Early Career Software Developers - Are You Sinking or Swimming?

Xin Zhao
xzhao1@seattleu.edu
Seattle University
Seattle, WA, USA

Narissa Tsuboi
tsuboin@seattleu.edu
Seattle University
Seattle, WA, USA

ABSTRACT

Background: Newbies in their early stage of software engineering careers suffer from unfitting task assignments, unclear job expectations, and insufficient communication with managers frequently, which leads to personal frustration, unsatisfactory team performance, and low employee retention. **Goals**: The goal of this research is to investigate new software developers' "sink or swim" early career experience from the following four dimensions: job assignment, newbie-manager pairing, job satisfaction, and thoughts and suggestions. **Methodology**: To achieve our research goal, we conducted an empirical study by distributing an online questionnaire that includes both qualitative and quantitative questions. **Results**: There are several factors contributing to a "sink or swim" early career experience, such as unclear about what to do, who to report to, lack of communication, and vague expectations, posing negative impacts on both individuals and the organization. In addition, we also propose a new community smell in our paper - Newbie Sink or Swim, based on our investigation. **Conclusions**: The early stage is critical to software developers' careers. A failing start phase has detrimental effects on software developers and development teams. Our study empirically examines software developers' early careers from various aspects, providing deeper insights into how to build a more supportive and productive working environment for entry-level developers in the software community.

LAY ABSTRACT

The significance of software engineering in our technology-driven world cannot be overstated. Software bridges the divide between technology and business, resulting in the creation of dependable, secure, and efficient software solutions to complex and cutting-edge problems in our society today.

Software developers are pivotal in developing such solutions. Even though the examination of software developers is not a new topic in the research community, the investigation into early career professionals from an empirical view is limited. To fill this knowledge gap, we surveyed early career software developers (newbies) with a varied demographic composition to investigate their "sink or swim" experiences. We scope this experience within four perspectives: 1) whether or not newbies know or are clearly informed of what to work on; 2) whether or not newbies know who to report to, or who is responsible for managing them; 3) newbie's job satisfaction, described as feeling valued and contributing to the organization; 4) thoughts and suggestions sourced directly from newbies to enhance early career experience.

Our study results revealed several factors that play a significant role in causing a "sink or swim" early career, such as new developers not knowing what to do and who to report to, lack of clear job expectations, and working on tasks that do not match their skills. "Sink or swim" early career experience leaves newbies distressed and demotivated, leading to a higher chance of quitting their jobs. Our participants also offered their suggestions to mitigate the "sink or swim" early career stage, such as self-advocate and frequent meetings with managers. Meanwhile, it is noteworthy to mention that organizations should be committed to building a supportive working environment and uplifting culture for all employees, especially for early-stage software developers.

CCS CONCEPTS

• **General and reference → Empirical studies**; • **Social and professional topics → Computing organizations**.

KEYWORDS

Empirical Software Engineering, Early Career Experience, Sink or Swim, Community Smells

ACM Reference Format:
Xin Zhao and Narissa Tsuboi. 2024. Early Career Software Developers - Are You Sinking or Swimming? . In *Software Engineering in Society (ICSE-SEIS'24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3639475.3640106

1 INTRODUCTION

The expert at anything was once a beginner. — Helen Hayes

Software Engineering continues to revolutionize our changing digital world, empowering organizations to swiftly expand their Information Technology infrastructure and create dependable, top-tier software solutions. This transformation is leading to improved operational performance, streamlined processes, and heightened efficiency in almost all domains today. Furthermore, software engineering also plays a pivotal role in cost reduction, enhanced communication, and task automation for different business fields. As the pioneer of software reliability engineering, John D. Musa put it, software engineering is "the future of a profession [27]."

Along with the rapid evolution and advancement of technology is the tremendous rise in demand for software developers. Based on the global data and business intelligence platform *Statista*, in the

Figure 1: Newbie's Trouble[3]

past five years, the software professional population grew by over 20% and is expected to amount to about 28.7 million in 2024[1]. A similar report from *Kinsta* also points out that "the demand for software developers has doubled since 2020"[2]. These statistics strongly indicate an ever-increasing need for software developers.

Early career software developers (in this paper, we also use **newbies** and **new grads** to refer to early career software developers) are essential to the software engineering community. First, an early career software developer is an investment that may take some time to produce returns in the form of productivity. The hiring bar is difficult in order to select candidates who will be able to learn quickly and contribute. Second, their team may not have time to adequately onboard them. Figure 1 refers to such situations in developing practice. Third, the culture of the team and organization may have a significant impact on a new grad's success, in addition to the technical skills they initially bring to the job

On the other hand, even though it is crucial to ensure early career software developers' success from both an individual level and organizational level in practice, research on newbies in the software engineering domain is scarce in the current literature. To fill this knowledge gap, we conducted this research aiming to **empirically investigate the "sink or swim" experiences of early career software developers**. To achieve this research goal, we designed an online questionnaire that contains both closed and open-ended questions from the following four aspects: job assignment, newbie-manager pairing, job satisfaction, and suggestions to enhance early career experience.

In the end, we collected 91 valid responses from participants with diverse backgrounds. By adopting both quantitative and qualitative analysis, we obtained enriched insights into newbies' early career experiences. In addition, we provided concrete suggestions to both newbies and organizations to uplift the early career stage for newbies. Last but not least, we also defined a new community smell: **Newbie Sink or Swim** that stemmed from our investigation.

[1]https://www.statista.com/statistics/627312/worldwide-developer-population/
[2]https://kinsta.com/software-engineering-statistics/
[3]Online resource: https://imgflip.com/memetemplate/168453057/High-five-drown

Detailed results, discussions, suggestions, as well as the broader impact of this study are presented in Section 4 and Section 5.

In a nutshell, our study offers three novel contributions to the software engineering community:

(1) An in-depth understanding of a "sink or swim" early career experience for new software developers from various perspectives.
(2) Concrete suggestions to both early-stage software developers and managers/organizations to avoid "sink or swim" experiences in practice.
(3) Definition of a new community smell to provide future research implications to both industry and research community to gain deeper insights into socio-technical aspects in the software engineering discipline.

Uniqueness of our paper. Compared with the existing literature in Section 2, our study stands out as a novel research from the following perspectives. First, in our study, we focus on new software professionals who **have less than five years of software development work experience**. Second, we conducted an **anonymous survey**. Compared with interviews or most case studies where personal identification may be revealed, survey respondents are more likely to provide honest and candid responses in anonymous surveys because they do not feel the pressure to conform to social norms or provide socially desirable answers. In interviews, participants may be inclined to give answers they believe the interviewer wants to hear [21].

