long-standing (i.e., more than half a year) issue reports. This drop might be due to such issue reports having become obsolete or being hard to address. Since bounties with a value of less than $100 will not be refunded to the backers if the issue report remains unaddressed, we suggest that backers be cautious when proposing small bounties on long-standing issue reports.

Backers should consider proposing a bigger bounty in first-timer bounty-projects. Although the issue-addressing likelihood is only 37.4% for projects with no bounty-usage experience, the first-timer model in Section 5.5.2 shows that the bounty value of an issue report is the most important factor in the first-timer projects, as the issue-addressing likelihood is higher for higher bounty values. The high ratio (2.5) of the bounty value of successful bounty issue reports to the bounty value of failed bounty issue reports also supports this finding. We suggest that backers of projects with no bounty-usage experience propose higher bounty values for issue reports.

Bounty platforms should allow for splittable multi-hunter bounties. In addition to a voluntary nature, open source projects have a collaborative nature. Some issues are hard for a developer to address alone. Hence, we encourage developers to work

together, especially for issue reports which have a high bounty value (as these issue reports are often harder to address). However, the current bounty workflow only allows **one** bounty hunter to claim the bounty, which goes against the collaborative nature of open source. It may also drive the developers, who want to collaboratively address the issue, away because not every participant will get a reward at the end. Therefore, bounty platforms should consider adding the ability for a bounty to be split across multiple hunters to encourage developers to work together on difficult bounty issues.

Bounties should be transferable. The total value of all addressed-unpaid bounties ($43,256) is "frozen" in Bountysource. In addition, the median number of days between the closing date of the issue report and the date of collecting our data is 372.5 (Figure 5.11), which means that more than half of the bounties from the ignored bounty issue reports were unclaimed for at least one year. By manually examining these 479 addressed-unpaid bounty issue reports, we found 31 cases in which someone reminded the bounty hunter to claim the bounty, however, the reminder was ignored. By reassigning these unclaimed bounties to other issue reports, a larger value could be created for these "stale" bounties. For example, Bountysource can suggest and enable backers to assign their long-standing unclaimed bounties to another unaddressed issue report, which has many comments (i.e., people care about it), to encourage developers to address the issue report. Interestingly, we also found suggestions from developers who did not want to receive the bounty but suggested the bounty backers transferring the bounty to other issue reports or to the project as a kind of funding.

5.7 Threats to Validity

In this section, we discuss the threats to the validity of our results.

Threats to **external validity** are related to the generalizability of our findings. We studied only bounty issue reports from GitHub and Bountysource. Future research should study issue reports from other bounty platforms, issue tracking systems and open source projects to determine whether our findings are generalizable to other types of issue reports (e.g., from commercial platforms), other bounty platforms and projects. Although our models have a high explanatory power, there might be additional factors that relate to the likelihood of an issue being addressed. Future studies should investigate more factors.

Threats to **internal validity** relate to the experimenter bias and errors. One threat is that we rely on manual analysis to identify the addressed-unpaid issues and to identify why developers did not claim a bounty in Section 5.6.1, which may introduce bias due to human factors. To mitigate the threat of bias during the manual analysis, two of the authors conducted the manual analysis and discussed conflicts until a consensus was reached. We used Cohen's kappa (Gwet et al., 2002) to measure the inter-rater agreement and the value is 0.86, which indicates a high level of agreement.

There are many additional factors which may have an association with our model, e.g., the type of a project. Since there is no clearly defined project type for a project in GitHub, we would need to manually identify the project type (which would introduce a bias as well). Future studies should consider this factor if the type of a project can be clearly defined.

Another threat is that we regarded all open issue reports as failed ones, which may introduce bias, since some issue reports could be worked on by one or more hunters

at the time we collected our data. However, it is not possible to distinguish between bounties which are worked on or actual failed bounties, since it is not mandatory for a hunter to update their progress on an issue report. To alleviate this threat, we updated the status of our studied issue reports after 200 days since the first time of our data collection. In other words, only the issue reports that remain unsolved for more than 200 days are regarded as failed ones in this study.

Threats to **construct validity** concern the relation between theory and observation. One threat relates to the project categorization in Section **??**, in which we used 50 bounty issue reports as a threshold to distinguish whether a project uses bounties moderately or frequently. To alleviate this threat, we redid the analysis of Section **??** with other thresholds for bounty-usage frequency (i.e., 40 and 60). The results show that our findings still hold (see Appendix C in our supplementary material (Zhou, 2018) for more details).

Threats to **conclusion validity** concern the relation between the treatment and the outcome. One threat is caused by the statistical tests that we performed. To alleviate the threat, we used non-parametric tests that do not make an assumption about the underlying data distribution. Another threat is that there may exist confounding factors that bias our conclusion. To alleviate this threat, we constructed multi-factor models to control for confounding factors.

5.8 Related Work

In this section, we discuss related work in the research area of improving the issue-addressing process.

5.8.1 Improving the issue-addressing process

Issue addressing is an essential activity in the life cycle of software development and maintenance. Therefore, a large amount of research was done to improve the issue-addressing process. One group of studies focused on providing insights into improving the issue-addressing process in aspects of the quality of issue reports, the effectiveness of developers and automated bug localization and fixing. For example, Bettenburg et al. (2008) and Hooimeijer and Weimer (2007) analyzed the quality of bug reports (i.e., a type of issue report) and provided some guidelines for users to generate high-quality reports so that developers can address issues more efficiently. Ortu et al. (2015) analyzed the relation between sentiment, emotions, and politeness of developers in comments with the needed time to address an issue. They found that the happier developers are, the shorter the issue-addressing time is likely to be. Zhong and Su (2015) performed an empirical study on real-world bug fixes to provide insights and guidelines for improving the state-of-the-art of automated program repair. Soto et al. (2016) performed a large-scale study of bug-fixing commits in Java projects and provided insights for high-quality automated software repair to target Java code. A number of studies helped developers locate the buggy code in projects using information retrieval techniques (Zhou et al., 2012; Saha et al., 2013; Wang and Lo, 2016; Wang et al., 2014a; Wang and Lo, 2014; Wang et al., 2011).

There is not much research to study the how to leverage bounties on improving the issue-addressing process. The work of Kanda et al. (2017) is closest to ours. They studied GitHub and Bountysource data but studied only 31 projects (compared to 1,203 in our study). They compared the closed-rate and closing-time between bounty issue and non-bounty issue reports. Their results showed that the closing-rate of bounty

issue reports is lower than that of non-bounty issue reports, and it takes longer for the bounty issue reports to get closed than non-bounty issue reports.

Different from prior studies, we perform an empirical study to understand the relationship between bounties and the issue-addressing process. We provide insights into how to better use bounties to improve the efficacy of the issue-addressing process.

5.9 Chapter Summary

In this chapter, we studied 5,445 bounties with a total value of $406,425 from Bountysource along with their associated 3,509 issue reports from GitHub to study the relationship between the bounty (e.g., timing of proposing a bounty, bounty value, and bounty-usage frequency) and the issue-addressing likelihood. We found that:

1. The timing of proposing bounties is the most important factor that is related to the issue-addressing likelihood. Issue reports are more likely to be addressed with a faster addressing-speed if bounties are proposed earlier.

2. In first-timer bounty-projects, the issue-addressing likelihood is higher for higher bounty values and in these projects, backers should consider proposing a relatively bigger bounty.

3. Backers should be cautious when proposing small bounties on long-standing issue reports as they risk losing money without getting their issue addressed.

Our findings suggest that backers should consider proposing a bounty early and be cautious when proposing small bounties on long-standing issue reports. Bounty platforms should allow dividing bounties between hunters, and transferring bounties to other issue reports.

CHAPTER 6

Studying monetary donations in GitHub open source projects

Operating an open source project requires not only intrinsic motivation (e.g., the joy of participation) but also extrinsic motivation (e.g., financial incentives). Almost 95% of open source projects are no longer maintained after a year. Nowadays, although donations start to play an important role in operating open source projects, there is little knowledge about the characteristics of donors and the usage of donations. A better understanding of the characteristics of donations, their donors, and the usage of donations in open source projects is needed to provide insights to the stakeholders of open source projects to help them operate their projects more sustainably. In this paper, we study the donations that are received through the Open Collective platform (i.e., an online crowdfunding platform) to support open source projects, to understand the characteristics of these donations, their donors, and the usage of these donations. To do so, we investigated 225 GitHub open source projects that received 54,889 donations with a total value of $2,537,281 through the Open Collective platform. We find that: 1) In general, corporate donors tend to donate more money than individual donors in a single donation. However, in a collective, the total donation amount from individual donors is more than corporate donors, suggesting the importance of individual donors. Moreover, individual donors are more likely to redonate to the same collective compared to corporate donors. 2) Non-engineering-related expenses take up to 54.0% of the total number of all expenses that are filtered against donation. For instance, "Web services", "marketing", and "travel" expenses are

the three most frequent and costly non-engineering-related expense types. For engineering-related expenses, the most frequent expenses are related to development and maintenance. Interestingly, we also observed that 18% of the engineering expenses were spent to propose bounties for addressing issues with a median cost of $95 per proposed bounty. We further analyze the differences between individual-supported collectives (i.e., collectives where more than 80% of their donation amount is from individual donors) and corporate-supported collectives (i.e., collectives where more than 80% of their donation amount is from corporate donors). We observed that corporate-supported collectives tend to receive a higher donation amount than individual-supported collectives. They have no significant difference in terms of popularity (e.g., the number of pull requests) of their associated GitHub projects.

6.1 Introduction

OPEN source projects are widely used by many companies, government agencies, and individuals. A prior study (Androutsellis-Theotokis et al., 2011) shows that 65% out of 1,313 surveyed companies relied on open source projects to expedite application development. However, operating open source projects is a challenging task. Operating open source projects (e.g., fixing bugs and maintaining documentation) requires a significant amount of effort from developers. However, "Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his project and distribute for free?", as Bill Gates once noted.[1] In other words, we cannot expect all developers to be willing to volunteer for maintenance tasks (Steinmacher et al., 2018). As a result, 64% of well-known and popular open source projects rely on one or two contributors to manage most of their tasks (Avelino et al., 2016) and almost 95% of open source projects are no longer maintained after a year (Rich Sands, 2012).

Financial incentives are an important extrinsic motivator for developers to sustain open source projects (Atiq and Tripathi, 2016). More and more individuals and corporations make donations to open source projects. For instance, the Linux Foundation plans to provide 10 million dollars to support open source projects on Community Bridge.[2] More than 6 million dollars donations have been made through the *Open Collective* platform,[3] an online platform that hosts the funding for more than 500 open source projects.

[1] https://genius.com/Bill-gates-an-open-letter-to-hobbyists-annotated

[2] https://www.linuxfoundation.org/press-release/2019/03/the-linux-foundation-launches-new-communitybridge-platform-to-help-sustain-open-source-communities/

[3] https://opencollective.com/

Nowadays, donations play an important role in the smooth operation of open source projects.[4] However, how donors make donations to open source projects and how the received donations are spent have not been examined in depth. With a better understanding of such questions, we can provide insights to the stakeholders of open source projects to help them operate their projects more sustainably. For instance, the stakeholders of an open source project can have a better estimation of the donations that they probably could receive and the potential donors. It also can help stakeholders estimate their budgets for operating an open source project more sensibly.

In this paper, we study 225 GitHub open source projects that set up collectives on the Open Collective platform for collecting donations. These collectives received 54,889 donations from 7,071 individual and 877 corporation donors, with a total value of $2,537,281. 84.6% (i.e., $2,192,439) of the received donations have been spent on various operational activities (e.g., development and maintenance). We examine how donors make donations and how the received donations are spent to sustain open source projects. We found that:

1. In general, corporate donors tend to donate more money (with a median value of $25) than individual donors (with a median value of $5) in one donation. However, in a collective, the total donation amount from individual donors ($833 in median) is more than corporate donors ($550 in median), suggesting the importance of individual donors. Moreover, individual donors are more likely to redonate to the same collective compared to corporate donors.

[4] https://opensource.guide/getting-paid/

2. Non-engineering-related expenses take up to 54.0% of the total number of all expenses. “Web services”, “marketing”, and “travel” expenses are the three most frequent and costly non-engineering-related expense types. For engineering-related expenses, the most frequent expenses are related to development and maintenance. Interestingly, we also observed that 18% of the engineering expenses were spent to propose bounties for addressing issues with a median cost of $95 per proposed bounty.

We further analyze the differences between individual-supported collectives (i.e., collectives where more than 80% of their received donation amount is from individual donors) and corporate-supported collectives (i.e., collectives where more than 80% of their received donation amount is from corporate donors). We observed that:

1. Corporate-supported collectives tend to receive a higher monthly and total donation amount than individual-supported collectives. There is no significant difference between corporate-supported and individual-supported collectives in terms of the popularity of their associated GitHub projects.

2. Both Individual/corporate-supported collectives are likely to spend donations on a small group of specific types of expenses (e.g., engineering and web services).

Our findings suggest that the stakeholders of GitHub open source projects should try to attract more individual donors since they will donate more money overall and more steadily than corporate donors. Collectives should not expect to receive a large amount of funds overall from donations (e.g., over $10,000) unless their projects are

very popular (e.g., more than 9,000 issue reports) and are mainly supported by corporations. Projects should budget for a reasonable amount (e.g., 13% of total funds) of non-engineering expenses (e.g., marketing and traveling).

[Jiayuan syas:Removed organization]

6.2 Background

In this section, we briefly introduce the donation platform, Open Collective.

