A Case Study of Implicit Mentoring, Its Prevalence, and Impact in Apache

Zixuan Feng
Oregon State University
Corvallis, OR, USA
fengzi@oregonstate.edu

Amreeta Chatterjee
Oregon State University
Corvallis, OR, USA
chattera@oregonstate.edu

Anita Sarma
Oregon State University
Corvallis, OR, USA
anita.sarma@oregonstate.edu

Iftekhar Ahmed
University of California
Irvine, CA, USA
iftekha@uci.edu

ABSTRACT

Mentoring is traditionally viewed as a dyadic, top-down apprenticeship. This perspective, however, overlooks other forms of informal mentoring taking place in everyday activities in which developers invest time and effort. Here, we investigate informal mentoring taking place in Open Source Software (OSS). We define a specific type of informal mentoring—*implicit mentoring*—situations where contributors guide others through instructions and suggestions embedded in everyday (OSS) activities. We defined implicit mentoring by first performing a review of related work on mentoring, and then through formative interviews with OSS contributors and member-checking. Next, through an empirical investigation of Pull Requests (PRs) in 37 Apache Projects, we built a classifier to extract implicit mentoring. Our analysis of 107,895 PRs shows that implicit mentoring does occur through code reviews (27.41% of all PRs included implicit mentoring) and is beneficial for both mentors and mentees. We analyzed the impact of implicit mentoring on OSS contributors by investigating their contributions and learning trajectories in their projects. Through an online survey (N=231), we then triangulated these results and identified the potential benefits of implicit mentoring from OSS contributors' perspectives.

CCS CONCEPTS

• **Human-centered computing** → *Open source software.*

KEYWORDS

Implicit mentoring, Open Source Projects

ACM Reference Format:
Zixuan Feng, Amreeta Chatterjee, Anita Sarma, and Iftekhar Ahmed. 2022. A Case Study of Implicit Mentoring, Its Prevalence, and Impact in Apache. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3540250.3549167

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3549167

1 INTRODUCTION

Open Source Software (OSS) projects are volunteer-led communities where contributors worldwide collaborate to create large, complex software [14, 95]. Mentoring plays a key role in ensuring the sustainability of OSS projects by training new (and current) contributors who need to learn the necessary technical skills and the process and cultural norms of the community.

Mentoring usually describes an interpersonal relationship where an experienced contributor (the *mentor*) provides functional advice and interpersonal guidance to an inexperienced individual (the *mentee*) [47]. Mentoring relationships between the two parties facilitate the transfer of declarative knowledge–technical facts that mentees need to accomplish their tasks–and procedural knowledge about navigating the contribution process and project culture [5, 74]. Research shows that mentoring is an effective means for newcomer training and improves the onboarding experience and retention of contributors in OSS [21, 74].

It is no surprise then that OSS foundations have heavily invested in mentorship programs such as the Google Summer of Code scholarship program, which paired 1,289 students with mentors from across OSS organizations in summer 2021 [32]. The Linux Foundation currently has seven mentorship programs and has invested over 2.5 million USD in support of first-time and underrepresented OSS contributors [52]. Additionally, OSS projects (e.g., Apache Mentoring Program [24]) connect newcomers to project mentors by providing some basic mentoring structure (e.g., mentor-mentee interest matching and guidance on communication, task scope, and progress checkpoints).

Numerous studies have investigated the benefits of formal mentoring [21, 83]. However, the dedicated time and effort needed to be a mentor can make formal mentorship programs impractical [21]. Mentors need to devote time and effort to guide newcomers, and themselves face a variety of challenges, such as mismatches between mentor-mentee's background and interests, difficulty finding tasks that match newcomers' skills and the project timeline, lack of time, etc [4]. Mentors can also find it challenging to keep the mentee engaged, especially if the project culture is harsh or the mentees are not proactive/intensely interested in OSS [5]. The additional effort reduces the technical output of mentors in OSS [20]. Further,

Labuschagne and Holmes [50] found that, although mentees valued mentorship programs, these often do not result in long-term retention of contributors.

Outside of such formal mentoring, contributors actively seek guidance and support from each other via informal channels such as direct contact through emails and video conferencing or meeting at conferences. However, such non-code contributions are rarely given the same acknowledgement as code contributions [86]. As a case in point, when interviewed, an experienced OSS mentor mentioned: "*there is no recognition, no kudos, no kind of positive reinforcement for them to continue being a mentor. So, usually, they are a mentor once and then they leave*" [P4[1]]. Mentoring as a side hobby, therefore, becomes difficult to sustain. As Bosu and Sultana [9] found that those who did lots of mentoring felt that as a result of their community service they "lost their engineering voice".

One way to overcome this issue is to acknowledge (and promote) informal mentoring that can implicitly occur in everyday technical activities, such as code reviews. It is well known that code reviews are not always supportive. Past work has found that the ratio of negative sentiments is higher than that of positive sentiments in code review comments [62], and that destructive criticism, negative feedback that is nonspecific and inconsiderate, is fairly common [34]. Thus, acknowledging those who take the additional effort to mentor by providing constructive feedback and explanation when suggesting changes or improvement is worthwhile. Such recognition is important not only to encourage "implicit" mentors, but also to sustain this mode of mentoring, which potentially: (1) requires less effort than that needed for a dedicated mentor-mentee relationship, (2) is topical and aligned with the mentor's technical interests, (3) is part of the mentor's development activity, and (4) allows both the mentors and mentees to "learn on the job".

To the best of our knowledge, no one has yet investigated the prevalence and the impact of such a form of mentoring, prompting us to ask:

RQ1: How can mentoring be implicitly provided in everyday development tasks?

RQ2: How prevalent is "implicit mentoring" in OSS projects?

RQ3: How does implicit mentoring impact OSS contributors–mentors and mentees?

Through a formative mixed-method study of literature review and interviews with five senior OSS developers, we defined the concept of implicit mentoring as a type of informal mentoring, which we then validated through member checking (RQ1). We then empirically investigated the occurrence of implicit mentoring via Pull Requests (PRs) in 37 Apache projects (RQ2). We opted to investigate Apache projects because these projects have well-documented discussion procedures. Apache projects follow the principles of open communication logging discussion online [25]. We opted to analyze PRs, as they are used in many scenarios beyond basic patch submission (e.g., PR-comments can be used to conduct code reviews, discuss new features [33], and provide feedback [44]). Then, we used Machine Learning classification (Random Forest) to identify the prevalence of implicit mentoring in these projects automatically.

We investigated the impact of implicit mentoring (RQ3) through a quantitative analysis of developers' contributions in our dataset of 37 Apache projects. We then surveyed 231 developers who had contributed to these projects to triangulate our results.

The significance of our contribution is threefold: (1) define implicit mentoring in the context of OSS, (2) automatically identify implicit mentoring from PR-comments, and (3) analyze the impact of implicit mentoring on OSS contributors. Our results lay the foundation for future research on supporting informal mentoring, as well as provide guidance on how OSS communities can create an appreciative community. As a survey participant aptly noted: "*Getting feedback from experts in a project you are interested in is priceless. The only drawback is that those experts are often strapped for time and can't devote a lot of attention...It's frustrating that [mentoring] is not recognized or rewarded at all*" [ASF-104[2]].

2 RELATED WORK

Mentoring has been extensively researched in various domains. In management and organizational literature, works have investigated the extent to which mentoring helps with organizational citizenship behaviors (defined as positive employee attitude to the organization) [18]. Payne and Huffman [63] investigated the relationship between mentoring and positive organizational attitudes and found it to have a strong association with effective commitment (employee's emotional attachment or identification with the organization) and continuance commitment with the organization. In addition, Fagerholm et al. [21] investigated how mentoring and project factors affect the onboarding process in OSS. In education literature, Mullen and Klimaitis [59] conducted a literature review of empirical studies on mentoring to identify other forms in addition to formal mentoring. These included student mentoring done in groups, among peers, and in collaborative or cross-cultural forms.

A primary advantage of mentoring to the mentee is the transfer of knowledge and subject matter [19]. In education literature, mentoring is one of the most effective strategies for mentees to improve technical skills centered on a complicated job. By completing daily tasks alongside professional mentors, mentees develop cognitive processes that aid in problem-solving and show the validity of their understanding [13]. In addition, when used in conjunction with real-world work, mentoring aids in the development of communication skills such as explaining, persuading, bargaining, and establishing understanding [54]. Mentoring may also help mentees to build strong personal connections. For example, mentees may grow to trust and respect their mentors as they work together on goals [17].

LaFleur and White [51] found that mentees and mentors have similar experiences and perceptions. Mentors may also benefit from learning new skills from their mentees, such as those related to emerging technologies [49]. Mentees can help their mentors improve their job performance by providing new perspectives and knowledge [19]. In addition, mentors may feel a sense of generative or immortality as they watch their mentees grow [71]. Mentors gain personal satisfaction from observing and participating in the success of their mentees [48].

[1] P(N) refers to interview participant number.

[2] ASF-N refers to survey participant number.