This paper is organized as follows: In Section 2, we present related work in the area of developer onboarding, community smell, and newbie free-riding. In Section 3, we introduce the background of our research. In Section 4, the methodology of this paper is explained. Section 5 describes the results of our empirical study, along with the discussions of our findings. Section 6 discusses threats to the validity of this paper and approaches we undertook to mitigate these threats. In Section 7, we present the broader impacts of this study. Finally, in Section 8, we conclude the paper and outline our future work.

2 RELATED WORK

The study on "sink or swim" in the field of software engineering is scarce. Yet, a handful of investigations are seen in other fields. For example, Varah et al. [41] discussed sink or swim experience for beginning teachers; Blazer et al. examined beginners' organizational socialization tactics to reflect sink or swim experience [7]. In this section, we introduce three related terms that have been discussed on a larger scale in the field of software engineering: onboarding, community smells, and newbie free-riding.

2.1 Onboarding

A related term to our study is **onboarding**. Defined by Begel and Simon [5], onboarding is "the orientation process by which new hires adjust to and become effective software developers within the corporation." Onboarding experience for software developers is not a new research topic in the software engineering discipline. Some previous investigations focused on one particular aspect during the onboarding process. For example, Pham et al. [29] conducted an empirical study to understand the **testing** skill of new developers and confirmed the knowledge gap between university graduates and industrial expectations. Ju et al. [19] employed an empirical study to examine how **tasks** influence the onboarding experience. Sharma and Stol [34] developed a theoretical model to examine the linkage between onboarding and turnover intention of software developers.

Another category of onboarding research is based on the characteristics of software systems. For example, Steinmache analyzed the onboarding experience in the open-source software community in their work [36]. Because many software applications, such as products from Google and Microsoft, are used worldwide, there are also several papers working on the understanding of onboarding of software developers in large-scale globally distributed systems. such as [9] [10], and [26]. More recently, Rodeghero and Zimmermann conducted a study [30] related to remote onboarding in the era of the Covid pandemic.

2.2 Community Smells

Another relevant concept related to our study is **Community Smells**. Community smells have been gaining more and more attention since the term was coined in 2015 by Tamburri et al. [39]. In their work, they defined community smells as "sets of organizational and social circumstances, having implicit causal relations [39]." In other words, community smells focus on the identification of poor socio-technical decisions together with their potential negative impact on the organizations.

Community smells are strongly correlated with social debt, the state of a software development organization, or a team accumulating sub-optimal decisions [39]. Just as code smells [17] (poor coding practice) are recognizable anti-patterns that serve as warning signs of technical debt, community smells warn of social debt and are connected to people and their interactions within their software development teams [39]. A recent systematic literature review on community smells can be referred to in Caballero-Espinosa's work [11].

2.3 Newbie Free-Riding

One of the community smells studied in the current literature related to newbies is **Newbie Free-Riding**, which indicates a socio-technical phenomenon that senior employees take advantage of newbies by claiming the ownership of development product without contortions [38]. If the newcomer is successful on their own, the team benefits from their productivity and "free-rides" on the newcomer's efforts. If not, the newcomer may have a much harder time finding success in their new role.

The problem of newbie free-riding also exists in other disciplines, such as social science [22] and management [14]. How to address the free-rider problem has been a popular research question in the field of social science [2] [22]. However, related research in software engineering is limited. More investigations are encouraged (such as exploring the causes and effects of newbie free-riding) to benefit the software engineering community.

3 BACKGROUND

In this section, we discuss the background of our study, including an explanation of the key terms we use in this paper, the uniqueness and novelty of our research, and research questions that steer our investigation.

3.1 Key Terms

3.1.1 Sink or Swim. The idiom "sink or swim" describes a situation where someone must find a way to succeed through their own efforts or else fail completely[4]. The expression has a similar meaning to other phrases like "thrown in the deep end" and "trial by fire" [7].

3.1.2 Early Career. It is hard to reach a conclusion on what is "early" in the context of research and practice [4]. In our investigation, we adopted **five years** as the threshold of an early career. This definition is consistent with the definition of "early" in a large number of academic research both in [15] [1] and out [25] [8] [18] of software engineering field.

3.1.3 Job Satisfaction. Job satisfaction is extensively researched in the history of psychology [20] [3]. There also exist numerous metrics to measure job satisfaction [44] [16] [40]. Inspired by these previous examinations, in our paper, we investigate job satisfaction from two dimensions: *feeling invested in and appreciated by managers* and *believing their contributions benefit their manager and team's goals.*

Last but not least, we aim to gain an understanding of **"sink or swim" early career software development experience**. Compared with onboarding which typically takes place during the initial days, weeks, or months of employment, by the examination of the first several years of a software developer's career, we inspect a longer period from different perspectives so that more knowledge will be gained about career development activities.

3.2 Research Questions

Our research goal is to *investigate the "sink or swim" experiences of early career software developers* from the following four perspectives: job assignment, newbie-manager pairing, job satisfaction, and thoughts and suggestions. To this end, we ask the following research questions.

3.2.1 Job Assignment. In the context of this paper, we consider job assignment from the standpoint of whether newbies are clear about what to do or not. When newbies are aware of their tasks and responsibilities, they can work more efficiently, resulting in higher productivity levels. Moreover, understanding what to do ensures that tasks are performed correctly and up to the desired quality standards. Inspired by this observation, the very first perspective we are interested in understanding the early career experience is:

> ***RQ 1:*** *How clear is it for newbies to know what they are supposed to be working on? If it is not clear, what are the causes to this, the effects on newbies, and the actions newbies take to address this?*

3.2.2 Newbie-Manager Pairing. In our paper, we define newbie-manager pairing as newbies "*know (do not know) who is supposed to report to and who is responsible for supervising them.*" Knowing who to report to at work is crucial for effective communication, task

[4]Online resource: https://www.merriam-webster.com/dictionary/sinkorswim

assignment, and alignment with organizational goals. Therefore, our second research question is defined as:

> ***RQ 2:*** *How clear is it for newbies to know who to report to and who is responsible for supervising them? If it is not clear, what are the causes to this, the effects on newbies, and the actions newbies take to address this?*

3.2.3 Job Satisfaction. Job satisfaction is a subjective measure and can vary from person to person. Inspired by the study in psychology and social science [37] [32], in our paper, we examine job satisfaction from two aspects: *feeling invested in* and *making contributions*. Therefore, our third research question is defined as:

> ***RQ 3.1:*** *To what degree do newbies agree with the statement "My manager (or team) is invested in my development and success?*
> ***RQ 3.2:*** *To what degree do newbies agree with the statement "My contributions benefit my manager (or my team) and their goals?*

3.2.4 Suggestions to improve early career experience. Several investigations have shown how to improve software developers from different angles, such as onboarding [42], productivity [28] and learning [45]. However, few studies have been conducted on how to improve software developers' early career experience. Therefore, we are interested in collecting newbies' opinions on how to boost their early career experience. To this end, we define our fourth research question as:

> ***RQ 4:*** *What do early career software developers suggest organizations and teams do to improve their early career experience?*

4 METHODOLOGY

This section provides the details of the methodology we employed in our study, including our survey design, pilot study, participant recruitment, and data collection and analysis.

