International Journal of Software Engineering and Knowledge Engineering
© World Scientific Publishing Company

Evaluating a Method to Select Software Developers from Source Code Analysis

JOHNATAN OLIVEIRA
Federal University of Lavras
Lavras-MG, Brazil
johnatansfc@gmail.com

MAURÍCIO SOUZA
Federal University of Lavras
Lavras-MG, Brazil
mauricio.ronny@ufla.br

EDUARDO FIGUEIREDO
Federal University of Minas Gerais
Belo Horizonte-MG, Brazil
figueiredo@dcc.ufmg.br

Software Developers are in high demand in today's fast-paced technological environment, and there is tough competition for the best candidates. Recruiters are tasked with the challenge of finding the best and most qualified software developers to meet the needs of their organizations. With the rise of technology and the increasing use of online platforms, such as GitHub, recruiters now have access to a vast pool of potential candidates. However, this also presents new challenges, as recruiters must navigate the overwhelming number of profiles and determine the true capability and credibility of the developer profiles they are considering. This paper investigates the difficulties faced by recruiters in finding skilled software developers and the role that GitHub plays in the recruitment process. By conducting interviews with 15 recruiters from 3 different countries, we aim to shed light on the challenges and provide a comprehensive method to help recruiters overcome these difficulties and find the right software developers for their organization. The results indicate that the method provides a more comprehensive understanding of profiles on GitHub, emphasizing aspects such as programming language, frequency of commits, and experience, in addition to filtering cluttered profiles.

Keywords: Recruitment Challenges, Software Engineers, GitHub, Hard Skills

1. Introduction

Software development has become one of the most crucial fields in the current technological landscape, and as a result, there has been a significant demand for qualified software developers [1, 2, 3, 4]. With the increasing need for software development professionals, the competition to hire the best talent in this field has become more intense [5, 6, 7]. One of the biggest challenges in the hiring process of software developers is evaluating their technical skills [7, 8, 9]. Unlike other traditional careers, the skills of software developers cannot be evaluated merely by observing their work experience, educational qualifications, or conducting interviews [4, 10, 11].

Free/Libre Open Source Software (FLOSS) software development is no longer restricted to small groups or communities of developers [12, 13]. FLOSS developers can collaborate on many projects to varied degrees from any geographical location, at any hour of the day, and with any educational background (e.g., adding features, writing patches, and enhancing documentation) [13, 14]. Since software developers come from a large talent pool with a wide range of backgrounds and levels of expertise [12, 15], finding developers with appropriate and specialized experience becomes challenging [9, 12, 16]. In fact, selecting a new software developer is complex and expensive [9, 13, 17]. From social platforms, such as GitHub, the recruiters can be use available information to support decisions to select a new developer for the team [9, 18, 16, 19]. In general, there is a lot of information on GitHub [9], but it is necessary to mine the data to know more about a specific developer [9]. GitHub is an open-source software hosting repository with extensive social networking features integrated with the development environment [20].

The Society for Human Resource Management estimates that it can cost $4,129 and take 42 days on average to fill a new position [a]. Due to a lack of developers, companies are forced to hire freelancers or take a chance on hiring not qualified software developers for the job, which may not work out in the long run [12]. While this might not be a considerable concern for some businesses, for startups, it might be a barrier to growth [b]. Although the scarcity of qualified workers in the tech sector is already widely acknowledged, this problem has other root reasons as well. According to Job Seeker Nation Report [c], 67% of recruiters still struggle to locate high qualified candidates with the necessary abilities, despite the rise in tech graduates in recent years.

In the case of a software development project, specific research methods targeting the search for experts have been conducted [21, 22, 23]. One of the first was the Expertise Recommender [21]. To find experts, it uses two heuristics. The first one, change history, assumes that the authors of the relevant revision are the file's subject-matter experts. The second one, dubbed tech help, uses a support database to find customers who have already found solutions to issues [21]. By examining the usage history of a specific method, Schuler et al. [22] have established a method to find Java method experts. They argue that programmers who use a technique should be regarded as having the same level of knowledge as programmers who alter and modify the method. However, these previous papers [21, 22, 23] do not investigate the central problem from the recruiters point of view. On the other hand,

[a]https://employa.com/blog/the-biggest-challenges-when-hiring-engineers/
[b]https://www.codingame.com/work/blog/hr-news-trends/tech-recruitment-2021-trends-statistics/
[c]https://cloudemployee.co.uk/blog/it-outsourcing/the-5-challenges-you-will-face-when-hiring-software-developers

our work investigates the problems of the recruitment process from the viewpoint of recruiters.

In this paper, we aim to explore the difficulty in the process of hiring software developers from the recruiters point of view and how GitHub can be used as a tool to indicate the hard skills of software developers. The purpose of this study is to understand the impact of GitHub on the hiring process and the potential it holds as a tool to indicate the technical skills of software developers. By examining the current practices in the hiring process and the use of GitHub, we aim to provide insights and recommendations for organizations seeking to hire software developers.

We conducted exploratory interviews with 15 recruiters from Brazil, Canada, and United States. Our study relies on GitHub to mine the software developers and present their profiles to recruiters. This means that prospective recruiters can see a developer profile of projects listed on the site and a history of their code-related actions over time on both these and other projects. Changes made by developers are communicated to other developers following the project as they are made [20]. In addition to discourse about changes in the form of comments, a history of commits (or contributions) to the code is compiled over time [20]. We select aleatory recruiters from LinkedIn and, to invite these participants, we sent e-mails to explain about the study. We scheduled the interview sessions to be forty minute long.

We interpret our interview results through coding inspired by Ground the Theory [24] to understand how and why specific situations were viewed as reliable signals of underlying characteristics of a potential hire. Our findings indicate that specific activity traces are considered more trustworthy by recruiters compared to others, mainly due to their ability to provide reliable signals of underlying traits that are typically challenging to evaluate through traditional interviews. For instance, programming languages, frequency of commits, and frequently used APIs are viewed as particularly valuable indicators of a developer's technical proficiency.

The results indicate that the method provides a more comprehensive understanding of profiles on GitHub, emphasizing aspects such as programming language, frequency of commits, and experience, in addition to filtering cluttered profiles. However, recruiters point out the need for additional information in the profiles, such as the popularity of projects and historical details, and the importance of seeking complementary data on other social networks for a more complete understanding of the candidates.

We have observed the following results. 1) Recruiters require more detailed information on developers, and a method that can provide these data would greatly assist the selection process. 2) Some developers create GitHub profiles with unnecessary information such as icons, figures, and generic text. 3) When recruiters have numerous candidates to consider, selecting a new developer becomes more challenging since it requires processing numerous profiles, which can be time-consuming. 4) Identifying a developer's programming language proficiency is not always straightforward. As a result, recruiters may need to use methods such as toy source-code to evaluate this.

The remainder of this paper is organized as follows. We describe the key concepts to understand this paper in Section 2. Section 3 shows the study design. Section 4 presents the results obtained from this paper. Section 5 describes the limitations of our study. Section 6 presents the related work. Finally, Section 7 concludes by presenting our key findings.

2. Background

In this section, we define the important concepts to understand the work. Section 2.1 shows details about hard and soft skills. Section 2.2 presents the way to select software developers. Furthermore, Section 2.3 shows details about the method used to compute programming skills.

2.1. *Hard Skills and Soft Skills*

Hard skills and soft skills are two crucial aspects of software development that organizations consider while hiring software developers [25]. Software developers need to have hard skills, which include technical know-how and aptitude in areas like programming languages, software tools, and frameworks [26, 27]. For a software engineer to properly carry out their daily job, these abilities are crucial. Technical interviews, coding tests, and reviews of a developer's contributions to open-source projects are frequently used to assess hard skills. Soft skills, on the other hand, are the non-technical competencies and character traits of a software development, including teamwork, communication, problem-solving, and flexibility [26, 27]. Non-technical talents or soft skills include psychological phenomena, including social interaction skills, communication, creativity, and teamwork [28]. These abilities are crucial to a software developer's success and overall effectiveness.

Although soft skills are relevant for selecting a developer candidate, our focus is on hard skills in this work. Because evaluating soft skills is subjective, a face-to-face meet is recommended to understand the candidates' soft skills in more details. In contrast, we can obtain extensive data about hard skills from GitHub. They cover the theoretical fundaments and practical experience one needs to complete the intended task [28]. Developers are frequently viewed as technical people [28, 29]. As a result, their technical competency is heavily stressed in both practical work and research studies.