Open Collective

Open Collective,[5] *Patron*,[6] and *Salt*[7] platforms are examples of online platforms for crowdfunding to support the operation of open source projects. The Open Collective platform is one of the most popular platforms. The platform facilitates a transparent mechanism for managing donations tracking their usage (i.e., expenses), which enables each donation and expense transaction transparent. In other words, it enables us to collect various information about donations, e.g., the corresponding donors and the usage of these donations. Hence, we study the donations that are collected on the Open Collective platform.

We provide below a brief background of the Open Collective platform along the following three dimensions: collective, donor, and how the platform works.

Collective: A collective is a group of people sharing the same mission to collect donations. People can set up their collectives on the Open Collective platform which is free of charge. There are many types of collectives according to their missions, such

[5] https://www.opencollective.com
[6] https://www.patreon.com
[7] https://salt.bountysource.com

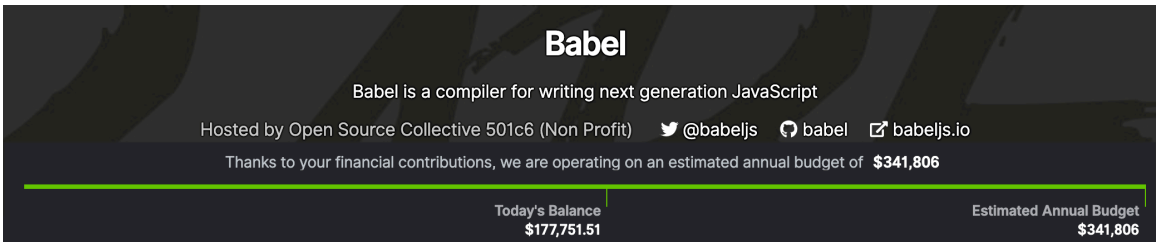


Figure 6.1: An example of the collective of an open source project.

as ones for supporting open source projects, meetups, and non-profits organizations. For example, the *WWCode Toronto* collective[8] is to support a non-profit organization aiming at inspiring women to succeed in technology careers, and the *Babel* collective[9] is created to support the *Babel* open source project. Collectives that are associated with open source projects can choose to add their associated GitHub repository link to the official homepage of their Open Collective (see Figure 6.1). We focus on collectives that are associated with open source projects. We introduce how we identify such collectives in Section 6.3.

Members of a collective can submit their expenses to a collective for their contributions or for reimbursements. When an expense is submitted, the expense will be labeled with a specific expense type to represent the main purpose of the expense. Unfortunately there is no uniform definition for expense types across collectives. Hence, we manually relabeled the expense types and Table 6.1 shows the definition of the relabeled 11 expense types along with corresponding examples. We elaborate on our relabeling process in Section 6.4.2. After an expense is submitted, administrators of the collective receive a notification and they need to decide whether to approve or reject the expense.

[8] https://opencollective.com/wwcodetoronto
[9] https://opencollective.com/babel

Table 6.1: The different types of expenses along with corresponding examples.

Expense Type	**Explanation of expenses**	**Examples from actual expense descriptions**
Engineering	Implementing new features, addressing reported bugs, and maintenance related costs.	"App development in October", "Community maintenance", and "April 2018: Documentation updates".
Web services	Web hosting and SaaS (i.e., Software as a service) subscription costs.	"GoDaddy domain name cost", "Heroku Hosting costs for July 2015", and "Canny.io annual subscription".
Design	Website design costs.	"Roots Website Redesign (Concept, Colors, First Designs)", "Open Source Design Rollup-Banner", and "Fiverr - Convert PSD design to HTML/JS BS4".
Donation	Donating to other collectives.	"Donation to the Python Software Foundation", and "Donation expanse for obfuscator.io domain".
Food & Beverage	Food and drink expenses for meetings or events.	"Food for the team meeting in Amsterdam", "Pizza for PDXNode Hack Night".
Legal	Bookkeeping, accounting, and brand registration costs.	"Brand registration", "Watson & Associates - Bookkeeping - March 2018", "Watson & Associates - Quarterly Accounting - January 2018".
Marketing	Advertisement and related costs (e.g., stickers, business cards) for attracting more users.	"New Logo Design", "Stickers for the conference", and "Printing signage and business cards".
Travel	Meeting and attending events (e.g., conferences) related costs (e.g., transportation, accommodation).	"Train for Vue.js conference", "Conference travel reimbursements for Q4 2018", and "Airbnb for Vue Sprint in Poland".
Team	Expenses for team activities (e.g., team t-shirts and video games).	"Core team T-shirts", "Mailing custom t-shirts to contributors", and "destiny 2 digital deluxe edition".
Supplies & Materials & Office	Office supplies and equipment costs.	"Mac USB hub", "Postage and Envelopes", and "Raspberry Pi Zero W + Case".
Other	Others expenses.	"Emergency expenses", "Portuguese translations from Urb-i", and "Transfering collective to EU host".

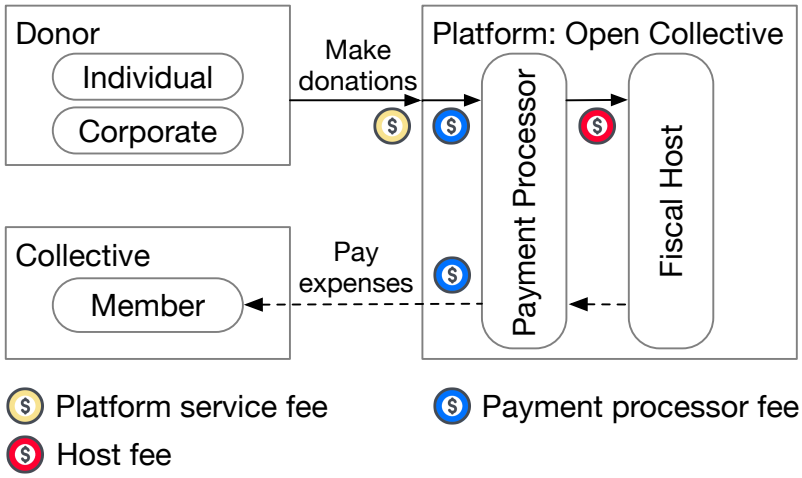


Figure 6.2: The transaction flow for donating or paying an expense on the Open Collective platform.

Donor: There are two types of donors: *individual* and *corporate* donors[10]. Donors can choose to donate one-off, monthly, or yearly. Donations can be made through a credit card, a gift card, or the balance of their collectives or organizations. Since October 06, 2017, donors can attach a brief message to their donations to explain the rationale for their donations.

The workflow of the Open Collective platform: The Open Collective platform plays the role of an agency between collectives and donors. The platform provides a payment processing service through several payment processors, such as *Stripe*[11] and *Paypal*,[12] so that donors can make donations conveniently. The platform also provides a fiscal sponsorship service by connecting several fiscal hosts, which help collectives to store their funds, generate invoices, and pay expenses, so that collectives do not need to create their own legal entity and bank account.

[10] https://docs.opencollective.com/help/about/terminology
[11] https://www.stripe.com
[12] https://www.paypal.com

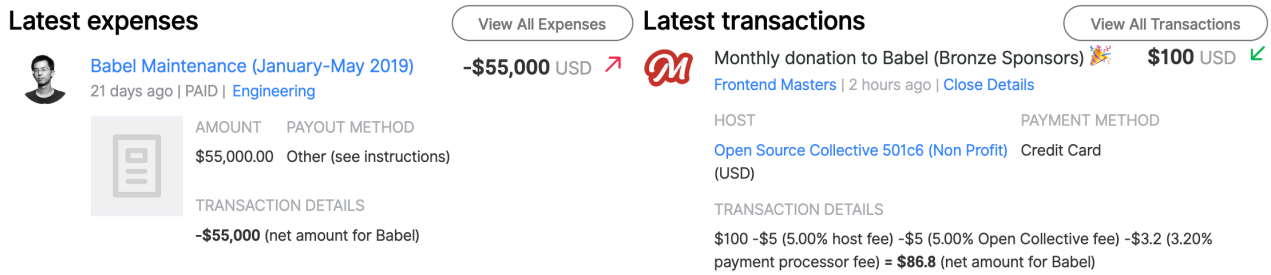


Figure 6.3: An example of transaction record from the *Babel* collective.

Figure 6.2 shows the transaction flows of donations and expenses. The colorful coins refer to the different types of fees that are charged for different transactions. When a donor makes one donation, the platform will charge a 5% service fee (i.e., the yellow coin) then the funds will go through a payment processor to a fiscal host. A payment processor will charge a payment processing fee (i.e., the blue coin), which is around 3%. After that, the fiscal host will host the funds and charge a hosting fee (i.e., the red coin). The hosting fee varies across fiscal hosts. For example, the *Open Collective Foundation* fiscal can host US-based charity projects and the fee is 5% for each donation.[13] When a submitted expense is approved, the expense will be paid from the platform to the expense submitter. During this transaction, only the payment processing fee will be charged. In general, the total service fee for a donation or an expense in 8% to 13% of the total amount of transaction.[14] For example, Figure 6.3 shows an expense transaction and a donation transaction along with their corresponding fees in detail.

The platform supports six currencies (e.g., USD, CAD, and EUR)[15] and all transactions (i.e., donations and expenses) within the platform are visible to the public.

[13] https://docs.opencollective.com/help/hosts
[14] https://docs.opencollective.com/help/collectives#how-much-does-it-cost
[15] https://docs.opencollective.com/help/product/currencies

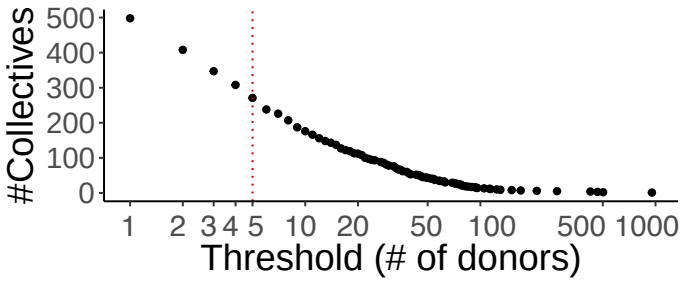


Figure 6.4: The number of collectives under different thresholds of the number of donors.

6.3 Data Collection

The Open Collective platform publishes its dataset[16] in the CSV format. The dataset consists of 804 collectives and their donation and expense records ranging from Jan. 23, 2015 to Jan. 31, 2019. The dataset also includes the donation messages from Oct. 06, 2017, to Apr. 12, 2019.

Because we wish to study donations and expenses that are for open source projects, we only focus on the collectives that are associated with open source projects. Hence, we filter out collectives, such as *WWCode Toronto*, that are not associated with open source projects. We automatically extracted 418 collectives that have GitHub repository addresses in the descriptions. For the rest 386 collectives that do not provide GitHub repository addresses in their descriptions, we checked for their GitHub repository by manually searching the associated projects through Google. If a GitHub repository exists for a collective, we collect the address for that collective. We found another 102 collectives that have GitHub repositories. In total, we collected 520 collectives which are associated with open source projects.

[16] http://drive.opencollective.com

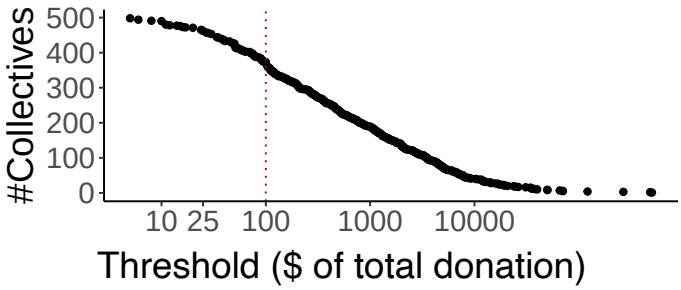


Figure 6.5: The number of collectives under different thresholds of donation amount.

We observe that some collectives receive a large total donation amount of donations from a small number of donors. We also observe that some collectives only received very few donations in total. For example, the *docker.io* collective only received one donation with an amount of $6. To reduce the bias that is introduced by collectives with a low donation frequency or a low donation amount in total, we use a threshold-based approach to further select proper collectives by following a prior study (Miura et al., 2016). We selected collectives in terms of the number of donors and the total donation amount for each collective. Figures 6.4 and 6.5 present the number of collectives against different number of donors and total donation amount of a collective. We selected collectives with more than five donors and more than $100 total donation amount. We ended up with 43.3% (225 out of 520) of the collected collectives. Our selection criteria ensure that the studied collectives are active ones which are used by open source projects for their operation.

Since donors can make donations in six different currencies, we convert all currencies into United States Dollar (USD) using the daily exchange rate provided by OANDA to better conduct our study in terms of donation and expense amounts.[17]

[17] https://docs.opencollective.com/help/product/currencies, https://www.oanda.com/fx-for-business/exchange-rates-api/daily-average-exchange-rates

Table 6.2: Dataset description.

Period	Nov. 23, 2015 to Jan. 31, 2019
Number of collectives	225
Number of expenses	2,213
Total amount of expenses	$2,192,439
Number of donations	54,889
Total amount of donations	$2,537,281
Number of donors	7,446
Number of donation messages	589

Finally, our studied dataset contains 225 collectives, 7,446 donors, 54,889 donations with $2,537,281 donation amount, and 2,213 expenses with a total value of $2,192,439. Table 6.2 gives an overview of our dataset.