4.1 Survey Design

We designed an online questionnaire using the tool Qualtrics[5] in our study. The survey contains five sections in total. The first section is to understand the demographics of our participants. The second to fifth sections are to answer our research questions that were discussed earlier. The format of the questions includes choice questions and open-ended questions.

To ensure the target audience of the survey aligns with the directive of this study, we define the relevant group of interest, the *sampling frame* [**kitchenham2015evidence**], by constraining the definition of "early career" from Section 3.1.2 to software developers or professionals in software adjacent roles (i.e. quality assurance, infrastructure, networking, security, and site reliability roles including any type of lead or manager roles) that held less than 5 years of professional experience excluding internships and education. Participants without professional experience were excluded from the sampling frame and discouraged from completing the survey in the survey's Informed Consent section. Participants who were not currently working but possessed the required amount of experience were included in the study.

4.2 Pilot Study

To ensure the validity and quality of our survey, two software professionals reviewed our survey as the pilot study. The first professional has 35 years of experience working in the industry with 21 years of management experience. The second professional has 7 years of experience working in the industry, with 3 years of management experience.

Both of them agreed with the importance of this work. As one of them put,

> *" ... I personally have watched the industry decline over the last 30 years in how it brings in new (early in career) developers and trains them up. We used to know, as an industry, that if you hired people you had a responsibility to grow them, and we took active steps to do this. Now we seem to expect them to sink or swim ... "*

Their feedback suggested minor revisions. Based on their feedback, we made the following changes to the survey: (1) better explained our use of the idiom "sink or swim" in the software development context; (2) added more opportunities for participants to share open-ended remarks related to a multiple-choice question.

Table 1 below shows our survey questions after revisions. Due to space limitations, some questions have been omitted in the table.

4.3 Participant Recruitment

To recruit participants, we employed non-probabilistic sampling and categorized our approaches using 4 techniques: purposive, convenience, self-selection, and snowball sampling. (1) We implemented purposive sampling by sending requests to individuals within the sampling frame whom we considered likely to respond. We utilized the direct message feature of LinkedIn[6], a professional network social media platform, and email to contact eligible professionals in our immediate professional networks. (2) We employed convenience sampling at the author's institution by personally inviting eligible professionals in the computer science graduate program to participate. Additionally, requests were sent to members of the authors' institutional Project Center where collaborations with businesses, government agencies, and nonprofit organizations are established within the Information Technology sector. These partnerships enabled us to disseminate our research invitations through their extensive networks. (3) In self-selection sampling, we posted invitations to complete the survey on online discussion forums that host software development discussion communities (LinkedIn, Discord, Reddit, and DEV.to). Upon encountering the survey invitation on those platforms, participants decided to take the survey or dismiss it. (4) Lastly, we used snowball sampling by requesting respondents to share the survey with others who fit the sampling frame by including a request at the end of the survey for the respondent to share the survey link with others.

[5]https://www.qualtrics.com/

[6]https://www.linkedin.com/

Table 1: Survey Questions

#	RQ	Question Body	Answer Choices
1	D	How old are you?	[Younger than 18, 19-24, 25-34, 35-44, 45-54, 55-64, Older than 65]
2	D	How do you describe yourself?	[Female, Male, Non-binary, Gender-fluid, Gender-queer, Other]
3	D	What is your highest level of education?	[Some High School, High School Diploma, GED, Associate's Degree, Bachelor's Degree, Master's Degree, Doctorate]
4	D	How many years of work experience in the software development industry do you have, excluding education and internships?	[Less than one year,1-2,2-3,3-4,4-5,5 years]
5	RQ 1	In general, I know what I was supposed to be working on.	[True, False]
6*	RQ 1	What factors caused you to be unsure of what you were supposed to be working on?	[I used to know what to work on, but now I am not being assigned much work, There is no process for assigning work, There weren't any tasks that matched my skill level, There is not enough work to go around, Other]
7*	RQ 1	How did not know (or being unsure) what you were supposed to be working on affect you?	
8*	RQ 1	What actions did you take, if any, to resolve the not knowing what you were supposed to be working on? Select all that apply.	[Asked my manager for guidance, Asked a member of my team for guidance, Independently looked for work to do, Started looking for a new job, Other]
9	RQ 2	After onboarding, I know who I am supposed to report to and who was responsible for supervising me.	[True, False]
10*	RQ 2	What factors caused you to be unsure of who you reported to and/or who was responsible for supervising you? Select all that apply.	[I was not assigned to a manager or team, Poor onboarding process, Company or team was unprepared for my arrival, Lack of formal management or team structure, Distributed (remote) team, Other]
11*	RQ 2	How did not know who you were supposed to report to or who was responsible for supervising you affect you?	
12*	RQ 2	What actions did you take, if any, to resolve the not knowing what you were supposed to be working on? Select all that apply.	[Contacted human resources or my recruiter, Asked a member of my team for guidance, Started looking for a new job I took no action, Other]
13	RQ 3.1	My manager is invested in my development and success.	[Strongly Disagree, Somewhat Disagree, Neither agree nor disagree, Somewhat agree, Strongly agree]
14	RQ 3.1	(Optional) Explain what influenced your agreement or disagreement with the previous question: "My manager is invested in my development and success."	
15**	RQ 3.1	My team is invested in my development and success.	[Strongly Disagree, Somewhat Disagree, Neither agree nor disagree, Somewhat agree, Strongly agree]
16**	RQ 3.2	My contributions benefit my manager and his/her goals.	[Strongly Disagree, Somewhat Disagree, Neither agree nor disagree, Somewhat agree, Strongly agree]
17**	RQ 3.2	My contributions benefit my team and its goals.	[Strongly Disagree, Somewhat Disagree, Neither agree nor disagree, Somewhat agree, Strongly agree]
18	RQ 4	What is the most impactful area companies can improve on to give early career developers a better start in their new roles?	[Management and staffing, Work distribution and planning methods, Training, Company culture, Other, Nothing]
19	RQ 4	(Optional) Expand on your answer to the previous question: "What is the most impactful area companies can improve on to give early career developers a better start in their new roles?"	
20	RQ 4	(Optional) Any last thoughts, experiences, or comments you'd like to share with us?	