2.2. *Recruitment Process*

Developers recruitment is defined as an employer's actions that are intended to 1) bring a job opening to potential job candidates who do not currently work for the organization, 2) influence whether these people apply for the job, 3) verify whether they maintain interest until a job offer is made, and 4) influence whether a job offer is accepted [30]. The recruitment process for selecting a new software developer is a crucial task for companies, as it determines the success of software development projects [4, 31, 30]. The process involves several strategies aimed at evaluating the hard and soft skills of software developers. Some common strategies used in the recruitment process of software developers include interviews and profile mining [30, 32]. Technical interviews are one of the most common strategies used to evaluate the technical abilities of software developers [7, 30]. The interviews usually consist of coding challenges, algorithms, and data structures, aimed at evaluating the candidate's problem-solving and coding skills [32, 33]. Another strategy is GitHub Profile Mining. GitHub is a Web-based platform for version control and collaboration that has become the facto standard for software development professionals [31, 20]. Recruiters can review a candidate's GitHub profile to evaluate their coding skills, contributions to open-source projects, and overall development activity [20]. In this study, we apply this strategy from a method [23] that compute the hard skills. It is possible to conduct other strategies, such as Behavioral Interviews. Behavioral interviews are aimed at evaluating a candidate's soft skills, such as communication, collaboration, problem-solving, and adaptability [31, 34]. The interviews usually consist of questions that require the candidate to provide examples of how they have handled difficult situations in the past [31, 34].

Even when applying these strategies, the recruitment process of software developers is a challenge. One of the biggest challenges is evaluating the technical abilities of software developers, as it requires a deep understanding of programming languages, tools, and frameworks [34, 35]. Another challenge is the lack of qualified software developers, which makes it difficult for organizations to find the right candidate [35]. The recruitment process of software developers is a crucial task for organizations, and recruiters should use a combination of strategies to evaluate the technical and interpersonal skills of software developers. The use of profile mining, technical interviews, and behavioral interviews can provide valuable insights into a candidate's abilities and help recruiters make informed decisions [12]. Despite the challenges, the recruitment process is essential for the success of software development projects, and organizations should invest the necessary time and resources to ensure they find the right candidate [12].

The hiring process might be straightforward, but the initial step finding potential candidates can be extremely difficult due to technological advancements [12]. A company's bottom line may be significantly impacted by practical recruitment activities that make it stand out and more desirable to prospective employees [30]. Software developers can post and share their work in online areas thanks to social networking platforms, for instance GitHub. These experts develop a reputation within a field of expertise, frequently intending to find a job [30].

2.3. *A Method to Identify Hard Skills*

This section describes the method [23] used to compute the programming skills of the software developer. This method implements two models, (i) one based on Changed Files (CF) and (ii) the other based on Changed Lines of Code (CLOC).

For each model, the method computes two programming skills: Programming Language and Application Programming Interface (API), most used by the developer. In addition, the method shows details about the developer profile, for example, since when they use GitHub, the number of repositories, and a short bio. The method is based on GitHub blame [d] and git log [e] services. GitHub blame is used to pinpoint which revision and author last modified each line of a file, thus providing a fine-grained view of authorship throughout the history of the file. This enables an in-depth analysis of individual contributions to a software project. Similarly, git log is employed to evaluate the volume and specifics of a developer's contributions, such as the number of commits made with a focus on particular programming languages. This evaluation aids in constructing the Commit Frequency Model (CF), which quantifies and assesses a developer's active engagement in software development tasks. On the other hand, the Code Lines Change Model (CLOC) utilizes the data from git log to measure the quantitative impact of changes to the codebase by developers. This includes tracking the total number of lines added, modified, or removed and analyzing the types of programming languages involved. Such metrics offer insights into the developers' technical proficiency, adaptability, and specialization across various technologies. These tools combined form a comprehensive solution to evaluating the hard skills of software developers, providing a detailed perspective on their contributions to collaborative software projects. Figure 1 presents the overview of the method from the use of the GitHub blame. The method computes programming skills in the following way.

(1) The method receives input from the name of the target developer.
(2) It runs GitHub blame from all developer repositories from GitHub to identify the number of commits by programming language, date, number of developers, and amount of lines of code changed (add, removed, and modified).
(3) The method computes programming languages and commits to a specific developer.
(4) From the last step, the method uses a strategy to identify APIs from keywords of the programming languages based on imports.
(5) The method presents as output a PDF file with CV of the target developer.

Figure 2 presents a CV generated by the method for a developer, including a short bio, name, and a section on their GitHub repositories with duration of use. It features two bar charts: one detailing lines of code changed per programming language, and another showing commits made in various languages. Additionally, it details the APIs most frequently used by the developer.

[d]https://git-scm.com/docs/git-blame
[e]https://git-scm.com/docs/git-log

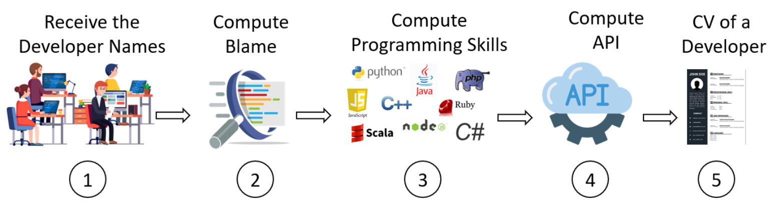
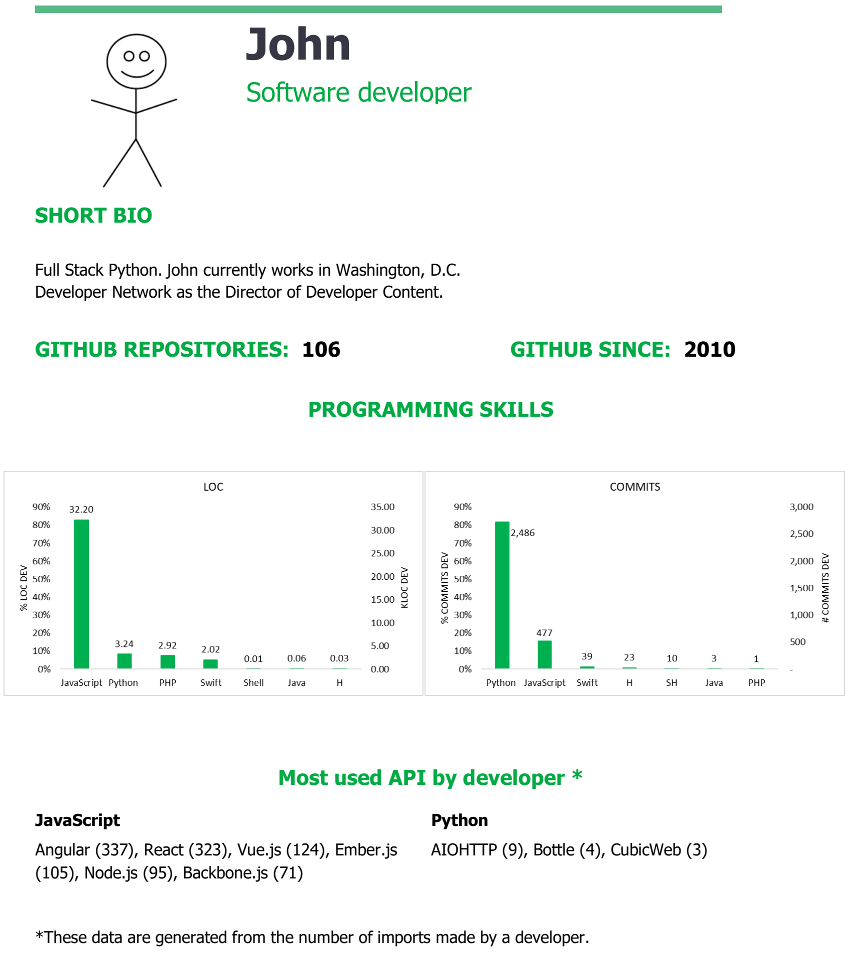


Fig. 1. Method to Compute Programming Skills



John

Software developer

SHORT BIO

Full Stack Python. John currently works in Washington, D.C. Developer Network as the Director of Developer Content.

GITHUB REPOSITORIES: 106 **GITHUB SINCE: 2010**

PROGRAMMING SKILLS

LOC
% LOC DEV
KLOC DEV
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
32.20
3.24
2.92
2.02
0.01
0.06
0.03
JavaScript
Python
PHP
Swift
Shell
Java
H
COMMITS
% COMMITS DEV
# COMMITS DEV
3,000
2,500
2,000
1,500
1,000
500
-
2,486
477
39
23
10
3
1
Python
JavaScript
Swift
H
SH
Java
PHP

Most used API by developer *

JavaScript

Angular (337), React (323), Vue.js (124), Ember.js (105), Node.js (95), Backbone.js (71)

Python

AIOHTTP (9), Bottle (4), CubicWeb (3)

*These data are generated from the number of imports made by a developer.

Fig. 2. Curricula example generated from the method used

3. Study Design

This section describes the protocol for interviewing recruiters of our empirical study. Section 3.1 presents the aims of our study and the research questions we address. Section 3.2 shows the protocol we follow to conduct the interviews. For instance, we adopt the concepts of Focus Interview [36] and Think Aloud [37]. Section 3.3 presents the steps to select the subjects. Section 3.4 summarizes the phases conducted to interview recruiters.