6.4 Study Result

In this section, we present the result of our empirical study of donations in GitHub open source projects. We first study the characteristics of donors and their donations. Then, we study the characteristics of expenses. Finally, we study the differences between individual-supported collectives and corporate-supported collectives. *[Jiayuan syas:maybe add a conclusion, like, our findings can also do xxxx.] [Jiayuan syas:Summary the rq in a higher level?]*

6.4.1 RQ1: What are the characteristics of donors and their donations?

Motivation: As we introduced in Section 6.2, there are two types of donors – individual and corporate donors. A corporate donor represents a legal entity instead of an individual. Due to their different nature, these two types of donors may exhibit different donation characteristics. For example, a corporate donor may make donations more frequently with higher amounts than an individual donor. Additionally, it is interesting to know the characteristics of donations within a collective. For instance, what is the proportion of the donations that are contributed by these two types of donors, and whether donors across these two types of donors tend to redonate to the same collective. With a better understanding of the characteristics of donors and their donations, the stakeholders (e.g., operators) of a collective can have a better estimation of the donations that they would typically receive and the potential types of donors that they might be able or wish to attract.

Approach: First, we compare the donation amount in terms of different donation styles (i.e., one-off, monthly, or yearly donations), then we compare the amounts and frequencies of donations that are made by individual and corporate donors among all collectives. We employ the Wilcoxon rank-sum test (Bauer, 1972) to measure whether or not the differences between individual and corporate donors are statistically significant. We calculate Cliff's delta d effect size (Long et al., 2003) to quantify the magnitude of the differences of the amount and frequency of donations between the two types of donors.

To further evaluate the likelihood of a donor redonating to a collective, we employ the sticky metric from a prior study (Yamashita et al., 2016). The value of the sticky

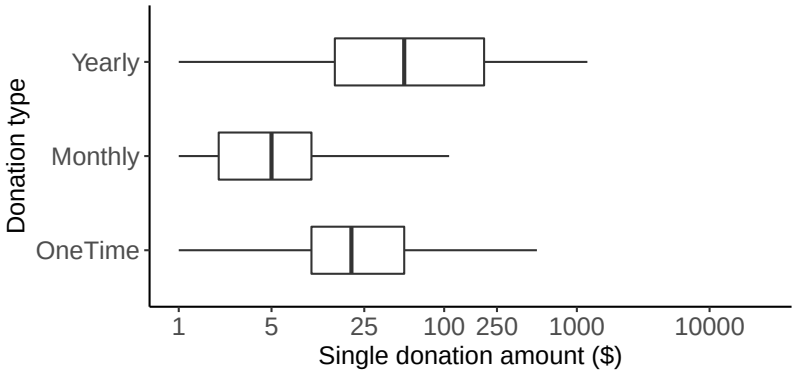


Figure 6.6: The boxplot of donation amounts for different donation styles. Note that if we distribute the yearly amount into each month, it comes to $4.2.

metric reflects the proportion of donors that donated in the prior period an redonate in the current period for the same collective. Similar to the prior study, we measure the retention of donors of a collective by calculating the proportion of donors that donated in the prior six months and still donate in the recent six months.

Then we study donors and their donations at the collective level. To do so, we first calculate the proportion of individual and corporate donors within each collective, and compare the proportion of individual and corporate donors across collectives. We also calculate the proportion of the total amount of the donations that were contributed by these two types of donors in each collective, and compare the distributions of donation amount proportion across collectives. We use the Wilcoxon rank-sum test and Cliff's delta d effect size to measure the significance and magnitude of distribution of these two types of donors.

Results: Donors tend to donate more money in a one-off style. Figure 6.6 shows the boxplot of donation amount for different donation styles. The median donation amount for "yearly", "monthly", and "one-off" is $50, $5 and $20, respectively. Note that if we distribute the yearly amount into each month, it comes to $4.2. We perform

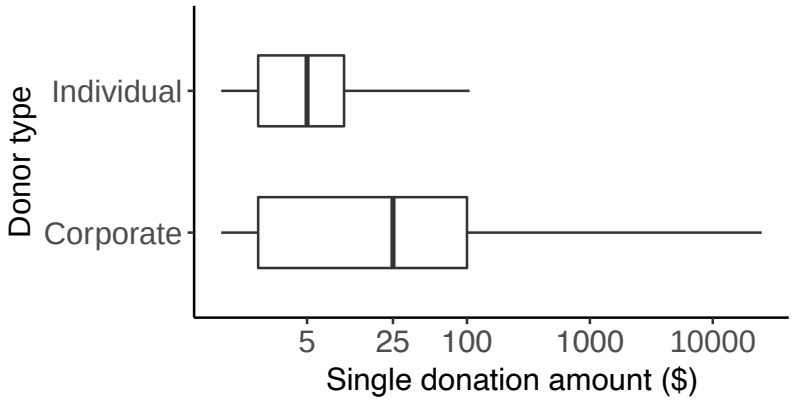


Figure 6.7: The distribution of donation amount for different donor types.

the Wilcoxon rank-sum test and Cliff's delta test to measure the differences between distributions of donation amount for "one-off" and "monthly" donation types. The result shows that two distributions are significantly different with a large Cliff's delta effect size, indicating that donors tend to donate more when they choose to make a one-off donation.

Corporate donors tend to donate significantly more money (a median of $25) in a single donation than individual donors (a median of $5). Figure 6.7 shows distributions of donation amount for individual donors and corporate donors. Corporate donors made donations with a median amount of $25, which is five times that of the individual donors' median amount. The Wilcoxon rank-sum test shows that there exists a significant difference between these two distributions with a medium Cliff's delta d effect size. The largest donation was made in Jan. 17, 2019, with an amount of $250,000 by the corporate donor *Modus Create*, which is a company aiming at digital transformation such as cloud migration. The median of the donation frequency for individual donors and corporate donors are both 3. There is no significant difference between the donation frequency distributions for individual and corporate donors .

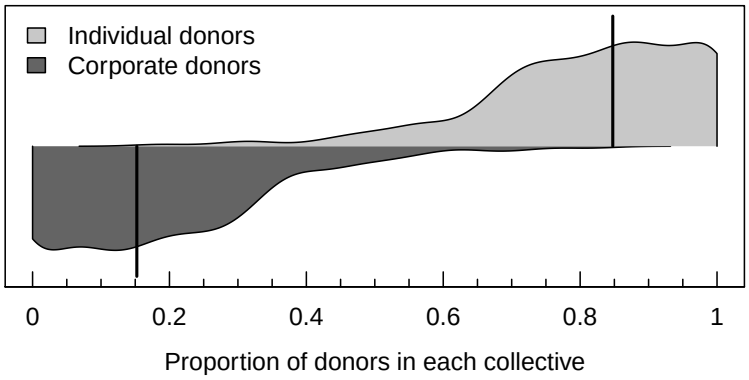


Figure 6.8: The distribution of the proportion for the individual and corporate donors across collectives.

There are significantly more individual donors (a median number of 14) than corporate donors (a median number of 3) in a collective. In general, the total donation amount from individual donors are significantly higher than that from corporate donors in a collective. Figure 6.8 shows the distributions of the proportion for individual and corporate donors in a collective. It is obvious that the number of individual donors is more than corporate donors in a collective. The median proportion of individual donors across all the studied collectives is 85% and that of corporate donors is 15%. The results of the statistical test show that individual donors are significantly more than corporate donors with a large effect size in one collective.

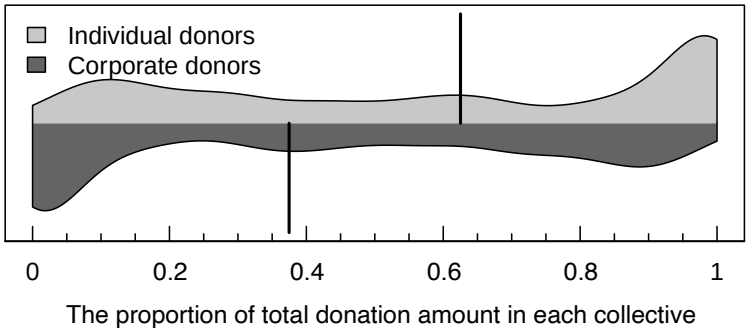


Figure 6.9: The distribution of the proportion of the donation amount from individual donors and corporate donors across collectives.

Figure 6.9 shows the distributions of the proportion of total donation amounts from individual and corporate donors in a collective. The median proportion from individual donors is 63% and that of a corporate donors is 33%. Across collectives, the median donation amount from individual and corporate donors are $833 and $550, respectively. Our observation highlights the *importance of individual donors*. Although the donation amount of an individual donor is less than a corporate donor, the total contribution from individual donors is significantly more than corporate donors. For example, 94.8% (3,917 out of 4,132) total donation amount in the *ImageSharp*[18] collective is from individual donors and the median donation amount of the individual donors is $5, which is smaller than the donation amount for corporate donors ($20).

Individual donors are more likely to continue to redonate to the same collective to which they previously donated compared to corporate donors. Figure 6.10 shows the distributions of the sticky value of individual donors and corporate donors.

[18] https://opencollective.com/imagesharp

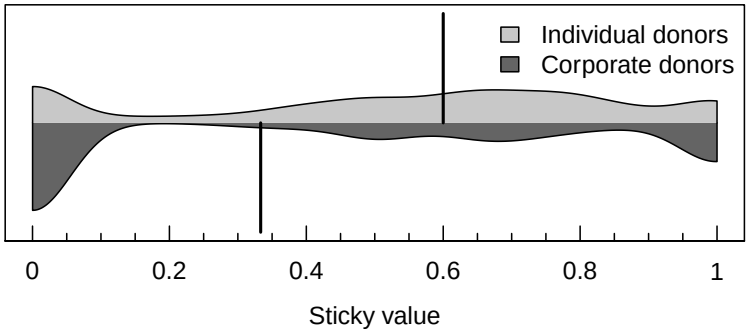


Figure 6.10: The distribution of sticky value for individual and corporate donors for each collective.

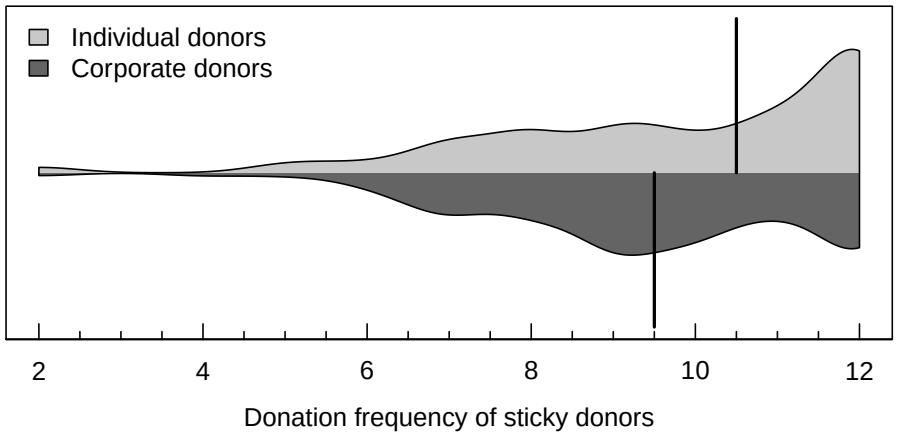


Figure 6.11: The distribution of donation frequency for individual and corporate donors who are sticky to a collective.

The median sticky value for individual donors is 0.60, while that of corporate donors is 0.33. The Wilcoxon rank-sum test shows that there exists a significant difference between these two distributions with a small Cliff's delta d effect size, indicating individual donors are more likely to redonate than corporate donors. In addition, to understand if such repeat donors are usually active in making donations, we examine the frequency of donations for such donors. Figure 6.11 shows the donation frequency of individual donors and corporate donors. In median, the repeat individual donors donate 10.5 times and repeat corporate donors donate 9.5 times, which is much more active than the average level of the donation frequency of donors, i.e., 4 and 5.2 for individual and corporate donors, respectively.

Summary: In general, corporate donors tend to donate larger amounts (with a median value of $25) than individual donors (with a median value of $5). However, in a collective, the total donation amount from individual donors ($833 in median) is more than corporate donors ($550 in median), which highlights the importance of individual donors (i.e., the value of working to attract more individual donors to one collective) Moreover, individual donors are more likely to redonate to a collective than corporate donors.

6.4.2 RQ2: What are the received donations spent on?

Motivation: Operating (i.e., developing and maintaining) open source projects encounters various types of expenses (e.g., development cost and website hosting cost). It is challenging to estimate the various expenses for operating an open source project.[19] Therefore, a study of the types of expenses for operating open source projects would

[19]https://thenewstack.io/survey-open-source-programs-are-a-best-practice-among-large-companies/

be of great value to the leader and stakeholders of open source projects. In this RQ, we first provide an overview of the types of expenses across collectives. Then we further analyze the non-engineering-related and engineering-related expenses, respectively. A better understanding of the cost of operating such projects can help open source project stakeholders estimate their budgets more sensibly.

6.4.3 Overview of engineering-related and non-engineering-related expenses

Approach

To provide an overview of the types of expenses across collectives, we first calculate the number of collectives that have expenses and no expense, respectively. Then we further compare the monthly expense amount across collectives between two families of expenses – non-engineering-related expenses and engineering-related expenses. We use a collective's median monthly expense amount to represent its monthly expense amount.

