Asynchronous Technical Interviews: Reducing the Effect of Supervised Think-Aloud on Communication Ability

Mahnaz Behroozi
Mahnaz.Behroozi@ibm.com
IBM, USA

Chris Parnin
cjparnin@ncsu.edu
North Carolina State University, USA

Chris Brown
dcbrown@vt.edu
Virginia Tech, USA

ABSTRACT

Software engineers often face a critical test before landing a job—passing a technical interview. During these sessions, candidates must write code while thinking aloud as they work toward a solution to a problem under the watchful eye of an interviewer. While thinking aloud during technical interviews gives interviewers a picture of candidates' problem-solving ability, surprisingly, these types of interviews often prevent candidates from communicating their thought process effectively. To understand if poor performance related to interviewer presence can be reduced while preserving communication and technical skills, we introduce *asynchronous* technical interviews—where candidates submit recordings of think-aloud and coding. We compare this approach to traditional whiteboard interviews and find that, by eliminating interviewer supervision, asynchronicity significantly improved the clarity of think-aloud via increased informativeness and reduced stress. Moreover, we discovered asynchronous technical interviews preserved, and in some cases even enhanced, technical problem-solving strategies and code quality. This work offers insight into asynchronous technical interviews as a design for supporting communication during interviews, and discusses trade-offs and guidelines for implementing this approach in software engineering hiring practices.

CCS CONCEPTS

• **Software and its engineering**; • **Human-centered computing** → **Empirical studies in HCI**;

KEYWORDS

technical interviews, asynchronous communication, skill evaluation, software engineering

ACM Reference Format:
Mahnaz Behroozi, Chris Parnin, and Chris Brown. 2022. 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 '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3540250.3549168

1 INTRODUCTION

A *technical interview* is a specialized form of job interview [2] where software engineering candidates are tasked with writing code to solve programming problems while simultaneously performing a *think-aloud*, or providing a vocal explanation of their problem-solving approach to potential employers [35]. Think-aloud has a major impact on software engineering hiring decisions. According to tech consultant Scott Hanselman, interviewers often look for "thoughtfulness, analysis, patience, calm, and humility" in think-aloud [25]. In addition, Refael Zikavashvili, co-founder and CEO of Pramp[1]—an online platform for practicing technical interviews—describes the significance of think-aloud stating, "more important than the solution is how [candidates] work at getting the solution." Further, interview guides from companies like Google and Microsoft expect candidates to "explain [their] thought process and decision making throughout the interview" [24] and "be prepared to share...the rationale behind your decisions" [36]. Thus, during technical interviews, candidates are evaluated on their ability to communicate their thought process, sometimes even moreso than their coding skills [13].

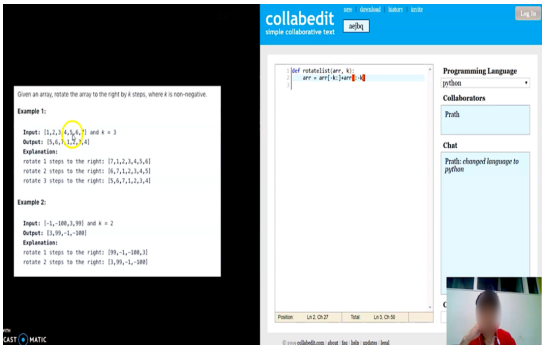
Unfortunately, supervised think-aloud often has adverse and negative effects on problem-solving and communication ability [10, 51]. One interview candidate shared their struggle with think-aloud: "Thinking out loud ends up with me spending brain cycles on reflecting on how what I say must be registering with the interviewer and in addition there's a fear of recognizing that I've gone down the wrong path and starting over...speaking takes more time [and] leads to self-consciousness" [12]. Another shared, "I made a few bugs I would never have made if I were concentrating solely on coding—honestly it's so much harder for me to think aloud." [30]. Further, supervised think-aloud allows for interruptions from interviewers, which can throw off candidates during their problem-solving process. Consider the following example from tech blogger Jesse Squires [43]: "I was familiar with the algorithm and could explain how it works conceptually. After only a few lines, the interviewer called out from behind me while comfortably sitting in his chair to make a joke that 'you must must be writing a lot of Swift' because I had accidentally omitted a few semicolons. That comment threw me off for the rest of our time... the rest of the hour or so in that room did not turn out well." Rather than gaining visibility into candidates' thought processes and problem-solving strategies, supervised think-aloud in technical interviews muddles and disturbs them. Instead of having a conversation with an interviewer—candidates report that it often feels like an interrogation.

