Hiring is Broken: What Do Developers Say About Technical Interviews?

Mahnaz Behroozi
North Carolina State University
Raleigh, NC, USA

Chris Parnin
North Carolina State University
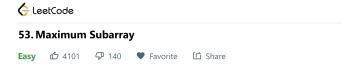Raleigh, NC, USA

Titus Barik
Microsoft
Redmond, WA, USA

Abstract—Technical interviews—a problem-solving form of interview in which candidates write code—are commonplace in the software industry, and are used by several well-known companies including Facebook, Google, and Microsoft. These interviews are intended to objectively assess candidates and determine fit within the company. But what do developers say about them?

To understand developer perceptions about technical interviews, we conducted a qualitative study using the online social news website, Hacker News—a venue for software practitioners. Hacker News posters report several concerns and negative perceptions about interviews, including their lack of real-world relevance, bias towards younger developers, and demanding time commitment. Posters report that these interviews cause unnecessary anxiety and frustration, requiring them to learn arbitrary, implicit, and obscure norms. The findings from our study inform inclusive hiring guidelines for technical interviews, such as collaborative problem-solving sessions.

Index Terms—diversity and inclusion, Hacker News, programming, software engineering, technical interviews, whiteboard

PRELUDE

Let's begin with a technical interview problem. Consider the following coding question from LeetCode,[1] an online platform for preparing software development candidates for interviews:

LeetCode

53. Maximum Subarray

Easy   👍 4101   👎 140   ❤ Favorite   ⬆ Share

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

```
Input: [-2,1,-3,4,-1,2,1,-5,4],
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

[1] You can solve the problem interactively at https://leetcode.com/problems/maximum-subarray/. The provably optimal solution to this question—called Kadane's algorithm—is described in Bentley's 1984 column, Programming Pearls [8]. The column presents various solutions to this question with cubic, quadratic, and linear time complexities. Bentley notes that the problem "is really a 'toy'—it was never incorporated into a system."

Before going further—and regardless of your coding proficiency—we'd like you to spend a few minutes and take a stab at this question.

Well, how did it go? Did you find an $O(n)$ solution?

Developers within the LeetCode community report that—within the past six months—this coding question has been used in technical interviews at well-known software companies such as Apple, Amazon, Microsoft, Google, Facebook, and Uber.

I. INTRODUCTION

A technical interview for software development consists of one or more stages within the interview life cycle [3], [26]. It begins with an initial screening of the candidate, usually conducted over the phone or through an online coding platform, such as CoderPad,[2] Skype Interviews,[3] and interviewing.io.[4] Depending on their performance, the candidate may be invited for an on-site visit. This on-site visit consists of a series of one-on-one interviews (each 45 minutes to an hour, over a period of half a day to several days) with engineers, and sometimes managers. The interviews primarily focus on technical coding or algorithms, either on the whiteboard or using a simple text editor on a computer. In other words, technical interviews are primarily a test of the candidates' problem-solving or "analytical ability" [26]. And if all goes well, the candidate can expect to receive an offer.

For hiring managers, technical interviews have considerable appeal [23]. First, hiring managers are able to ask questions directly from their companies' question bank, instead of having to design their own interview questions. Second, the format purports to reduce variation between interviewers and teams, since candidates can be evaluated through objective scoring criteria. Finally, the interview process becomes scalable: new interviewers are straight-forward to train, and these interviewers can interchangeably ask coding questions to *any* software engineering candidate. All of these, in theory, result in a more-or-less standardized and meritocratic technical interview pipeline.

[2] https://coderpad.io
[3] https://www.skype.com/en/interviews/
[4] https://interviewing.io

The collective experiences of interview candidates, however, appear to tell a very different story—even a cursory glance reveals that "technical interviews are broken" [41], that they are an "antagonistic" [38] form of high-pressure "whiteboard algorithm hazing" [18] that have "nothing to do with real day-to-day developer work" [22], are "humiliat[ing] professionally" [35], assess candidates through an "algorithm question lottery [of] luck and chance" [44], and require substantial "upfront investment" from the candidate [37] to learn the "cultural norms necessary to get themselves into a desk at a technology firm" [29]. Still others argue that technical interviews may even "promote exclusion and discrimination, serving only as a barrier to entry for qualified underrepresented candidates" [1].

The goal of this paper is to take meaningful, personal, and yet disjoint anecdotes such as these—and amplify them into a principled theoretical foundation to support research towards improving technical interviews in software development. To that end, we conducted a qualitative study in which we obtained over forty-six thousand authored comments from Hacker News—a social website for software practitioners focusing on computer science, software development, and entrepreneurship—pertaining to the topic "interviews." We framed these comments through the analytical lens of small stories [4]—stories of their personal experiences and their past events—and through thematic analysis [11] identified *concerns* software developers have about technical interviews. The contribution of this paper is a state-of-the-practice synthesis of concerns from the Hacker News community about technical interviews for software developers, reflected through their own words.

Our analysis of Hacker News identifies several concerns in current software engineering practices with regards to technical interviews. Though hiring managers justify these practices as being meritocratic, our findings suggest that candidates perceive these practices as subjective, arbitrary, unnecessarily stressful, non-inclusive—and at times—demeaning to their sense of self-worth and self-efficacy. We propose guidelines to make hiring more inclusive and equitable without sacrificing interviewing effectiveness, for example, providing candidates with explicit evaluation criteria in advance.