To identify the engineering-related and non-engineering-related expenses, we manually examine the expenses and the labels of each expense. These labels are provided by users when submitting expenses. We observed 15 expense labels (namely, the food, beverage, supplies and material, office, team, design, web services, engineering, marketing, communications, travel, donation, legal, fund, and other) by users from 139 collectives that have expenses. After examining the expenses and their labels, we observed some original expenses are labeled inaccurately due to the following three reasons:

1. **Expenses have different labels, while their purposes are similar.** For example, the description of an "office" type expense is "mac usb hub", and the description of a "supplies and materials" type expenses is "hardware renewal". However, the purposes of both expenses are the same, buying supplies and materials. Therefore, for consistency, these two expenses should be tagged with the same label.

2. **Expenses have the same label, while purposes are different.** For example, the descriptions of three "communication" type expenses are "travel to Poland sprint" (i.e., onsite meeting), "MailChimp email service" (i.e., the web communication service), and "community maintenance" (e.g., triage issues). However, "travel to Poland sprint" is for traveling, "MailChimp email service" is for SaaS, a type of web service, and "community maintenance" relates to maintenance. For consistency, these three types of expense labels should be labeled with three labels.

3. **Expenses have the wrong labels.** We also observed that some expenses were labeled with wrong labels. For example, the expense with a description "project maintenance and enhancement" should be labeled with "engineering" rather than "design".

In order to reduce the bias from these inaccurate labels when analyzing different types of expenses, the first and the third authors manually relabeled all expenses using the existing original expense types provided by users. Note that we merged type "office" type into type "supplies and materials" since they share a similar purpose. For the "communication" type expenses, we split and merged them into three other expense types, respectively. Then we removed the "communication" expense type. The expenses that were for the onsite meeting (e.g., "Berlin Meetup Organizer Costs") were

merged into "travel" type expenses, the expenses that related to web communication services were merged into "Web services" type expenses, and the expenses that were for maintenance were merged into "engineering" type expenses. Finally, we ended up with 12 expense types. Table 6.1 shows the explanations and examples for these expense types. Cohen's Kappa is 0.91, which indicates a high level of inter-rater agreement.

We consider all expenses of type "Engineering" as engineering-related expenses and the rest of types of expenses as the non-engineering-related expenses.

Results:

Overall, 38.2% (86 out of 225) collectives have no expenses. The possible explanation is that such collectives do not receive enough donations to pay the expenses. Figure 6.12 shows the distributions of the total received donation amount for the collectives that have expenses and ones without expenses. The median amount of the received donations for the collectives without expenses and collectives with expenses are $824.5 and $3,504, respectively. The Wilcoxon rank-sum test shows that the difference between these two distributions is statistically significant and non-negligible (i.e., Cliff's delta d is medium).

Non-engineering-related expenses occur more frequently than engineering-related expenses. However, the amount of engineering-related expenses are higher than non-engineering-related expenses. 75.0% (104 of 139) of the collectives with expenses have non-engineering-related expenses, which is higher than the proportion of collectives that have engineering-related expenses (55.4%). Such non-engineering-related expenses take up 45.6% (i.e., 665 out of 1,459) of all expenses. In terms of the

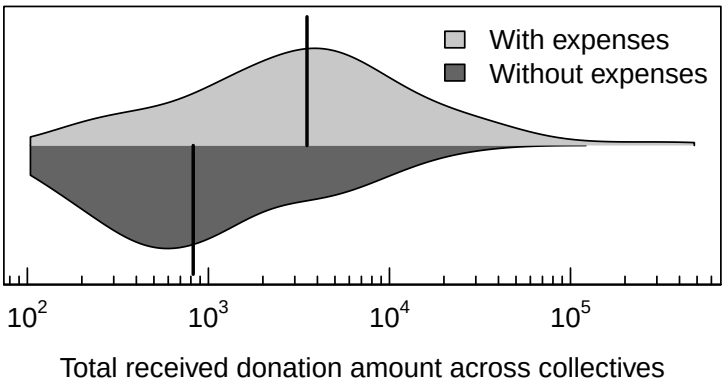


Figure 6.12: The distribution of total received expense amount for the collectives with expenses and the ones without expenses.

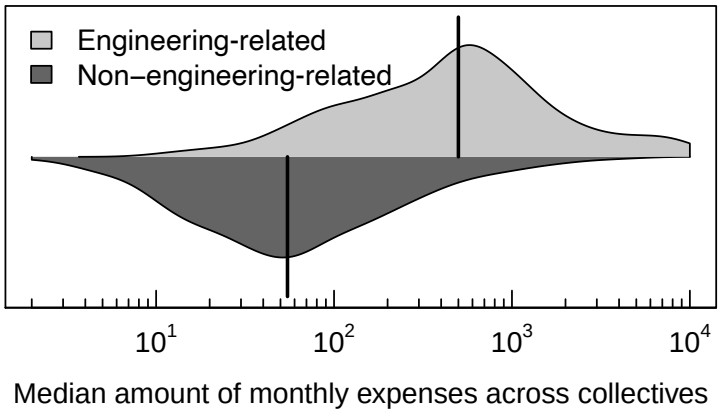


Figure 6.13: The distribution of collectives' median monthly expense amount that were used for engineering-related versus non-engineering-related expenses.

expense amount across all collectives, 87.0% (median) of the total expense amount is spent on engineering-related expenses and 13.0% (median) of the total expense amount is spent on non-engineering-related expenses. Figure 6.13 shows the distribution of median monthly expense amount of a collective for engineering-related and non-engineering-related expenses across collectives. The median amount of engineering-related expenses is $500 and that of non-engineering-related expenses is $54.75. The Wilcoxon rank-sum test shows that the difference between these two distributions is statistically significant with a large Cliff's delta d effect size, indicating that collectives spent significantly more funds on engineering-related expenses than non-engineering-related expenses.

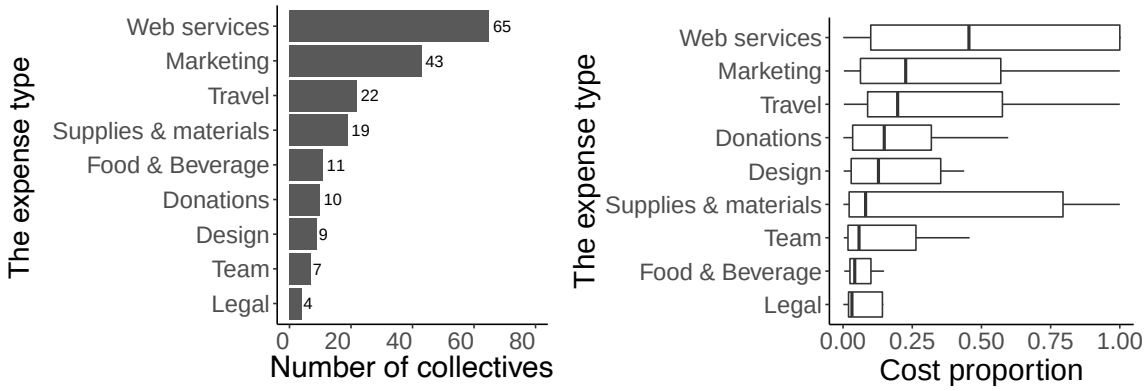
We perform further analysis on the engineering-related and non-engineering-related expenses and elaborate on the results in the following sections.

6.4.4 Non-engineering-related expenses

Approach

To study how collectives spent money on non-engineering-related expenses, we first analyze how frequently a non-engineering-related expense type is spent across collectives. For each non-engineering-related expense type, we calculate the number of collectives that have ever spent money on this type. Then for each collective, we calculate the proportion of the amount of each type of non-engineering-related expenses.

To further understand the purpose of non-engineering-related expenses and how widely such purpose is applied across collectives, we calculate the frequency of words appearing in expense descriptions. More specifically, we count a word only once even if it appears more than one time in the description of a collective.



(a) (b)

Figure 6.14: (a) The number of collectives that have each non-engineering-related expense types. (b) The distribution of cost proportion of each non-engineering-related expense types across collectives.

We perform preprocessing on the raw description of expenses before analyzing them. We perform tokenization, stemming, and stop word removal on the raw description of each expense. We use the **tokenizers**[20] *R* package for tokenization and stemming. To remove stop words, we not only remove the stop words listed in the **stopwords**[21] *R* package, but also consider the collective names as stop words and remove them. To reduce any potential bias due to the synonym words, we also replace synonyms or short forms manually. For instance, we replace "development engineering" with "development" and "bounty program" with "bounty".

[20]https://cran.r-project.org/web/packages/tokenizers/index.html
[21]https://cran.r-project.org/web/packages/stopwords/index.html

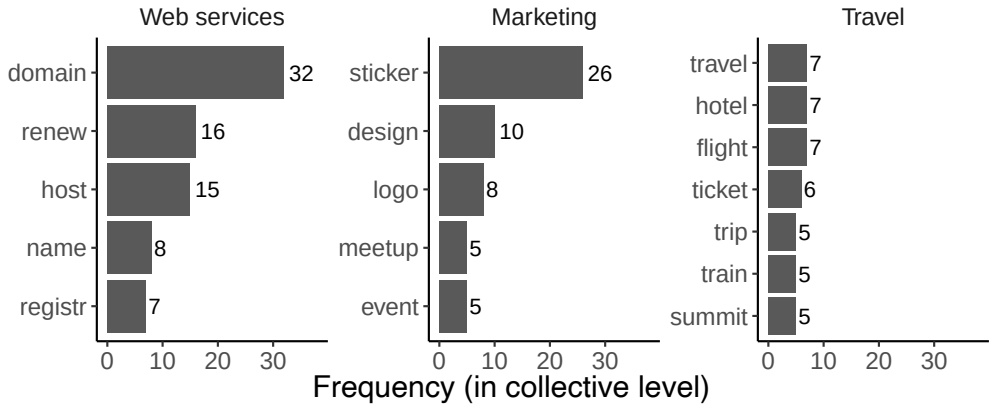


Figure 6.15: The frequency of the top five frequent keywords for "web services", "marketing", and "travel" expenses, respectively, at the collective level (i.e., the frequency of a keyword is counted once for each collective).

Results

"Web services", "marketing", and "travel" are the three most frequent and costly expense types among the nine non-engineering-related expense types. Different collectives use their received donations for different purposes. For example, the *Storybook* collective used all received donations on marketing. Figure 6.14a shows that "web services", "marketing", "travel", and "supplies & materials" are the four most widely used expense types. Especially, 29% (65) collectives have "web services" expenses and 19% (43) collectives have "marketing" expenses. Figure 6.14b shows the distribution of cost proportions of non-engineering-related types of expenses across collectives. We observe that 'web services", "marketing", and "travel" still are the top three non-engineering-related expense types with the highest median cost proportion. However, the median cost proportion of "supplies & materials" is low (i.e. 5%). Overall, "web services", "marketing", and "travel" are the three most common and costly non-engineering-related expense types.

Figure 6.15 shows the collective-level frequency of the top five frequent words for "web services", "marketing", and "travel" expenses, respectively. From the figure, we observe some possible purposes for "web service" expenses such as the registration or renewal of domain name, service, server hosting, or digital license renewal. Possible purposes for "marketing" expenses were the making stickers (from stickermule.com), designing logos, or hosting onsite events (e.g., meetups). Interestingly, **47% of marketing expenses are for making stickers.** Possible purposes for "travel" expenses were for developer on offsite activities (e.g., conferences and summits), transportation fees (e.g., flight, train, and lyft), and accommodation fees (e.g., hotel).

6.4.5 Engineering-related expenses

Approach

To study the purpose of engineering-related expenses, we applied the same text preprocessing process on the expense descriptions. We calculate the word frequency for the words of expense descriptions. We assume that such frequent words reflect the purposes of engineering expenses. Table 6.3 presents the top 10 most frequent words (in stemmed form) appearing in the engineering-related expenses. We notice that except for the stemmed words "support" and "contribut" (marked with *) that are vague, the other eight stemmed words represent a software engineering task. Note that we consider an expense to involve a software engineering task if the expense description contains task-related words. For example, if an expense description contains the frequent words "development" and "documentation", we consider this expense to involve two tasks that are related to development and documentation.

Table 6.3: The top 10 most frequent stemmed words and a corresponding example of expense for each of these words. The stemmed words which are marked with a '*' are not that specific, however, the other eight stemmed words can be mapped to software engineering tasks.

Freq. (%)	**Stemmed word**	**Example**
268 (40.3%)	develop	"App development in October".
97 (14.6%)	mainten	"Community Maintenance".
72 (10.8%)	contribut*	"Contribution to webpack".
68 (10.2%)	bounti	"Bug Bounty claim $100".
55 (8.1%)	issu	"Work on PR #805 (issue #787)".
51 (7.7%)	document	"Documentation on webpack".
40 (5.8%)	support*	"General Support".
33 (5.0%)	communic	"Development and Communication"
24 (3.6%)	releas	"v0.19.0 Release"
16 (2.3%)	test	"JHipster VueJS - Add entity client unit tests"

In some cases, one expense could be associated with several purposes (i.e., one expense description may contain more than one of the frequent words in Table 6.3) and we do not know the cost portion for each purpose. For example, we cannot estimate the cost of maintenance and development cost for a $3,500 expense, with the description that says "Maintenance & Development 10/2017". Hence, we focus on expenses that have only one single purpose (i.e., only contain one of the frequent words that are listed in Table 6.3) when analyzing the cost of a specific purpose of an expense. For example, for a $100 expense, with the description that says "Webpack development", we consider the development cost to be $100. After filtering out the expenses with more than one purpose, we ended up with 349 expenses. According to the most frequent words in Table 6.3, we estimate the cost of the eight software engineering tasks that are mentioned in these 349 expenses.