Alternative formats for technical interviews exists. For example, Behroozi et al. [10] demonstrated that in *private interviews*—without supervised think-aloud—participants reported feeling at ease, having time to understand the problem and reflect on their solution. As a result, technical problem-solving performance was improved

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

ACM ISBN 978-1-4503-9413-0/22/11.
https://doi.org/10.1145/3540250.3549168

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



(a) Asynchronous Technical Interviews (Screencast)

(b) Public Whiteboard Interview Setting

Figure 1: (a) Solving a problem and recording their screen while thinking aloud. (b) Solving a problem in presence of an experimenter while thinking aloud. (Note: *These images are not taken from our actual experiment.*)

nearly two-fold. While private interviews have considerable advantages for candidates, such as reducing stress and improving technical problem-solving abilities, they lack the ability to foster think-aloud while solving the problem, which eliminates valuable information for interviewers in their evaluation of candidates. Fully private interviews without think-aloud inhibit interviewers from adequately evaluating potential employees, and thus have limited value in the hiring processes.

In this paper, we evaluate *asynchronous technical interviews*, where candidates submit audio and screencast recordings of themselves solving a technical interview coding problem and explaining their approach throughout the process without the interviewer(s) present (see Figure 1-a). This approach allows candidates to freely decide when and how to communicate about their thought process, while eliminating negative effects of supervision as achieved with private interviews. To this end, we conducted a study where we compare asynchronous technical interviews to synchronous whiteboard interviews, or a traditional technical interview setting with supervised think-aloud. The goal of our study is to understand whether asynchronicity can enhance think-aloud quality, while preserving technical performance.

To analyze asynchronous technical interviews, we evaluated candidates' think-aloud and technical abilities through a retrospective evaluation of recorded screencasts without live observation. We found that communication ability and coding performance are both improved by eliminating interviewers' direct supervision, and the clarity of think-aloud was significantly higher in asynchronously recorded session compared to traditional whiteboard interviews, while preserving technical solving performance. The contribution of this work is to elucidate the effects of interviewers' direct supervision on candidates' thought process communication ability. The implications of this work include several trade-offs to implementing asynchronous technical interviews and guidelines for more effective administrations of coding interviews.

2 METHODOLOGY

A technical interview is a hiring assessment that involves job-seeking candidates completing *think-alouds* and expressing *technical skills* in front of potential employers. To understand the impact of supervised think-aloud on technical interview performance, we compare traditional technical interview environments (our control) with asynchronous technical interviews (our treatment). In this section, we describe our research questions, experimental settings, and the data collected and analyzed to measure the differences of participants' communication and coding abilities in each setting.

2.1 Research Questions and Hypotheses

To understand the impact of asynchronicity in technical interviews, we investigated the following research questions:

RQ1: What are the effects of asynchronous technical interviews on think-aloud performance?

To observe candidates' thought process during hiring evaluations, interviewers ask them to think aloud while solving programming problems. Think-aloud, as a communication skill, is essential for evaluating candidates in technical interviews [13]. To investigate whether asynchronicity impacts the quality of candidates' communication ability, we compared think-aloud informativeness and stress indicators in asynchronous technical interviews against traditional whiteboard interview settings with synchronous interviewer supervision. We speculate that asynchronous technical interviews are less stressful due to their ability to increase privacy for candidates and eliminate direct supervision [10]. Thus, we hypothesize that our approach will improve candidates' think-aloud quality and reduce stress in vocalizations.

RQ2: What are the effects of asynchronous technical interviews on technical ability?