3.1. *Goal and Research Questions*

The main goal of this study is to understand the problems in hiring a new software developer to fill a specific position from the point of view of recruiters. More specifically, we would like to know if the selection of software developers is appropriate and how profile mining can help in this step. Since we are interested in comprehending the recruiters' opinions about this process, we conducted exploratory interviews with them. This means prospective recruiters can see a developer's profile of projects listed on GitHub and a history of their code-related actions over time on both these and other people's projects. We show below the 4 Research Questions (RQ) of this study.

RQ1-How do companies recruit software developers?
RQ2-What are the possible application of the method to mine a developer profile?
RQ3-How do the method results complement GitHub information?
RQ4-What are the opportunities for improving the hiring process?

From the RQ1, our primary goal is to comprehend the variety of strategies and processes companies implement to recruit software developers. This inquiry aims to unearth the diverse methods employed for assessing software developers' hard skills. Such methods often include technical and behavioral interviews, as well as evaluations of candidates' GitHub profiles. This question endeavors to elucidate the prevalent practices in the recruitment landscape. The second question focuses on exploring the practicality and effectiveness of integrating novel methods into the existing framework for selecting software developers. This involves assessing if alternative strategies can effectively augment the current recruitment process and contribute to a more robust selection of candidates. Our third query delves into the potential for enhancing GitHub profiles with supplementary information. By investigating additional data extracted from GitHub repositories, we aim to understand if these insights can aid recruiters in making more informed decisions when selecting software developers. The final question seeks to pinpoint potential areas for enhancement within the method employed in this study, as well as within the broader hiring process. Our goal is to discern the limitations and areas ripe for improvement, assessing how the method can adapt to evolving technological trends and requirements. Insights gleaned from this inquiry are intended to refine the recruitment process, making it more efficient and effective in securing top-tier software development talent.

Figure 3 shows the four steps conducted to answer the research questions: 1) Instrumentation. In this step, we selected two developers from GitHub with similar JavaScript profiles and conducted a semi-structured interview. 2) Selecting Subjects. In this step, we investigate software developer selection with 2 randomly selected developers and recruiters with at least 2 years of experience invited from LinkedIn and networking contacts, speaking either Portuguese or English. 3) Interview with Recruiters. This step is responsible to describe the online semi-structured interviews with recruiters, focusing on their use of GitHub during the hiring process. 4) Data Analysis. This step is responsible for coding and analyzing the data from the interviews. Each step of this figure is a section presented in the following.

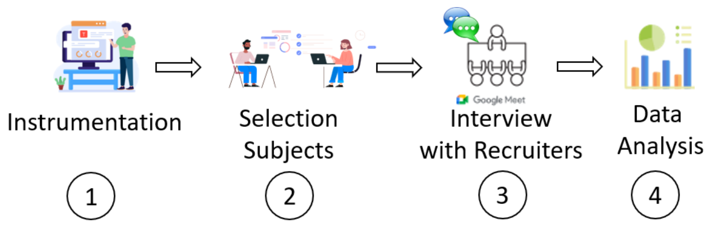


Fig. 3. Study Steps

3.2. *Instrumentation*

In this section, we describe the instrumentation used to conduct the study. We create one fictitious job opportunities based on open jobs from LinkedIn. Next, we randomly select, two developers from GitHub. Who have similar profiles concerning the programming languages (in this case, JavaScript). It was not mandatory that the developers have the same programming skill level. We then compute the programming skills of these two developers from the method selected in this study [23] and generate each developer's CV. Before conduct the interview with recruiters, we select them. We select recruiters with at least 2 years of experience invited from LinkedIn and networking contacts. After that, we conducted a semi-structured interview with recruiters. In this type of interview, open questions can be applied off-order. This way, the participants can express themselves more freely and elaborate on their answers. To obtain more details about the opinion of the recruiters, we adopt the think-aloud protocol [37, 38, 39].

Participants in think-aloud protocols speak their thoughts out loud as they carry out a series of predetermined tasks [37, 38, 39]. In our case, we observe the use of GitHub from the recruiters actions. Participants are invited to say whatever comes to mind as they finish the activity. This could involve what they are seeing, contemplating, acting upon, and feeling. For instance, when a recruiter looks at the developer's repositories, we need to understand the recruiter's viewpoint. Making thought processes as explicit as feasible throughout task execution allows observers to gain insight into participants' cognitive processes rather than just their end product. Every verbalization is recorded, transcribed, and examined. We also request that the interviewees share their screens with us. This step is essential to view points evaluated by recruiters. Before starting the recorder screen and voice, we solicit the consent of the interviewees to record. Finally, we analyze the data generated by interviews using the open and axial coding inspired to Ground the Theory [21, 40, 41]. A replication package for this paper is made available at Zenodo [42].

3.3. *Selection Subjects*

This study rely on two profiles of developers and selected 15 recruiters to investigate the software developers selection process. This way, we select aleatory two software developers from GitHub with similar abilities but not necessarily of the same level of programming skills. We selected two developers by which the GitHub profiles had information about the developer which matches the job description. As a result, we invite 20 recruiters from our personal network and mined from LinkedIn of known, and 15 recruiters accepted participated of the study. To invite a recruiter, they need to have experience in at least two years as a recruiter. This period is an important factor in being interviewed to be able to contribute to the study.

3.4. *Interview with Recruiters*

We used interviews for data collection conducted online in November and December 2022. We adopt the concept of Focus Interview [36]. This type of interview emphasizes the interviewees subjective and personal responses, where the interviewer engages in eliciting more information [43]. According to Merriam [44], interviews effectively elicit information about things that cannot be observed. We used semi-structured interviews with open-ended questions because this approach gathers richer responses compared to structured interviews [45]. These were conducted in Portuguese or English. Interviewees: we contacted each participant by e-mail. To invite these participants, we select aleatory recruiters from LinkedIn or from our personal networks. We, too, used networking from companies to invite recruiters. Each interview occurred in a private Google Meeting room. We asked the recruiters to describe a recent past hire, focusing on how they used GitHub during the hiring process. We request what information on the site was relevant and what that information is saying about the candidate.

Interviewees voluntarily accepted to participate in the research. They had to agree with the Informed Consent Form, which guarantees the confidentiality of the data provided, the anonymity of the participants, and the right to withdraw from the research at any moment. We collected data and enhanced the interview in two rounds. During the first round, we piloted the interview design with two recruiters. We then discarded the pilots' data. Through the pilot tests, we understood that it was necessary to change some interview questions, enhance CVs of the developers, and emphasize the focus on hard skills to conduct the interview. Therefore, our final pool of interviewing was composed of 15 recruiters because two were used in the study pilot. Table 1 presents the Interview Questions (IQ) in detail. The first column indicates the research questions used to create the interview question. The second column indicates the number of IQs, and the last column presents the interview questions. Note that from IQ1 to IQ4 are questions about data exclusive of GitHub. Already, from IQ5 to IQ8 are questions about the method data.

Table 1. Interview Questions

RQ	IQ	Interview Questions
RQ1	IQ1	How does the company evaluate a profile of a new software developer to compose the team? What are the steps?
RQ1, RQ2	IQ2	Based on GitHub, how does the company evaluate the candidate's skills? Can you compare Candidate 1 with Candidate 2 using GitHub data?
RQ1	IQ3	What the characteristics do you observe on these profiles of the developers ?
RQ3,RQ4	IQ4	Which developers would the company choose between these two developers? Why?
RQ3,RQ4	IQ5	From the PDF file of the mined profiles, what data do you consider as important?
RQ3,RQ4	IQ6	From PDF file of the mined profiles, how do you compare the two developers ?
RQ3,RQ4	IQ7	From PDF file of the mined profiles, what developer do you choose for this job position? Why?
RQ3,RQ4	IQ8	From the PDF file of the mined profiles, what are the recruiter's observations? How can the PDF file help the recruiter to select a software developer from GitHub?

3.5. *Data Analysis*

We conducted a series of semi-structured interviews. When we finished all interviews, we compile the results. To compile the results, we create a transcript of the interview in details and made highlights in crucial points about opinion and actions of the recruiters. In our analysis, we coded the interview transcripts to identify the various ways in which profiles were employed during the hiring process, as well as the diverse types of inferences drawn about individuals under evaluation based on the GitHub environment.

Using HyperResearch [30], a qualitative analysis software tool, we identified relevant sentences or broader segments in interview transcripts related to each research question. We used open coding to analyze the interview transcripts. We start by reading the transcripts, identifying key points, and assigned them a code (i.e., a 2–3 words statement that summarizes the key point). In the context of this work, the 15 attributes identified in the previous step were used as seeds for this analysis.

By constantly comparing the codes [40], we grouped them into four categories that gave a high-level representation of the codes. This coding process was conduced by one researcher and constantly discussed with other two researchers.