Table 6.4: The frequency and five-number summary of the expense amount for the most frequent eight software engineering tasks. Note that we cannot estimate the expense amount for the "communication" task since it is always mentioned with other tasks in the same expense description.

Purpose (Stemmed word)	**Freq.**	**Quantile value of corresponding expense amount**					
		Min.	**1^{st} Qu.**	**Median**	**Mean**	**3^{rd} Qu.**	**Max.**
Development (develop)	147	10	300	650	1,352	1,470	10,000
Bounty (bouti)	67	10	100	100	107	100	200
Maintenance (mainten)	48	50	178	500	1,654	1,400	9,000
Issue (issu)	34	20	293	465	493	690	1230
Documentation (document)	25	50	398	975	771	1,063	1,318
Release (releas)	15	50	100	120	289	375	1,000
Testing (test)	13	50	90	180	339	600	1,180
Communication (communic)	0	0	0	0	0	0	0

Results:

40% (268) engineering-related expenses involved development tasks (i.e., expenses that mention the word "develop") and the median cost of such expenses is $650. Table 6.3 shows the development tasks that are involved in most of the expenses (40%). Development tasks also co-occur with other software engineering tasks in the same expense. For the 268 engineering expenses that involved development, 121 of them mention other software engineering-related tasks (e.g., maintenance and documentation) as well. For example, the description of an expense says: "development and docs update in Nov". Table 6.4 shows that, in the expenses that are for development, the median expense amount is $650, which is the second-highest compared with that of the other tasks abovementioned.

18% of engineering expenses are due to a bounty or a specific issue (see Table 6.3). Both of the two stemmed words "bounti" and "issu" are related to the task of addressing issues. For the bounty expenses, collective maintainers first propose bounties on some issue reports of their GitHub projects, to motivate developers to address these

issues (Zhou et al., 2020b). After a developer addresses the issues, they submit an expense to claim the associated bounty. For issue expenses, developers first address the issue, then submit an expense to claim compensation for their effort.

The median cost for addressing one bounty issue is between $95 and $100, while for some specific issues, the cost can be as high as $930. There are 10 collectives with expenses that are related to bounties. Table 6.4 shows that the median cost of a bounty expense is $100 and the median cost for an issue expense is $465. Every bounty expense is proposed for addressing one issue, while an issue expense may represent the cost of addressing several issues. For example, the description of an issue expense says: "worked on issues #524, #549, #564, #558, #556". Hence, we manually extracted 85 issues by using the identified issue id (e.g., #54) from 24 (out of 34) issue expenses. And we estimated the expense amount for each specific issue by the amount of the expense divided by the number of issue ids in the expense. The calculated median issue expense amount is $95, which is close to the median bounty expense amount.

14% (97) of the engineering expenses involve maintenance tasks and the median cost of such expenses is $500. Table 6.3 shows that maintenance is the second most frequently mentioned word among engineering expenses. Table 6.4 shows that 48 engineering expenses are only for maintenance tasks and the median expense amount is $500. Documentation is a maintenance-related task. Table 6.4 shows that the median cost of documentation-related expenses is the highest at $975.

Summary: Non-engineering-related expenses represent 54.0% of all the expenses. "Web services", "marketing", and "travel" are the three most frequent and costly non-engineering-related expense types. For instance, 47% of marketing expenses are used for making stickers. For engineering-related expenses, the most frequent expenses are related to development and maintenance. Interestingly, we observe that 18% of the engineering expenses were spent to payout bounties for addressing issues with a median cost of $95.

6.4.6 RQ3: What are the differences between individual-supported collectives and corporate-supported collectives?

Motivation: In Section 6.4.1, we observed different characteristics between individual and corporate donors. In this section, we investigate the differences between collectives that are mainly supported by individual donors (i.e., individual-supported collectives) and those that are mainly supported by corporate donors (i.e., corporate-supported collectives). For simplicity, we refer individual-supported and corporate-supported collectives as to *Ind_Collectives* and *Corp_Collectives*, respectively. For example, do Ind_Collectives and Corp_Collectives receive different donation amounts? Do they spend their funds differently? With a better understanding of the differences between Ind_Collectives and Corp_Collectives, the stakeholders of collectives could have a better expectation of their potential donors and expenses.

Approach: In order to categorize Ind_Collectives and Corp_Collectives, we consider the collectives in which more than 80% of their donation amount are from individual donors as Ind_Collectives and collectives in which more than 80% of their donation

amount are from corporate donors as Corp_Collectives. We do so to ensure the collectives that we selected are primarily supported by either individual donors or corporate donors.

We first study the differences between Ind_Collectives and Corp_Collectives in terms of the received total donation amount. Because different open source projects set up their collectives for receiving donations at different times and with different frequencies, we use the average received monthly donation amount (referred as to *monthly-donation-amount*) of a collective to represent its general received monthly donation amount. To determine whether the *monthly-donation-amount* between two types of collectives is statistically significant, we use the Wilcoxon rank-sum test and Cliff's delta d effect size.

Then we further compare these two groups of collectives in terms of the popularity of their associated projects on GitHub. For example, the more watches of an open source project has, the more users are interested in that project. We collected seven project-related metrics (namely, the number of issues, pull requests, watches, forks, contributors, stars, and commits) from GitHub to reflect the popularity of an open source project in GitHub.

Finally, we study the usage of different expense types for Ind_Collectives and Corp_Collectives. Similar to prior work (Hassan, 2009), we employ Shannon's entropy to quantify the usage of expenses for each collective in terms of the expense amount across different expense types. The expense entropy of a collective quantifies the distribution of the expenses across the different expense types in a collective. A low entropy value for a collective indicates that the collective spent most of its funds on a small number of expense types. For example, if a collective's entropy is zero, the

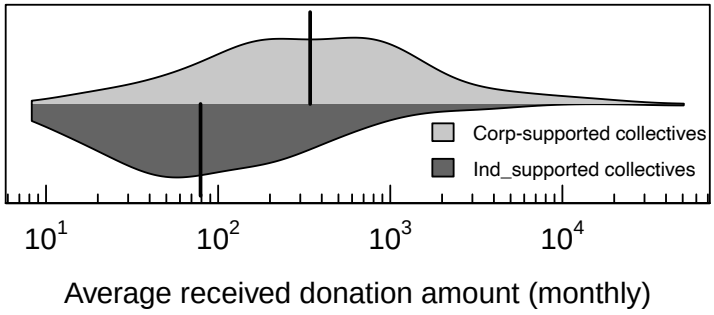


Figure 6.16: The distribution of monthly-donation-amount for Ind_Collectives and Corp_Collectives.

collective only spent funds on one specific expense type. A high expense entropy value for a collective indicates that the collective does not have a concentration for spending funds on specific expense types. For example, the expense entropy of a collective is one, indicating the collective spent funds on all occurrence expense types of the collective equally.

Results: The monthly-donation-amount and total donation amount of Corp_Collectives are significantly higher than those of Ind_Collectives. Figure 6.16 shows the distribution of the monthly-donation-amount for Ind_Collectives and Corp_Collectives. The median amount is $343 for Corp_Collectives, while $79 for Ind_Collectives. The Wilcoxon rank-sum test shows that there is a significant difference between them with a large Cliff's delta effect size, indicating the collectives driven by corporate donors received significantly more funds than the collectives driven by individual donors monthly. The median received donation amount of Ind_Collectives and Corp_Collectives are $5,094 and $1,406, respectively. The Wilcoxon rank-sum test

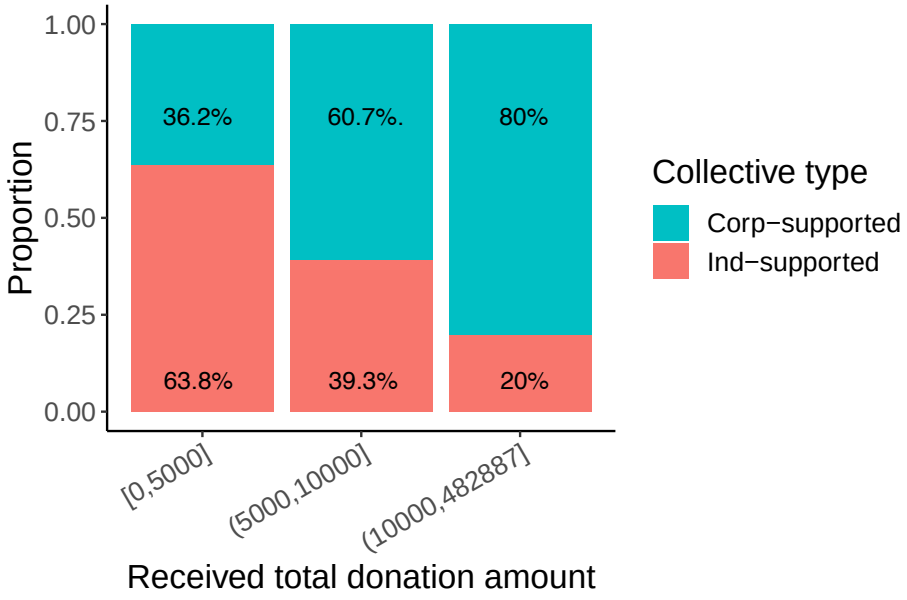


Figure 6.17: The proportion of Ind_Collectives and Corp_Collectives under different ranges of total received donation amount.

shows that distributions of the total received donation amount are significantly different between Ind_Collectives and Corp_Collectives with a medium Cliff's delta effect size. Figure 6.17 shows the proportion of Ind_Collectives and Corp_Collectives under different ranges of total received donation amount. We observed that there are more Corp_Collectives than Ind_Collectives in the range that have larger received total donation amount. For example, when looking at the collectives with a total donation amount larger than $10,000, 80% (24 out of 30) of them are Corp_Collectives. In other words, **Corp_Collectives are much more likely to get a large amount of donations (i.e., $10,000 compared with Ind_Collectives**).

There is no significant difference between Ind_Collectives and Corp_Collectives in terms of the popularity of their associated GitHub projects. The Wilcoxon-rank test shows that there is no significant difference between Ind_Collectives and

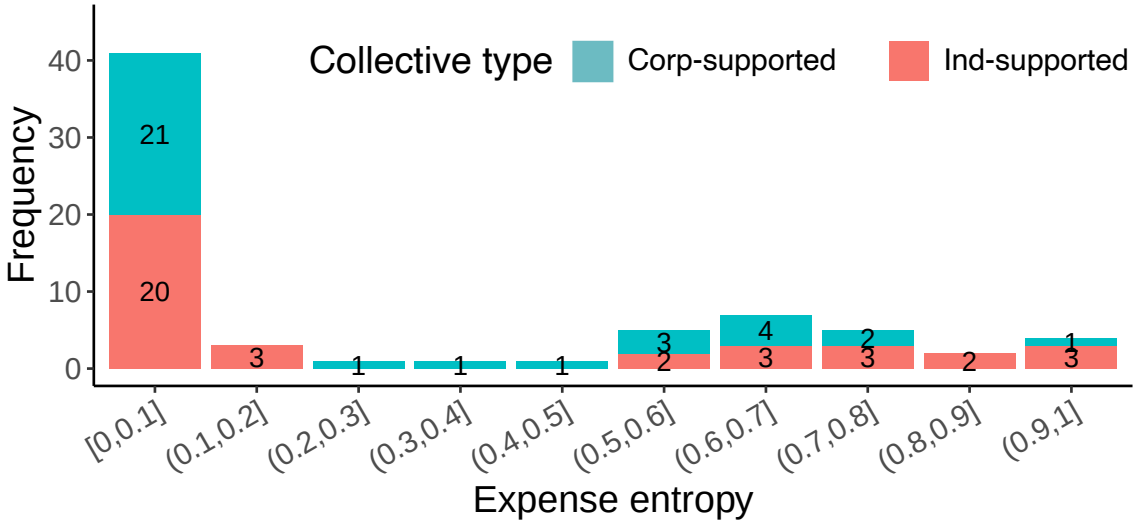


Figure 6.18: The frequency for Ind_Collectives and Corp_Collectives' expense entropy.

Corp_Collectives in terms of the number of issues, pull requests, watches, forks, contributors, stars, and commits of their associated project on GitHub. **In general, there is a positive correlation between the total donation of a project and its popularity.** We compute the correlation between total received donation amount and seven project-related metrics (i.e., the number of issues, pull requests, watches, forks, contributors, stars, and commits) using Spearman's rank correlation coefficient (Daniel et al., 1978). The correlations are 0.435, 0.404, 0.404, 0.386, 0.370, 0.339, and 0.260, respectively. The number of issues has the highest correlation with the total received donation amount of an open source project, indicating that the more issues in a project the higher the likelihood that project will receive higher donation amounts.