Technical interviews also incorporate coding tasks to evaluate the programming skills and technical expertise of potential employees. We aim to discover if asynchronous technical interviews can improve think-aloud ability without negatively impacting the technical performance of candidates solving coding challenges. To measure technical ability, we observed problem-solving strategies identified as best practices for technical interviews and analyzed code quality using standard industry hiring evaluation metrics, namely correctness and optimality. We hypothesize that candidates

Table 1: Description of the Coding Problems

Question	Description
Q1 (Valid Parentheses)	Given a string *s* containing just the characters '(', ')' , '{' , '}', '[' and ']', determine if the input string is valid. An input string is valid if: 1- Open brackets must be closed by the same type of brackets. 2- Open brackets must be closed in the correct order.
Q2 (Jump)	Given an array of non-negative integers *nums*, you are initially positioned at the first index of the array. Each element in the array represents your maximum jump length at that position. Determine if you are able to reach the last index.
Q3 (Increasing Subsequece)	Given an integer array *nums*, return the length of the longest strictly increasing subsequence. A subsequence is a sequence that can be derived from an array by deleting some or no elements without changing the order of the remaining elements. For example, [3,6,2,7] is a subsequence of the array [0,3,1,6,2,2,7].
Q4 (Rotate Array)	Given an array, rotate the array to the right by *k* steps, where *k* is non-negative.
Q5 (Longest Substring)	Given a string *s*, find the length of the longest substring without repeating characters.

will be able to maintain similar problem-solving strategies and produce comparable coding solutions to programming challenges in asynchronous and synchronous technical interview settings.

2.2 Participants

We retrieved data for our control group from the public whiteboard setting from Behroozi et al.'s study [10] analyzing public and private technical interview environments. From their study, we considered the audio and video recordings of 24 student participants who completed the technical interview task in the public setting. This sample includes 19 male and 5 female subjects. To obtain participants for our treatment group using asynchronous technical interviews, we randomly sampled 24 participants enrolled in a graduate-level Software Engineering course. Our treatment group sample consists of submissions from 9 female and 15 male students.

All study participants in both experimental settings had knowledge of Java and/or at least one other high-level programming language. Prerequisite classes for the graduate Software Engineering course at the authors' institution gave students the typical knowledge required to complete technical interviews, including constructing data structures, implementing search and sort algorithms, and characterizing run time and space complexities. Furthermore, graduate students represent potential technical interview candidates, as many were actively engaged in software engineering internship and full-time job searches or had prior professional experience. This study was approved by the local ethics board.

2.3 Tasks

In work by Behroozi et al. [10], tasks are selected in such a way that it satisfies three main criteria: to be solvable within the time limit of the experiment; to demonstrate sufficient difficulty such that cognitive load can be induced while not being too trivial to solve; and to have ecological validity—the task should reflect actual technical interviews. In our evaluation, the study tasks selected maintain the same criteria. Table 1 shows the coding questions used for our interview tasks with descriptions of each problem. All of the tasks utilized in our study can be found in "Elements of Programming Interviews in Java" [2], a resource to help candidates prepare for technical interview with practice programming challenges. These questions are of a similar level of difficulty and represent a class of problems related to string manipulation and usage of arrays, stacks, or hash maps, which are fundamental Computer Science concepts commonly tested during technical interviews [2]. Furthermore, the problems contain a large solution space with multiple approaches, including brute force and more sophisticated approaches with different complexities, as shown in Table 2.

2.4 Procedure

2.4.1 Asynchronous Technical Interviews. For our treatment group, students were randomly assigned one of four coding problems in Table 1 (Q1-Q4) via email. Of the 24 randomly sampled submissions, nine subjects completed Q1, five completed Q2, six completed Q3, and four completed Q4. Each problem statement included example test cases, provided in Table 2, which indicated the expected program output given certain input values. Students were asked to provide a reasonable solution to their assigned task using pseudocode or a programming language of their choice on Collabedit,[2] an online code editor with syntax highlighting but without a compiler to run the code. The assignment also stated that thought process communication and correctness of the solution were the most important criteria, while efficiency and syntax were secondary.