4. RESULTS

This section details the results of our investigation. Section 4.1 provides descriptive statistics on the sample demographics. Section 4.2 to Section 4.5 presents the results to answer the four research questions investigated in this paper.

4.1. *Participants' Demographic Information*

This section shows the demographic information of the interview participants. Table 2 shows the 17 participants of this study, including two in the pilot. In the first column, we present the participant ID. We remove the real name of participants to preserve their identity. The second column shows the gender of the participants. The third column shows the company size. We separate the data by company size into four groups: a) Micro Enterprises: fewer than 10 people, b) Small Enterprises: between 11 and 50 people, c) Medium-Sized Enterprises: between 51 and 250, and d) Large Enterprises: from 251 people or more. In the fourth column, we show the country of the recruiter company. Finally, in the last column, we present the approximate number of employees in each company. The last two rows of Table 2 present the demographic data about the pilot study. In general, the companies of participants in the study are Large enterprises. For instance, participant P1 from Brazil works in a company with $\sim$ 16,500 employees. In contrast, we also interviewed recruiters from micro-enterprises, for instance, participant PS2 (pilot study). Of the 15 interviewed, four are female.

4.2. *Recruitment Channels*

In this section, we answer the RQ1: How do companies recruit software developers?

Our first research question focused on how recruiters use recruitment channels to evaluate new candidates. We investigate the channels and strategies used to select software developers from the interviews. In general, all interviewees believed that a GitHub profile provided insight into an individual's technical abilities and personal qualities more reliably than resumes or code samples taken out of context. The GitHub profiles provide recruiters with a history of individual contributions over time. In some situations, searching for complementary information about the candidate from other channels, for instance, LinkedIn, is necessary; for instance, when the candidate is junior and the GitHub is clean or very simple. On the other hand, we observe that some companies like to receive curricula in PDF of a candidate and conduct a face-to-face interview. In the same case, they provide a candidate with a fictitious programming problem. In this way, the candidate needs to implement a source-code as a solution from a time and programming language stipulated by the

Table 2. Participant demographics

Participant	Gender	Company Size	Country	Employees
P1	M	Large Enterprises	Brazil	16,500
P2	M	Large Enterprises	Canada	9,700
P3	M	Large Enterprises	Canada	9,700
P4	M	Large Enterprises	Brazil	8,452
P5	M	Large Enterprises	Brazil	5,571
P6	M	Large Enterprises	Brazil	4,000
P7	F	Large Enterprises	Brazil	3,600
P8	F	Large Enterprises	Brazil	2,990
P9	M	Large Enterprises	Brazil	2,990
P10	F	Large Enterprises	Brazil	2,990
P11	M	Large Enterprises	Brazil	2,788
P12	M	Large Enterprises	Brazil	1,873
P13	M	Large Enterprises	United States	500
P14	F	Large Enterprises	Brazil	174
P15	M	Medium-Sized Enterprises	United States	52
Pilot Study				
PS1	M	Large Enterprises	Brazil	4,419
PS2	M	Micro Enterprises	Brazil	3

recruiters. In this case, the recruiter evaluates the source-code quality. In general, according to interviewees, companies look at soft skills in the first moment and then select a developer based on hard skills. However, if a developer has bad soft skills, this developer cannot pass for the next steps. Therefore, the soft and hard skills are complementary.

Nevertheless, this situation depends on also the requirements of the open job. For example, if the company searches for a junior developer, it is optional that this developer knows many technologies, such as Java, Python, JavaScript, and frameworks. On the other hand, if the company is searching for a senior developer, this candidate needs excellent soft and hard skills because, generally, this developer is more expensive and is expected to have more experience. Besides, they will guide the team to solve a problem; therefore, if the developer has excellent hard skills, but their soft skills are not good, then this candidate is weak. As an example to support this claim, recruiter P2 said:

> *[...] The best software developer is not someone mastering JavaScript or React, for instance. The best software developer is one who can contribute to the team. The best software developer needs a balance between soft and hard skills. The best software developer needs to listen to the team for giving and receiving help. To develop a software is hard then, we require people with the same propose. As said the popular saying "one bad apple can spoil the whole barrel".*

Finding a new software developer can be difficult, especially if the recruiter only looks at the curricula and promote test with toy problems. During the recruitment process, whether it is done using paper or online platforms like LinkedIn, important factors like cultural fit, communication skills, and cooperation skills can often be overlooked. However, these factors are crucial for success in a collaborative work environment. Furthermore, although source-code exams can give a candidate a look of their technical proficiency, they might not fully capture their problem-solving abilities or their capacity to design clear, maintainable code. To get a more thorough assessment of a candidate's prospective fit the business, it is crucial to combine these methodologies with other evaluations, like behavioral interviews and real-world projects.

RQ1 summary. *We have observed that software developers need to balance both soft and hard skills. Moreover, we have found that recruiters consider GitHub to be an important platform for obtaining reliable information about job candidates.*

4.3. *Applicability*

In this section, we answer the RQ2: What are the possible application of the method to mine a developer profile?

Table 3 shows coding computed to help us to answer this question. This table has four columns: Category, Codes, Participants, and Frequency. The column Category indicates a group created to aggregate the similar codes. The column Codes are abstract models that emerge during grounded theory analysis's sorting and demonstration stages. They conceptualize the integration of substantive codes as hypotheses of a theory. This study uses coding inspired by grounded theory but does not necessarily apply all ground theory steps. The column Participants indicates the number of participants that cited something related to the code. Finally, the last column, Frequency, shows the number of times a code was cited.

Table 3 shows four codes: 1) "Effort Reduction", 2) "First Step to Filter", 3) "Rework" and 4) "Profile Type". Concerning "Effort Reduction", we identify from the interviews with recruiters that it is necessary to optimize the process of recruiting a developer. Sometimes, there are many candidates for a job position. We identify this code from the view point of the 13 (86.66%) participants with frequency of 17 occurrences. The number of candidates can harder the selection process because it is expensive in time and resources. That is, the team needs to stop some tasks to help the recruiter to select a new developer. Therefore, the hiring process may require less effort to find a new candidate by using the method to generate the CV of the candidates from hard skills. The recruiter interviewed P1 said:

> *[...] GitHub is a rich source of data about candidates for job vacancies. However, we need to spend time identifying the candidate's hard skills, for example, the primary programming language and whether the candidates know a specific API / framework. From the generated CV, we reduced the work to identify the developer's abilities. In addition, we are able know more details about the candidate.*

The code "First Step in Filtering" is about recruiters' viewpoint of using the CV as an instrument to pre-select the candidates. We observed that the recruiters usually have a considerable effort to conduct the hiring process because, generally, there are many candidates. Sometimes, to find a suitable candidate in a universe of twenty candidates, for instance, is exhausting. This way, from the viewpoint of recruiters, it is possible to reduce the effort by using the CV generated by the method that computes hard skills. The idea is to use the CV as a pre-selection of the candidates. This code was manifested by 12 (80%) participants in 14 times. In that case, it is possible to adopt the method, but If the job opening is for a junior position, the recruiter needs to evaluate in more detail because the GitHub profile of the candidate may be less important. In the following, we quote recruiter P4.

> *[...] This CV can help us to pre-selecting the candidates. It is possible to evaluate the developer from the specific hard skill as a target of our interest, for example, knowing React. Therefore, the sector of the humans resource can be streamlined in search specific abilities and conduct the filter to the next steps. As such, the recruiter needs less work to conduct the process of selecting a new candidate.*

The other code identified was "Rework". This code summarizes the situation required to restart the recruitment process. We note that 8 (53.33%) of the 15 recruiters point out the problems and costs associated with the flawed selection process. Therefore, it is necessary to be the most assertive as possible in selecting a software developer because conducting the recruitment process again is expensive and exhausting. In addition, the team generally needs to support the newly hired developer or offer training. This way, the costs to the company can be high, because it may be necessary to fire the developer and hire another. Through the method, it is possible to help in the selection process with more details about the candidate are available from source-code analysis. In this way, the recruiter has more data to decide about the software developer, for instance, the candidate's main programming language or knowing APIs. In the following, we show the quote of recruiter P7.