** Question 6-8 and Question 10-12 only show up when participants choose True for Question 5 and Question 9, respectively.*

*** There is an optional follow-up question after Question 15 - 16. These questions are similar to Question 14. For example, the follow-up question for Question 15 is Explain what influenced your agreement or disagreement with the previous question: "My team is invested in my development and success.".*

4.4 Data Collection and Analysis

4.4.1 Dataset. There are 124 participants who took part in our survey. Among these 124 participants, 20 participants read the Informed Consent but did not continue further with the survey, leaving a total participant pool of 104 participants. After manual examinations of all these 104 responses, we removed another 13 incomplete or invalid responses. Finally, our dataset consists of 91 responses. The completion rate of our survey is 91/104 (87.5%). Our data analyses are based on these 91 responses.

4.4.2 Qualitative Data Analysis. In this study, we adopted qualitative coding to analyze qualitative data to capture the main ideas and conceptions in the data. The idea of the Open Coding process is to break down data into discrete parts and assign each part meaning by labeling it with "codes" representing that meaning. Open coding has been widely used in the domain of qualitative data analysis, such as politics, [46], management [49], and computer science [33] [23].

Our coding process includes four stages. First, the first author and second author coded separately on the first one-third of the responses to all the open-ended questions. Second, the two authors then met to discuss to agree upon a set of codes to be used for the remaining data. Third, the first author and second author applied the agreed codes for the second one-third of all the responses to all the open-ended questions separately, meeting again to update the codes. Fourth, this process is repeated on the last one-third of the responses to all the open-ended questions. In the end, a final set of codes are established by this incremental open coding approach. Last, to further increase the accuracy and reduce the bias of the codes, in our fourth step, we invited another researcher (but not the author of this paper) who is experienced in Open Coding to examine our final codes. This researcher agreed with our final code set.

5 RESULTS AND DISCUSSION

This section presents the results of our investigation and related discussions. To provide the context for the following discussions, We start with the narration of participants' demographics. Following are the detailed discussions of each research question we posed in Section 2.4.

5.1 Demographics

When selecting their most recent role, 79 participants selected Software Developer and 12 participants entered alternate developer roles including Software in Test, Software Support, and Development and Operation (DevOps). Regarding participants' age, 33 participants are between 18 and 24 years old, 50 participants are between 25 and 35 years old, and seven participants are 35 and above. One participant was under 18 and one declined to answer this question.

Another demographic information we collect is the participant's years of experience. Due to the nature of our study, we only accepted participants with between 0 and up to 5 years of industry experience. The distribution is shown in Figure 2. The average years of working experience is 2.01 years, and the median is 2.0 years.

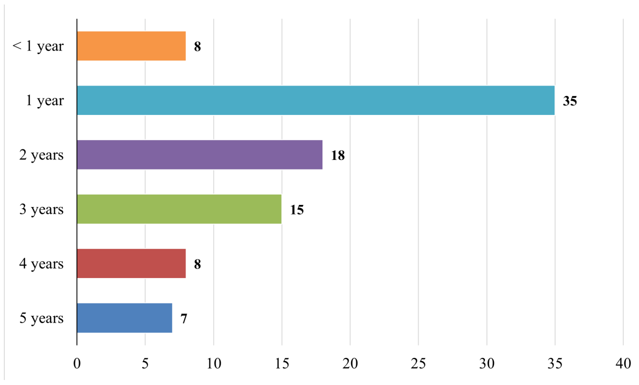

Figure 2: Years of Software Development Experience

91 participants provided their educational background. The majority of participants hold a Bachelor's Degree (55/91 ≈ 60.44%), followed by Graduate and professional degrees (55/91 ≈ 60.44%). Additionally, there are two participants with associate or technical degrees and eight who had not completed a degree. Another demographic we consider in this study is gender identity. Of the 91 complete responses, 49 described themselves as male, 34 as female, three as non-binary or third gender, and five either chose Prefer Not to Say or did not answer this question.

This demographic information demonstrates that our participants come from diverse backgrounds. Interestingly, there is almost an even proportion of participants who identify as male to those who identify as female. We are glad to see this gender distribution considering the fact that most software developers identify themselves as male [43] [35]. Additionally, since the study targeted participants with less than 5 years of experience, the higher percentage of respondents being in the *current early career developer* sub-group for work experience is expected and observed.

5.2 Job Assignment

To answer Research Question 1 (RQ1), we quantify how many participants felt they received clear job assignments during their early career experience. To do this, we asked, "In general, I know what I was supposed to be working on." Of 91 participants, 31 participants (38.46%) reported that in general, they did *not* know to work on, while 56 participants (61.54%) were aware of their job assignments. To ascertain the causes of newbies' unclear job assignment, we ask them the reason by providing a half-open half-closed question in our survey. Participants primarily attributed two causes to not having a job assignment: (1) Their team lacked a formal process for assigning work and (2) There were no tasks that matched their skill level. Interestingly, no selections were made for the answers "I'm not being assigned much work" or "There is not enough work to go around," which suggests that although the newbie may not be directly assigned to a task, there is work being assigned amongst those in their team.

Regarding the effects of not knowing what to do, 21 participants answered this question. Our open coding analyses identified that newbies feel **distress** when, in general, they are not given a job assignment or are unclear on what to work on. We define **distress** in this study as the *amalgamation of the terms used by its own participants: stress, anxiousness, and feeling lost and confused.* One new developer shared personal experience dealing with job assignment related to stress:

> *"It made me feel lost and confused. I was not sure what I am supposed to do... whenever I wasn't assigned work and received no communication this made me feel that I was failing."*

Finding 1: Unclear or absence of job assignments cause feelings of distress in newbies.

Open coding of the responses to the effects of *not knowing what to do* also revealed that "wasting time" was a common byproduct. Newbies report that they spend time trying to figure out what to work on, instead of actually progressing toward completing a distinct set of tasks due to the unclear job assignment problem.

Finding 2: Newbies suffer from decreased productivity when job assignments are not clearly defined.

The result of the question "What actions did you take to resolve not knowing what you were supposed to work on?" is shown in Figure 3. We can see that newbies will more often try to resolve the job assignment problem by reaching out to their manager or team for guidance.