Students were required to record their screen and voice while solving the problem (see Figure 1-a). However, participants were not allowed to consult other resources to complete their task, including but not limited to online resources, books, notes, and other people. To limit preparation for the coding problem beforehand, students signed up for a one-hour slot to perform their task, but they did not know their coding problem ahead of time. This information was emailed to them at the start of the time slot, and then students had

[2] http://collabedit.com/

Table 2: Sample Solution Approaches, Time Complexities, and Test Cases for Coding Problem Tasks

Question	Approaches' Time Complexity	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
Q1	Stack: $O(n)$	input: " " Output: False	input: [] Output: True	input: }} Output: False	input: }{ Output: False	input: (()) Output: True	input: ()([]) Output: True
Q2	Greedy: $O(n)$ Dynamic Programming: $O(n^2)$	input : [2,3,1,1,4] Output: true	Input: [3,2,1,0,4] Output: false	Input: [2] Output: true	Input: [0, 4, 8, 1, 17, 3] Output: False	Input: [1, 1, 1, 1, 0, 1] Output: False	Input: [5, 0, 0] Output: True
Q3	Dynamic Programming with Binary search: $O(nlogn)$ Dynamic Programming: $O(n^2)$ Recursive with memoization: $O(n^2)$	Input: [10,9,2,5,3,7,101,18] Output: 4	Input: [0,1,0,3,2,3] Output: 4	Input: [7,7,7,7,7,7,7] Output: 1	Input: [4,3,2] Output: 1	Input: [] Output: 0	
Q4	Brute Force: $O(n)$	Input: [1,2,3,4,5,6,7], 3 Output: [5,6,7,1,2,3,4]	Input: [1,2,3,4,5,6,7], 6 Output: [2,3,4,5,6,7,1]	Input: [-1,-100,3,99], 2 Output: [3,99, -1, -100]	input: [28], 3 Output: [28]	Input:[50,51,52,53], 121 Output: [53, 50, 51, 52]	
Q5	Sliding Window: $O(n)$ Sliding Window Optimized: $O(n)$ Brute Force: $O(n^3)$	Input: s = "abcabcbb" Output: 3	Input: s = "bbbbb" Output: 1	Input: s = "pwwkew" Output: 3			

to complete their coding assignment within one hour and upload their recorded screencast; otherwise, the submission would not be considered valid. All participants were able to complete their tasks and upload their submissions within the allotted time slot.

2.4.2 Traditional Whiteboard Interviews. To investigate how supervised think-aloud impacts communication ability, we compared students' submitted screencast recordings to data collected from Behroozi et al.'s [10] *public whiteboard setting*. The whiteboard setting resembled a traditional technical interview, where student participants stood at a whiteboard to complete a programming challenge (Q5) without knowledge of the problem beforehand, and performed a supervised think-aloud in front of an observing researcher (see Figure 1-b). Participants were similarly given sample test cases, prohibited from using external resources, and permitted to use any coding language. They had to complete the task within a 30 minute time limit, and were encouraged to verbalize their thought processes as if participating in a real job interview.

All participants who completed the technical interview task in the public whiteboard setting within the given time limit were included in this research experiment ($n = 24$). In each session, participants' voice and work on the whiteboard were captured using head-mounted mobile eye-tracking devices and made available to the research team. In this study, we analyzed participant think-aloud and technical abilities in recordings and compared them against the data collected from our asynchronous technical interviews setting.

3 ANALYSIS

To test our hypotheses on the effects of asynchronicity on technical interview proficiency, we collected data observing think-aloud and technical performance of participants completing interview-related tasks in asynchronous and synchronous interview settings.

3.1 Measuring Think-Aloud

To answer RQ1, we analyzed think-aloud by examining communication quality while also observing speech patterns to identify possible sources of stress or cognitive load.

3.1.1 Quality. To evaluate think-aloud quality, we analyzed speech pattern informativeness and evaluated thoughts vocalized in the asynchronous and synchronous technical interview settings.

Informativeness. Think-aloud provides insight into candidates' thought processes through communication between interviewees and interviewers. Yet, not all vocalized thoughts associated with problem-solving are worthy of communication, and verbalizing all thoughts is not the same as communicating the thought process. Furthermore, complex thought processes cause individuals to use more filler words, or non-informative vocalizations, to seem less stressed during uncertain pausing moments [1, 32].