> *[...] The selection process is challenging for us recruiters because we need to select a developer to help our team. If we select wrong, we will have problems in the future because it will be necessary to hire other developers and restart the process. Therefore, this problem can delay the project. The method cannot substitute the face-to-face meeting or the recruiter, but the generated CV can help to understand the developer's profile in more details.*

The source-code available in GitHub can provide idea of the profile of the developer. The code "Profile Type" indicates the suitability to back or front-end, for example. This code was manifested by 7 (46.66%) participants in 7 times. We observe that recruiters analyzing the CV generated by the method had more details of the candidate. For example, if a candidate like more Python or made more commits and wrote many lines of code to SQL, then this developer has characteristics of the back-end and database manager. These data could be obtained from GitHub without the mining method. However, to investigate all repositories of a specific developer manually is very exhausting to recruiter. Recruiter P6 said:

> *[...] Sometimes we need to look at many CVs by day. This way is inevitable to lose some information. In addition, if we need deep information into each developer's repository manually, selecting a developer will be more complicated. Therefore, sometimes we look at GitHub profiles in general, only the tip of the iceberg after selecting a candidate from the best. We apply, for example, source-code tests to obtain more information about the hard skill of the developer. This summarized PDF file can help us in this step to avoid looking only at the tip of the iceberg. From the CV, it is possible to understand the developer's profile and know if they can work with us. In addition, if the job requires a junior, we immediately understand the deficiency of some technologies, and we can provide training, for example. Therefore, the presented method can help decision to select a candidate.*

Table 3. Applicability of the Method Evaluated

Category	**Codes**	**Participants (%)**	**Frequency**
Application	Effort Reduction	13 (87%)	17
	First Step to Filter	12 (80%)	14
	Rework	8 (53%)	8
	Profile Type	7 (47%)	7
		Total	46

RQ2 summary. *Usually, recruiters need to evaluate many profiles of the candidates for a job. This task is complex and extremely relevant to the selection of an excellent developer for the team. Therefore, through the method, it is possible to obtain more details about each profile from GitHub. This way, the recruiter decision for a candidate is better supported. In addition, the method can be used as a first step to select a developer and provide more details about them.*

4.4. *Alternative Visualization*

In this section, we answer the RQ3: How do the method results complement GitHub information?

To respond to this research question, we asked the recruiters about the data in the PDF file and GitHub. Table 4 shows the category Alternative Visualization. This table presents the exact configuration of Table 3. The code Programming Language represents the recruiters' viewpoint about the details they did not observe on GitHub but noted from the PDF file of mined profile. During the interview, recruiters analyzed the developer's programming language from repositories in GitHub. GitHub presents the primary programming language of a repository. However, it does not present the programming language that the developer used in their repositories. For instance, if a developer creates a script only in Python, but the primary programming language of the repository is JavaScript, only observing the primary programming language informed by GitHub is insufficient. That way, the recruiters praised the alternative visualization of the programming language provided in the method presented in the PDF file. This code was manifested by 13 (86.66%) participants, 15 times. As an example, recruiter P15 said.

> *[...] In fact, it is more convenient to observe the programming language that a specific developer uses with the PDF file. Because from the GitHub repository, we observe other programming languages that are not the developer's skills.*

Another interesting code found was "Frequency of Commits". This code was manifested by 12 (80%) participants, 15 times. "Frequency of Commits" is presented in a PDF file generated by the method about the commits made by a specific developer concerning a programming language. The "Frequency of Commits" made by a developer can indicate the amount of work them is doing on a project for a specific a programming language. However, it is essential to remember that quantity is not necessarily a direct measure of the quality of work, as Recruiter P11 said.

> *[..] The number of commits made by a developer can be visualized from GitHub too. However, the PDF file complements this view because it puts a bar chart with commits by programming language side by side. This way is more convenient to analyze.*

"Time Experience" is a code concerning a developer's period using GitHub. This code was manifested by 11 (73.33%) participants, 13 times. "Time Experience" is data essential to understand the knowledge of developers about the version control, for example, Git. Note that these data are available from GitHub, but sometimes needs to be highlighted to recruiters. From the PDF file, the recruiter notes these data and can be associated with other data, for example, programming language, API, and frequency of commits. The recruiter P9 said.

> *[...] These data is available on GitHub. However, from the PDF file, it is possible to complement the data presented by GitHub because it is possible to compare the period that the developer used GitHub and the technologies adopted by them.*

"API/Framework" is a code about the technologies developers use together with a programming language, for instance, Hadoop. This code was manifested by 8 (53.33%) participants, 12 times. GitHub shows a pool of data that sometimes can confuse the recruiter and can be hard to be analyzed, for example, the API used by them. The PDF file shows the API/Framework most used by developers from a specific programming language. The use of API by a developer can indicate familiarity with the technologies. Developers often use APIs and frameworks that they are familiar with, as this can help them work more efficiently and produce higher-quality code. Familiarity with a particular API or framework makes it easier for developers to find solutions to problems they encounter during development. In addition, the recruiter can take these data to understand more about the developer because the API can be associated with the application's performance, for example. The performance of an API or framework can significantly impact the overall performance of the software application. Therefore, it is possible to check if a developer knows API most adopted by the company. It is crucial to consider the speed and efficiency of the API or framework used and ensure it meets the project's performance requirements. Recruiter P7 said.

> *[...] Recruiters face a non-trivial task when searching for specific APIs in source-code via import. As the method shows the mostly used API, it is more practical for a recruiter to understand what is being used and, if necessary, ask questions to the developer. Since this import can be presented in many files, and they must be opened individually, the process can be time-consuming when made manually.*

The code "Repositories" indicates the number of repositories of a developer. The number of repositories a developer has on GitHub can be one factor to consider when assessing their skills and experience. However, it should be evaluated alongside other factors, including code quality and collaborative skills. This code was manifested by 7 (46.66%) participants 12 times. Recruiter 6 said.

> *[...] It is easy to look at these data from GitHub. However, combining these data with the level of commits by programming language helps the recruiter analyze the developer's profile. Moreover, this mechanism in GitHub is not trivial. Therefore, the data from the PDF file complements the data presented in GitHub. These data is vital because the number of open-source projects a developer has contributed to can positively indicate their commitment to the community and willingness to share knowledge and expertise. The developer is more comfortable working in a distributed environment and is skilled at navigating open-source processes and tools.*

"Pollution Profile" is a code about the presentation problems in GitHub. The pollution of GitHub profiles refers to developers cluttering their profiles with unnecessary pictures and emojis. While GitHub is primarily used for hosting and sharing code repositories, many developers use their profiles to showcase their personalities and interests. However, some developers take this to the extreme and fill their profiles with many images and emojis, making it challenging to navigate and find relevant information. Additionally, these unnecessary elements can slow down the page's loading time and make it less accessible. This practice harms the recruiter in finding the data about the developer. This code was manifested by 3 (20%) participants 3 times. Recruiter 4 said.

> *[...] I think that the pollution of GitHub profiles to be a significant problem in the candidate selection process. While I appreciate that developers want to showcase their personality and interests, excessive pictures and emojis on their GitHub profiles can be distracting and make it difficult for me to evaluate their technical skills and experience. It can also slow down the process of reviewing profiles, mainly if I am working with many candidates. Additionally, cluttered profiles can make it challenging to find essential information, such as links to code repositories, projects, and contributions to open-source software. Therefore, I urge developers to keep their profiles clean and straightforward, highlighting their technical accomplishments and experience clearly and concisely. This will help them stand out as a serious candidate and increase their chances of being selected for the next stage of the hiring process.*

Figure 4 shows the distribution of recruiter preferences when selecting methodologies for the hiring process of new developers. The bar chart presents a significant inclination towards the method, with 87% of the participants showing a preference for its alternative data visualization. In contrast, only 13% favored using GitHub. This significant difference shows the method's utility and effectiveness in the decision-making process of recruitment.

Table 4. Category Visualization

Category	**Codes**	**Participants (%)**	**Frequency**
Alternative visualization	Programming Language	13 (87%)	15
	Frequency of Commits	12 (80%)	15
	Time Experience	11 (73%)	13
	API / Framework	8 (53%)	12
	Repositories	7 (47%)	11
	Pollution Profile	3 (20%)	3
		Total	69

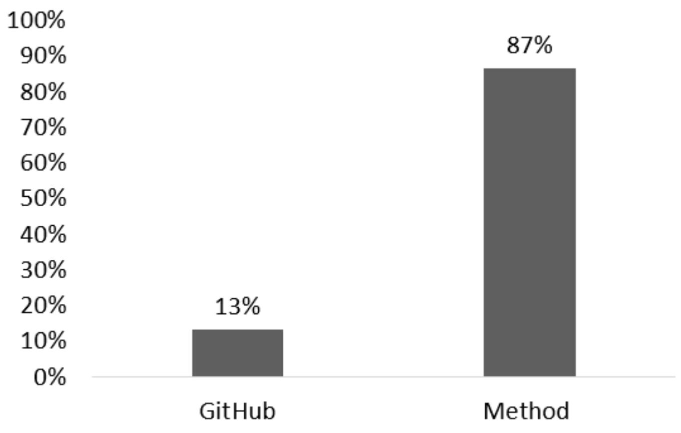


Fig. 4. GitHub vs Method

RQ3 summary. *In general, the method to summarize profiles is able to show an alternative visualization of the data provided by GitHub. For example, pollution profile was observed by the recruiters. The pollution profile refers to cluttered profiles with unnecessary pictures and emojis, making it difficult to navigate and find relevant information. The method is able to filter out this problem.*

4.5. *Miss Information*

In this section, we answer the RQ4: What are the opportunities for improving the hiring process?