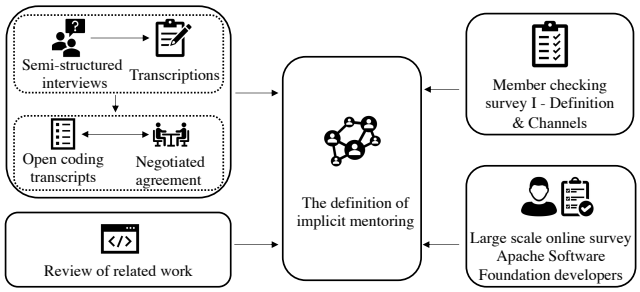

Figure 1: Overview of research method (RQ1).

In the context of research in OSS, most research only focuses on formal mentoring, especially onboarding activities. Researchers had found that mentoring allowed for a more effective onboarding experience than when newcomers entered a project through a natural, non-deliberate process [20, 21]. Google, through its formal Google Summer of Code (GSoC) mentorship program, aims to facilitate onboarding to OSS [40]. However, several works have identified the challenges that contributors face in OSS [4, 5].

Another form of mentoring is informal mentoring. Informal mentoring is a valuable tool for employee grooming because it occurs in a relationship that is voluntary and created by both parties [39]. According to Inzer and Crawford [39], informal mentoring occurs in a relationship between two people where one gains insight, knowledge, wisdom, friendship, and support from the other. Nandi and Mandernach [60] found that students improve significantly when they are mentored informally by analyzing student academic records and source-code commit logs. In addition, studies have found that formal mentoring is less effective than informal mentoring [39, 60]. However, to the best of our knowledge, no prior work has explored the benefits of participating in informal mentoring in OSS. Our paper fills this gap in understanding by investigating one form of informal mentoring, i.e., implicit mentoring.

3 IMPLICIT MENTORING IN OSS

Our goal was to identify implicit mentoring in OSS. In the following subsections, we detail the applied methodology and the results.

3.1 Methodology

We start our investigation of implicit mentoring by exploring how informal mentoring takes place in OSS (RQ1). We used a formative mixed-method study design, where we first reviewed literature on mentoring, following up with semi-structured interviews with OSS contributors and member checking the results of our analysis. We then surveyed 231 Apache Software Foundation developers to triangulate our results. Figure 1 shows an overview of the methodology.

Literature review: We surveyed existing literature in the context of mentoring in OSS to understand and contextualize existing work on this topic [43]. We first conducted a pilot search on two well-known digital libraries, IEEE and ACM, to identify optimal keywords. This helped us identify relevant words used in mentoring literature in Software Engineering, especially in OSS. Our final list of search keywords included: "mentor", "mentee", "mentoring". Our initial search resulted in 54 publications. Then, the first and second authors read the titles and abstracts and only selected papers discussing mentoring in the abstract (N=17 papers). In case our search criteria resulted in leaving out some relevant studies, we performed a single iteration of backward snowballing [91] looking for additional studies published in journals and other conferences, as suggested by Keele et al. [42]. This resulted in six further articles in the educational literature outlining the extensive definition, effects, implications of mentoring. Our final list included 23 papers. By reviewing these publications, we gained a comprehensive understanding of the different flavors of mentoring, including its types, channels, activities, and challenges.

Table 1: Demographic information of interview participants.

			Mentoring experience			
ID	**Gender**	**OSS experiences**	**Mentor**	**Informal /Formal**	**Mentee**	**Informal /Formal**
P1	Woman	Over 10 years	Y	Both	Y	Both
P2	Man	Over 10 years	Y	Both	Y	Both
P3	Woman	6-10 years	Y	Both	Y	Both
P4	Woman	6-10 years	Y	Both	Y	Both
P5	Woman	6-10 years	Y	Both	Y	Both

Formative Interviews: We then conducted semi-structured interviews with five OSS developers to learn about their experiences regarding mentoring—both from the perspective of being a mentor and a mentee. We recruited two developers, one from the Apache Software Foundation and another from the Linux Foundation, both of whom have spearheaded mentoring programs in OSS. We then used snowball sampling method to recruit three additional participants (two from the Linux Foundation, and one from the Apache Software Foundation). We stopped after five interviews, since we reached saturation—participants' responses did not provide any new details about the mentoring channels, activities, or challenges that we already identified from the literature review.

The interviews were done remotely, lasting around 30 minutes each. Each interview was recorded with participants' consent (following university-approved IRB protocol) and transcribed. Participants were offered a $50 gift card as compensation for their time. Table 1 presents the demographic information of our participants. During the interview, we asked participants questions about their roles and experience in OSS, their mentoring experience, especially that happening in daily development activities, and their opinions regarding formal and informal mentoring in OSS. See supplementary [84] for interview questions.

We performed and analyzed the interviews incrementally, that is, after each interview we transcribed and analyzed the data. We qualitatively analyzed the transcripts, following the open coding protocol [30]. Two researchers performed the analysis by independently coding the transcript resulting in codes regarding types of mentoring and challenges, how and where mentoring was provided, and recommendations for mentoring programs. All five participants referred to situations where an OSS contributor provided detailed technical guidance and support when reviewing technical contributions, which helped them learn. These comments served as the base for our definition of "implicit mentoring". When doing the analysis, each emerging code was compared with the existing codes to determine if the emerging code was a discrete category or a subset of an existing code. We carried out the whole procedure

via continuous comparison throughout the coding sessions and through negotiated agreement [23]. Researchers discussed the rationale for applying specific codes and reached a consensus during the negotiated agreement process. Two authors agreed on the (1) definition of conventional formal mentoring in OSS, (2) definition of implicit mentoring in OSS, (3) the challenges of mentoring in OSS, (4) the activities of formal mentoring, (5) activities of implicit mentoring, (6) frequency of implicit mentoring, (7) benefits of implicit mentoring.

Member checking survey I: To validate our understanding of implicit mentoring that we attained through literature review and formative interviews, we conducted a member checking survey. We contacted each of our interview participants through email and conducted a face-to-face survey through a teleconference tool (Zoom). The survey (created in Qualtrics [70]) included three demographics questions, three questions about what constitutes implicit mentoring (e.g., Please identify activities you consider a part of mentoring), and where/when it occurs (e.g., Please identify where implicit mentoring takes place). Participants could either fill out their responses or verbalize their thoughts, a majority chose the latter. See supplementary [84] for survey questions.

Large scale online survey: To acquire a more generalized perception about implicit mentoring and further validate our definition, we surveyed a larger population of OSS developers. Specifically, we surveyed OSS developers who had contributed to Apache projects. We selected Apache developers since Apache Software Foundation is dedicated to promoting and enhancing mentorship in OSS. Additionally, Apache has explicit guidelines for mentors that contributors have to adhere to [24], making Apache developers a suitable target for our survey.

Survey design: Our survey comprised 19 questions, a mix of multiple-choice, Likert scale, and open-ended questions (see supplementary for the survey questions [84]). The survey included demographics questions (Q1-Q5), validation questions of implicit mentoring definitions (Q6, Q9), participant experiences/satisfactions with it (Q7-Q8, Q13-Q14, Q18-Q19), and finally questions about participants' perceptions of the impact of implicit mentoring (Q10-Q12, Q15-Q17). We conducted five pilot studies with graduate students and professionals with OSS experience using snowball sampling [31]. After each pilot study, we collected feedback and refined the survey based on the feedback.

Participant selection: We focused on software developers who had contributed to PRs in Apache projects. We decided to focus on Apache for several reasons. First, Apache Software Foundation oversees one of the most popular formal mentorship programs (Google Summer of Code [78]), which would mean that the philosophy of mentoring is instilled in Apache contributors. Second, Apache projects have well-defined policies and a philosophy of transparency where all discussions should be conducted in online forums. This is essential for us to extract implicit mentoring from archived communication. Finally, Apache projects are often studied in scientific research, which allows our work to be placed in the context of existing research [11, 29, 41]. Mannan et al. [55] curated a set of 37 active Apache projects that used PR-comments as a discussion channel. Therefore, we decided to use these projects as our "test subject". We used the GitHub API [58] to mine contributor emails from these 37 projects. After removing emails of accounts that were either deleted or private, we were left with 3,699 developer email addresses in total.

Survey responses: We used Qualtrics [70] as a distribution platform to deploy our survey. We emailed the survey to these 3,699 developers (following university-approved IRB protocol), and 70 emails bounced (giving 3,629 valid emails). The survey was open for two weeks, during which we received 231 responses or a response rate of 6.37%. These response rates are consistent with other studies in software engineering [90].

Survey data analysis: In our survey, 219 out of 231 survey respondents identified as men (94.81%), and the majority age group was 25 to 34 years old (44.59%). Most of our respondents had a master's degree (42.42%). 29.87% were senior contributors with over 20 years of programming experience. In addition, over half (54.98%) of our respondents had within 2 to 10 years of open-source experience.