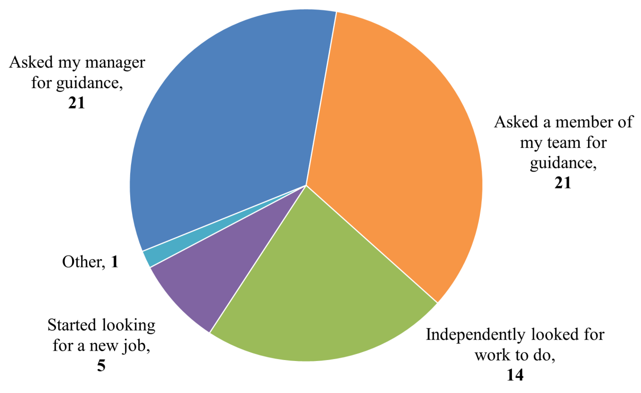

Figure 3: Actions Taken on Unclear Job Assignment

Of the 35 participants who answered this question, 32 participants, or 91.47% reached out to either or both their manager and team for guidance. More impressively, of these 32 participants who asked for help, 12 of them, or 37.50% also independently looked for work to do. This shows that newbies are proactive in attempts to improve their experience and performance, even if they work in a "sink or swim" environment.

Finding 3: Newbies ask for work and guidance on their work when there is no clear job assignment.

5.3 Newbie-Manager Pairing

To answer Research Question 2, we first quantify the number of participants who, after onboarding, still did not know who to report to or who was responsible for supervising them. 18 of 91 (19.78%) participants who answered this question reported they were not paired with a manager. Further, we asked these 18 participants to identify what they think caused them to finish onboarding without a newbie-manager pairing. The majority attributed the cause of this disconnection to a lack of formal management and team structure.

We also set up an open-ended question to seek the effect of not knowing who to report to or who is responsible for supervising newbies. We received 8 participants who shared their thoughts. Our analysis of these responses found that this experience causes the newbie to feel distress through the underlying sense of feeling lost, similar to the negative effects of the job assignment problem in Section 5.2.

Finding 4: Lack of a newbie-manager pairing causes feelings of distress in newbies.

Moreover, we asked the 18 participants who didn't have a manager what actions they took in response to this situation. From the results in Figure 4, we found that the most popular action newbies take is to ask for guidance from their team. These results demonstrate that newbies advocate for themselves when placed in this disadvantageous situation.

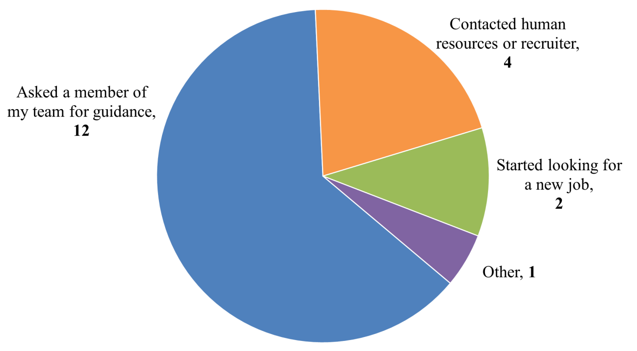

Figure 4: Actions Taken on Missing Newbie-Manager Pairing

5.4 Job Satisfaction

To answer Research Question 3 (RQ 3), in this section, we focus on the newbie's experience of feeling invested in and feeling that their contributions benefit their manager and team, resulting in job satisfaction.

5.4.1 *RQ 3.1: To what degree do you agree with the statement "My manager (or team) is invested in my development and success?*

To answer RQ 3.1, we focus on the participant's **feeling of being invested in** their manager and team, as one part of feeling satisfied about their early career experience. To this end, we measured participants' sense of feeling invested in by their manager with Likert scale agreement questions of the statements "My **manager** is invested in my development and success." Following is an open-ended question asking the participant to explain their level of agreement. Furthermore, this pair of questions was mirrored in terms of the participant's feeling of **team** investment. Participant's strength of agreement with these statements for their manager and team are shown in Figure 5.

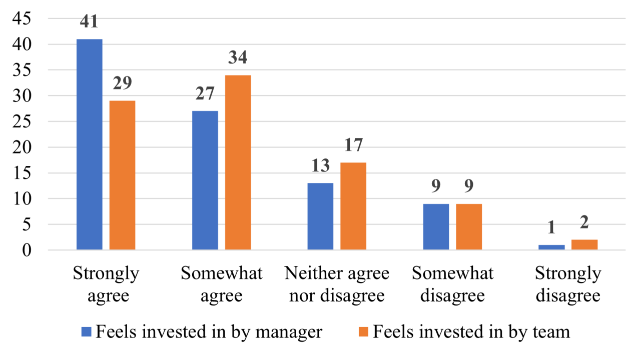

Figure 5: Newbie Feels Invested in by Manager and Team

In terms of feelings of investment by their managers, 74.73% percent of participants agreed that their manager is invested in their development and success and 85.71% feel the same about their team. Therefore, another finding of this study is:

Finding 5: Most newbies feel invested in by their manager and team.

We also noticed that this feeling of investment has positive impacts on a developer's growth. One participant describes investment in action when they said:

> *"My manager has met with me frequently, checking in and helping guide my effort to grow as a software engineer and professional in general."*

5.4.2 *RQ 3.2: To what degree do you agree with the statement "My contributions benefit my manager (or my team) and their goals?*

To answer RQ 3.2, we examined the participant's feeling that **their contributions benefit their manager and their team's goals**. We investigate this experience using the same structure of questions discussed in the previous section. First, we investigate agreement with the statement "My **contributions** benefit my *manager* and their goals," followed by an open-ended follow up question asking the participant to explain their level of agreement. The understanding of the contributions to the *team* and its goals follows the same format. Participant's strength of agreement with these statements for their manager and team are shown together in Figure 6.

In terms of feeling that they contribute to their manager's goals, 85.71% of participants agreed that their contributions benefit their manager's goals and 85.71% feel the same about their team's goals. This result illuminates the newbie's ability to provide value to their organization.

Finding 6: Newbies contribute to achieving their manager and team's goals.