To respond to this research question, we asked the recruiters about the need for more data from PDF generated by the method. Table 5 shows the category "Lacking Information". This table presents the exact configuration of Table 4. We create a code "Starts". Stars on GitHub are a way for users to indicate that they liked a particular project and follow its development. For software developers, having many stars in their projects can indicate that their work is popular and well-received by the community and can be a positive factor when looking for work or receiving contributions from other developers. In addition, stars also help make a project stand out on GitHub, making it more visible to other developers looking for solutions to a particular problem or inspiration for their own projects. This code was manifested by 11 (73.33%) participants, 12 times. In general, the recruiters missed these data from the PDF file, as recruiter 6 said:

> *[...] I have the responsibility of looking for the best professionals for open positions. And, to evaluate a developer's experience, in addition to analyzing the profile of the developer, it is important to check other relevant data, such as the number of stars that their projects have on GitHub. This information can indicate the popularity and quality of the work done by the candidate.*

The code "Dev History" means the historical from GitHub of a developer, for example, actual job, country of origin, and complete read-me. The PDF file limits the developer's text and does not present the data about the contact, for example, address or country, when informed by the developer. This limitation from the viewpoint of the recruiters could be better. This code was manifested by 5 (33.33%) participants 8 times. Recruiter 8 said:

> *[...] For me it is interesting to understand the developer's culture, and it is possible to see this from, for example, from the country of origin. Another fascinating data is about the actual job position of the candidate. From this, it is possible to know more details about the candidate. These data need to be presented.*

The code "Seeking Information" represents the necessity of the recruiter to search for more details about the candidate from other social networking, by lack of data from GitHub and PDF presented, for instance, research from LinkedIn. This code was manifested by 5 (33.33%) participants in 6 times. The recruiter P6 said.

> *[...] Sometimes, GitHub does not have reliable data about the candidate, for example, when they are a junior. Supplementing a candidate's profile with additional research, such as reviewing their online presence, including their LinkedIn profile, is essential. The method could support this research to help the recruiter.*

The code "Followers" indicates a developer's popularity and interest in their projects. Generally, a higher number of followers means that a developer has a larger audience and their work is well-regarded within the GitHub community. This code was manifested by 4 (26.66%) participants 4 times. The PDF file generated by the method does not show these data. From the viewpoint of the recruiters, these data are essential, mainly if the job opening is for senior. Recruiter P3 said.

> *[...] The number of followers is a data relative but very important to analyze, mainly if the job opening is for a senior. Overall, senior has the characteristics of a guide to the team, and the number of followers could be a point of attention for us.*

From the interviews, we note the recruiters are interested in analyzing the source-code about the add, delete, and changes made by a developer. This way, we create the "Code Churn". On GitHub, code churn is a metric that measures the number of changes made to a source-code repository. This metric can be used to understand how the code in a project changes over time, helping developers identify issues and opportunities for improvement. This code was manifested by 2 (13.33%) participants 2 times. The PDF file generated by the method does not show this metric. Recruiter P14 said.

> *[...] I believe that Code Churn is a valuable metric for evaluating the quality of a developer's work on GitHub. Code churn can help understand how code evolves over time, identify problems and opportunities for improvement, and even predict quality risks in software projects.*

Table 5. Category Lacking information

Category	**Codes**	**Participants (%)**	**Frequency**
Lacking Information	Star	11 (73%)	12
	Dev History	5 (33%)	8
	Seeking Information	5 (33%)	6
	Followers	4 (27%)	4
	Code Churn	2 (13%)	2
		Total	32

Figure 5 shows the preferences of recruiters in the developer selection process, highlighting the importance of stars for nine recruiters as a key factor in assessing the reputation of a developer. Dev History was a marker of proven experience for three recruiters, while Seeking Information was an indicator of a candidate's initiative in self-improvement for another three recruiters. Notably, two recruiters considered the combination of all three factors to be crucial, suggesting a comprehensive approach to evaluating prospective developers.

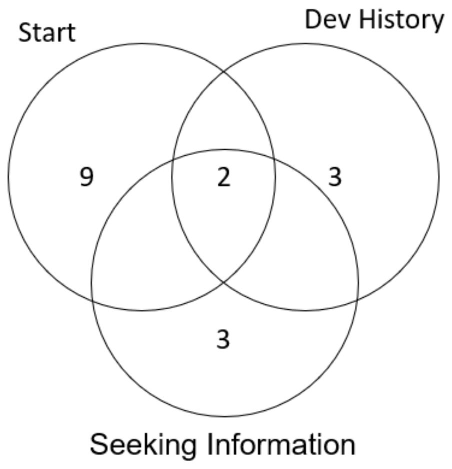


Fig. 5. Viewpoint of the recruiters

RQ4 summary. *The recruiters reported the lacking of data from the PDF file, including information on a developer's popularity, historical data from GitHub, and data about the developer's contact information. The recruiters also expressed the necessity of seeking information from other social networking sites when there is a lack of data from GitHub and PDF. Overall, recruiters believe that improvements in the method could better support their recruitment processes by providing more comprehensive data about developers on GitHub.*

5. THREATS TO VALIDITY

Due to some limitations and threats to the study validity, we need a cautious interpretation of the results of the present study. The results could be different in other geographical and cultural areas. Thus, further work is needed to replicate the results in other geographical areas and software development. In the following, we present the main threats to validity, organized in four typical groups [46].

Construct. To answer our research questions, we asked recruiters open questions. These questions may not cover all data about the recruitment process. However, we scheduled the interview sessions to be relatively long (forty minutes), making sure that we gave the participants enough time to express their ideas and share their thoughts. At the beginning of each interview section, we asked the participants to answer the questions in their own words and provide as much detail as they feel is relevant to address each question. We also placed an open question at the end of the interview to allow the participants to share any additional information about the topic. In addition, interviews were structured to elicit an open expression of deep truths with freedom from leading or biased questions. Neutral and non-influential language was preserved for the whole text in order to present the forum as a common ground for uninfluenced, real opinions and experiences.

Internal. The method we used in our study, as the interviews, can be affected by bias and inaccurate responses. This effect could be intentional or unintentional. We repeatedly and constantly used phrases to encourage the participants to provide their honest opinions, using the phrase "based on your experience" in most of the questions. We also indicated that the participants should "feel free to change/add/delete information or not." Sometimes, we also indicated that "there is no right or wrong answer; we are interested in what you think and your perspective." Using a study pilot, we also took multiple steps to reduce potential confirmation bias [47]. We asked participants to describe their examples of hiring a new software developer. Another threat to the validity of our study is drawing conclusions based on recovered memories [48]. We are interested in capturing recruiters' opinions about what components constitute rationale, independently of how accurate their memories are. We encouraged participants to take their time to recall situations and to report the hire that mattered in their experience. Thus, all interviews were conducted by at least two team members excluding the bias of an individual interviewer influencing the data and the analysis phase involved multiple rounds of independent reviews for ensuring that data were not distorted by the preconceptions.

External. Our studied recruiters may only partially represent part of the employers' population. To mitigate this threat, we recruited a diverse sample of the population with diverse types and amounts of experience. In addition, our interviewees are from three different countries: Brazil, Canada, and United States. Although the data analyzed provide us some clues, we recommend carrying out more studies to confirm and go beyond these results across various geographic and cultural locations. Our results may not generalize beyond the scope of our research. Therefore, another significant area for further research is to explore different settings or subgroups in order to assess the relevance and applicability of our assumptions in other contexts.

Conclusion. The participation of the authors who followed the Grounded Theory procedures poses another threat. Their beliefs might have caused some distortions when interpreting the data. To mitigate this threat, the Grounded Theory coding activities were shared with the other co-authors. Moreover, the identification of the constructs and the depicting of propositions were performed separately by the first author and other co-authors. In fact, three authors participated in the Grounded Theory procedures independently; then we merged their results to shape the theory. Thus, the contents were compared and discussed by the researchers until reaching a consensus.

6. Related Work

In this section, we review previous studies that have investigated hard skills of software developers, as well as the recruitment process for these developers through GitHub. Understanding the skills that are important for software development and how they are evaluated can help organizations make better recruitment decisions and ensure they are hiring the best talent for their teams. Additionally, exploring the recruitment process on GitHub can clarify how developers are discovered and selected by companies, and provide insights into the evolving landscape of software development.

Constantinou and Kapitsaki [49] proposed an approach in two-step. The first step was to gauge developers' GitHub commit activity by looking at both the volume and consistency of their long-term contributions to separate projects. The second phase involved comparing the created developers' expertise profiles to known StackOverflow answering activity using a data set of individuals who were active on both StackOverflow and GitHub. On the contrary side, our research is about the recruitment process which deals with the problems recruiter face in understanding and evaluating the quality of programmers. We provide a more comprehensive and integrative GitHub analysis, including the qualitative information that the quantitative analysis of contributions may lose.