We quantitatively analyzed the closed-ended questions to understand developers' perceptions of the definition and impact of implicit mentoring. We used a card sorting method [98] to conduct a qualitative analysis of the responses to the open-ended questions in our survey. Two researchers used a two-step card sorting process. Each of them evaluated the responses (cards) and categorized them according to their codes in the first step. Then these two researchers met to discuss each code and categories. After this, the responses were categorized into high-level groups. The researchers next investigated the categories to refine them and organize them into more useful groups and themes.

3.2 Results

Our results show that in OSS both formal and informal mentoring take place.

Formal mentoring happens where mentors and mentees are formally connected either through scholarships or mentorship programs [4, 21, 82]. However, despite the formal programs having guidance and financial support, it is difficult for mentees to figure out how to seek guidance [4], or follow up with their mentors: "*There was no continuous progress of mentoring, we didn't know how to follow up on things*" [P4] [4, 80, 81]. Sustaining the mentor-mentee relationship is further difficult because of: (a) diverging interests: "*...interests tend to diverge, you tend to look for new mentors, new areas, and new people to serve*" [P5] [4, 5], (b) limited resources: "*if the mentee leaves...a more likely outcome. Am I going to be left with code to maintain that I might have been better for me to write in the first place*" [P5] [78], (c) difficulty in seeking guidance: "*I really got overwhelmed with all the information...I didn't have much idea what to ask for my mentor to guide me with*" [P4] [4], and (d) cost-benefit assessment: "*It's hard to assess...whether the return on investment is going to be effective*" [P2] [77].

Informal mentoring, perhaps because of the above challenges, can be more effective than formal mentoring [39, 60]. Informal mentoring is "interest-driven" when a mentor or mentee reaches out to the other to seek/give guidance in a particular area. Informal mentoring occurs frequently: "*I have always had mentors, all informal mentors because I chose to learn from them*" [P1] [39]. These informal mentoring included cases where the mentee or mentor reaches out to the other to ask for guidance.

Table 2: Mentoring in OSS and its characteristics.

		Provenance	
		Interview	**Related Work**
Mentoring Tasks	Suggestions	P1, P3, P5	[5, 20, 86, 97]
	Instructions	P2, P3, P4, P5	[5, 20, 86, 97]
	Mechanisms to fix errors	P1,P3, P5	[5, 9, 20, 97]
Mentoring Channels	Email, Slack, Skype, etc	P1, P3, P4, P5	[5, 21, 97]
	Code review comments	P1, P3, P4	[5, 9, 20]
	In person/Remote	P4, P5	[4, 21]

Implicit mentoring, a form of informal mentoring where training was "implicitly" provided as a part of contributors' technical activities, such as code review: "*When somebody reviews a patch, that gives a feedback to you, that's a form of mentoring*" [P1]. We specifically define implicit mentoring as: "mentoring that occurs in everyday development activities such as code reviews, where a mentor provides an underlying explanation when providing suggestions, instructions, or mechanisms to address errors". As P2 described: "*what you do in your **day-to-day activities** where you mentor...teachable moments, where you **explain** why you are doing certain things. So you're essentially communicating knowledge about the system that goes beyond the knowledge necessary to act on the particular item that you have*". Such mentoring can be through multiple channels: emails, PR-comments, in person meetings, or through online communication tools. As P3 commented, "*There are going to be mentors who are more like code reviewers or the design reviewers...mentorship that happens every single day*". A caveat, not all code reviews include mentoring. Gunawardena et al. [34] found that destructive criticism, where changes are requested without explanations, is common and negatively impacted the receiving developer and lowered their willingness to continue to work.

To validate our definition of implicit mentoring, we asked large scale online survey participants to confirm if they agreed with the above definition of implicit mentoring. The question had five options ranging from: "Strongly agree" to "Strongly disagree". We considered a respondent to agree with the definition, if they selected "Agree" or "Strongly agree". A majority (214 out of 231, 92.64%) agreed with our definition of implicit mentoring in OSS.

PR-comment(s) was a popular choice (selected 27.78% times) as a channel for implicit mentoring, followed by issue-list comments (18.62%), in-person conversation (15.32%), Email (15.17%), or comments on Stack-Overflow posts (14.11%). Note, a participant could select multiple channels. Respondents also identified other channels (39 entries) such as, "Instant Messaging (Slack or Skype)" (17 votes), "video meeting (Zoom)" (6 votes), and "Reddit"(2 votes). One of the respondents mentioned, "*Every single type of interaction is an opportunity for mentoring*" [ASF-150]. Therefore, mentoring need not always be formal and can be done during contributors' everyday activities, such as code review: "*When somebody reviews a patch, that gives a feedback to you, that's a form of mentoring*" [P1]. Table 2 presents the different aspects of mentoring our interviews and literature survey identified, along with the channels where mentors can "teach" in an OSS project.

Observation 1: Implicit mentoring occurs in OSS projects as a part of everyday development activities through code reviews (e.g., PR-comments).

Satisfaction with implicit mentoring: *Perspectives of implicit mentees:* Analysis from our survey indicated that 183 out of 231 (79.22%) respondents confirmed they have received implicit mentoring, and over 68.31% of these contributors claimed they have been implicit mentees for at least two years or more. Among these implicit mentees, 86.34% contributors were satisfied with their experiences ("Extremely satisfied" 15.85%, "Satisfied" 49.18%, "Somewhat satisfied" 21.31%).

Participants could give the reasons for their satisfaction rating through an open-ended text; 46 participants provided did so. 24 of them mentioned that implicit mentoring helped them in their professional development. One respondent mentioned, "*Being an implicit mentee allowed me to learn so much from talented folks, both soft and technical skills*" [ASF-94]. Six respondents addressed that being an implicit mentee is a way to contribute to the community, "*Being an implicit mentee is satisfying to me because it makes you feel like you belong to a caring community, and you do not have to do it alone*" [ASF-69]. Two respondents wrote about the reasons for their dissatisfaction, including "*public shame*" [ASF-121] and "*communication skills/issues*" [ASF-129].

Perspectives of implicit mentors: Analysis from our survey showed that, 175 out of 231 (75.76%) respondents confirmed they have being implicit mentors. 68.57% of them claimed to being implicit mentors for over two years or more. Among these implicit mentors, 72.00% were satisfied with their mentoring experiences. The reasons for their satisfaction included, making a contribution to the project (4), a way to improve their technical skills (5), and altruism (9). For example, respondent mentioned: "*I never get tired of seeing 'the lights come on' when the mentee 'gets it'*" [ASF-140]. One respondent reported the reason for their dissatisfaction: "*I think it has helped me grow substantially as an engineer, but it is frustrating that it is not recognized or rewarded at all by my employer*" [ASF-104].

Observation 2: A majority of respondents have either been an implicit mentor or mentee, and are satisfied with their experiences; mentee satisfaction (86.34%) and mentor satisfaction (72.00%).

4 PREVALENCE AND IMPACT OF IMPLICIT MENTORING IN OSS

We investigated the prevalence of implicit mentoring (RQ2) and its impact (RQ3) on OSS contributors using a mixed-methods approach (see Figure 2). We began by developing an automated machine learning classifier to identify instances of implicit mentorship in PR-comments. Following that, we investigated the impact of implicit mentoring by analyzing the number of PRs submitted by developers, the number of mentoring comments contributors received, and developers' contribution complexity. We also analyzed the responses from the contributor survey (described in Section 3) to gain an understanding of the benefits of implicit mentoring from OSS contributors' perspectives (Q10-Q12, Q15-Q17) and to triangulate our findings from the mining analysis.

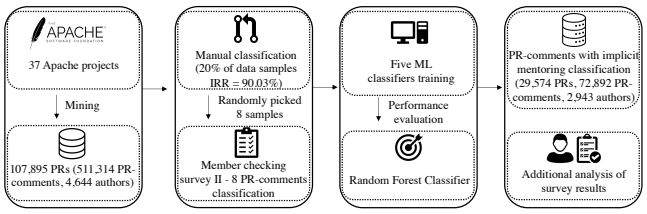

Figure 2: Overview of research method for investigating prevalence and impact of implicit mentoring.

4.1 Methodology

We start by selecting the 37 Apache projects from the list that was curated by Mannan et al. [55], as explained in Section 3. This dataset comprised 107,990 PRs with 836,729 PR-comments, logged by 12,668 contributors. As our analysis required GitHub profile data, we removed contributions of 42 user profiles whose GitHub accounts were deleted at the time of data collection. Additionally, we filtered out the PR-comments made by the PR author as we were interested in mentoring comments made by others. These steps resulted in 107,895 PRs with 511,314 PR-comments, and 12,626 contributors (out of which 4,644 contributors were PR-comment authors). See supplementary [84] for further details about each project in the dataset.

Manual Classification of Training Data: Our goal was to identify implicit mentoring instances. Since manual classification of 511, 314 PR-comments is not a practical option, we decided to use machine learning. To train the machine learning classifier, we manually classified a training set. We determined the size of this training dataset by using a 95% confidence interval and a margin of error of 5% [46] on the dataset, giving us a sample size of 384 PR-comments. We then did a random selection of 384 PR-comments to create the training set.