II. METHODOLOGY

Research context. We used Hacker News, a social website for software practitioners, to conduct our investigation. As a community, Hacker News contains over 1.5 million user-submitted comments on a variety of cultural and technical topics of significance to the hacker community (for example, "F.C.C. Repeals Net Neutrality Rules," "CIA malware and hacking tools," and "How to Pass a Programming Interview," to convey a sense of the diversity of topics). Wu and colleagues [42], through a survey with software developers who use GitHub, found that Hacker News serves as an important venue for software developers to exchange ideas as part of a broader cultural ecosystem. Barik and colleagues [6]

conducted a formative study using Hacker News to demonstrate that investigations within the online community can yield insights into qualitative research topics, for example to understand how software developers interpret programming and play [5]. We adopt this approach to investigate concerns with technical interviews.

Data collection. We used the Algolia[5] search engine API, which indexes all of Hacker News, to retrieve JSON-formatted topics containing "interview." The results were sorted by popularity, and a limitation of this search engine is that it returns a maximum of 1,000 results—which it did.

Data cleaning. We automatically filtered the results with some standard heuristics, such as "interview with," as these topics tend to be about interviews with people, not about the activity of technical interviews. This procedure removed 262 topics from consideration. Two authors then independently went through the remaining topics manually, and, using the title alone, excluded topics that were not related to technical interviews (Cohen's $\kappa = 1$; these topics are easy to identify but difficult to write a systematic expression for, for example, "Interviewing my mother, a mainframe COBOL programmer," or "AT&T CEO interrupted by a robocall during a live interview"). After data cleaning, 456 topics remained, containing a total of 46,115 comments.

Characterizing the data on technical interviews. The relevant topics on technical interviews spanned the time period from March 3, 2008 through April 25, 2019. The least popular topic had 29 points (essentially, votes), and the most popular topic had 1020 points ($u = 146$, $sd = 151$). The number of comments per topic ranged from 0 to 997 ($u = 101$, $sd = 104$). Points also strongly correlate with comments ($r = 0.78$), such that more popular topics tend to have more comments. To get a high-level sense of the diversity of topics on technical interviews within the Hacker News community, Table I presents a list of the most polarizing topics. To obtain this list, we applied a rule-based sentiment analysis tool, called VADER [19], to the titles of the Hacker News topics. 86 of the titles had a positive polarity (greater than 0 and less than or equal to 1, "Best interview questions to spot ideal employees"), and 98 of the titles had a negative polarity (less than 0 and greater than or equal to -1, "Programmers are confessing their sins to protest a broken job interview process"). For the remaining 272 posts, VADER did not identify a polarity in either direction (0, "The GitHub Job Interview"). In short, the Hacker News community has quite a bit to say about technical interviews.

Qualitative analysis. We imported the Algolia JSON comments for the top five topics by posted comments (or approximately 10% of the data in a Pareto distribution) into the ATLAS.ti data analysis software.[6] The topics are: "The latest trend for tech interviews: Days of unpaid homework" (997), "I interviewed at six top companies in Silicon Valley in six days" (692 comments), "How to Pass a Programming

Title	Points	Comments	Polarity[1]
Most Negative Titles			
Programmers are confessing their sins to protest a broken job interview process	303	226	-0.80 🔴
UBER ATC is disguising research as fake job interviews?	133	31	-0.75 🔴
Ask HN: Failed interview, feeling unemployable and depressed—what do I do?	377	234	-0.73 🔴
My Day Interviewing for the Service Economy Startup from Hell	604	274	-0.68 🔴
The Programming Interview from Hell	174	136	-0.68 🔴
Most Positive Titles			
Best interview questions to spot ideal employees	41	22	0.82 🟢
Thanks HN: Developers and YC companies video speed interview for free	38	12	0.74 🟢
Ask HN: What should an ideal developer interview process look like?	261	278	0.70 🟢
Ask HN: What are good tech jobs that don't require being good at interviewing?	92	84	0.70 🟢
Video games beat interviews to recruit the very best	129	122	0.67 🟢

[1] Polarity computed by VADER, with scores ranging from -1 (most negative) to 1 (most positive).

Concerns about...	Description
RELEVANCE (Section III-A)	Problem-solving is not grounded in real-world code, constraints, or scenarios.
ANXIETY (Section III-B)	Stress associated with problem-solving in conjunction with time pressure and surveillance by the interviewer.
AFFECT (Section III-C)	Emotions, such as frustration, and humiliation associated with the technical interview experience.
AFFORDANCES (Section III-D)	A lack of naturally occurring resources and mechanisms typically available in a normal programming environment. For example, coding with a marker or within a word document.
PRACTICE (Section III-E)	Time commitment needed to practice ("grind") various problems and solutions in order to be competitive with other candidates.
MISSING OUT (Section III-F)	Evaluation criteria and proxies that filter out candidates, unfairly.

isk with the comment's identifier in the following URL: `https://news.ycombinator.com/item?id=*`.

III. CONCERNS FROM CANDIDATES

In this section, we present expressions of concern about technical interviews from candidates, organized through themes. The complete list of concerns is found in Table II.