Both Ind_Collectives and Corp_Collectives are likely to spend funds on a small group of specific types of expenses (e.g., engineering and web services). Figure 6.18 shows that the expenses of Ind_Collectives and Corp_Collectives have a similar pattern in terms of expense entropy. The entropy of 41 (59%) of the collectives is no more than

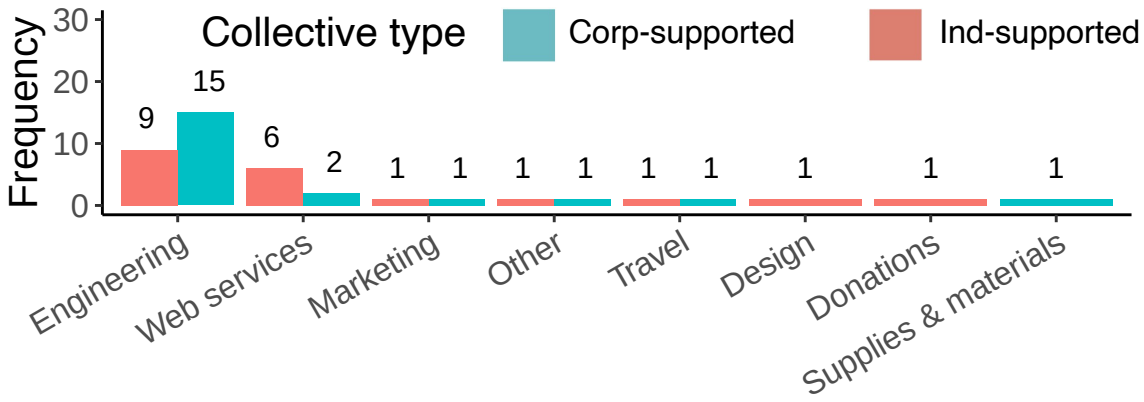


Figure 6.19: The frequency of the most costly expense type in low expense entropy Ind_Collectives and Corp_Collectives.

0.1, indicating those collectives spent funds on a very small group of specific expense types. In particular, 9 Ind_Collectives and 12 Corp_Collectives only spent funds on the engineering expense. Figure 6.19 shows the frequency of the most costly expense types in the 41 Ind_Collectives and Corp_Collectives with an entropy of no more than 0.1. We observed that the type of engineering is the most costly expense type in 45% (9 out of 20) Ind_Collectives and 71.4% (15 out of 21) Corp_Collectives. Except for the engineering expense, the web services expense is the main expense type for six of the Ind_Collectives.

Summary: Corp_Collectives tend to receive a higher total and monthly donation amount than Ind_Collectives. However, Corp_Collectives have no significant difference in terms of the popularity of their associated GitHub projects. Both Ind_Collectives and Corp_Collectives are likely to spend funds on a small variety of expense types (e.g., engineering and web services).

6.5 Discussion

Given that our study is one of the first studies to explore the use of donations in an open source projects setting, we now discuss some peculiarities about donations and the use of donated funds in an open source setting.

In particular, we discuss some interesting findings in terms of rejected expenses, a successful case of bounty expenses, payment options of donors and the purposes of donation. Then we highlight the implications of our findings.

6.5.1 Rejected expense submissions

168 expense submissions were rejected in our data set. To analyze the rationale for such rejections, we first manually identified and filtered out 35 invalid rejected expense submissions which were done for testing the Open Collective platform's expense rejection feature. Then we analyzed the 133 remained rejected expenses.

"Donation" expenses are more likely to be rejected. 42% of the "donation" expenses were rejected (i.e., a collective donating to another collective). In 13 collectives, 19 expenses were proposed as "Donation" type and eight of the 19 expenses were rejected.

There exists several users who submit fraudulent expenses. Before filtering out any rejected expenses, we identify 37 users whose expenses are always rejected. Some rejected expense amounts are large and the descriptions are meaningless. For example, a user *wassana-homchuen* submits three expenses to *Webpack* with the same value of $58,902 in one day with meaningless descriptions, i.e., "Available balance:", "http://www.90minlive.com", and "azuer". The user *japan-hunter* had similar behaviors to *Webpack*. Besides, both of these two user accounts were created

on the date they submitted their expenses. We suspect that these two users want to "steal" donations from *Webpack*.

6.5.2 A successful case of bounty expenses

Bounty is a type of monetary incentive in open source projects. Users can propose bounties to motivate developers to complete tasks, which can be a bug-fixing task or a documentation task. In Section 6.4.2, we observed that bounty is a frequent purpose of expenses in engineering-related expenses. We observed that three collectives (i.e., the *Buttercup*, *Boostnote*, and *JHipster* collectives) proposed bounties on issue reports. These bounties were paid out using the received donations of their corresponding collective. In total, there are 68 expenses related to a bounty with 91% of them being done in the *JHipster* collective. Especially, 77% (i.e., 62 out of 81) of the expenses in the *JHipster* collective were done to cover the costs of a bounty. The administrators of the *JHipster* collective explained that, with the growing user number, bounties were introduced to help manage the growing larger and more complex situation.[22] With a well-designed bounty rule[23] and the financial support from donations, 90.3% (i.e., 62 out of 67) bounty issue reports were addressed, which is much higher than the average addressing rate (i.e., 43.0%) of bounty issue reports (Zhou et al., 2020b).

[22] https://medium.com/open-collective/jhipsters-bounty-system-and-how-it-saved-the-project-cc118888f642

[23] https://www.jhipster.tech/bug-bounties/

Figure 6.20: The word cloud of the top 10 frequent words from donation messages left by donors.

6.5.3 Payment options for donors

Donors can make donations using *Stripe*, *PayPal*, credit card, debit card, and gift card. **Stripe is the most popular payment processor for donors.** 97.2% donations are made through Stripe,[24] which is an online payment processors. **The gift card is a suggested payment method but still not a popular one.** Comparing with donating through credit cards or prepaid cards, the gift card is a more flexible payment method for corporations. By using gift cards, corporations can let their employees choose the collective that they might wish to support. Besides, we find that the median donation amount of gift cards is $20 which is higher than the median donation amount (i.e., $5) by other payment methods. Although the use of a gift card is officially recommended by the Open Collective platform officially,[25] only 1.2% of the donations were made using gift cards.

[24] https://stripe.com/
[25] https://docs.opencollective.com/help/backers-and-sponsors/gift-cards

6.5.4 Purposes of donation

43% of the donation messages express their gratitude to collectives. There are 41,471 donations after Oct. 6, 2017, of which only 584 of them had a donation message. Figure 6.20 visualizes the frequency of the top 10 frequent words in the redonation messages. We observed that the top three frequent words are "thank", "work", and "great". 43% (i.e., 256 out of 589) of the messages contain either one of them. These three words are used to express gratitude to the collectives, e.g., "Thanks for doing great work". Especially we found six messages which express their appreciation to collectives for special release versions, for example, "GitExtension 3.0 release congrats". We also observed that the keyword "keep" in 11% (i.e., 64 out of 589) messages and this keyword expresses encouragements from donors, for example, "Keep going! Awooooo". Besides all these good words, we also observed a few cases which using the message for a successful donation contains an advertisement. For example, the contents of the six messages are about an online casino website.

6.5.5 Implications

Collectives should consider attracting more individual donors. In Section 6.4.1, we found that in general corporate donors donate more funds than individual donors in a single donation, but individual donates more money than corporate donors in total for a collective. One possible explanation is that individual donors tend to donate to a collective more consistently than corporate donors. This observation shows the importance of individual donors. Hence, the stakeholders of collectives should consider attracting more individual donors over corporate donors.

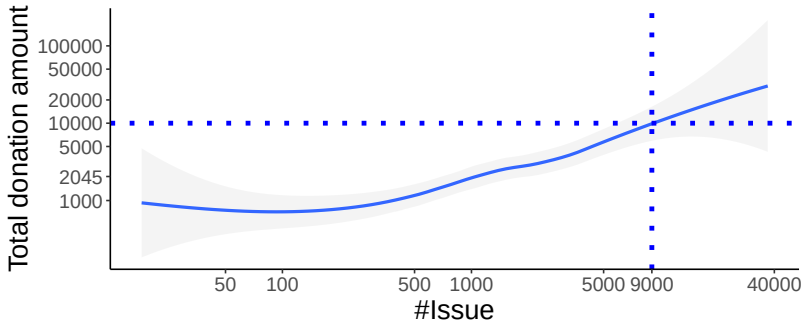


Figure 6.21: The relationship between the total received donation amount of collectives and the number of issues of their associated GitHub projects. The donation amount is $10,000 when the number of issues reaches 9,000.

Collectives should not expect to receive a large amount of donations unless their associated projects are very popular and their projects are mainly supported by corporations. Figure 6.21 shows the trend of received donation amount of collectives against the number of issues of the in-associated open source projects. We observe that when the number of issues of a project reaches 9,000, the received donation amount is $10,000. However, the likelihood of a GitHub project to reach such level of popularity is low. The mean and median number of issue requests of GitHub projects for different languages vary from 2.0 to 64.4 and 1 to 25, respectively (Bissyandé et al., 2013). Hence, collectives may not receive many donations from the community unless their projects are very popular. In addition, we observe that Corp_Collectives are much more likely to get a large amount of donations (i.e., $10,000) compared with Ind_Collectives. Therefore, our findings suggested that collectives may not receive a large amount of funds from donations unless their projects are very popular and have corporations to support them.

Projects should budget for a reasonable amount (e.g., 13% of total funds) of non-engineering expenses when operating an open source project (e.g., marketing and

travel). In Section 6.4.3, we observe 13% of the total expense amount is spent on non-engineering-related expenses. In Section 6.4.4, we show that 75.0% (104 of 139) of the collectives with expenses have non-engineering-related expenses and such expenses take up 45.6% (i.e., 665 out of 1,459) of their total expenses. In other words, non-engineering expenses are quite common in open source projects. For instance, two types of very frequent non-engineering expenses are marketing and travel. Therefore, we suggest that open source projects should allocate a reasonable amount of budgets for such non-engineering expenses.

6.6 Threats to Validity

Open source project donation is an area which is rapidly becoming a crucial area with the strong industrial support and involvement (**?**). This study is the first step in such an area. Threats to **external validity** are related to the generalizability of our findings. Since we focus on open source projects on the Open Collective platform, our results might not generalize to other platforms. Future work is needed to explore the generality of our observations.

Threats to **internal validity** relate to the experimenter's bias and errors. One threat to internal validity is that we manually identified GitHub repositories of 102 studied collectives in Section 6.3, which may introduce bias due to human factors. Another internal validity is that we manually relabeled expenses types for expenses in Section 6.4.2 which may introduce bias. To mitigate the threat of bias during the manual analysis, two of the authors conducted the manual analysis and discussed conflicts until a consensus was reached. We used Cohen's kappa (Gwet et al., 2002) to measure the inter-rater agreement. Before discussing differences, the Cohen's kappa

coefficients are 0.94 for GitHub repositories identification and 0.91 for expense type relabeling, both indicating a high level of agreement.

One threat to the internal validity of our study is that we choose 80% as the threshold to tag a collective as being an individual or corporate supported collective. To alleviate this threat, we conducted a sensitivity analysis with a higher threshold of 90%. Our findings still hold for the higher threshold level.

6.7 Related Work

6.7.1 Financial Incentives in Software Engineering

Financial incentives (e.g., donations and bounties) are being offered in open source projects to improve development and sustainability of open source projects. For example, open source projects collect donations to cover maintenance cost and maintainers propose bounties to attract and motivate developers to complete software development. Several of studies have investigated how to improve the donations in software engineering. Krishnamurthy et al. (2014) studied why some open source developers accept financial rewards while others do not. They found that intrinsic (e.g., the need for a creative task) and extrinsic motivations (e.g., rewards) positively influence their willingness to accept monetary rewards, while community motivation (e.g., contributing to a social community) negatively influences their willingness to accept rewards. Krishnamurthy and Tripathi (2009) investigated the factors that impact community members' donation on open source platform (i.e., *Sourceforge*)[26] and found that the community members' time length of association with the community and rational

[26] https://sourceforge.net/

commitment affect their donations. Yukizawa et al. (2019) investigated two psychological theories (i.e., social proof and legitimization of party contributions) to promote donations in open source projects. From an interview, they found that these two theories help promote and improve donations to open source projects. Nakasai et al. (2017) studied donations in the Eclipse project and found that the offered benefits to donors can motivate donations.

Another group of studies focuses on studying the impact of bounties on software engineering tasks (e.g., bug-fixing). Nakasai et al. (2018) studied how donation badges (i.e., a widget used on the website to show a user has made donations) impact developers' responses to bug reports by a donor. They observed that Eclipse developers respond faster to the bug reports which are reported by users that have donation badges. Kanda et al. (2017) studied the issue reports that were offered bounties of GitHub projects and showed that the closing-time of bounty issue reports is longer than that of non-bounty issue reports. Zhou et al. (2020b) performed a large scale study on more than 3,000 issue reports and found that the timing and bounty-usage frequency of a project are important factors in increasing the issue-addressing likelihood. Various studies analyzed the usage of bounties to motivate developers to detect and report software security vulnerability (Finifter et al., 2013; Maillart et al., 2017; Zhao et al., 2014). For example, Finifter et al. (2013) studied the vulnerability rewards program for Chrome and Firefox and found that compared to the cost of hiring a full-time security researcher, the vulnerability program is cheaper. Maillart et al. (2017) suggested that project managers should dynamically adjust the value of rewards (e.g., money) according to the market situation (e.g., increase rewards when releasing a new

version). In our study, we observed several cases that applied donations to propose bounties to address issue reports.

Different from prior studies, we investigate the donors and their behaviour, and the usage of these donations on operating open source projects.