Next, the first two authors manually labeled a subset of the PR-comments in the training set with binary labels based on whether the PR-comment included implicit mentoring (or not). When labeling, they followed the definition of implicit mentoring introduced in Section 3: a PR-comment was labeled as implicit mentoring if it included an "explanation" in addition to giving suggestions, instructions, or helping fix errors. Table 3 shows the rule book with examples of both positive and negative cases of PR-comments including implicit mentoring (See supplementary [84] for more). The authors selected 20% from training dataset through random selection to calculate Inter-Rater Reliability (IRR) [36], which resulted in a high agreement (90.03% Cohen Kappa [88]). The remaining 80% of the training dataset was split evenly between the two authors who individually classified the PR-comments.

Member Checking Survey II: To ensure the validity of our manual classification, we contacted the five interviewees (Section 3) to validate our PR-comment labeling. We randomly selected eight PR-comments (four implicit mentoring, four not implicit mentoring) from the manually classified set of 384 PR-comments. The survey was created in Qualtrics [70] and conducted face-to-face using Zoom [3]. The survey presented the PR-comment and included three response options: Mentoring, Not mentoring, and Not sure. For the first two options, participants were asked to give a reason for their response. As before, participants could either type in or verbalize their answers (See supplementary [84] for further details about Member Checking Survey II).

Table 3: Classification rule book.

Mentoring	PR-comment sample	Mentoring Action
YES	"...run [tool] on the project before creating a PR. You would have noticed [problem]..."	Instruction
YES	" I would still duplicate [action] like I did in [certain PR] because it's widely used in [tests]. Maybe this could be removed after [situation]."	Suggestion
YES	" Would you mind just doing [action] again to kick off [framework]? I think [framework] is just not happy when it has a lot of loads."	Mechanisms to fix errors
NO	"LGTM, Merging into master and release-[version number]"	NA
NO	"This PR currently has merge conflicts, but [#PR] is next in line, so you may want to wait till it is merged before you fix these conflicts."	NA
NO	"This is not ready. Missing apache header on the new file and there is no test. No idea what this is fixing."	NA

The member checking survey responses were 90% in agreement with our classification (the remaining 10% was when participants selected "Not Sure"). This suggests that our manual classification is reliable. Moreover, their explanations in the open-ended replies indicate that our rule book is reliable. For example, P3 explained why a PR-comment was mentoring: "*This is helpful feedback in explaining that these changes are of low value*" And P4 explained why a PR-comment was not mentoring: "*This is asking for information [and not mentoring]*".

Training Machine Learning Classifier: Using the manually classified training data, we trained five common supervised machine learning classifiers as they have been used in similar studies for classifying discussion comments [10, 55]. They were: Random Forest [61], Bernoulli [79], Support Vector Machine [35], KNeighbors [65], and Decision tree [45]. Our training data consisted of the comments in the PR. We used the usual text cleaning steps which included using Porter's stemming [66]. We eliminated all of the terms from the standard stop word list in order to do suffix stripping. To ensure the best performance, we applied hyper-parameter adjustments from the Python Scikit learn library [64] to all five classifiers. By applying *ScikitLearnRandomizedSearchCV*, we found the optimal parameters for each classifier. The models were trained and evaluated using a 10-fold cross validation methodology. That is, the data was randomly divided into 10 equal sets, and nine sets were used for training and one for testing. We trained our model using this method 10 times and report the mean scores.

Table 4 shows the precision, recall, F1, and AUC scores of the five classifiers [67]. Random Forest Classifier (RFC) had the best overall performance when considering both the F-measure (0.88) as well as the AUC scores (0.94). Therefore, we used RFC for further analysis. The final tuned parameters for our classifier RFC were $n\_estimators$=2800, $max\_features$=*auto*, max_depth=73, $min\_samples\_split$=20,$min\_samples\_leaf$=2, and $bootstrap$=*True*.

Manual Analysis of Misclassification: Any systemic error from misclassified PR-comments could be critical to its acceptance. We investigated the confusion matrix [89] of our RFC model as shown in Figure 3. We can infer from the confusion matrix that

Table 4: Classification results per classifier.

	Precision	Recall	F1	AUC
RandomForest	0.87	0.90	0.88	0.94
Support Vector Machine	0.84	0.84	0.84	0.91
NaiveBayes	0.81	0.79	0.78	0.90
DecisionTree	0.74	0.74	0.73	0.81
K-neighbors	0.72	0.62	0.57	0.71

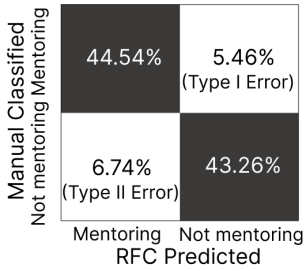

Figure 3: Confusion Matrix.

neither class (Mentoring, Not-mentoring) is disproportionately impacted by the RFC's misclassifications.

Impact of Implicit Mentoring–data mining: We used three different metrics to investigate the impact of implicit mentoring.

(i) Reduction in guidance: Prior research has identified several benefits of mentoring. For example, mentees require less and less guidance to accomplish their goals over time [72]. We wanted to check if this holds true for implicit mentoring as well. As OSS contributors are mostly voluntary and can have sporadic contributions, we eschew using time as a metric but amount of contributions. We started by sorting the PRs for each contributor within projects based on the time when it was created and calculating the number of mentoring comments received per PR. Next, we grouped the PRs based on when the first mentoring comment was received. The number of comments for a PR can vary depending on the complexity of the PR, and outliers can influence the statistical analysis. So we used a normalized value of the mentoring ratio (number of implicit mentoring comments/number of comments) instead of the number of mentoring comments. We used Pearson correlation [6] test to check if the mentoring ratio decreased with each progressing PR.

(ii) Contribution complexity: Both mentors and mentees benefit in their technical knowledge and job performance gains as they help each other [19, 76, 78]. "True mastery" is defined by Zhou and Mockus [96] as an indicator of a developer's ability to perform complex tasks. We posit that with increased technical knowledge, contributors will work on more central and complex parts of the project. The complexity of contribution can be measured using (a) code metrics and (b) centrality metrics.

(a) Code metrics such as, Lines Of Code (LOC) and McCabe cyclomatic complexity [56] are commonly used to measure the complexity of source code [92, 94]. We used Understand [87] to compute LOC, McCabe cyclomatic complexity.

(b) Centrality metrics focus on the relationship between files in a project. Ahmed et al. [1] mentioned that the more times a file is referenced by other files, the more central and hence more important that file is. We used degree centrality and eigenvector centrality to measure the centrality of each file in the project. Degree centrality quantifies the number of connections a file has with other files [26]. Eigenvector centrality, on the other hand, calculates a file's relevance in a network and indicates the impact of a file on the entire project [68]. We used Understand [87] to compute the number of references (in and out dependencies) for each file associated with a PR. We used Igraph library [15] to compute the degree centrality and eigenvector centrality from this data.

As the contribution complexity measures are per file, we have to extract which files were changed for which PR (and by whom). To do so, we first identified the commits made by a PR-author. Note, a single PR can include multiple commits, and a single commit can change multiple files. Next, we used Git to identify the authors' names and files committed per PR. We analyzed total of 292,252 files in 11,551 PRs by 1,284 PR-authors.

(iii) *Increase in productivity*: Prior research has shown that mentors also benefit from mentoring by "continuous learning" by supporting others, which results in increased productivity [7]. We wanted to check if the same holds for implicit mentoring. We use the number of PRs submitted by contributors as a proxy for productivity.

Impact of implicit mentoring–contributor survey: Recall that in Section 3 we deployed a survey to 231 Apache contributors. A goal of this survey was to collect developers' perceptions about the impact of implicit mentoring, both as a mentor and a mentee. The survey (Qualtrics) included closed-form questions listing a set of benefits, which we created by reviewing the related work on mentoring. The related work was identified as described in Section 3. The first two authors individually reviewed the related work about mentoring and its benefits by analyzing the abstract and result sections, extracting the benefits listed. Through negotiated agreement they finalized the list of benefits, which were then categorized into four themes, using open card sorting [98]. Table 5 shows the potential benefits for being OSS implicit mentors/mentees that were included in the survey. We also included a "Text entry" option to allow developers to provide additional perceived benefits, both per theme as well as overall.

4.2 Results

Prevalence of implicit mentoring in OSS projects: We applied the RFC classifier on our dataset to investigate the prevalence of implicit mentoring in PR-comments. In our dataset, on average, a PR had 4.74 PR-comments (sd = 8.81). However, not all comments in a PR are mentoring, specifically implicit mentoring. So we checked what percent of the PR-comments are actually implicit mentoring. Our results show that, 27.41% of PRs had at least one implicit mentoring comment. These comments were made by 2,943 out of 4,644 PR-comment authors (63.37%). This shows that contributors routinely participate in implicit mentoring. This finding is consistent with our survey results as we present in Section 3.