A. Relevance

"Building a great and useful app rarely requires Herculean feats of logic and puzzle solving," says $HN_{18944553}$. Indeed, adds $HN_{18945198}$, "the number of times I've seen things like dynamic programming come up in a real world application are vanishingly small." $HN_{18944499}$ describes an interview experience: "I'm a data scientist, and Google asked me to sum all values of nodes at each height of a tree. I had to implement the tree, BFS, and the algo (which was easy once you have BFS) in 25 minutes, minus any talky time. BFS is not something I thought about much in the last 5 years, and quite frankly could care less about. I got stuck when I knew I needed 'something' to finish implementing BFS, but couldn't remember and the Google interviewer offered no help."

These expressions capture the discrepancy between the skills needed for performing a software development job and the problems they are asked to be solved to get that job. $HN_{18946686}$ elaborates, "this then becomes representative of your experience in spite of the fact that you are never likely to be confronted with that kind of problem with that kind of time-frame. Whatever is on your CV, and whatever you can say about what you've learned over the years, becomes totally irrelevant in the face of that."

A noteworthy undercurrent we found within the discussion of relevance is the view that developers should be able to solve technical interview problems "from scratch" or find them to be enjoyable: "If you're a programmer, you like to solve puzzles,"

Interview" (552 comments), "I will not do a tech interview" (545 comments), "'Clean your desk': My Amazon interview experience" (509 comments).

We conducted coding over multiple iterations. In the first cycle, we used descriptive coding, and assigned short codes and labels to capture and summarize the salience of the comments [32]. We framed these comments through "small stories" [4]—an epistemological lens that permits analysis of small vignettes of everyday stories and experiences—such as Hacker News comments [5]. In the second iteration, we conducted a thematic analysis to organize the comments into *concerns* [11].

To further characterize the themes, we performed an additional purposive sampling, or non-probabilistic sampling, on comments in technical interview threads from Hacker News and authored memos [9]. These memos captured interesting exchanges that promoted depth and credibility of the concerns, and framed the posters' concerns through their self-reported experiences. That is, the memos provide a thick description to contextualize the findings [31].

Supporting verification. In this paper, quotations from Hacker News are referred as $HN_{identifier}$. Each post can be accessed on Hacker News by substituting the aster-

says $HN_{11248511}$. $HN_{18944989}$ explains their belief, "I wrote a custom HashMap for Java, used BFS and DBS on graphs, wrote a top down custom parser, custom string searching algorithms to solve real business problems. I would rather work with someone who knows how neural networks really work rather than [someone] who knows pytorch or keras, because they can be learned rather easily. In some industries knowledge of algorithms can be more valuable than knowing a myriad of frameworks that change every few years any way." Finally, $HN_{18944164}$ offers that relevance could vary based on experience: "There are career stages. For candidates just out of school or with little experience, asking algorithmic questions totally makes sense. For more senior candidates (who actually progressed to the next stage, not just spent a lot of years), the questions become more real-life, more open-ended and with more than one (or sometimes none) *correct* answer."

B. Anxiety

"The problem is that interviews are high stress affairs," says $HN_{6251756}$. They continue, "this is great when you've got to climb a tree to get away from a tiger. This is horrible if you are trying to demonstrate your ability to function mentally. It does not matter how reasonable your questions are. If this is what someone faces, you do not get an accurate picture of how good anyone with interview anxiety is. And a lot of people suffer from this." $HN_{6253596}$ also describes anxiety that arose during an interview with their own team: "I once interviewed for a job within the company I already worked for [with] my existing boss, and a colleague." Then, "I was asked a question about ASP.NET ViewState which I drew a complete blank on. I just couldn't wrap my ahead around the question because of nerves. It was only when my boss reminded me that this was something I had actually taught him and my colleague only 6 months ago that my nerves cleared, I relaxed, and then I could suddenly think clearly again."

Having stress can cause disruptive effects on memory [13], [21], [24], [30] which can negatively affect candidates' performance during interviews, even with people they already know: "The interview process creates a dynamic relationship between interviewer and interviewee that never exists between colleagues and bosses," continues $HN_{6253596}$. One possible cause of anxiety during the interview is being watched and judged by the interviewer. $HN_{6252459}$ says, "I also get very anxious during exams and always make silly mistakes. It saddens me to see some interviewers on this thread that naively believe that the kind of stress a person feels during an exam (especially one where the tester is sitting in front of you and watching your every move) is the same kind of stress you might experience while trying to fix some problem in front of your computer." Some developers were not sympathetic with the concern of anxiety and $HN_{6253513}$ believes the problem is fixable: "I think the real problem is in your head: your anxiety about job interviews is sabotaging something you're otherwise perfectly good at," and continues by adding "You need to be more zen about this. More laid back, relaxed, confident, or something like that."

C. Affect

"I think it's offensive and I don't like how the industry has standardized on basically assuming everyone's a bullshitter," says $HN_{18946686}$, adding, "What's worse is that you'll have to repeat this over and over again for any company you interview with." Sometimes candidates see themselves as being a gladiator fighting in the Colosseum for entertainment. "But made-up *puzzles*? For which the asker already knows the answer, so they sit back and watch us dance?," questions $HN_{1260547}$.