5.5 Suggestions to Improve Early Career Experience

To answer Research Question 4 (RQ 4), we ask newbies for *their* opinions on how to improve the early career experience so that they will not be left to a sink or swim environment. There were 80 participants who selected the most impactful areas companies can

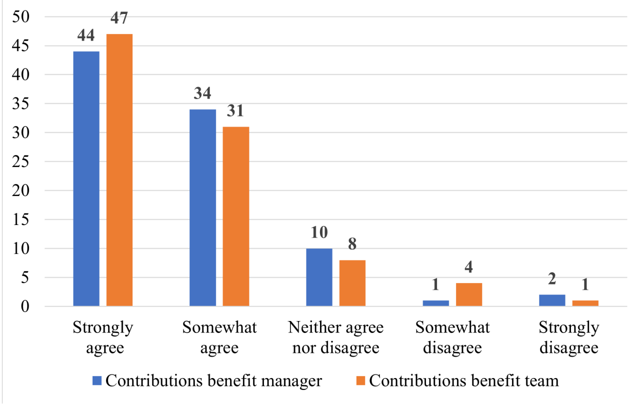

Figure 6: Newbie's Contributions Benefit Manager and Team Goals

improve on to give new developers a better start in their new roles. From Figure 7, we see that Training (36.35%) and Work distribution and planning methods (31.25%) were rated the most impactful. Company culture was the next highest rated at 13.75%.

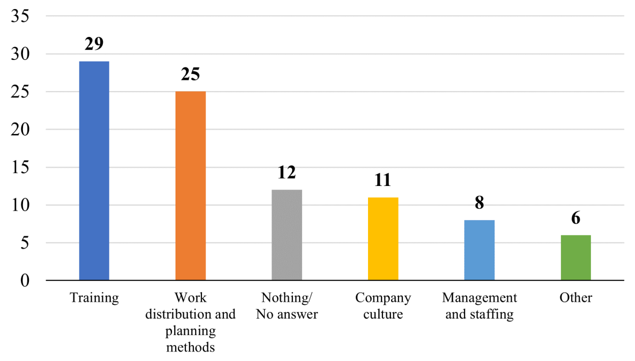

Figure 7: Areas of Improvement

We analyzed 41 open-ended responses to follow up on the reasoning behind participant's selection. Sentiment on the area of Training as an improvement was represented in 41.46% of responses. Many of these responses were laced with frustration, with one participant even criticizing the state of training in the software industry in general:

> *"Training tends to be terrible. New developers are expected to figure out everything on their own."*

In regards to how company culture improves new grad experience, a participant shared,

> *" ... [company] culture - you want to make sure developers feel like they belong even while they're figuring things out..."*

To conclude, we present another finding from the discussion of this section:

Finding 7: Newbies feel that Training, Work Distribution Methods, and Company Culture are the most impactful areas companies can improve.

In addition, during the process of analyzing participants' final thoughts, we realized that a sink or swim experience can exacerbate **Imposter Syndrome**. Imposter Syndrome is a psychological phenomenon that causes intense feelings of intellectual fraudulence, even in the face of the sufferer's success [48]. Imposter syndrome compounds stress and burnout, and decreases job performance over time [48].

We identified several pertinent examples of imposter syndrome in participants experiencing performance issues in their roles. For instance,

> *" I really want to be good enough for my current job... I'm now on a self-driven plan to improve or be let go, I don't know if I can actually win or become an engineer after all... But now I have mainly high anxiety since I started [this] job ... while it appears those who started with me are managing it well and getting stronger. "*

Finding 8: Sink or swim experience for newbies may lead to Imposter Syndrome.

6 THREATS TO VALIDITY

This section discusses threats to the validity of our study and the mitigation strategies we employed to address these threats based on the three types of research validity threats: internal, external, and construct threats [12] [13].

6.1 Internal Threats to Validity

This research design may hold internal threats to validity: (1) It is possible that qualitative responses could be misinterpreted. Particularly because the study solicited responses based on how participants *feel* or *felt* at a time that may be recent or distant from memory. To mitigate this threat, we used a systematic qualitative data coding process. (2) The number of participants and their demographic diversity may have affected the results. Our study is based on the responses from 91 participants. We may get different results when surveying a larger number of participants. Additionally, there may be demographics we did not measure that may have affected the results. To mitigate this threat, we plan to conduct a more demographically expansive study in the future.

6.2 External Threats to Validity

External threats refer to whether the participants included in this study represent the perspectives of a majority of early-career software developers. In an empirical study, it is not possible to make a definitive conclusion because it is not possible to include every person who meets the research participant criteria in the study. To mitigate this threat, we solicited responses from an array of sources. Our participant demographics have a representative mixture of years of experience across 0 through 5 years, and an impressive balance of perspectives across genders. Another external threat is perspectives may be skewed by the state of employment for early-career developers that existed at the time of this study. Recently, big tech companies and non-tech companies alike have performed layoffs and decreased early career hiring. As our study is unable to benchmark responses against a normalized environment, we plan to perform this study again in a neutral hiring and employment environment.

6.3 Construct Threats to Validity

Construct threats to validity refer to the threat the method of measurement may not align with the study construct. For this study, construct threats arise from survey questions that may inaccurately present our research intentions. To mitigate this threat, we obtained and implemented feedback from two software professionals with experience in academia and industry. Additionally, from the manual reviews of participant responses, we concluded that participants answered each question as we intended.

7 BROADER IMPACT

As we delve into the intricate realms of early career experience, it becomes increasingly imperative to recognize the profound societal impacts and implications that our work may hold. In this section, we elucidate the broader impact of our research, both in terms of advancing the understanding of the early career stage for software professionals and proposing future research directions in the software community.

7.1 Suggestions to Newbies

We found that newbies who are suffering in a "sink or swim" work environment make concerted attempts at improving their situation by balancing two approaches: (1) asking for support and (2) making progress independently. One suggestion we received from one newbie for fellow newbies is as follows,

> *"Being an early developer is all about a balance between time-boxing your individual effort and collaborating with your peers to ensure the work gets done."*

Newbies should constrain their independent efforts to a certain amount of time, also known as "**time-boxing**" one's efforts. Then, if they're still stuck, reach out for support. This describes our first suggestion for newbies:

Suggestion 1: Balance your efforts between bids for support and working independently.

Our study also indicates that when newbies face challenges in work, they sometimes choose to work on their own, even if they may need to invest a large amount of time outside work hours. One reason is that newbies are worried that asking too many questions may make senior colleagues or managers annoyed. To address this concern, we suggest newbies follow several practices at work:

(1) Be Proactive: Don't wait for opportunities to come to you. Identify areas where you can make a difference and take the initiative to lead or participate in relevant projects.

(2) Know Your Worth: Understand your strengths, skills, and the value you bring to your organization. Recognize your accomplishments and contributions.

(3) Seek Feedback: Regularly ask for feedback on your performance. Constructive feedback can help you identify areas for improvement and showcase your commitment to personal and professional development.