Marlow and Dabbish [13] investigated how employers evaluate developers based on their GitHub accounts, which give hints about their activities, talents, motivations, and values. These indicators were thought to be more trustworthy than resumes. Open source contributions, according to employers surveyed by Marlow and Dabbish [13], are a sign that a developer has the correct values and is not just in the industry for the money. Also, it was stated that contributions to high status projects showed some level of proficiency. While the authors in the first place stress employer's perspective, our aim is to engage them in the process of detailed methodology of analyzing GitHub profiles to support the process of selection. We apply a systematic and sequential approach which increases the degree of objectivity as it helps recruiters to make the right decisions by screening relevant and surplus information of profiles.

Similar research was done by Singer et al. [50]. However, they focused on the social media accounts of developers in general. These include Stack Overflow, Twitter, Coderwall, and Masterbranch, as well as GitHub and other profile aggregators. They discovered that some recruiters may find profiles challenging to comprehend or assess. A developer's public activities must always be judged against their actual artifacts, such as their code, tests, documentation, or debates, in order to be truly evaluated. Also, developers evaluated employers and organizations to determine their legitimacy and acceptability as partners. The majority of developers and recruiters are aware that a developer's lack of activity does not necessarily indicate something significant; they may simply be a private developer or focus only on closed source projects. Singer et al. [50] discuss the evaluation of a broad range of social media accounts, our study focuses specifically on GitHub profiles, offering a focused and detailed analysis. We develop a method that could assist recruiters in overcoming the challenges highlighted by Singer et al. by providing analytical tools to accurately assess technical skills and reduce the subjective interpretation often associated with social media profiles.

The results of a study done to examine employers' experiences in employing computer science graduates for software startup companies are presented by Adnin et al. [51]. For this study, 23 software businesses' employers were interviewed. The findings demonstrate that employers have difficulty finding qualified people because they lack the real-world experience and specialized skill sets needed for startup jobs. The importance of soft skills like communication, teamwork, and a willingness to learn was also emphasized by the companies.

In our previous work [23], we investigated the use of GitHub to check hard skills. We surveyed experienced developers to investigate their opinions about the ability of source-code analysis to indicate programming skills. The results showed that source-code analysis is valuable for assessing programming skills. However, it should complement other assessment forms, such as technical interviews and code reviews. We discuss practical implications and future research for using source-code analysis in assessing programming skills.

Montandon et al. [31] used a combination of social network analysis, machine learning, and expert identification techniques to identify the top contributors and experts in specific libraries and frameworks. They first extracted data from GitHub repositories related to the libraries and frameworks, such as user profiles, contributions, and interactions. Then, they constructed a social network of developers based on their interactions, and applied machine learning techniques to classify developers as experts or non-experts based on their contributions and interactions. The authors evaluated their methodology on four popular libraries and frameworks: React, Angular, Vue.js, and Express.js. They found that their methodology was able to identify experts with high precision and recall, and that the identified experts had a higher impact and activity level compared to non-experts. The paper concludes that the proposed methodology can be useful for various applications, such as recommending experts for code review, identifying potential contributors, and analyzing the evolution of the library or framework over time. Contrasting with Montandon et al. [31], our study uses a direct recruitment lens to enhance the hiring process, emphasizing the practical application of similar methodologies to streamline the recruitment process and improve the accuracy of candidate evaluations in real-world settings.

Our paper focuses on investigating the process of detecting experts in software development, improving the recruitment process, and exploring the challenges and problems of current methods. Unlike the related work mentioned, this paper interviewed recruiters of software developers in three countries: Brazil, the United States, and Canada. The study aimed to gain insights into improving the detection of experts, creating a good curriculum, and addressing the challenges faced in the recruitment process. The proposed methodology in this paper can help to compute programming skills and can be used to improve the recruitment process of the software developer.

7. Conclusion and Future Work

Selecting the right software developer is crucial for any organization that wants to develop a software product with quality and on time. A good software developer not only possesses the hard skills required to write efficient and effective code, but also can understand the project requirements and work collaboratively with other development team members. Additionally, a qualified developer can help identify potential problems early in development, saving time and resources in the long run. The importance of selecting a good developer is particularly evident in the fast-paced and ever-changing software industry, where keeping up with new technologies and methodologies is essential. Investing time and resources in recruiting and retaining talented software developers can make the difference between a successful software product and a failed project.

In this study, we aimed to gain insight into the perceptions of software recruiters from Brazil, United States, and Canada on detecting software experts, crafting an effective resume, and the challenges and problems associated with current recruitment methods. Through interviews with recruiters from various organizations, we identified valuable strategies to assess potential candidates and their skills. We highlighted the importance of a well-crafted resume, which should include clear and concise information about the candidate's experience, projects, and contributions. Our research also identified challenges recruiters face in evaluating candidates, such as assessing hard skills, needing more standardization in the recruitment process, and time constraints. Our study provides practical guidance for recruiters and developers to improve their recruitment processes and increase their chances of success in the highly competitive software industry.

Based on the findings from this study, we recommend that organizations invest in advanced software tools that can analyze GitHub profiles for coding quality, project diversity, and collaboration patterns, offering a more comprehensive view of candidate capabilities. Developing standardized protocols for evaluating both hard and soft skills ensures a consistent and fair assessment process across all candidates, including structured interviews, standardized coding tests, and systematic review of candidates' project portfolios.

The recruitment process for software development positions can be improved by considering candidates' hard skills and soft skills. Future work could investigate the most important hard skills for software developers in different domains, such as web development, mobile development, or data science. Additionally, the relationship between hard skills and project success could be better explored to understand the impact of technical expertise on project outcomes. The research could examine software development teams' most valuable soft skills, such as communication, teamwork, and adaptability. Developing assessment tools and methods for soft skills is also an area of future research. Furthermore, investigating the relationship between hard and soft skills in the recruitment process could provide insight into the optimal balance of technical and non-technical skills for successful software development teams.

Besides that, another future work possibility is adding the machine learning methods in order to identify software engineer skills on GitHub as a predicting methods. Machine learning comes into play by making it possible to automatically classify and predict developers' expertise levels considering their coding patterns, commit records, and project involvement. For example, supervised learning algorithms could be trained on collections of labeled datasets where developers are divided by different skill levels, from which the system could learn relevant complexity features extracted from the developers' interactions with the repositories. Additionally to this, natural language processing can be used to analyze commit messages and comments for the purpose of obtaining information about developers' problem-solving skills and collaborative teamwork. Such method would not only increase the precision of the skills test but also provide scalability and molding to new trends in software development niche.

References

[1] S. Gupta, H. K. Singh, R. D. Venkatasubramanyam and U. Uppili, Scqam: A scalable structured code quality assessment method for industrial software, in *Proceedings of the 22nd International Conference on Program Comprehension*, *ICPC 2014*, (Association for Computing Machinery, New York, NY, USA, 2014), p. 244–252.

[2] V. Veloso and A. Hora, Characterizing high-quality test methods: A first empirical study, in *Proceedings of the 19th International Conference on Mining Software Repositories*, *MSR '22*, (Association for Computing Machinery, New York, NY, USA, 2022), p. 265–269.

[3] A. C. D. Batista, R. M. de Souza, F. Q. B. da Silva, L. de Almeida Melo and G. Marsicano, Teamwork quality and team success in software development: A non-exact replication study, in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, *ESEM '20*, (Association for Computing Machinery, New York, NY, USA, 2020).

[4] A. N. Meyer, T. Zimmermann and T. Fritz, Characterizing software developers by perceptions of productivity, in *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, *ESEM '17*, (IEEE Press, 2017), p. 105–110.

[5] A.-N. Mehdi, P. Urso and F. Charoy, Evaluating software merge quality, in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, *EASE '14*, (Association for Computing Machinery, New York, NY, USA, 2014).

[6] H. Zhang, H. Huang, D. Shao and X. Huang, Fireteam: A small-team development practice in industry, in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *ESEC/FSE 2020*, (Association for Computing Machinery, New York, NY, USA, 2020), p. 1365–1375.

[7] D. Chinn and T. VanDeGrift, Gender and diversity in hiring software professionals: What do students say?, in *Proceedings of the Fourth International Workshop on Computing Education Research*, *ICER '08*, (Association for Computing Machinery, New York, NY, USA, 2008), p. 39–50.

[8] S. A. ao and E. Insfran, Evaluating software architecture evaluation methods: An internal replication, in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, *EASE'17*, (Association for Computing Machinery, New York, NY, USA, 2017), p. 144–153.

[9] S. Kourtzanidis, A. Chatzigeorgiou and A. Ampatzoglou, Reposkillminer: Identifying software expertise from GitHub repositories using natural language processing, in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, *ASE '20*, (Association for Computing Machinery, New York, NY, USA, 2021), p. 1353–1357.

[10] L. Zhang and C. Wang, Rclassify: Classifying race conditions in web applications via deterministic replay, in *Proceedings of the 39th International Conference on Software Engineering*, *ICSE '17*2017, p. 278–288.