Some candidates perceive that companies try to negatively affect their self-esteem by conveying that their standards are way higher than candidates' abilities. $HN_{11248119}$ complains about this with saying: "I have come to believe it is part of an industry-wide negging style to keep people in their place." $HN_{11251499}$ also shares a related story: "it is like some companies like to feel special. After being invited twice for Google interviews, I mean really invited by their HR, not me applying for them. On both occasions I failed the process with their stupid questions. I started replying to their HR, if I am so good to be invited but in their eyes unable to devise a inode search algorithm for unlimited hard disk sizes with a specific set of hardware and search time constraints, over the phone interview, then why couldn't they just please stop inviting me!? That was the last time I heard from them and I don't care a bit about it."

Candidates also report concerns with unveiled emotion from interviewers as well as casual and open disinterest. $HN_{11251441}$ recalls, "I was literally asked, 'What is the time complexity of the moving window average algorithm over an array?' and when I asked for clarification, I could hear an edge of... I guess frustration in my interviewer's voice. Granted, by this time, we'd been through a couple of other problems, and time was running short, but I still think it was pretty unprofessional of the interviewer to let frustration or any other sort of negative emotion show during the interview." $HN_{6245567}$ adds, "the thing I found strangest is that some interviewers would walk in the room and throw up a coding exercise without any introduction at all. They literally wouldn't give their names and what projects they worked on." However, some candidates also have had happy experiences. "I've interviewed for Google and Apple internships and the process has been extremely pleasant, with the interviewers happy to give their time. With Apple, I got to meet the entire team and spend time with them. I've heard similar stories about the Microsoft interview process (I mean, they even fly you out to Redmond)" ($HN_{13131599}$).

D. Affordances

"Expecting perfectly correct code on a whiteboard seems to me to be a slight abuse of the medium. Whiteboards and chalkboards specifically exist to sketch things out in an adhoc fashion, often in a collaborative and easy-to-edit way," says $HN_{1247743}$. Another developer adds, "every time an interviewer has told me something like this, they then nitpick syntax and appear to be primarily concerned with 'does my whiteboard code compile' sorts of problems" ($HN_{17728663}$).

Affordance issues also extended to interviews over the phone. "I've had that experience during a technical phone screen with a different 'hires only the best' company. I was asked to write (over the phone) a trivial statistical algorithm and started to describe the algorithm: 'Function F returns a double and has two parameters, pointer to the start of the double array P and integer N for length of array.' Apparently on the other end of the line was a human compiler that kept rejecting my input and preferred 'double F open parens double star P comma int n close parens'!" ($HN_{17727794}$).

Candidates were concerned that mediums such as whiteboards, shared Google documents, and phone communication did not take advantage of their skills built over many years typing in computers and IDEs, with aids such as syntax highlighting, and auto-completion. "I wouldn't pass then since I live in post 2000 and am used to letting the IDE handle the nitty gritty details while I focus on the actual meat of creating software," says $HN_{11247663}$. $HN_{11247490}$ adds, "This happened to a friend. He confirmed that pseudocode would be acceptable, but then as he was writing it out the interviewer got on him about not terminating lines with semicolons (I suppose the pseudocode looked C-ish)."

Some developers try to circumvent this concern. $HN_{11247779}$ suggests, "You can always preempt the whiteboard issue by bringing a laptop along. *Hey, I'm a lot more comfortable writing code on a keyboard and with an IDE. Let's program this together in a text editor instead of a whiteboard.*" But, $HN_{11247895}$ cautions, "Speaking as an interviewer, don't do this to me without prior discussion. Being able to discuss things on a whiteboard is a necessary skill for working in a co-located office. This includes pseudocode."

E. Practice

"*It's not about getting the right answer but the way you think.* I've never found that to be true. If you don't get to the right answer, you're gone. If they planned to ask two questions and you only got through one, you're gone no matter how you 'thought' about it. A huge part of this is LeetCode practice. If you can't solve most algorithm questions on a whiteboard in less than an hour (because you haven't practiced) then you won't pass any interviews," warns $HN_{17729934}$. I never

Candidates were concerned about the time commitment required to practice and/or memorize algorithms. "Now that I am 51, I feel annoyed that all of these stories of interviews involve asking questions about algorithms that rarely come up in real coding," laments $HN_{18944082}$. They continue, "I cannot spend hours and hours studying up on these algorithms, there are much more important things (real coding-related things) which I need to learn about, to the extent I have time to do that." $HN_{11249643}$ adds, "To me it was a bunch of rote memorization, just like a biology course. I never—*never*—have needed to know how bubblesort/heapsort/mergesort actually work, except to appease interviewers."

Candidates highlighted the potential bias associated with availability to practice. "Honestly, there are so many posts like yours on HN, it's a surprise companies don't change

this ridiculous algorithms thing," questions $HN_{18944244}$ and continues, "However, algorithms bias towards younger people, recently out of college, math hobbyists and people with a lot of free time." $HN_{18944519}$ adds, "They used to interview using the kind of brainteasers found in books like the ones Mensa used to make. The algorithms approach, I suspect, is just a CS proxy for a test just like their old approach was. It would also filter for youth, which they semi-openly advertise as well (see chess literature on brain age for what I mean). Conformance too (due to the prep time)."

Finally, $HN_{11251399}$ warns about the dangers of hiring developers based on ability to practice: "Even once you get past the outright bozos, there are quite a few programmers who can program quick one-off things, but have no sense of design or maintainability. They can deliver functionality, but deliver in a way that piles on technical debt and damages the long term health of the codebase. I think the traditional technical interview format ironically encourages this sort of behavior, by encouraging applicants to focus on narrowly solving the problem at hand, as quickly as possible, both in terms of machine time and programmer time, even if that means the code is an unmaintainable mess in the long run."