Observation 3: Implicit mentoring occurs in 27.41% of PRs, with a majority of PR-comment contributors having served as implicit mentors.

Table 5: Potential benefits for being an implicit mentee/mentor as identified from related work.

	Mentee	Mentor
Personal Growth	1. Improving my technical expertise [28, 78]	1. Improving my technical expertise [28, 38, 85]
	2. Improving my ability to transfer knowledge [93]	2. Developing confidence in leading/managing [73]
	3. Improving the art of asking questions [22]	3. Improving my other *non-technical skills [38, 53, 73]
	4. Improving my communication skills [22, 76]	
	5. Improving my other *non-technical skills [5, 76]	
Career Growth	1. Increasing my chances of promotion/higher salary [28]	1. Increasing my chances of promotion/higher salary [28]
	2. Learning experiences beneficial to my job/organization [75]	2. Improving my reputation [28, 75]
	3. Helping me grow my career [28, 78]	3. Improving my project's reputation [75]
Social skills	1. Creating my collaboration network [27, 28, 76]	1. Identifying potential collaborators [2, 16, 28]
	2. Identifying people who are domain experts [28, 78]	2. Identifying other mentors [2, 16]
	3. Identifying people who are programming experts [28, 78]	3. Identifying potential contributors that can lead to career opportunities [16, 28]
Altruism	1. Enjoying sharing experience [28, 76]	1. Receiving gratitude/appreciation from implicit mentees [2, 16, 28]
	2. Building friendship [27, 28, 78]	2. Improving my ability to impart knowledge and experience [2, 28, 85]
		3. Enjoying observing the growth of implicit mentees [2, 28]
		4. Developing long-term friendship [16, 27, 28]

*non-technical skills: critical thinking, problem-solving, public speaking, professional writing, teamwork, digital literacy, leadership

Impact of Implicit Mentoring on Mentees: We analyzed the impact on mentees using the first two metrics: (i) reduction in guidance, and (ii) contribution complexity.

(i) Reduction in guidance: One of our goals was to test the hypothesis ($H1$) that there is a correlation between amount of contributions spent on the project and amount of guidance (implicit mentoring comments for contributors). We sorted contributions (PRs) by time for each contributor instead of time as a metric due to the fact that OSS contributors are mostly volunteers and can make contributions irregularly. Pearson correlation test between mentoring ratio (Number-mentoring-comments/ Number-comments) and sorted contributions (PR) by time in the project indicates that as time progresses, the mentoring ratio decreases (Pearson correlation coefficient = -0.16, P value < 0.001). It is possible that the number of comments are reduced anyway in the project due to attrition of contributors, which can be a confounding factor for our analysis. So we checked the number of comments in the projects over time and found that it increases over time (Pearson correlation coefficient = 0.21, P value < 0.001).

(ii) Contribution complexity: The following hypothesis is to evaluate the different complexity of code metrics ($H2$) or centrality metrics ($H3$) submitted by implicit mentees and non-mentees in the project. We split the PRs into two groups: the most complex PRs submitted by contributors who at least received one mentoring comment and the most complex PRs submitted by contributors who did not receive mentoring comment in the project. Past works have shown mentoring improves technical knowledge, so we used PR complexity as a proxy for analysis [28, 78]. As explained in Section 4.1, we measured complexity using two different code metrics. When measured using LOC, contributors who received at least one implicit mentoring comment wrote more LOC as compared to those who did not receive implicit mentoring comment (Welch's Two Sample t-test, t =7.14, df = 1,022, P value < 0.001). The effect size was small (Cohen d = 0.36) [12]. We see similar results for cyclomatic complexity as well. Contributors who received at least one implicit mentoring comment created code with higher cyclomatic complexity compared to those who did not receive implicit mentoring comment (Welch's Two Sample t-test, t =8.06, df = 975, P value < 0.001). The effect size was medium (Cohen $d = 0.41$).

For testing Hypothesis ($H3$), we used two centrality measures: eigenvector centrality and degree centrality. Our results show that, the eigenvector centrality of contributions from contributors who received at least one implicit mentoring comment is higher compared to those who did not receive implicit mentoring comment (Welch's Two Sample t-test, $t = 7.63$, $df = 1,080$, P value < 0.001). The effect size was medium (Cohen $d = 0.38$). For degree centrality, we see similar results, degree centrality of contribution from contributors who received at least one implicit mentoring comment is higher compared to those who did not receive implicit mentoring comment (Welch's Two Sample t-test, t =5.69, df = 1,095, P value < 0.001) with a small effect size (Cohen $d = 0.29$).

Findings from the survey corroborate these results. As Figure 4 (left pane) shows, the majority of participants believe that being an implicit mentee aids in personal development, especially in terms of "improving [their] technical skills" (98.31% respondents when combining "Strongly agree" and "Agree" options).

The survey results also highlight additional benefits, those that cannot be captured from mining repositories. Respondents mentioned that implicit mentoring helps develop technical skill as well as non-technical skills such as, network building and communication. As one respondent mentioned, "*Contributing to an unfamiliar codebase provides useful experience operating outside of your comfort zone and encourages intellectual humility. This translates to more effective working relationships in a professional setting*" [ASF-47]. Another benefit of implicit mentoring is the ability to share experiences. "*Being an implicit mentee is satisfying to me because it makes you feel like you belong to a caring community, and you do not have to do it alone*" [ASF-69].

Observation 4: Mentees benefit from implicit mentorship in terms of both technical and non-technical abilities, as well as career advancement.

Impact of Implicit Mentoring on Mentors: An impact on mentors can be increased productivity for mentors because of their implicit mentoring actions (Hypothesis $H4$). We analyzed the differences between the number of PRs submitted by contributors

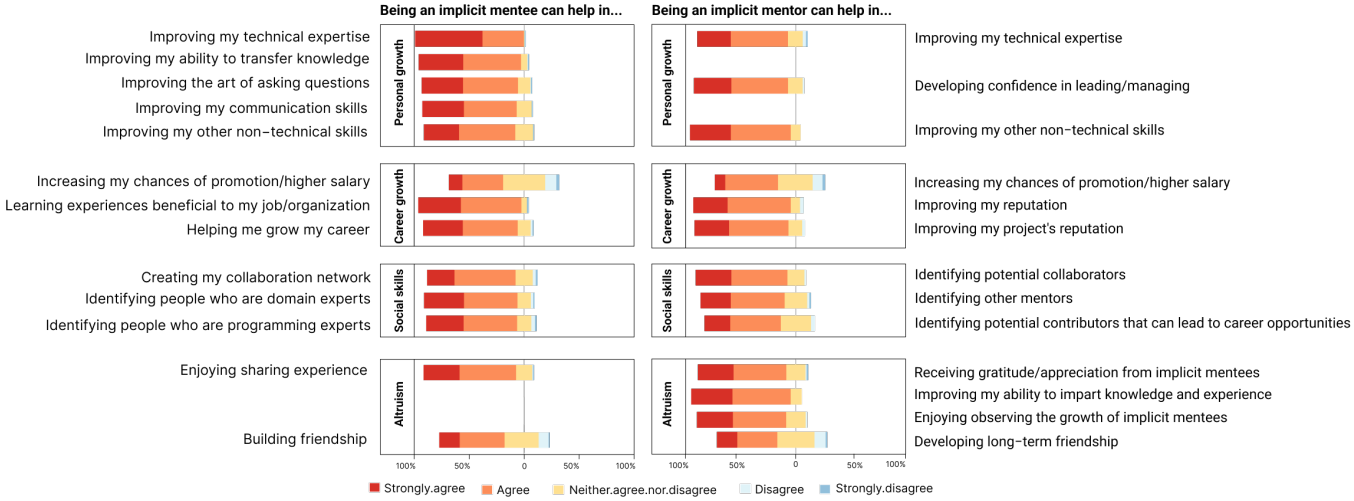

Figure 4: Potential benefits for being: an implicit mentee (left pane) or a mentor (right pane) in OSS.

who were mentee-only (at least received one implicit mentoring comment) and those who served both roles (at least received and gave one implicit mentoring comment). Welch's Two Sample t-test, t = 11.16, df = 1,767, P value < 0.001 (effect size using Cohens d = 0.44 (small)).

Hudson [38] mentioned that mentors can improve their own professions by participating in mentorship programs. We also evaluated the different complexity of code metrics and centrality metrics submitted by mentee only and contributors who have served both mentee and mentor in the project. To test this hypothesis ($H5$), we split the PRs into two groups: the most complex PRs submitted by contributors who were mentee-only and the most complex PRs submitted by contributors who have served both roles. When considering the LOC as a metric for code complexity, Welch's Two Sample t-test (t =7.21, df = 422, P value < 0.001) shows the differences between the two groups to be significant. The same holds true for cyclomatic complexity (Welch's Two Sample t-test, t =7.08, df = 532, P value < 0.001), eigenvector centrality (Welch's Two Sample t-test, t =6.12, df = 657, P value < 0.001) and degree centrality (Welch's Two Sample t-test, t =5.43, df = 491, P value < 0.001). The effect sizes are small to medium for all of metrics (LOC: Cohen d = 0.52; cyclomatic complexity: Cohen d = 0.51; eigenvector centrality: Cohen d = 0.44; degree centrality: Cohen d = 0.39).