To measure the informative parts of speech vocalized by participants in each setting, we extracted non-filler words using the Voice Activity Detector (VAD) module from Google's open-source WebRTC[3] code to convert speech to text. VAD utilizes a Gaussian Mixture Model (GMM) to detect human voice in the presence of background noise–such as keyboard typing or marker noise while writing on a whiteboard. As the recordings contained relatively long periods of silence, we also removed silent moments by converting audio recordings in both settings into a 16 bit Pulse-code modulation (PCM). We then sent 30-millisecond long frames using a sliding window technique to VAD to filter out the frames that did not contain human voice. Finally, we used the Google Web Speech API[4] to convert speech to text on each chunk in recordings.[5]

The Google Web Speech API considers vocalizations such as 'um' and 'uh' as silence. We considered the following list as additional common non-informative filler words: 'so', 'huh', 'oh', 'okay', 'then', 'now', 'oops', 'yeah', 'yes', 'alright', 'sorry', and 'cool'. We filtered filler words out of candidates' thought process text to evaluate their think-aloud. From the text generated for each participant, we calculated participants' non-filler words rate (NF_{Rate}) as:

$$NF_{Rate} = \frac{NF_t}{d}, \quad (1)$$

[3] https://webrtc.org/
[4] https://wicg.github.io/speech-api/
[5] The Python script and the generated texts from audio recordings can be found here: https://rb.gy/iermrc

where, NF_t is the total number of non-filler words, and d is the duration of the session in minutes.

Thought Vocalization. The ability for interviewees to clearly vocalize their thought process and problem-solving approach plays a major role in their technical interview performance, as companies evaluate prospective employees on the technical competency of coding solutions as well as their ability to communicate about it [20]. Indeed [6]—an employment company—emphasizes the importance of communication during technical interviews and note think-alouds strongly factor into candidates getting a job offer [28].

To rate the thought vocalization of participants, two authors acted as evaluators to retroactively analyze audio recordings from both settings. A 5-point Likert scale ranking was implemented to assess the *effectiveness* of participants' think-aloud during their interview—in the analysis, ratings of 4 or higher were considered effective, 3 or lower were considered ineffective. This scoring reflects the typical binary feedback candidates often get while completing technical interviews—hired or not hired. To evaluate participants' think-aloud, we used criteria based on communication expectations during technical interviews from industry employers [21], including *engagement*, or the confidence and articulateness in which participants express their technical knowledge (i.e. how easy it was to follow), and *smoothness*, or the clarity and consistency of their think-aloud while solving the problem (i.e., avoiding long periods of silence). Then, the two coders came together to discuss their individual rankings and come to an agreement on the quality of participants' think-aloud.

3.1.2 Stress. Speech can be affected by *stress*, a psychological state in response to task demand or fear [26]. To understand the impact of the technical interview settings on speech, we analyzed speech patterns, such as speech rate and the presence of stress cues.

Speech Rate. According to the National Center of Voice and Speech (NCVS),[7] the average conversation rate for English speakers in the United States is approximately 150 words per minute. However, stressful conditions have been shown to impact human speech rate. For instance, stress can increase speech rate and word productivity in humans [14]. On the other hand, humans may slow their talking speed by adding pauses [23] or increasing usage of filler words [32] during stress-inducing activities, such as public speaking [53]. Research suggests secondary tasks, such as a technical interview problem-solving tasks, require attentional resources that impact the spatiotemporality of vocal systems and can reduce speech rate and fluency [3, 14, 17–19, 49].

To evaluate speech rate, we used VAD to convert the think-aloud verbalized by participants into text. We used these results to analyze audio recordings of interviews and measure the overall speech rate of participants in each setting, compiling the total number of words (non-filler and filler) uttered during think-aloud divided by the total duration of a particular session to complete the technical interview task. We anticipate participants in the traditional whiteboard setting will have discrepancies in rate of speech compared to those completing asynchronous technical interviews.

Stress Cues. Research shows activities such as public speaking induce anxiety in humans and impacts their speech [53]. The public nature of supervised think-aloud has also been shown as a source of stress, negatively impacting candidates' communication ability and technical performances [10]. Further, studies suggest stress cues and speech can be perceived by listeners [22], which may impact the reaction and synthesis of information acquired by the audience.