F. Missing out

"I used to work at Google. I saw a lot of good candidates get rejected. I myself was rejected multiple times before I got an offer. I was talking to my manager who was on the Hiring Committee about this dilemma, and at the end of the day the fact is that good companies don't give a shit about their false negative rate—only their net positives. By having an efficient technical interview process, yes, you let good candidates go. Just as you do by only having certain target universities or requiring certain experience. But they don't give a fuck. They get 1000 applications a day. Hundreds of internal referrals," complains $HN_{6252419}$.

Developers frequently shared stories where candidates would have normally been filtered out if not for considerable interventions. $HN_{12860682}$ shares their experience: " I have a buddy who I have dragged along with me (many times staking my reputation on his abilities), to every engagement I go on and this guy could not pass a *how to use Microsoft Word interview*. He has Asperger's and locks up and fails miserably in the interviewing process, but the honest reality is, he is 10 times the developer I am, the guy sees patterns instantly and has a knack for code organization. He can master a new technology in a week and is hands down the best developer I have ever met. That being said, over the years watching him has lead me to the conclusion that [technical interviewers] only see the world through their limited experiences. It should be classified as a form of confirmation bias."

Developers also wondered how current practices might be filtering out candidates with more diverse backgrounds and skills: "I cannot help but think that these big tech companies (FAANG, et. al) are missing out on diversifying and increasing their engineering expertise by passing over developers like you. I often think what Google/Facebook would be like if they

hired in some experienced engineers that may not be able to whiteboard a BFS tree or can tell you Djikstra's algorithm, but have proven business track records of getting projects done, on budget, and on time. Real, pragmatic, get-it-done types of engineers. That's not to say whiteboard expert engineers can't also be this way—it's just that whiteboard interviews don't hire for this in particular—technical expertise comes first" ($HN_{18943168}$). $HN_{18944016}$ shares never being able to find gender parity in interviews: "Also... I mentioned that working on teams with other women was important to me... but every technical onsite I've had has been given by a man. They've pitched teams led by women, and my HR/recruiting contacts have been nearly all women. But for the interview itself? All men."

Conventional wisdom, as reflected in "Cracking the Code Interview," [26] has claimed that a false positive (bad hire) is much more expensive than a false negative (missing a good hire). $HN_{18945043}$ adds, "The top companies with these LeetCode tests probably don't care that good people are being rejected or [that candidates are] avoiding them because of the amount of preparation required. Middle sized companies and startups doing LeetCode tests are missing good people and probably can't afford the same number of false negatives as someone like Google with an endless supply of candidates." Several developers, including $HN_{18944154}$, offered counter-arguments to conventional wisdom: "You think people that pass technical interviews can't be false positives? I think they weed out a few, but completely ignore practical development skills, work ethic, soft skills, design and architecture skills, etc. Of course maybe this explains why most of the big tech companies have seemed pretty stagnant for the last decade, largely failing with products and decisions that have poor execution and market fit outside of the products that made them big in the first place."

IV. LIMITATIONS

The nature of small stories analysis, and our application of it to Hacker News, introduces several limitations.

Representativeness. Our study into technical interviews were conducted through an analysis of a single source of practitioner experiences, Hacker News. There are some substantial biases in terms of demographics.[7] Specifically, in a survey conducted in 2011 with 4643 respondents, 89% self-reported being under the age of 40, with 43% of respondents being between the ages of 26-30. In a similar gender survey conducted in 2009 with 1487 respondents, 95% reported as male. We did not find any demographic information on race.

The implication of this demographic is that it may not accurately reflect the concerns of a more diverse population, particularly with respect to underrepresented minorities. Nevertheless, our findings in some ways reflects a lower bound on the concerns of developers. If non-marginalized groups have substantial concerns about technical interviews, then it is very

likely that the impact of technical interviews on marginalized groups is even more severe.

Groupthink in online communities. Another effect from online communities may arise as a result of the moderation and points system used within Hacker News to rank and display comments, in which individuals in the community internalize their true opinions and instead converge to a form of *groupthink*. Fearing reprisal from other members of the community, individuals may be compelled to only share experiences that they believe would be positively scored by their peers [25].

It is also possible that other communities may have different perspectives than Hacker News. Consequently, the set of identified concerns may not be complete.

Qualitative interpretations. Finally, we acknowledge that qualitative research, however rigorously conducted, involves not only the qualitative data under investigation but also a level of subjectivity and interpretation on the part of the researcher as they frame and synthesize the results of their inquiry [10], [27]. In particular, though many posters express concerns about technical interviews, posters whose thoughts are better articulated tend to be given greater representation in the results. Thus, we emphasize that our own findings should be examined as only one of many possible presentations with respect to technical interviews.

Additional studies are needed to mitigate these limitations, such as interviews, surveys, and other instruments to triangulate our concerns [12], [17], [39]. Our findings can be used as a starting point for conducting such studies.

V. RELATED WORK

Despite their importance, technical interviews are understudied in the scientific literature. Ford and colleagues [14] conducted a study from the perspective of hiring managers and University students participating in mock technical interviews. The study identified a mismatch of candidates expectations between what interviewers assess and what they actually look for in a candidate—specifically, through implicit norms in how interviewers expected candidates to explain their solutions, such as using "concrete examples" and "asking relevant questions." In contrast, our study in this paper focuses on professional developers, rather than students.