6.7.2 Understanding and Improving the Sustainability of Open Source Software Projects

Keeping open source projects sustainable is a challenging task. Therefore, researchers performed a significant number of studies on this topic to understand the sustainability of open source projects (Gamalielsson and Lundell, 2014; Valiev et al., 2018; Coelho and Valente, 2017; Eghbal, 2016). Valiev et al. (2018) performed an empirical study to understand the relationship between the sustainability of a project and its surrounding projects (i.e., dependent projects or projects on which it depends) in the ecosystem. They showed that the number of project ties and the relative position in the dependency network has a significant impact on the sustainability of a project. Coelho and Valente (2017) investigated the reasons why modern open source projects fail and they found that failures are due to various reasons (e.g., lack of interest and time, low maintainability, and conflicts among developers). Eghbal (2016) reported the risks and challenges that are associated with maintaining open source projects, and argued that open source projects still lack a reliable and sustainable source of funds.

A number of prior studies studied the sustainability of open source projects from the angle of contributors (Lee et al., 2017; Ye and Kishida, 2003; Avelino et al., 2016; Canfora et al., 2012; Pinto et al., 2016). For example, Avelino et al. (2016) found that 65% of their studied projects rely on one or two developers to survive. Lee et al. (2017)

studied the motivations, and barriers that are experienced by the one-time code contributors. They found that the main motivation for one time contributors is to fix bugs that impeded their work. Such one-time contributors did not plan on becoming long term contributors due to various barriers, e.g., entry difficulties and lack of time. Ye and Kishida (2003) investigated the motivation of developers to participate in open source projects and found that the desire to learn is one of the major motivation and they also provided insights to improve the sustainability of open source projects based on their findings, e.g., creating a friendly environment and culture for newcomers to learn from the community. To help newcomers, Canfora et al. (2012) proposed an approach to identify and recommend mentors in software projects by mining data from mailing lists and version control systems. Steinmacher et al. (2016) proposed a portal, namely FLOSScoach, to support newcomers to open source projects.

In our study, we provide insights on how projects used their donated funds and find that development and web services expenses are the major expenses for open source projects.

6.8 Chapter Summary

In this paper, we studied 225 open source software projects that collected a total donation amount of $2,537,281 through the Open Collective platform. We first analyzed how donors make donations and how collectives use these donated funds. We found that:

1. In general, corporate donors tend to donate more funds than individual donors for an individual donation, while the total donation amount from individual

donors are more than corporate donors in a collective, which suggests the influence of individual donors. Moreover, individual donors are more likely to redonate to a collective compared to corporate donors.

2. Non-engineering-related expenses take up 54.0% of the total number of all expenses. “Web services”, “marketing”, and “travel” are the three most frequent and costly non-engineering-related expense types. For engineering-related expenses, the most frequent expenses are related to development and maintenance.

3. Corp_Collectives are more likely to receive a larger total donation amount than Ind_Collectives collectives.

Our findings suggest that open source projects should try to attract more individual donors. Projects should not expect to receive a large amount of donations unless they are very popular and are mainly supported by corporations. Projects should budget for a reasonable amount of non-engineering expenses (e.g., marketing and traveling).

CHAPTER 7

Conclusion and Future Work

THE Summary

7.1 Thesis Contributions

The goal of this thesis is to highlight the promising outcome of mining online distribution platforms for games to provide practical suggestions for game developers. In our literature survey, we showed that extensive research has been conducted in the field of mining mobile app stores. However, prior studies confirmed the large differences between developing a traditional non-game and a game software system, raising threats to directly applying knowledges from prior mobile app store research to game

engineering. Below, I summarize the main contributions of this thesis around the four studied aspects of online distribution for games:

1. **The choice of update strategy is associated with the proportion of 0-day updates.** Urgent updates are usually released in a state of emergency, causing unnecessary stress on developers. The stress of these so-called "fire-fighting conditions" can not only lead to inefficient problem solving, but also introduce changes that can easily create new problems (**?**). In this thesis, We empirically studied urgent updates of games on the Steam platform to understand the causes behind urgent updates. We observed that games that release frequently also release a higher proportion of 0-day updates than games that use a traditional build-up candidate update strategy. Our findings are consistent with the findings of **?**, who observed that releasing frequently leads to a higher proportion of patches that must be reverted.

2. **The early access model helps developers elicite early feedback and more positive reviews to attract additional new players.** In order to get a better understanding of the impact and limitations of the early access model, we conducted an in-depth empirical study on early access games on the Steam platform. We observed a correlation between early access games and a higher positive review rate. While the early access model is not a fix for low-quality games, the early access model appears to be a valuable tool for developers that want to improve their games by interacting with their players.

3. **Although negative reviews contain more valuable information for developers, the portion of useful information in positive reviews should not be ignored by**

developers and researchers. The competition within the gaming industry, and the hard-to-please user base has made the quality of games an increasingly important issue. As game reviews are a direct reflection of user concerns, a better understanding of reviews can help developers produce games with an improved user-perceived quality. In this thesis, we performed an empirical study on the reviews of games on the Steam platform. We observed that 29% of the positive reviews discuss cons of the games, and 7% of the positive reviews report bugs in the games. Moreover, positive reviews contain a higher portion of pros of the games, and a slightly higher portion of suggestions, than negative reviews. Hence, developers and researchers should not dismiss the information that can be extracted from positive reviews.

4. **User-recorded gameplay videos that are available online can be used as a supplemental source of bug reports for games.** In recent years, gameplay videos have become popular in the gaming community. The high amount of videos around game bugs opens a new opportunity for developers to collect intuitive information of bugs. In this thesis, we proposed an approach that uses the metadata of gameplay videos to train a random forest classifier, to determine the likelihood that a gameplay video showcases a game bug. Our approach achieves a mean average precision at 100 of 0.91, and shows a good generalizability.

7.2 Future Research Directions

The results of our empirical studies highlighted the promising outcome of mining online distribution platforms for games to provide practical suggestions for game developers. In addition to the results stated in this thesis, there are other aspects of online distribution that we think should be examined by future research efforts.

7.2.1 The impact of rapid updating of games on well-established software engineering practices

Prior work (**???**) on update strategies focused mostly on the Mozilla Firefox project, in which the update strategy changed from traditional build-up candidate updates to frequent updates (i.e., every six weeks). In this thesis, we showed that most games update much more frequently than once every six weeks (see Chapter **??**), a phenomenon that was recently observed for mobile apps (**?**). The unique distribution mechanism (e.g. online store) of games and mobile apps allows developers to release updates for their software at an increasingly rapid pace. Future research efforts need to carefully reconsider how such rapid pace of updating software influences our well-established understandings of software engineering practices. For example, how should the practice of requirement engineering and testing be adjusted to cope with such rapid update pace for games.

7.2.2 A deep understanding is needed for the relation between using the early access model and the satisfaction of both players and developers.

In this thesis, we observed a correlation between early access games and a higher positive review rate (see Chapter **??**). One possible explanation for this correlation is that players who buy EAGs are friendlier towards developers. Another explanation is that developers that use the early access model are good at keeping their players satisfied. Future studies should use methods such as developer surveys, user studies, and controlled experiments to examine in more depth the causality between using the early access model and the satisfaction of both players and developers.

7.2.3 The early access model for non-game software.

In this thesis, we investigated the early access model for games and provided game developers with suggestions on how to leverage the early access model. Some traditional software has also adopted similar development models, for example, the "Office Insider" program for Microsoft Office (**?**). Future studies should verify if such models for non-game software share similar characteristics as the early access model.

7.2.4 Adjusting existing automated techniques for analyzing mobile app reviews so they can be applied successfully to game reviews.

Throughout our study of reviews for games on the Steam platform, we observed that in several aspects, game reviews are different from mobile app reviews (see Chapter **??**).

In addition, we noticed that reviews for games tend to be sarcastic (e.g., "Went to washroom for one minute and lost the game. 10/10."), posing threats to directly applying automated sentiment analysis or information retrieval techniques for analyzing mobile app reviews on game reviews. Hence, in this thesis we conducted manual categorization of statistically representative samples of reviews to ensure the quality of our findings. Future studies should investigate further how to adjust existing automated techniques for analyzing free form reviews of games and mobile apps.

7.2.5 Revisiting prior work on mobile app reviews to take missing factors into account.

Prior research on mobile app reviews only had access to the number of owners of an app, while our study analyzed both the number of owners and the number of active users (players) of games, and observed that the number of players has a stronger relation with the number of reviews received per day. In addition, prior mobile app studies that do study paid apps, tend to ignore the impact of discounts. These studies of mobile app reviews should be revisited, as we observed that a sales event has the strongest relation with an increase in the number of received game reviews.

7.2.6 Comparing mobile games with PC games.

Our studies focused on mining the Steam platform, which is an online distribution platform dedicated for PC games. Prior studies on mobile games (**????**) show that developing mobile games has its special software and hardware challenges and peculiarities, compared to developing PC games. Future studies should investigate whether the findings and suggestions in our thesis hold for mobile game development.

7.2.7 Surveying and interviewing game developers to evaluate the suggestions from this thesis.

In this thesis, we studied four aspects of online distribution for games to provide valuable insights and practical suggestions. Our research received widespread coverage from several prestigious media outlets in the gaming industry, including PC Gamer (**?**), Kotaku (**??**), and Gamasutra (**?**). These outlets receive up to 50 million visitors per month and are the leading venues in the gaming community. In addition, our publications have been read for more than 18,000 times on ResearchGate alone. The overwhelming attention and positive feedback that we received from the gaming community showcase the relevance of our studies to this community. Nevertheless, future studies should consider other more structured avenues (e.g., interview and surveys) to gather feedback from the gaming community about our work and our suggestions.

Bibliography

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C. K., and Schneider, K. A. (2018). Classifying Stack Overflow posts on API issues. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 244–254.

Alelyani, T., Mao, K., and Yang, Y. (2017). Context-centric pricing: early pricing models for software crowdsourcing tasks. In *Proceedings of 13th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE)*, pages 63–72.

Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2012). Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 850–858. ACM.

Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2013). Steering user behavior with badges. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 95–106.

Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M., Gousios, G., et al. (2011). Open source software: A survey from 10,000 feet. *Foundations and Trends in Technology, Information and Operations Management*, 4(3–4):187–347.

Arya, D., Wang, W., Guo, J. L., and Cheng, J. (2019). Analysis and detection of information types of open source software issue discussions. In *Proceedings of the 41st International Conference on Software Engineering*, pages 454–464. IEEE Press.

Atiq, A. and Tripathi, A. (2016). Impact of financial benefits on open source software sustainability. In *International Conference on Information Systems (ICIS)*, pages 1–10.

Avelino, G., Passos, L., Hora, A., and Valente, M. T. (2016). A novel approach for estimating truck factors. In *IEEE 24th International Conference on Program Comprehension (ICPC)*, pages 1–10.

Bauer, D. F. (1972). Constructing confidence sets using rank statistics. *Journal of the American Statistical Association*, 67(339):687–690.

Berger, P., Hennig, P., Bocklisch, T., Herold, T., and Meinel, C. (2016). A journey of bounty hunters: Analyzing the influence of reward systems on stackoverflow question response times. In *IEEE/WIC/ACM international conference on web intelligence (WI)*, pages 644–649. IEEE.

Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., and Zimmermann, T. (2008). What makes a good bug report? In *Proc. of the Int'l Symp. on Foundations of Software Engineering*, pages 308–318. ACM.

Bissyandé, T. F., Thung, F., Lo, D., Jiang, L., and Réveillère, L. (2013). Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *2013 IEEE 37th annual computer software and applications conference*, pages 303–312. IEEE.

Bonferroni, C. (1936). Statistical class theory and probability calculus. *Publications of the R Higher Institute of Economic and Commercial Sciences of Florence*, 8:3–62.

Canfora, G., Di Penta, M., Oliveto, R., and Panichella, S. (2012). Who is going to mentor newcomers in open source projects? In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE)*, pages 1–11.

Chen, C., Chen, X., Sun, J., Xing, Z., and Li, G. (2018). Data-driven proactive policy assurance of post quality in community Q&A sites. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW):33:1–33:22.

Chen, C., Xing, Z., and Liu, Y. (2017). By the community & for the community: A deep learning approach to assist collaborative editing in Q&A sites. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW):32:1–32:21.

Cochran, W. G. (2007). *Sampling techniques*. John Wiley & Sons.

Coelho, J. and Valente, M. T. (2017). Why modern open source projects fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 186–196.

Daniel, W. W. et al. (1978). *Applied nonparametric statistics*. Houghton Mifflin.

Efron, B. (1986). How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81(394):461–470.

Eghbal, N. (2016). *Roads and bridges: The unseen labor behind our digital infrastructure*. Ford Foundation.

Farrar, D. E. and Glauber, R. R. (1967). Multicollinearity in regression analysis: the problem revisited. *The Review of Economic and Statistics*, pages 92–107.

Finifter, M., Akhawe, D., and Wagner, D. (2013). An empirical study of vulnerability rewards programs. In *USENIX Security Symposium*, pages 273–288.

Ford, D., Lustig, K., Banks, J., and Parnin, C. (2018). "We don't do that here": How collaborative editing with mentors improves engagement in social Q&A communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 608:1–608:12.

Gamalielsson, J. and Lundell, B. (2014). Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved? *Journal of Systems and Software*, 89(1):128 – 145.

Grant, S. and Betts, B. (2013). Encouraging user behaviour with achievements: an empirical study. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 65–68. IEEE.

Gwet, K. et al. (2002). Inter-rater reliability: dependency on trait prevalence and marginal homogeneity. *Statistical Methods for Inter-Rater Reliability Assessment Series*, 2(1):1–9.