As shown in Figure 4, right pane shows the result, which validates the quantitative results. Participants believed that being a mentor helps them improve their technical expertise (81.58% responded as "Strongly agree" or "Agree") and managerial skills (85.52% responded as "Strongly agree" or "Agree"). A reason can be that mentors continuously improve themselves as they teach. "*To be a good mentor, I can make an effort and keep my ability improving*" [ASF-152]. Participants also mentioned that mentoring has an impact on personal growth "*Being a mentor in OSS is all about sharing your experience, regardless of background. It is also tremendously self-rewarding as you gain popularity which helps career path evolution*" [ASF-94].

Table 6: Triangulation of the definition, channels, and labeling of code reviews with implicit mentoring.

Characteristics of implicit mentoring				
	Interview	**Member checking**	**Survey**	**Related work**
Definition	✓	✓	✓	✓
Channels	✓	✓	✓	✓
Implicit mentoring PR-comments labeling				
		IRR		**Member checking**
Classification		✓		✓

The survey results also revealed that being an implicit mentor can help in improving social skills and career growth. A majority of survey respondents mentioned that they did implicit mentoring because of altruism. "*I tend to find that people who take advice ultimately produce better results. If I can play a small part in that I am extremely satisfied*" [ASF-208].

Observation 5: Implicit mentoring helps mentors continuously improve their technical and non-technical skills as a result of assisting others.

5 DISCUSSION

5.1 Triangulation

Sections 3 and 4 alluded to a multiple-triangulation validation strategy involving (1) triangulating the definition of implicit mentoring via member checking with interview participants as well as a larger survey, and (2) triangulating our labeling of PR-comments with a short survey with interview participants. In this section, we bring these different kinds of triangulation together, and summarize it in Table 6.

As we are the first to define implicit mentoring, we triangulated our definition through multiple ways. First, we sought for consistency checking by comparing the open coding of the interview data, first within the research team (negotiated agreement between two researchers) and then via member checking with the interview

participants. Consistency checking is a kind of "internal validity" check to show if the authors' understanding of implicit mentoring matched that of (interview) participants. Next, we validated our definition through a large scale survey of Apache contributors, helping us get "external validity".

We triangulated the components of the mentoring tasks and their channels similarly. The literature review was the data source, and then we triangulated the results through interviews and member checking for internal validity, and then externally validated the results through the contributor survey.

Finally, we triangulated whether we labeled PR-comments with implicit mentoring correctly; we achieved internal consistency by using Inter-Rater Reliability between the first and second author. Then we got external validity by asking the interview participants (Member Checking-II) to flag which PR-comments they considered to include implicit mentoring (Recall, we included eight PR-comments, four of which were labeled to have contained implicit mentoring).

5.2 Creating an Appreciative Community

Research has shown that traditional mentoring takes effort, which reduces the technical productivity of mentors [5, 20]. Currently, mentors from implicit mentoring are unacknowledged and that causes mentors to disengage. Creating an appreciative community where mentoring activities are visible and mentors are acknowledged is an important consideration for OSS projects. As interviewee P4 said: "*there is no recognition, no kudos, no kind of positive reinforcement for them to continue being a mentor. So, usually, they are a mentor once and then they leave*".

OSS communities can experiment with various techniques for "baking in" mentor appreciation into their projects.

1 The code of conduct for OSS can explicitly state that mentors should be thanked for various efforts, including code reviews.
2 A lightweight mechanism to acknowledge mentors could be creating an attribution tag, such as @mentor to allow contributors to formally acknowledge mentoring they received when contributing.
3 OSS communities may utilize our technique to discover implicit mentorship in their projects, which they can then use to award "karma" points (or other non-code related activities). As one survey respondent explained their motivation for implicit mentoring: "*Getting karma in the project. Typically becoming a committer or PMC member*" [ASF-83].
4 OSS communities that host projects, such as GitHub, may also identify implicit mentors and include implicit mentoring into contributor profile pages.
5 OSS projects can emphasize the extent to which the project receives (implicit) mentorship by automatically identifying these instances. Previous research has identified markers that attract newcomers to a project, with a friendly community ranking high on the list [69].

5.3 Impact of Implicit Mentoring...

...on mentees: We examined the impact of implicit mentoring on mentees by investigating the relation between contribution complexity and implicit mentoring. Furthermore, we also surveyed the people who had been implicitly mentored to validate our findings. A majority of participants perceived implicit mentoring to be helpful in developing their technical and non-technical skills, as well as in their career growth. The survey results show that implicit mentoring helps mentees create their collaboration network and identify the domain experts. Social networks are important to create a sense of belonging, Bosu and Carver [8] found that informal discussions that occur through code review can help create friendships. It would be interesting to investigate how these networks are created and how they evolve over time, whether positions in these social networks correlate with reputation in the community and/or career growth, and whether these networks reflect organizational hierarchies (e.g., the Apache Software Foundation roles as PMC, chair, VP, board member).

...on mentors: was seen in improvement in their technical and non technical skills. A majority of mentors took on mentoring for its altruistic purposes, but a recurring problem was mentor "burnout" and work overload, making it difficult to retain mentors [5, 20, 86]. Since, by its very nature, implicit mentoring constitutes brief, topical interactions, it is possible that if OSS projects reward implicit mentoring, more mentors will participate. Many of mentoring benefits identified in formal mentorship programs were also seen as benefits in implicit mentoring, thus, acknowledging implicit mentoring, which requires less effort that dedicated mentor-mentee relationship, can help make mentoring sustainable in OSS projects. Mentor networks of individuals created by bringing together individuals who share topical interests, can help mentors learn from each other and identify potential collaborators.

...on projects: As one of the survey respondents addressed, "*Implicit mentoring generates benefits for the mentee, mentor, and their organization*" [ASF-205], having a mentor for purposeful practice is essential in software development efforts [37]. We classified the 85 open-ended answers to the question "why did you choose to be an implicit mentor?" into four groups. 42% of contributors expressed a desire to give back to the community and project, and 34% felt that mentoring is beneficial to the project's health, including quality, maintenance, and onboarding of newcomers. 13% of participants said they love assisting others, and 11% believe implicit mentoring occurs spontaneously. As such, more research would help answer questions about how implicit mentoring affects project quality, how implicit mentoring effects attracting newcomers, and how implicit mentoring affects keeping contributors to OSS projects.

6 THREATS TO VALIDITY

Our study, like any other empirical research, is not without its risks. We have taken all reasonable steps to mitigate the effect of these potential threats, which we will describe in detail below.

Construct validity: It is possible that some of the analyzed projects may have used other communication channels such as JIRA that we did not detect from the PR analysis. However, respondents to the survey confirmed that PR is the most popular channel they engaged in implicit mentoring, which reduces the risk of losing information due to only focusing on PR.

There is always a threat to the construct validity if participants misunderstand our survey questions. To mitigate this threat, we conducted pilot studies with developers with different experience

levels from OSS community. We updated the survey based on the feedback of these pilot studies.

Internal validity: The manual analysis applied while labeling PR-comments could have introduced unintentional bias. To address this concern, two researchers individually labeled a significant portion of the data. We established a high Inter-Rater Reliability of 90%, which, according to McHugh [57], is considered as a substantial level of agreement. We then took samples from manually categorized data and used a member checking survey to validate our classifications. In addition, the definition and channels of implicit mentoring that had been triangulated with related works, member checking surveys, and online surveys. We discuss our methods in depth in the methodology of Section 3 and 4. We believe these steps have minimized this threat.

The set of metrics used to measure contributions complexity are widely used in literature. However, other metrics could have been used for this purpose. Thus, our evaluation is not exhaustive, but we believe that the metrics we used provide a fair assessment of contributions' complexity.

External validity: We only covered PRs on Github Apache projects, but we obtained large samples. The dataset used for the study consisted of only Apache projects. It is possible that the conclusions from our analysis may not apply to other OSS projects. Similarly, we surveyed developers only from the Apache Software Foundation. The characteristics of these developers may not be representative of all developers in other OSS projects.

7 CONCLUSION

In this work, we define the concept of implicit mentoring, where a mentor provides mentoring in the form of suggestions, instructions, or mechanisms during everyday development activities such as code reviews. Through an empirical investigation, we also showed a widespread prevalence of implicit mentoring.

Our results highlighted that implicit mentoring is beneficial for both the mentor and mentee in terms of learning, engagement, and social network growth. However, given that a large amount of implicit mentoring happens on a regular basis, it mostly goes unnoticed and unacknowledged. The research community needs to focus on developing mechanisms that can facilitate creating an appreciative community where mentoring activities are visible, and mentors are acknowledged to ensure sustained mentoring.