[11] F. Fagerholm, M. Ikonen, P. Kettunen, J. Münch, V. Roto and P. Abrahamsson, How do software developers experience team performance in lean and agile environments?, in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, *EASE '14*, (Association for Computing Machinery, New York, NY, USA, 2014).

[12] S. Amreen, A. Karnauch and A. Mockus, Developer reputation estimator (dre), in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*, *ASE '19*, (IEEE Press, 2020), p. 1082–1085.

[13] J. Marlow and L. Dabbish, Activity traces and signals in software developer recruitment and hiring, in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, *CSCW '13*, (Association for Computing Machinery, New York, NY, USA, 2013), p. 145–156.

[14] A. S. M. Venigalla and S. Chimalakonda, Understanding emotions of developer community towards software documentation, in *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Society*, *ICSE-SEIS '21*, (IEEE Press, 2021), p. 87–91.

[15] Y. Wu, J. Kropczynski, P. C. Shih and J. M. Carroll, Exploring the ecosystem of software developers on GitHub and other platforms, in *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work amp; Social Computing*, *CSCW Companion '14*, (Association for Computing Machinery, New York, NY, USA, 2014), p. 265–268.

[16] S. L. Vadlamani and O. Baysal, Studying software developer expertise and contributions in stack overflow and GitHub, in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 312–323.

[17] C. Hauff and G. Gousios, Matching GitHub developer profiles to job advertisements, in *Proceedings of the 12th Working Conference on Mining Software Repositories*, *MSR '15*, (IEEE Press, 2015), p. 362–366.

[18] C. Brown and C. Parnin, Understanding the impact of GitHub suggested changes on recommendations between developers, in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *ESEC/FSE 2020*, (Association for Computing Machinery, New York, NY, USA, 2020), p. 1065–1076.

[19] V. N. Subramanian, An empirical study of the first contributions of developers to open source projects on GitHub, in *Proceedings of the ACM/IEEE 42nd International*

Conference on Software Engineering: Companion Proceedings, *ICSE '20*, (Association for Computing Machinery, New York, NY, USA, 2020), p. 116–118.

[20] J. T. Liang, T. Zimmermann and D. Ford, Towards mining oss skills from GitHub activity, in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, *ICSE-NIER '22*, (Association for Computing Machinery, New York, NY, USA, 2022), p. 106–110.

[21] D. W. McDonald and M. S. Ackerman, Expertise recommender: A flexible recommendation system and architecture, in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, *CSCW '00*, (Association for Computing Machinery, New York, NY, USA, 2000), p. 231–240.

[22] D. Schuler and T. Zimmermann, Mining usage expertise from version archives, in *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, *MSR '08*, (Association for Computing Machinery, New York, NY, USA, 2008), p. 121–124.

[23] J. Oliveira, M. Souza, M. Flauzino, R. Durelli and E. Figueiredo, Can source code analysis indicate programming skills? a survey with developers, in *Quality of Information and Communications Technology*, eds. A. Vallecillo, J. Visser and R. Pérez-Castillo (Springer International Publishing, Cham, 2022), pp. 156–171.

[24] F. Rafii, S. F. H. Oskouie and Y. Nikravesh, Grounded theory data analysis, *Iran Journal of Nursing* **16** (2003) 43–49.

[25] A. Sarma, X. Chen, S. Kuttal, L. Dabbish and Z. Wang, Hiring in the global stage: Profiles of online contributions, in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 2016, pp. 1–10.

[26] F. Q. da Silva, C. Costa, A. C. C. França and R. Prikladinicki, Challenges and solutions in distributed software development project management: A systematic literature review, in *2010 5th IEEE International Conference on Global Software Engineering*, 2010, pp. 87–96.

[27] B. Al-Ani, M. J. Bietz, Y. Wang, E. Trainer, B. Koehne, S. Marczak, D. Redmiles and R. Prikladnicki, Globally distributed system developers: Their trust expectations and processes, in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, *CSCW '13*, (Association for Computing Machinery, New York, NY, USA, 2013), p. 563–574.

[28] J. Jia, Z. Chen and X. Du, Understanding soft skills requirements for mobile applications developers, in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, **1**2017, pp. 108–115.

[29] A. Mockus and J. Herbsleb, Expertise browser: a quantitative approach to identifying expertise, in *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, 2002, pp. 503–512.

[30] J. Marlow and L. Dabbish, Activity traces and signals in software developer recruitment and hiring, in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, *CSCW '13*, (Association for Computing Machinery, New York, NY, USA, 2013), p. 145–156.

[31] J. ao Eduardo Montandon, L. L. Silva and M. T. Valente, Identifying experts in software libraries and frameworks among GitHub users, in *Proceedings of the 16th International Conference on Mining Software Repositories*, *MSR '19*, (IEEE Press, 2019), p. 276–287.

[32] M. Muratbekova-Touron and G. Galindo, Leveraging psychological contracts as an hr strategy: The case of software developers, *European Management Journal* **36**(6) (2018) 717–726.

[33] M. Cherubini, G. Venolia, R. DeLine and A. J. Ko, Let's go to the whiteboard: How and why software developers use drawings, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *CHI '07*, (Association for Computing Machinery, New York, NY, USA, 2007), p. 557–566.

[34] T. Bhasin, A. Murray and M.-A. Storey, Student experiences with GitHub and stack overflow: An exploratory study, in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, IEEE2021, pp. 81–90.

[35] N. Assyne, Hard competencies satisfaction levels for software engineers: a unified framework, in *Software Business: 10th International Conference, ICSOB 2019, Jyväskylä, Finland, November 18–20, 2019, Proceedings 10*, Springer2019, pp. 345–350.

[36] M. Bloor, *Focus groups in social research* (Sage, 2001).

[37] M. Behroozi, C. Parnin and C. Brown, Asynchronous technical interviews: Reducing the effect of supervised think-aloud on communication ability, in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *ESEC/FSE 2022*, (Association for Computing Machinery, New York, NY, USA, 2022), p. 294–305.

[38] A. M. Gill and B. Nonnecke, Think aloud: Effects and validity, in *Proceedings of the 30th ACM International Conference on Design of Communication*, *SIGDOC '12*, (Association for Computing Machinery, New York, NY, USA, 2012), p. 31–36.

[39] E. L. Olmsted-Hawala, E. D. Murphy, S. Hawala and K. T. Ashenfelter, Think-aloud protocols: A comparison of three think-aloud protocols for use in testing data-dissemination web sites for usability, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *CHI '10*, (Association for Computing Machinery, New York, NY, USA, 2010), p. 2381–2390.

[40] A. L. Strauss and J. M. Corbin, *Basics of qualitative research: techniques and procedures for developing grounded theory* (Sage Publications, Thousand Oaks, Calif, 1998).

[41] R. Hoda, Decoding grounded theory for software engineering, in *Proceedings of the 43rd International Conference on Software Engineering: Companion Proceedings*, *ICSE '21*, (IEEE Press, 2021), p. 326–327.

[42] J. Oliveira, M. Souza and E. Figueiredo, Evaluating a method to select software developers from source code analysis (2023).

[43] M. Bärring, D. Nåfors, D. Henriksen, D. Olsson, B. Johansson and U. Larsson, A vsm approach to support data collection for a simulation model, in *Proceedings of the 2017 Winter Simulation Conference*, *WSC '17*, (IEEE Press, 2017).

[44] S. B. Merriam and E. J. Tisdell, *Qualitative research: A guide to design and implementation* (John Wiley & Sons, 2015).

[45] H. J. Rubin and I. S. Rubin, *Qualitative interviewing: The art of hearing data* (sage, 2011).

[46] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in software engineering* (Springer Science & Business Media, 2012).

[47] R. Pohl and R. Pohl, *Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory* (Psychology Press, 2004).

[48] A. Koriat, M. Goldsmith and A. Pansky, Toward a psychology of memory accuracy, *Annual Review of Psychology* **51**(1) (2000) 481–537, PMID: 10751979.

[49] E. Constantinou and G. M. Kapitsaki, Identifying developers' expertise in social coding platforms, in *42th Euromicro Conf. on Software Engineering and Advanced Applications (SEAA)*, (IEEE, Limassol,Cyprus, 2016), pp. 63–67.

[50] L. Singer, F. F. Filho, B. Cleary, C. Treude, M.-A. Storey and K. Schneider, Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators, in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, *CSCW '13*, (Association for Computing Machinery, New York, NY, USA, 2013), p. 103–116.

[51] R. Adnin, S. Afroz, M. Ulfat and A. Iqbal, A hiring story: Experiences of employers in hiring cs graduates in software startups, in *Companion Publication of the 2022 Conference on Computer Supported Cooperative Work and Social Computing*, *CSCW'22 Companion*, (Association for Computing Machinery, New York, NY, USA, 2022), p. 126–129.