To measure stress cues in speech, we collected one-minute samples from the beginning, middle, and the end of each voice recording. To avoid biased evaluations, the raters were blinded from the interview condition and evaluated 144 generated one-minute voice samples that were randomized to prevent recognition of participants and the setting. Two authors of this paper independently evaluated the extent of stress in participants' voice using a 5-point Likert scale (1 being Not at All Stressed and 5 being Extremely Stressed). Then, they reported voice features that influenced their perception of participants' stress during think-aloud. The common features taken into consideration were: shortness of breath, mouth dryness, shaky voice, faster or slower than normal rate of talking, higher or lower than normal voice pitch, and usage of non-informative filler words. We categorized participants with Likert rankings of three or less "*not stressed*" and rankings of four or five as "*stressed*".

3.2 Measuring Technical Performance

To answer RQ2, we observed technical aspects of interviews including problem-solving strategies and solutions' code quality.

3.2.1 Problem-Solving Strategies. To study problem-solving strategies, we analyzed participants' adoption of beneficial technical interview activities before and after programming. These best practices were derived from hiring teams, programming blogs, and resources to help users prepare for interviews.

Defining Approach before Coding. This strategy refers to technical interview candidates outlining their approach to solving a programming challenge question before starting to write code. During technical interviews, planning and explaining the approach to solve a coding problem before starting to code provides benefits for participants. For example, freeCodeCamp recommends interviewees should "communicate your approach to the interviewer even before you start to code" [45] to validate the approach with hiring managers, FullStack Academy notes describing approaches before coding provides insight into your thought process and increases understanding for interviewers [52], and Scaler suggests large tech companies like Google, Amazon, and Facebook are looking for candidates with "the patience and intellect to analyze the problem at hand thoroughly before jumping into providing a solution" [42].

To analyze this in our study, we observed recordings to determine whether participants clearly explained their approach before starting the problem-solving process with code or pseudocode in each setting. We investigated participant recordings to analyze their think-aloud and determine when they began to write code for their assigned problem. Participants who provided a preliminary algorithm to approach the problem before starting to write the program, even if their algorithm changed during the actual coding, were indicated to have adopted this problem-solving strategy.

Checking Code against Test Cases. After completing the code for technical interview programming questions, resources indicate interviewees should test their solution to validate their approach. In general, testing is vital for ensuring software works

[6] https://www.indeed.com/
[7] http://www.ncvs.org/ncvs/tutorials/voiceprod/tutorial/quality.html

correctly [50]. Testing is also an important step for validating solutions during in technical interviews. For example, Tech Interview Coach and Google Software Engineer Anthony D. Mays argues testing is something interviewees should "always" and "absolutely need to do" during technical interviews [34]. Similarly, Ibrahim Irfan—a former developer at Google and Facebook and co-founder of the Superpowered app[8]—emphasizes the importance of testing your code during interviews stating, "The biggest piece of advice I can give you here is to test your code, not your algorithm. Go through the exact code you wrote down. This is especially important in whiteboard interviews where you can't run your code" [29].

To explore the impact of interview settings on participants' testing their solution against test cases, we manually analyzed interview recordings to observe whether subjects tested their solution with at least one of the provided test cases after completing an initial solution to the programming question. Those that went through an input and produced an output value to verify their approach after completing a solution were regarded as implementing this technical interview problem-solving strategy, regardless of whether or not their solution was correct.

3.2.2 Code Quality. To evaluate code quality, we used industry standards for evaluating candidates. Specifically, we measured the correctness and complexity of participants' solution to ensure asynchronous technical interviews sustain technical performance.

Correctness. We evaluated participant code solutions in each setting with the specific test cases provided to participants. Table 2 summarizes the coding questions along with the test cases used to evaluate solutions. The screencast participant solutions were evaluated on test cases based on their randomly assigned question from Q1-Q4, while the whiteboard participants from Behroozi et al.'s study were evaluated against Q5 (see Table 1). To analyze program correctness, we manually translated all of the solutions from each setting into program code, filling in incomplete or incorrect syntax when necessary. Then, we executed the programs with the given input values to determine if the output of the code matched the expected output of the test cases provided. We scored participants in each setting based on the percentage of test cases passed out of the total number of test cases to indicate how well completed solutions met the requirements of the original problem.