The results reported in this paper lay the foundation for our future work. In the future, we plan to investigate how implicit mentoring impacts facets such as newcomer onboarding and participation diversity in OSS. We envision that our work will help to elevate the long-standing problem of lack of diversity in OSS.

The research artifacts for this study are available publicly at the companion website [84].

ACKNOWLEDGMENTS

We thank all interviewees and survey respondents for contributing to this research. This work is supported by the National Science Foundation under Grant No.: 1901031 and Google—Award for Inclusion Research Program (https://research.google/outreach/air-program/).

REFERENCES

[1] Iftekhar Ahmed, Caius Brindescu, Umme Ayda Mannan, Carlos Jensen, and Anita Sarma. 2017. An empirical examination of the relationship between code smells and merge conflicts. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, IEEE, New York, NY, USA, 58–67.

[2] Tammy D Allen, Mark L Poteet, and Susan M Burroughs. 1997. The mentor's perspective: A qualitative inquiry and future research agenda. *Journal of vocational behavior* 51, 1 (1997), 70–89.

[3] Mandy M. Archibald, Rachel C. Ambagtsheer, Mavourneen G. Casey, and Michael Lawless. 2019. Using Zoom Videoconferencing for Qualitative Data Collection: Perceptions and Experiences of Researchers and Participants. *International Journal of Qualitative Methods* 18 (2019), 1609406919874596.

[4] Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A Gerosa, Igor Steinmacher, and Anita Sarma. 2020. Recommending tasks to newcomers in oss projects: How do mentors handle it?. In *Proceedings of the 16th International Symposium on Open Collaboration*. Association for Computing Machinery, New York, NY, USA, 1–14.

[5] Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. 2018. Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in OSS projects. *CSCW* 27, 3 (2018), 679–714.

[6] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, Berlin, Heidelberg, 1–4.

[7] Suan Loy Zoe Boon. 1998. Principalship mentoring in Singapore: who and what benefits? *Journal of Educational Administration* 36 (1998), 29–43.

[8] Amiangshu Bosu and Jeffrey C. Carver. 2013. Impact of Peer Code Review on Peer Impression Formation: A Survey. In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, New York, NY, USA, 133–142.

[9] Amiangshu Bosu and Kazi Zakia Sultana. 2019. Diversity and inclusion in open source software (OSS) projects: Where do we stand?. In *ESEM*. IEEE, IEEE, New York, NY, USA, 1–11.

[10] João Brunet, Gail C. Murphy, Ricardo Terra, Jorge Figueiredo, and Dalton Serey. 2014. Do Developers Discuss Design?. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. Association for Computing Machinery, New York, NY, USA, 340–343.

[11] Tadeusz Chełkowski, Peter Gloor, and Dariusz Jemielniak. 2016. Inequalities in open source software development: Analysis of contributor's commits in Apache software foundation projects. *PLoS One* 11, 4 (2016), e0152976.

[12] J. Cohen. 1988. Statistical Power Analysis for the Behavioral Sciences.

[13] Allan Collins, John Seely Brown, and Susan E Newman. 1988. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *Thinking: The Journal of Philosophy for Children* 8, 1 (1988), 2–10.

[14] Catarina Costa, Jair Figueirêdo, João Felipe Pimentel, Anita Sarma, and Leonardo Murta. 2021. Recommending Participants for Collaborative Merge Sessions. *IEEE Transactions on Software Engineering* 47, 6 (2021), 1198–1210.

[15] Gabor Csardi, Tamas Nepusz, et al. 2006. The igraph software package for complex network research. *InterJournal, complex systems* 1695, 5 (2006), 1–9.

[16] Shoshana R Dobrow, Dawn E Chandler, Wendy M Murphy, and Kathy E Kram. 2012. A review of developmental networks: Incorporating a mutuality perspective. *Journal of Management* 38, 1 (2012), 210–242.

[17] David L DuBois, Nelson Portillo, Jean E Rhodes, Naida Silverthorn, and Jeffrey C Valentine. 2011. How effective are mentoring programs for youth? A systematic assessment of the evidence. *Psychological Science in the Public Interest* 12, 2 (2011), 57–91.

[18] Lillian T Eby, Marcus M Butts, Brian J Hoffman, and Julia B Sauer. 2015. Cross-lagged relations between mentoring received from supervisors and employee OCBs: Disentangling causal direction and identifying boundary conditions. *Journal of Applied Psychology* 100, 4 (2015), 1275.

[19] Lillian T Eby and Angie Lockwood. 2005. Protégés' and mentors' reactions to participating in formal mentoring programs: A qualitative investigation. *Journal of vocational behavior* 67, 3 (2005), 441–458.

[20] Fabian Fagerholm, Alejandro Sanchez Guinea, Jay Borenstein, and Jürgen Münch. 2014. Onboarding in open source projects. *IEEE Software* 31, 6 (2014), 54–61.

[21] Fabian Fagerholm, Alejandro S Guinea, Jürgen Münch, and Jay Borenstein. 2014. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*. Association for Computing Machinery, New York, NY, USA, 1–10.

[22] Denae Ford, Kristina Lustig, Jeremy Banks, and Chris Parnin. 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*. Association for Computing Machinery, New York, NY, USA, 1–12.

[23] Jane H. Forman and Laura J. Damschroder. 2007. Qualitative Content Analysis.

[24] The Apache Software Foundation. 2020. Mentoring Programme. [Online; accessed 2022-03-16].

[25] The Apache Software Foundation. 2022. Briefing: The Apache Way. [Online; accessed 2022-03-16].

[26] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[27] Coral Elizabeth Gardiner. 2008. *Mentoring: towards an improved professional friendship.* Ph. D. Dissertation. University of Birmingham.

[28] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. 2021. The shifting sands of motivation: Revisiting what drives contributors in open source. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, IEEE Press, New York, NY, USA, 1046–1058.

[29] Mohammad Gharehyazie, Daryl Posnett, Bogdan Vasilescu, and Vladimir Filkov. 2015. Developer initiation and social interactions in OSS: A case study of the Apache Software Foundation. *Empirical Software Engineering* 20, 5 (2015), 1318–1353.

[30] Barney G Glaser. 2016. Open coding descriptions. *Grounded theory review* 15, 2 (2016), 108–110.

[31] Leo A. Goodman. 1961. Snowball Sampling. *The Annals of Mathematical Statistics* 32, 1 (1961), 148–170.

[32] Google. 2022. Google Summer of Code. [Online; accessed 2022-03-16].

[33] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*. IEEE Press, New York, NY, USA, 358–368.

[34] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Garden, Emerson Rex Murphy-Hill, and Kelly Blincoe. 2022. Destructive Criticism in Software Code Review Impacts Inclusion.

[35] Steve R Gunn et al. 1998. Support vector machines for classification and regression. *ISIS technical report* 14, 1 (1998), 5–16.

[36] Kevin A Hallgren. 2012. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology* 8, 1 (2012), 23.

[37] Gary Hsieh, Youyang Hou, Ian Chen, and Khai N. Truong. 2013. "Welcome!": Social and Psychological Predictors of Volunteer Socializers in Online Communities. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. Association for Computing Machinery, New York, NY, USA, 827–838.

[38] Peter Hudson. 2013. Mentoring as professional development:'growth for both'mentor and mentee. *Professional development in education* 39, 5 (2013), 771–783.

[39] Lonnie D Inzer and Chris B Crawford. 2005. A review of formal and informal mentoring: Processes, problems, and design. *Journal of leadership Education* 4, 1 (2005), 31–50.

[40] Maggie Johnson and Max Senges. 2010. Learning to Be a Programmer in a Complex Organization: A Case Study on Practice-Based Learning during the Onboarding Process at Google. *Journal of Workplace Learning* 22 (2010), 180–194.

[41] Suhas Kabinna, Cor-Paul Bezemer, Weiyi Shang, and Ahmed E. Hassan. 2016. Logging Library Migrations: A Case Study for the Apache Software Foundation Projects. In *Proceedings of the 13th International Conference on Mining Software Repositories*. Association for Computing Machinery, New York, NY, USA, 154–164.

[42] Staffs Keele et al. 2007. *Guidelines for performing systematic literature reviews in software engineering.* Technical Report. Technical report, ver. 2.3 ebse technical report. ebse.

[43] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-based software engineering and systematic reviews.* Vol. 4. CRC press, Berlin, Heidelberg.

[44] Oleksii Kononenko, Tresa Rose, Olga Baysal, Michael Godfrey, Dennis Theisen, and Bart de Water. 2018. Studying Pull Request Merges: A Case Study of Shopify's Active Merchant. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. Association for Computing Machinery, New York, NY, USA, 124–133.

[45] RAVI Kothari and MING Dong. 2001. Decision trees for classification: A review and some new results. , 169–184 pages.

[46] JWKJW Kotrlik and CCHCC Higgins. 2001. Organizational research: Determining appropriate sample size in survey research appropriate sample size in survey research. *Information technology, learning, and performance journal* 19, 1 (2001), 43.