Halavais, A., Kwon, K. H., Havener, S., and Striker, J. (2014). Badges of friendship: Social influence and badge acquisition on stack overflow. In *2014 47th Hawaii international conference on System Sciences (HISS)*, pages 1607–1615. IEEE.

Hann, I.-H., Roberts, J., Slaughter, S., and Fielding, R. (2002). Economic incentives for participating in open source software projects. *International Conference on Information Systems (ICIS)*, page 33.

Hanrahan, B. V., Convertino, G., and Nelson, L. (2012). Modeling problem difficulty and expertise in stackoverflow. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*, pages 91–94. ACM.

Harrell, Jr., F. E. (2006). *Regression modeling strategies*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Hassan, A. E. (2009). Predicting faults using the complexity of code changes. In *IEEE 31st international conference on software engineering (ICSE)*, pages 78–88. IEEE.

Hata, H., Guo, M., and Babar, M. A. (2017). Understanding the heterogeneity of contributors in bug bounty programs. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 223–228.

Hooimeijer, P. and Weimer, W. (2007). Modeling bug report quality. In *Proc. of the Int'l Conf. on Automated Software Engineering*, pages 34–43. ACM.

Izquierdo, J. L. C. and Cabot, J. (2018). The role of foundations in open source projects. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society (ICSE SEIS)*, pages 3–12.

Jan, S. T., Wang, C., Zhang, Q., and Wang, G. (2017). Towards monetary incentives in social Q&A services. *arXiv preprint arXiv:1703.01333*.

Jarczyk, O., Jaroszewicz, S., Wierzbicki, A., Pawlak, K., and Jankowski-Lorek, M. (2018). Surgical teams on GitHub: Modeling performance of GitHub project development processes. *Information and Software Technology*, 100:32–46.

Kanda, T., Guo, M., Hata, H., and Matsumoto, K. (2017). Towards understanding an open-source bounty: Analysis of bountysource. In *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 577–578.

Katmada, A., Satsiou, A., and Kompatsiaris, I. (2016). Incentive mechanisms for crowdsourcing platforms. In *International Conference on Internet Science (ICIS)*, pages 3–18. Springer.

Krishnamurthy, S. (2006). On the intrinsic and extrinsic motivation of free/libre/open source (floss) developers. *Knowledge, Technology & Policy*, 18(4):17–39.

Krishnamurthy, S., Ou, S., and Tripathi, A. K. (2014). Acceptance of monetary rewards in open source software development. *Research Policy*, 43(4):632–644.

Krishnamurthy, S. and Tripathi, A. K. (2006). Bounty programs in free/libre/open source software. In *The Economics of Open Source Software Development*, pages 165–183. Elsevier.

Krishnamurthy, S. and Tripathi, A. K. (2009). Monetary donations to an open source software platform. 38(2):404–414.

LaToza, T. D. and Van Der Hoek, A. (2015). Crowdsourcing in software engineering: Models, motivations, and challenges. *IEEE software*, 33(1):74–80.

Lee, A., Carver, J. C., and Bosu, A. (2017). Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: A survey. In *IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 187–197.

Li, Z., Huang, K.-W., and Cavusoglu, H. (2012). Quantifying the impact of badges on user engagement in online Q&A communities. *International Conference on Information Systems (ICIS)*, (10).

Liu, J., Zhou, P., Yang, Z., Liu, X., and Grundy, J. (2018). Fasttagrec: fast tag recommendation for software information sites. *Automated Software Engineering*, 25(4):675–701.

Long, J. D., Feng, D., and Cliff, N. (2003). Ordinal analysis of behavioral data. *Handbook of psychology*, pages 635–661.

Lotufo, R., Passos, L., and Czarnecki, K. (2012). Towards improving bug tracking systems with game mechanisms. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 2–11. IEEE.

Maillart, T., Zhao, M., Grossklags, J., and Chuang, J. (2017). Given enough eyeballs, all bugs are shallow? Revisiting Eric Raymond with bug bounty programs. *Journal of Cybersecurity*, 3(2):81–90.

Mao, K., Capra, L., Harman, M., and Jia, Y. (2017). A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, 126:57–84.

Mao, K., Yang, Y., Li, M., and Harman, M. (2013). Pricing crowdsourcing-based software development tasks. In *35th International Conference on Software Engineering (ICSE)*, pages 1205–1208. IEEE.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (2016). An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering (EMSE)*, 21(5):2146–2189.

Miura, K., McIntosh, S., Kamei, Y., Hassan, A. E., and Ubayashi, N. (2016). The impact of task granularity on co-evolution analyses. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, page 47.

Nakasai, K., Hata, H., and Matsumoto, K. (2018). Are donation badges appealing?: A case study of developer responses to eclipse bug reports. *IEEE Software*, 36(3):22–27.

Nakasai, K., Hata, H., Onoue, S., and Matsumoto, K. (2017). Analysis of donations in the eclipse project. In *8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, pages 18–22.

Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., and Tonelli, R. (2015). Are bullies more productive?: Empirical study of affectiveness vs. issue fixing time. In *Proc. of the Working Conf. on Mining Software Repositories*, pages 303–313.

Overney, C., Meinicke, J., Kästner, C., and Vasilescu, B. (2020). How to not get rich: An empirical study of donations in open source. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE)*.

Palomba, F., Zanoni, M., Fontana, F. A., De Lucia, A., and Oliveto, R. (2017). Toward a smell-aware bug prediction model. *IEEE Transactions on Software Engineering*.

Pinto, G., Steinmacher, I., and Gerosa, M. A. (2016). More common than you think: An in-depth study of casual contributors. In *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 112–123.

Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M., and Fullerton, D. (2014). Improving low quality Stack Overflow post detection. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 541–544.

Rajbahadur, G. K., Wang, S., Kamei, Y., and Hassan, A. E. (2017). The impact of using regression models to build defect classifiers. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*, pages 135–145.

Rich Sands (2012). Open Source By The Numbers. https://www.slideshare.net/blackducksoftware/open-source-by-the-numbers.

Romano, J., Kromrey, J. D., Coraggio, J., and Skowronek, J. (2006). Appropriate statistics for ordinal level data: Should we really be using t-test and cohensd for evaluating group differences on the nsse and other surveys. In *annual meeting of the Florida Association of Institutional Research*, pages 1–33.

Saha, R. K., Lease, M., Khurshid, S., and Perry, D. E. (2013). Improving bug localization using structured information retrieval. In *Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering*, pages 345–355.

Soto, M., Thung, F., Wong, C.-P., Le Goues, C., and Lo, D. (2016). A deeper look into bug fixes: Patterns, replacements, deletions, and additions. In *Proc. of the Int'l Conf. on Mining Software Repositories*, pages 512–515.

Srba, I. and Bielikova, M. (2016). Why is Stack Overflow failing? Preserving sustainability in community question answering. *IEEE Software*, 33(4):80–89.

Stack Exchange (2017). Stack Exchange. https://archive.org/details/stackexchange. (last visited: Dec. 20, 2017).

Stack Overflow (2019). Stack Overflow: User Privileges. https://stackoverflow.com/help/privileges. (last visited: Jan. 23, 2019).

Steinmacher, I., Conte, T. U., Treude, C., and Gerosa, M. A. (2016). Overcoming open source project entry barriers with a portal for newcomers. In *IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 273–284.

Steinmacher, I., Pinto, G., Wiese, I. S., and Gerosa, M. A. (2018). Almost there: a study on quasi-contributors in open source software projects. In *Proceedings of the 40th International Conference on Software Engineering, (ICSE)*, pages 256–266.

Tantithamthavorn, C. and Hassan, A. E. (2018). An experience report on defect modelling in practice: Pitfalls and challenges. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE SEIP)*, pages 286–295. ACM.

Thongtanunam, P., McIntosh, S., Hassan, A. E., and Iida, H. (2016). Revisiting code ownership and its relationship with software quality in the scope of modern code review. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 1039–1050. IEEE.

Tian, Y., Nagappan, M., Lo, D., and Hassan, A. E. (2015). What are the characteristics of high-rated apps? A case study on free android applications. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 301–310. IEEE.

Treude, C., Barzilay, O., and Storey, M.-A. (2011). How do programmers ask and answer questions on the web?: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 804–807. IEEE.

Valiev, M., Vasilescu, B., and Herbsleb, J. (2018). Ecosystem-level determinants of sustained activity in open-source projects: A case study of the pypi ecosystem. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 644–655.

Wang, L., Yang, Y., and Wang, Y. (2019). Do higher incentives lead to better performance?-an exploratory study on software crowdsourcing. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. IEEE.

Wang, S., Chen, T.-H., and Hassan, A. E. (2018a). Understanding the factors for fast answers in technical Q&A websites. *Empirical Software Engineering*, 23(3):1552–1593.

Wang, S., Chen, T.-H. P., and Hassan, A. E. (2018b). How do users revise answers on technical Q&A websites? A case study on Stack Overflow. *IEEE Transactions on Software Engineering*.

Wang, S. and Lo, D. (2014). Version history, similar report, and structure: Putting them together for improved bug localization. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 53–63. ACM.

Wang, S. and Lo, D. (2016). Amalgam+: Composing rich information sources for accurate bug localization. *Journal of Software: Evolution and Process*, 28(10):921–942.

Wang, S., Lo, D., and Lawall, J. (2014a). Compositional vector space models for improved bug localization. In *IEEE Int'l Conf. on Software Maintenance and Evolution*, pages 171–180.

Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2014b). Entagrec: An enhanced tag recommendation system for software information sites. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 291–300. IEEE.

Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2018c). Entagrec ++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, 23(2):800–832.

Wang, S., Lo, D., Xing, Z., and Jiang, L. (2011). Concern localization using information retrieval: An empirical study on Linux kernel. In *2011 18th Working Conference on Reverse Engineering*, pages 92–96. IEEE.

Wei, X., Chen, W., and Zhu, K. (2015). Motivating user contributions in online knowledge communities: virtual rewards and reputation. In *48th Hawaii International Conference on System Sciences (HICSS)*, pages 3760–3769. IEEE.

Wu, Y., Wang, S., Bezemer, C.-P., and Inoue, K. (2018). How do developers utilize source code from Stack Overflow? *Empirical Software Engineering (EMSE)*, pages 637–673.

Xia, X., Lo, D., Wang, X., and Zhou, B. (2013). Tag recommendation in software information sites. In *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, pages 287–296.

Xu, L., Nian, T., and Cabral, L. (2020). What makes geeks tick? a study of stack overflow careers. *Management Science*, 66(2):587–604.

Yamashita, K., Kamei, Y., McIntosh, S., Hassan, A. E., and Ubayashi, N. (2016). Magnet or sticky? measuring project characteristics from the perspective of developer attraction and retention. *Journal of Information Processing*, 24(2):339–348.

Yanovsky, S., Hoernle, N., Lev, O., and Gal, K. (2019). One size does not fit all: Badge behavior in q&a sites. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization (UMAP)*, pages 113–120.

Ye, Y. and Kishida, K. (2003). Toward an understanding of the motivation open source software developers. In *Proceedings of the 25th International Conference on Software Engineering (ICSE)*, pages 419–429.

Yukizawa, U., Tsunoda, M., and Tahir, A. (2019). Please help! A preliminary study on the effect of social proof and legitimization of paltry contributions in donations to OSS.

In *26th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 609–613.

Zhang, H., Wang, S., Chen, T.-H. P., and Hassan, A. E. (2019). An empirical study of obsolete answers on Stack Overflow. *IEEE Transactions on Software Engineering*.

Zhao, M., Grossklags, J., and Chen, K. (2014). An exploratory study of white hat behaviors in a web vulnerability disclosure program. In *Proceedings of ACM workshop on security information workers (SIW)*, pages 51–58.

Zhao, M., Grossklags, J., and Liu, P. (2015). An empirical study of web vulnerability discovery ecosystems. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1105–1117.

Zhao, M., Laszka, A., and Grossklags, J. (2017). Devising effective policies for bug-bounty platforms and security vulnerability discovery. *Journal of Information Policy*, 7:372–418.

Zhong, H. and Su, Z. (2015). An empirical study on real bug fixes. In *Proc. of the Int'l Conf. on Software Engineering*, pages 913–923.

Zhou, J. (2018). Supplementary material for the study of issue bounties in github open source projects. https://github.com/SAILResearch/wip-18-jiayuan-bountysource-SupportMaterials/blob/master/appendix.pdf.

Zhou, J. (2019). Supplementary material for the study of reputation bounties on stack overflow. https://github.com/SAILResearch/wip-18-jiayuan-SO-bounty-SupportMaterials/blob/master/appendix.pdf.

Zhou, J., Wang, S., Bezemer, C.-P., and Hassan, A. E. (2019). Bounties on technical q&a sites: a case study of stack overflow bounties. *Empirical Software Engineering (EMSE)*, 25(1):139–177.

Zhou, J., Wang, S., Bezemer, C.-P., Zou, Y., and Hassan, A. E. (2020a). Studying the association between bountysource bounties and the issue-addressing likelihood of github issue reports. *IEEE Transactions on Software Engineering (TSE)*. Preprint on webpage at https://ieeexplore.ieee.org/document/9000923.

Zhou, J., Wang, S., Bezemer, C.-P., Zou, Y., and Hassan, A. E. (2020b). Studying the association between bountysource bounties and the issue-addressing likelihood of github issue reports. *IEEE Transactions on Software Engineering (TSE)*.

Zhou, J., Zhang, H., and Lo, D. (2012). Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports. In *Proceedings of the 34th International Conference on Software Engineering*, pages 14–24.