Advocating for yourself is not about being self-centered but about ensuring that you receive fair treatment, opportunities, and recognition in your working environment. It is a skill that, when used effectively, can benefit both you and your organization. Therefore, the second suggestion stemming from this study for newbies is:

Suggestion 2: Know your worth, be proactive, and seek feedback.

7.2 Suggestions to Managers

During our analyses, the importance of 1:1 meetings (or One-on-one meetings) drew our attention. Many newbies who had a sink or swim early career experience reported that their managers did not have regular 1:1 meetings.

One-on-one meetings provide an opportunity for newbies to establish a personal connection with their manager, helping newbies feel more comfortable in their new environment and fostering a sense of belonging. Managers can also use these meetings to discuss newbie's learning and development plans. These meetings contribute to a smoother working process and can set the stage for long-term career growth within the organization. Therefore, the first suggestion we have for managers is:

Suggestion 3: Have regular One-on-One meetings with newbies because they are central to newbies' early career success.

Our study also revealed that one factor leading to a sink or swim early career is unfit job assignment and unclear job expectations. Many participants complained that they did not work on the project/task that matched their skills. On the other hand, in the field of Psychology, there is a term called "*Pygmalion Effect*" or "*Rosenthal Effect*" that was proposed by Rosenthal and Jacobson in 1968, indicating expectations are related to performance [31]. The idea behind the Pygmalion Effect is that increased expectations from team leaders will indeed result in better performance from team members [47] [24]. Combined with what we found in this study and the Pygmalion Effect, the second suggestion we have for managers is:

Suggestion 4: Assign tasks based on newbies' skills; set clear and moderately high expectations.

7.3 Suggestion to Organizations

Our participants also stressed the importance of uplifting the company culture in the early career experience. As one participant put,

> *"... Company culture is a crucial and valid point because, in my opinion, early career developers can get off to a better start if they work in an environment that promotes a positive work culture ..."*

Uplifting company culture is a significant undertaking that requires a concerted effort from leadership and employees at all levels. We found out that it is vital that leaders and managers embody the desired cultural traits. Their behavior sets the tone for the entire organization. Also, organizations should encourage open discussions and solicit feedback from all levels of the organization. Employees who feel heard and valued are more likely to engage in culture improvement efforts [6]. Therefore, our suggestion to organizations is:

Suggestion 5: Continuous and dedicated effort is needed to uplift organizational culture to help newbies avert a sink or swim early career.

7.4 Implications to Research

Our study obtains a deeper understanding of the early career experience of software professionals. Based on what we found, in this paper, we define another community smell - **Newbie Sink or Swim**, which depicts an environment where a brand new junior developer is not given the necessary support to learn their role. Rather, the implicit or explicit expectation is for them to succeed or fail by primarily their own efforts, without formulated assistance from others. Newbie Sink or Swim causes heightened feelings of distress, anxiety, and confusion, and can exacerbate existing feelings of Imposter Syndrome in its sufferers. Therefore, another broader impact raised from this paper is the definition of the new community smell: Newbie Sink or Swim.

Newbie Sink or Swim: A community smell indicating new professionals' success or failure largely depends on their individual efforts.

As a new community smell, we look forward to the exploration of this concept and related research surrounding Newbie Sink or Swim from the software engineering community.

8 CONCLUSION AND FUTURE WORK

Our study sheds light on software developer's early career experiences from different perspectives. Based on our limited knowledge, this is the first investigation to understand the sink or swim early career for beginning software professionals in the current literature. We summarized the causes of a sink or swim early career experience. In addition, we also delved into the impacts of sink or swim experiences on newbies and the actions newbies take to address the issues. Our study also provides several suggestions to different stakeholders to enhance newbie's early career experience. At last, we present a new community smell - Newbie Sink or Swim, indicating the problem our paper inspects.

In the future, we are interested in the following research on the new community smell we proposed in this paper: Newbie Sink or Swim. In addition, as a future study, we are also interested in the extension of this investigation by conducting **an empirical examination of "sink or swim" early career experiences for new software professionals from *managers*' perspective**. This examination will provide us with an opposite but pertinent view to examine newbies' early career experience. We believe by considering "sink or swim" from multiple angles, we are better equipped to develop solutions to address the current challenges, thus enhancing a more humane and integral workplace for all the professionals in the software engineering community.

REFERENCES

[1] Muhammad Ovais Ahmad, Iftikhar Ahmad, and Fawad Qayum. 2022. Early career software developers and work preferences in software engineering. *Journal of Software: Evolution and Process*, e2513.

[2] Peter Asch and Gary A Gigliotti. 1991. The free-rider paradox: theory, evidence, and teaching. *The Journal of Economic Education*, 22, 1, 33–38.

[3] Brikend Aziri. 2011. Job satisfaction: a literature review. *Management research & practice*, 3, 4.

[4] Pat Bazeley. 2003. Defining'early career'in research. *Higher Education*, 45, 3, 257–279.

[5] Andrew Begel and Beth Simon. 2008. Struggles of new college graduates in their first software development job. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 226–230.

[6] Benisa Berry. 2004. Organizational culture: a framework and strategies for facilitating employee whistleblowing. *Employee Responsibilities and Rights Journal*, 16, 1–11.

[7] Maddy Blazer, Greg A. Chung-Yan, and Debra Gilin. 2023. Sink or swim: developing an alternative measure of employee socialization. *Employee Responsibilities and Rights Journal*, 3, 1, 16–20. DOI: 10.1007/s10672-023-09457-2.

[8] Agnes Bosanquet, Alana Mailey, Kelly E Matthews, and Jason M Lodge. 2017. Redefining 'early career'in academia: a collective narrative approach. *Higher Education Research & Development*, 36, 5, 890–902.

[9] Ricardo Britto, Daniela S Cruzes, Darja Smite, and Aivars Sablis. 2018. Onboarding software developers and teams in three globally distributed legacy projects: a multi-case study. *Journal of Software: Evolution and Process*, 30, 4, e1921.

[10] Ricardo Britto, Darja Smite, Lars-Ola Damm, and Jürgen Börstler. 2020. Evaluating and strategizing the onboarding of software developers in large-scale globally distributed projects. *Journal of Systems and Software*, 169, 110699.

[11] Eduardo Caballero-Espinosa, Jeffrey C. Carver, and Kimberly Stowers. 2023. Community smells—the sources of social debt: a systematic literature review. *Information and Software Technology*, 153, Article 107078, (Jan. 2023), 25 pages. DOI: 10.1145/1219092.1219093.

[12] Donald T Campbell and Julian C Stanley. 2015. *Experimental and quasi-experimental designs for research*. Ravenio books.