A slightly distant study by Ford and colleagues [15] identified barriers for female participants in Stack Overflow, an online programming community. Although not conducted in a technical interview setting, several of the identifies barriers resonate with those we identified in our study, in particular, impersonal interactions such as fear of negative feedback, discomfort from lack of diversity in the interviewers, an imposter syndrome of feeling that they didn't have the necessary expertise or qualifications, and time constraints that prevented investment in the site beyond their work day. A survey study with both male and female developers confirmed these barriers as being common across genders.

Using head-mounted eye trackers, Behroozi and colleagues [7] conducted a preliminary study of the public whiteboard interview setting and found that this setting pressures

---

[7]https://news.ycombinator.com/item?id=4397332

candidates into keeping shorter attention lengths and experiencing higher levels of cognitive load compared to solving the same problems privately on paper. The paper concludes that "programming is a cognitively intensive task that defies expectations of constant feedback that todays interview processes follow." Zhou and colleagues [45] investigated both technical and social competencies through GitHub and Stack Overflow data dumps. They found that collaboration competency skills are strongly associated with enhanced coding abilities as well as the quality of code.

Wyrich and colleagues [43] conducted an exploratory qualitative study with 32 software engineering students and found that coding challenge solvers also have better exam grades and more programming experience. Moreover, conscientious as well as sad software engineers performed worse.

Our study complements this prior work by offering qualitative context that explains technical interview performance.

Examining the grey literature of software engineering—that is, non-published, nor peer-reviewed sources of practitioners—provides some additional, though contradictory insights. Lerner [16] conducted a study of over a thousand interviews using the interviewing.io platform, where developers can practice technical interviewing anonymously. Their significant finding is that performance from technical interview to interview is arbitrary, and that interview performance is volatile—only 20% of the interviewees are consistent in their performance, and the rest are all over the place in terms of their interview evaluation. In contrast, a study conducted at Google by Shaper [34] investigated a subset interview data over five years to determine the value of an interviewers feedback, and found that the four interviews were enough to predict whether someone should be hired at Google with 86% confidence. Regardless, our study finds that developers perceive these interview practices to be arbitrary.

A study by Minor [28], conducted across eleven firms in various industries, reported the desire to minimize false positives from "toxic hires"—hires who may steal, commit fraud, bully other workers, or engage in sexual harassment. They found that "toxic workers are actually much more productive than the average worker, which can perhaps explain why they tend to stick around an organization longer than they should." Interestingly, the study found that although greater confidence predicts increased productivity, greater confidence also predicts greater likelihood of becoming toxic. As found in our study, this is an important aspect of hiring that is elided when focusing primarily on the analytical ability of the candidate.

VI. DISCUSSION

In this section, we mitigate some of the concerns related to technical interviews from software developers through a set of inclusive interview guidelines.

Guideline I—Use rudimentary questions for screening. When applying for software development positions, candidates may have limited notice before having to participate in a phone screen. Consequently, they may not have the time available to adequately prepare for the interview (Section III-E). The goal of the phone screen shouldn't be to fully assess the candidates' capabilities. Instead, the interview should be a rudimentary filter to assess whether the candidate can program at all, through what Atwood [2] describes as "blindingly, intentionally easy" questions. A second goal of the phone screen is informational: to share what the team does and to identify whether the candidates' advertised skill set fits these needs.

Our suggestion is that algorithms at this stage of the interview be no more complicated than the Rainfall Problem—a programming task that has been used in a number of studies of programming ability [33], [36].[8]

When complex algorithms are requested in interviews, hiring managers may be unnecessarily excluding candidates simply because of their time commitments, and not because of their lack of technical qualifications (Section III-F). If hiring managers need further evaluation, they can propose a small take-home project, where the candidate has more flexibility and autonomy in how they conduct this work.

Guideline II—Share the interview description in advance. To make technical interviews more equitable for all candidates, the hiring manager should share the details of the interview procedure with them. This includes not only the length of the interview, but also the types of questions that they will be asked. If certain resources are useful for being successful in the interview, these should be recommended to the candidate. Importantly, the scoring rubric for the technical interview should not be opaque to the candidate. Opaque hiring criteria gives an unfair advantage to those with prior interviewing experience, and can be frustrating to candidates who are unaware of the otherwise unwritten rules particular to the organization (Section III-C) [14].

Guideline III—Offer alternative interview formats. Allow candidates to opt-out of certain interview formats or make minor adjustments to the format, without reducing the ability to assess problem-solving skills. Simple changes to existing interview formats could reduce the anxiety associated with public performance in front of an interviewer, for example, by offering the candidate the opportunity to initially think about the problem in private (Section III-B) [7]. Moreover, having to think-aloud while performing a cognitively demanding task has been shown to inhibit task performance [20].

Similarly, while some candidates may prefer conducting the technical interview on a whiteboard, others may feel more comfortable working within an integrated development environment on the computer, or find it more natural to explain and sketch a problem on pencil-and-paper (Section III-D) [40].

---

[8]The original wording of this problem is simple enough, though variations exist: "Write a program that will read in integers and output their average. Stop reading when the value 99999 is input" [36].