[47] Kathy E Kram. 1988. Mentoring at work: Developmental relationships in organizational life.

[48] Kathy E Kram and Douglas T Hall. 1989. Mentoring as an antidote to stress during corporate trauma. *Human Resource Management* 28, 4 (1989), 493–510.

[49] Kathy E Kram and Douglas T Hall. 1994. Mentoring in a context of diversity and turbulence.

[50] Adriaan Labuschagne and Reid Holmes. 2015. Do Onboarding Programs Work?. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, New York, NY, USA, 381–385.

[51] Ann K LaFleur and Bonnie J White. 2010. Appreciating mentorship: the benefits of being a mentor. *Professional case management* 15, 6 (2010), 305–311.

[52] Linux. 2022. Diversity Inclusivity - Linux Foundation. [Online; accessed 2022-03-16].

[53] Antonio M Lluch, Clàudia Lluch, María Arregui, Esther Jiménez, and Luis Giner-Tarrida. 2021. Peer mentoring as a tool for developing soft skills in clinical practice: A 3-year study. *Dentistry Journal* 9, 5 (2021), 57.

[54] Marilyn M Lombardi and Diana G Oblinger. 2007. Authentic learning for the 21st century: An overview. *Educause learning initiative* 1, 2007 (2007), 1–12.

[55] Umme Ayda Mannan, Iftekhar Ahmed, Carlos Jensen, and Anita Sarma. 2020. *On the Relationship between Design Discussions and Design Quality: A Case Study of Apache Projects.* Association for Computing Machinery, New York, NY, USA, 543–555.

[56] T.J. McCabe. 1976. A Complexity Measure. *IEEE Transactions on Software Engineering* SE-2, 4 (1976), 308–320.

[57] M. L. McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia Medica* 22 (2012), 276 – 282.

[58] Thaís Mombach and Marco Tulio Valente. 2018. GitHub REST API vs GHTorrent vs GitHub Archive: A comparative study.

[59] Carol A Mullen and Cindy C Klimaitis. 2021. Defining mentoring: a literature review of issues, types, and applications. *Annals of the New York Academy of Sciences* 1483, 1 (2021), 19–35.

[60] Arnab Nandi and Meris Mandernach. 2016. Hackathons as an Informal Learning Platform. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. Association for Computing Machinery, New York, NY, USA, 346–351.

[61] Mahesh Pal. 2005. Random forest classifier for remote sensing classification. *International journal of remote sensing* 26, 1 (2005), 217–222.

[62] Rajshakhar Paul, Amiangshu Bosu, and Kazi Zakia Sultana. 2019. Expressions of sentiments during code reviews: Male vs. female. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, IEEE, New York, NY, USA, 26–37.

[63] Stephanie C Payne and Ann H Huffman. 2005. A longitudinal examination of the influence of mentoring on organizational commitment and turnover. *Academy of Management Journal* 48, 1 (2005), 158–168.

[64] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[65] Leif E Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2 (2009), 1883.

[66] Martin F. Porter. 1980. An algorithm for suffix stripping. *Program* 40 (1980), 211–218.

[67] D. M. W. Powers. 2011. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies* 2, 1 (2011), 37–63.

[68] Xingqin Qi, Eddie Fuller, Qin Wu, Yezhou Wu, and Cun-Quan Zhang. 2012. Laplacian centrality: A new centrality measure for weighted networks. *Information Sciences* 194 (2012), 240–253.

[69] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. 2019. The signals that potential contributors look for when choosing open-source projects. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–29.

[70] Qualtrics. 2015. Qualtrics XM - Experience Management Software. [Online; accessed 2022-03-14].

[71] Belle Rose Ragins and Terri A Scandura. 1999. Burden or blessing? Expected costs and benefits of being a mentor. *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior* 20, 4 (1999), 493–509.

[72] Hilary Sanfey, Celeste Hollands, and Nancy L Gantt. 2013. Strategies for building an effective mentoring relationship. *The American Journal of Surgery* 206, 5 (2013), 714–718.

[73] Terri A Scandura and Chester A Schriesheim. 1994. Leader-member exchange and supervisor career mentoring as complementary constructs in leadership research. *Academy of management Journal* 37, 6 (1994), 1588–1602.

[74] Andreas Schilling, Sven Laumer, and Tim Weitzel. 2012. Train and Retain: The Impact of Mentoring on the Retention of FLOSS Developers. In *Proceedings of the 50th Annual Conference on Computers and People Research*. Association for Computing Machinery, New York, NY, USA, 79–84.

[75] Andreas Schilling, Sven Laumer, and Tim Weitzel. 2014. Stars matter: how FLOSS developers' reputation affects the attraction of new developers. In *Proceedings of the 52nd ACM conference on Computers and people research*. Association for Computing Machinery, New York, NY, USA, 5–10.

[76] Jefferson Silva, Igor Wiese, Daniel M German, Christoph Treude, Marco Aurélio Gerosa, and Igor Steinmacher. 2020. A theory of the engagement in open source projects via summer of code programs. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 421–431.

[77] Jefferson De Oliveira Silva, Igor Scaliante Wiese, Daniel M German, Igor Fabio Steinmacher, and Marco Aurélio Gerosa. 2017. How long and how much: What to expect from Summer of Code participants?. In *2017 IEEE International Conference*

on Software Maintenance and Evolution (ICSME). IEEE, IEEE, New York, NY, USA, 69–79.

[78] Jefferson O Silva, Igor Wiese, Daniel M German, Christoph Treude, Marco A Gerosa, and Igor Steinmacher. 2020. Google summer of code: Student motivations and contributions. *Journal of Systems and Software* 162 (2020), 110487.

[79] Gurinder Singh, Bhawna Kumar, Loveleen Gaur, and Akriti Tyagi. 2019. Comparison between multinomial and Bernoulli naïve Bayes for text classification. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*. IEEE, IEEE, New York, NY, USA, 593–596.

[80] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work Social Computing*. Association for Computing Machinery, New York, NY, USA, 1379–1392.

[81] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology* 59 (2015), 67–85.

[82] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Why do newcomers abandon open source software projects?. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, IEEE, New York, NY, USA, 25–32.

[83] Igor Steinmacher, Igor Scaliante Wiese, and Marco Aurélio Gerosa. 2012. Recommending Mentors to Software Project Newcomers. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*. IEEE Press, New York, NY, USA, 63–67.

[84] Supplementary. 2022. A Case Study of Implicit Mentoring, its Prevalence, and Impact in Apache. https://doi.org/10.5281/zenodo.6367126

[85] Sylvia Yee Fan Tang and Pik Lin Choi. 2005. Connecting theory and practice in mentor preparation: Mentoring for the improvement of teaching and learning. *Mentoring & Tutoring: Partnership in Learning* 13, 3 (2005), 383–401.

[86] Bianca Trinkenreich, Mariam Guizani, Igor Wiese, Anita Sarma, and Igor Steinmacher. 2020. Hidden Figures: Roles and Pathways of Successful OSS Contributors. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–22.

[87] SciTools Understand. 2020. Understand static code analysis tool.

[88] Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med* 37, 5 (2005), 360–363.

[89] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. 2011. Confusion matrix-based feature selection. *MAICS* 710 (2011), 120–127.

[90] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco A Gerosa. 2020. What to expect from code review bots on GitHub? a survey with OSS maintainers. In *Proceedings of the 34th Brazilian symposium on software engineering*. Association for Computing Machinery, New York, NY, USA, 457–462.

[91] Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *EASE '14: 18th International Conference on Evaluation and Assessment in Software Engineering*. Association for Computing Machinery, New York, NY, USA, Article 38, 10 pages.

[92] Sheng Yu and Shijie Zhou. 2010. A survey on metric of software complexity. In *2010 2nd IEEE International Conference on Information Management and Engineering*. IEEE, New York, NY, USA, 352–356.

[93] Ahmad Nizam Mohd Yusof, Norlela Ahmad, Nirmala, and Lishudzaimah. 2012. Quality and effectiveness of knowledge management transfer using of Mentor-mentee Program and on Job Training in work place. In *2012 International Conference on Innovation Management and Technology Research*. IEEE Press, New York, NY, USA, 50–55.

[94] Hongyu Zhang, Xiuzhen Zhang, and Ming Gu. 2007. Predicting defective software components from code complexity measures. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*. IEEE, IEEE, New York, NY, USA, 93–96.

[95] Luyin Zhao and Fadi P Deek. 2004. User collaboration in open source software development. *Electronic Markets* 14, 2 (2004), 89–103.

[96] Minghui Zhou and Audris Mockus. 2010. Developer Fluency: Achieving True Mastery in Software Projects. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 137–146.

[97] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, IEEE, New York, NY, USA, 518–528.

[98] T. Zimmermann. 2016. Card-sorting: From text to themes. In *Perspectives on Data Science for Software Engineering*, Tim Menzies, Laurie Williams, and Thomas Zimmermann (Eds.). Morgan Kaufmann, Boston, 137–141.