[13] Thomas D Cook, Donald Thomas Campbell, and Arles Day. 1979. *Quasi-experimentation: Design & analysis issues for field settings*. Vol. 351. Houghton Mifflin Boston.

[14] Matteo Corciolani and Daniele Dalli. 2014. Gift-giving, sharing and commodity exchange at bookcrossing. com: new insights from a qualitative analysis. *Management Decision*, 52, 4, 755–776.

[15] Michelle Craig, Phill Conrad, Dylan Lynch, Natasha Lee, and Laura Anthony. 2018. Listening to early career software developers. *Journal of Computing Sciences in Colleges*, 33, 4, 138–149.

[16] Pavitra Dhamija, Shivam Gupta, and Surajit Bag. 2019. Measuring of job satisfaction: the use of quality of work life factors. *Benchmarking: An International Journal*, 26, 3, 871–892.

[17] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. 2012. *Refactoring: Improving the Design of Existing Code. Addison-Wesley Object Technology Series*. Pearson Education. ISBN: 9780133065268. https://books.google.com/books?id=HmrDHwgkbPsC.

[18] Michael Gebel. 2010. Early career consequences of temporary employment in germany and the uk. *Work, employment and society*, 24, 4, 641–660.

[19] An Ju, Hitesh Sajnani, Scot Kelly, and Kim Herzig. 2021. A case study of onboarding in software teams: tasks and strategies. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 613–623.

[20] Timothy A Judge and Allan H Church. 2000. Job satisfaction: research and practice. *Industrial and organizational psychology: Linking theory with practice*, 166, 198.

[21] Sarah Knox and Alan W Burkard. 2009. Qualitative research interviews. *Psychotherapy research*, 19, 4-5, 566–575.

[22] Peter Kollock and Marc Smith. 1996. Managing the virtual commons. *Computer-mediated communication: Linguistic, social, and cross-cultural perspectives*, 109–128.

[23] Amanda Lee and Jeffrey C Carver. 2019. Floss participants' perceptions about gender and inclusiveness: a survey. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 677–687.

[24] Fred C Lunenburg. 2011. Self-efficacy in the workplace: implications for motivation and performance. *International journal of management, business, and administration*, 14, 1, 1–6.

[25] Jane Mills, Cindy Woods, Helena Harrison, Jennifer Chamberlain-Salaun, and Ben Spencer. 2017. Retention of early career registered nurses: the influence of self-concept, practice environment and resilience in the first five years post-graduation. *Journal of Research in Nursing*, 22, 5, 372–385.

[26] Nils Brede Moe, Viktoria Stray, and Marcus R Goplen. 2020. Studying onboarding in distributed software teams: a case study and guidelines. In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, 150–159.

[27] John D. Musa. 1985. Software engineering: the future of a profession. *IEEE Software*, 2, 1, 55.

[28] AZRYNA AZLEN MOHD Nordin and NMA Rodziahlatih. 2020. Using saas to enhance productivity for software developers: a systematic literature review. *Journal of Theoretical and Applied Information Technology*, 98, 24, 4107–4128.

[29] Raphael Pham, Stephan Kiesling, Leif Singer, and Kurt Schneider. 2017. Onboarding inexperienced developers: struggles and perceptions regarding automated testing. *Software Quality Journal*, 25, 4, 1239–1268.

[30] Paige Rodeghero, Thomas Zimmermann, Brian Houck, and Denae Ford. 2021. Please turn your cameras on: remote onboarding of software developers during a pandemic. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 41–50.

[31] Robert Rosenthal and Lenore Jacobson. 1968. Pygmalion in the classroom. *The urban review*, 3, 1, 16–20.

[32] Teresa J Rothausen and Kevin E Henderson. 2019. Meaning-based job-related well-being: exploring a meaningful work conceptualization of job satisfaction. *Journal of Business and Psychology*, 34, 357–376.

[33] Stephan Salinger, Laura Plonka, and Lutz Prechelt. 2008. A coding scheme development methodology using grounded theory for qualitative analysis of pair programming. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*.

[34] Gaurav G Sharma and Klaas-Jan Stol. 2020. Exploring onboarding success, organizational fit, and turnover intention of software professionals. *Journal of Systems and Software*, 159, 110442.

[35] Karina Kohl Silveira and Rafael Prikladnicki. 2019. A systematic mapping study of diversity in software engineering: a perspective from the agile methodologies. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 7–10.

[36] Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Workshop on Global Software Development in a CSCW Perspective*. Vol. 16, 20.

[37] Adhe Rachman Sulistyo and Suhartini Suhartini. 2019. The role of work engagement in moderating the impact of job characteristics, perceived organizational support, and self-efficacy on job satisfaction.

[38] Damian A. Tamburri, Rick Kazman, and Hamed Fahimi. 2016. The architect's role in community shepherding. *IEEE Software*, 33, 6, (Nov. 2016), 70–79. DOI: 10.1109/MS.2016.144.

[39] Damian A. Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2015. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, 6, 1, Article 10, (May 2015), 17 pages. DOI: 10.1186/s13174-015-0024-6.

[40] N Van Saane, Judith K Sluiter, JHAM Verbeek, and Monique HW Frings-Dresen. 2003. Reliability and validity of instruments measuring job satisfaction—a systematic review. *Occupational medicine*, 53, 3, 191–200.

[41] Leonard J Varah, Warren S Theune, and Linda Parker. 1986. Beginning teachers: sink or swim? *Journal of teacher education*, 37, 1, 30–34.

[42] Jianguo Wang. 2012. Supporting developer-onboarding with enhanced resource finding and visual exploration.

[43] Yi Wang and David Redmiles. 2019. Implicit gender biases in professional software development: an empirical study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 1–10.

[44] John P Wanous and Edward E Lawler. 1972. Measurement and meaning of job satisfaction. *Journal of applied psychology*, 56, 2, 95.

[45] Shao-Fang Wen. 2023. Context-based support to enhance developers' learning of software security. *Education Sciences*, 13, 7, 631.

[46] Jared Wesley. 2014. The qualitative analysis of political documents. *From text to political positions: Text analysis across disciplines*, 135–160.

[47] Susan S White and Edwin A Locke. 2016. Problems with the pygmalion effect and some proposed solutions. In *Organizational influence processes*. Routledge, 182–208.

[48] Catherine Wilkinson. 2020. Imposter syndrome and the accidental academic: an autoethnographic account. *International Journal for Academic Development*, 25, 4, 363–374.

[49] Michael Williams and Tami Moser. 2019. The art of coding and thematic exploration in qualitative research. *International Management Review*, 15, 1, 45–55.