Guideline IV—Use a real problem. Several comments on Hacker News were critical of the use of artificial, puzzle-like problems that did not reflect the types of engineering tasks that candidates would do in their day-to-day software development activities (Section III-A). Such problems were also perceived as biased towards junior candidates just out of school, as these junior candidates were more likely to have recently solved these kinds of homework-style problems. For senior developers, our suggestion is to use technical interview problems that have real-world scenarios, and resemble programs that the candidate would actually write as a software developer within the team. For example, consider the purely academic problem of performing a depth-first traversal of a graph structure versus framing this problem as determining the order in which to install packages from a package management system when provided with an input of a hierarchy of dependencies. In the latter scenario, it may even be sufficient that the candidate can simply identify that this problem is an instance of depth-first search, without needing them to actually write code.

Guideline V—Solve problems as colleagues, not as examiners. Not all candidates can effectively solve technical algorithms in a fast-paced, high-pressure whiteboard setting—nor do these candidates often work in such stressful situations in their day-to-day software development jobs (Section III-B). Instead, the conversation between the interviewer and candidate should be less of an antagonistic interrogation, and more of a conversation in which both the candidate and the interviewer work together to solve the problem. Preferably, the candidate should be interviewed by the team that they intend to join, so that they can experience first-hand how they would work with one or more of their potential teammates.

VII. Conclusion

The technical interview has become commonplace within the software development industry as a means to assess candidates. However, despite its importance for hiring qualified candidates, the effectiveness and perceptions of the technical interview are understudied within the software engineering research community.

To understand how software developers perceive technical interviews, we conducted a qualitative study using the online social news website, Hacker News. By framing comments as small stories, we identified critical concerns from candidates regarding these interviews. These concerns include the relevance of these interviews as well as their impact on the candidates' anxiety, affect, and time commitments. We find that candidates who use technical interviews as a primary assessment instrument may unfairly filter out otherwise qualified candidates. We propose inclusive interview guidelines towards improving the technical interview process. The findings from this study underscore the need for additional research in this area, especially towards understanding how technical interviews impact underrepresented minorities within the software development community.

VIII. Acknowledgement

References

[1] C. Alvino, "Technical Interviews an Instrument of Exclusion and Discrimination," Jun 2014. [Online]. Available: https://careerconservatory.com/technical-interviews-an-instrument-of-exclusion-and-discrimination/

[2] J. Atwood, "Why Can't Programmers Program?" Feb. 2007. [Online]. Available: https://blog.codinghorror.com/why-cant-programmers-program/

[3] A. Aziz, T.-H. Lee, and A. Prakash, *Elements of Programming Interviews in Java: The Insiders' Guide*, 2015.

[4] M. Bamberg and A. Georgakopoulou, "Small stories as a new perspective in narrative and identity analysis," *Text & Talk*, vol. 28, no. 3, pp. 377–396, 2008.

[5] T. Barik, "Expressions on the nature and significance of programming and play," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2017, pp. 145–153.

[6] T. Barik, B. Johnson, and E. Murphy-Hill, "I heart Hacker News: expanding qualitative research findings by analyzing social news websites," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 882–885.

[7] M. Behroozi, A. Lui, I. Moore, D. Ford, and C. Parnin, "Dazed: measuring the cognitive load of solving technical interview problems at the whiteboard," in *2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*. IEEE, 2018, pp. 93–96.

[8] J. Bentley, "Programming Pearls: Algorithm Design Techniques," *Commun. ACM*, vol. 27, no. 9, pp. 865–873, Sep. 1984.

[9] M. Birks, Y. Chapman, and K. Francis, "Memoing in qualitative research: Probing data and processes," *Journal of Research in Nursing*, vol. 13, no. 1, pp. 68–75, Jan. 2008.

[10] E. Bott, "Favourites and others: reflexivity and the shaping of subjectivities and data in qualitative research," *Qualitative Research*, vol. 10, no. 2, pp. 159–173, 2010.

[11] V. Braun, V. Clarke, N. Hayfield, and G. Terry, "Thematic Analysis," in *Handbook of Research Methods in Health Social Sciences*, P. Liamputtong, Ed. Singapore: Springer, 2019, pp. 843–860.

[12] N. Carter, D. Bryant-Lukosius, A. DiCenso, J. Blythe, and A. J. Neville, "The use of triangulation in qualitative research." in *Oncology Nursing Forum*, vol. 41, no. 5, 2014.

[13] D. M. Diamond, C. R. Park, K. L. Heman, and G. M. Rose, "Exposing rats to a predator impairs spatial working memory in the radial arm water maze," *Hippocampus*, vol. 9, no. 5, pp. 542–552, 1999.

[14] D. Ford, T. Barik, L. Rand-Pickett, and C. Parnin, "The tech-talk balance: what technical interviewers expect from technical candidates," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2017, pp. 43–48.

[15] D. Ford, J. Smith, P. J. Guo, and C. Parnin, "Paradise unplugged: Identifying barriers for female participation on stack overflow," in *Proceedings of the 24th International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 846–857.

[16] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2016, p. 26.

[17] N. Golafshani, "Understanding reliability and validity in qualitative research," *The Qualitative Report*, vol. 8, no. 4, pp. 597–606, 2003.

[18] D. H. Hansson, "Horses for courses," Feb. 2017. [Online]. Available: https://m.signalvnoise.com/horses-for-courses/

[19] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[20] R. Jääskeläinen, "Think-aloud protocol," *Handbook of Translation Studies*, vol. 1, pp. 371–374, 2010.

[21] C. Kirschbaum, O. T. Wolf, M. May, W. Wippich, and D. H. Hellhammer, "Stress-and treatment-induced elevations of cortisol levels associated with impaired declarative memory in healthy adults," *Life Sciences*, vol. 58, no. 17, pp. 1475–1483, 1996.

[22] Q. Larson, "Why is hiring broken? It starts at the whiteboard," Apr. 2016. [Online]. Available: https://medium.freecodecamp.org/why-is-hiring-broken-it-starts-at-the-whiteboard-34b088e5a5db

[23] A. Lerner, "You can't fix diversity in tech without fixing the technical interview," Nov. 2016. [Online]. Available: https://medium.freecodecamp.org/you-cant-fix-diversity-in-tech-without-fixing-the-technical-interview-here-s-the-data-93130f977da2

[24] S. Lupien, S. Gaudreau, B. Tchiteya, F. Maheu, S. Sharma, N. Nair, R. Hauger, B. McEwen, and M. Meaney, "Stress-induced declarative memory impairment in healthy elderly subjects: relationship to cortisol reactivity," *The Journal of Clinical Endocrinology & Metabolism*, vol. 82, no. 7, pp. 2070–2075, 1997.

[25] C. McCauley, "The nature of social influence in groupthink: Compliance and internalization." *Journal of Personality and Social Psychology*, vol. 57, no. 2, p. 250, 1989.

[26] G. McDowell, *Cracking the Coding Interview*, 2015.

[27] B. Mehra, "Bias in qualitative research: Voices from an online classroom," *The Qualitative Report*, vol. 7, no. 1, pp. 1–19, 2002.

[28] D. Minor, "Just how toxic are toxic employees?" Jan. 2016. [Online]. Available: https://rework.withgoogle.com/blog/how-toxic-are-toxic-employees/

[29] K. Monterroso, "Real talk: The Technical Interview is Broken," Jun. 2016. [Online]. Available: https://medium.com/racial-equity-in-tech/real-talk-the-technical-interview-is-broken-b84b8375dccb

[30] J. W. Newcomer, S. Craft, T. e. Hershey, K. Askins, and M. Bardgett, "Glucocorticoid-induced impairment in declarative memory performance in adult humans," *Journal of Neuroscience*, vol. 14, no. 4, pp. 2047–2053, 1994.

[31] J. Ponterotto, "Brief note on the origins, evolution, and meaning of the qualitative research concept thick description," *The Qualitative Report*, vol. 11, no. 3, pp. 538–549, 2006.

[32] J. Saldaña, *The Coding Manual for Qualitative Researchers*. SAGE Publications, 2009.

[33] O. Seppälä, P. Ihantola, E. Isohanni, J. Sorva, and A. Vihavainen, "Do we know how difficult the rainfall problem is?" in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. ACM, 2015, pp. 87–96.

[34] S. Shaper, "How many interviews does it take to hire a Googler?" Apr. 2017. [Online]. Available: https://rework.withgoogle.com/blog/google-rule-of-four/

[35] S. Shogren, "Interview Humiliation," Oct. 2015. [Online]. Available: http://deliberate-software.com/on-defeat/

[36] E. Soloway, "Learning to program= learning to construct mechanisms and explanations," *Communications of the ACM*, vol. 29, no. 9, pp. 850–858, 1986.

[37] F. Thawar, "Technical interviews are garbage," Oct. 2017. [Online]. Available: https://be.helpful.com/https-medium-com-fnthawar-helpful-technical-interviews-are-garbage-dc5d9aee5acd

[38] R. Thomas, "How to Make Tech Interviews a Little Less Awful," Mar 2017. [Online]. Available: https://medium.com/@racheltho/how-to-make-tech-interviews-a-little-less-awful-c29f35431987

[39] S. J. Tracy, "Qualitative quality: Eight big-tent criteria for excellent qualitative research," *Qualitative Inquiry*, vol. 16, no. 10, pp. 837–851, 2010.

[40] B. Tversky, "What do sketches say about thinking," in *2002 AAAI Spring Symposium, Sketch Understanding Workshop, Stanford University, AAAI Technical Report SS-02-08*, 2002, pp. 148–151.

[41] P. Veluvolu, "Technical interviews are broken," Nov. 2017. [Online]. Available: https://blog.midweststartups.com/technical-interviews-are-broken-636a39c65209

[42] Y. Wu, J. Kropczynski, P. C. Shih, and J. M. Carroll, "Exploring the ecosystem of software developers on GitHub and other platforms," in *CSCW Companion*, 2014, pp. 265–268.

[43] M. Wyrich, D. Graziotin, and S. Wagner, "A theory on individual characteristics of successful coding challenge solvers," *PeerJ Computer Science*, vol. 5, p. e173, 2019.

[44] S. Yalkabov, "Fuck You, I Quit—Hiring Is Broken," Apr. 2016. [Online]. Available: https://medium.com/@evnowandforever/f-you-i-quit-hiring-is-broken-bb8f3a48d324

[45] C. Zhou, S. K. Kuttal, and I. Ahmed, "What makes a good developer? an empirical study of developers' technical and social competencies," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2018, pp. 319–321.