Optimality. Although we did not require an optimal solution, we did examine the programs submitted by participants in each setting to determine their performance based on runtime complexity, or the amount of computer time necessary to run an algorithm. For example, a program using two nested for loops iterating through each value in an array would have an $O(n^2)$ runtime complexity. The runtime complexities for potential solutions to the programming questions participants completed for the study are available in Table 2. To analyze the complexity of solutions, we converted solutions provided by participants into executable code and analyzed the time complexity of programs by observing the data structures and algorithms utilized. In general, employers expect efficient code with low complexity during technical interviews [35]. Thus, we evaluate optimality as a metric to determine the number of participants who provided the most optimal solution in each setting.

[8] https://superpowered.me/

4 RESULTS

4.1 Impact on Think-Aloud

4.1.1 Communication Quality.

Informativeness. We found participants in the traditional whiteboard interview setting were much less articulate based on higher usage of non-informative words in their vocalizations. Using the informativeness formula in Equation 1, which calculates the rate of non-formative words, we found the average rate of informativeness was 63.73 for whiteboard participants compared to 75.54 for participants in the screencast setting. This indicates that candidates in the asynchronous setting uttered less filler words and their speech contained more substantial statements. Further, a Mann–Whitney U test found the difference between the rate of non-filler words usage per minute to be significant between the two interview settings (U=203.0, p=0.041, d=0.647), as presented in Figure 2. Thus, we conclude the overall informativeness of participants' think-aloud was higher in asynchronous technical interviews than in the traditional whiteboard interview setting.

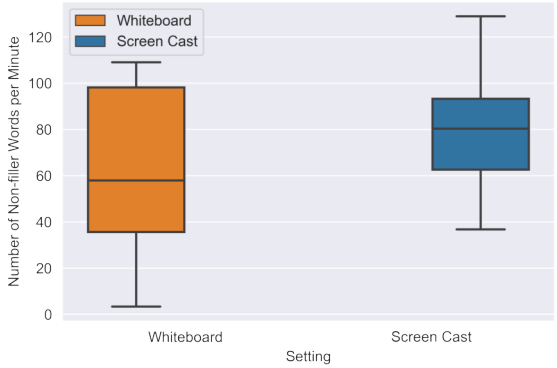


Figure 2: Non-filler words rate in synchronous (whiteboard) and asynchronous (screencast) interview settings

Thought Vocalization. Participants in the asynchronous setting were more often rated as effective, 83% (n = 20) compared with 54% (n = 13). Furthermore, approximately 88% (n = 21) of asynchronous technical interviews think-aloud sessions were categorized as *engaging* and 79% (n = 19) walked through their process *smoothly*. However, of the whiteboard participant thought vocalizations, only 58% (n = 14) were considered engaging and 67% (n = 16) smooth. The differences in effectiveness ratings were significantly different (χ^2 = 4.75, p = 0.0292). Overall, we find that asynchronous interview participants communicated their thoughts significantly more effectively than those in the traditional whiteboard setting.

4.1.2 Stress.

Speech Rate. Generally, participants in the whiteboard setting used fewer words and a slower speech rate to communicate compared to those using screencast recordings. We found the speech rate of asynchronous participants was approximately 81.00 words

per minute compared to 69.40 for whiteboard participants, indicating increased pauses and cognitive load. We also found the talking rate of participants was significantly slower in the whiteboard setting (U=204.0, p=0.043, d=0.615). In whiteboard settings, not only is information being transformed at a slower rate, this slower rate is consistent with higher cognitive load and stress levels.

Stress Cues. We found 86.36% (n = 19) of the 22 one-minute samples labeled as *stressed* were from participants in the whiteboard setting. Further, we observed that 25% (n = 6) of participants in the whiteboard setting were categorized as stressed in their overall think-aloud, while *none* of the screencast participants were considered to exude stress cues in their thought process vocalizations. The difference of perceived stress from voice was statistically significant between the settings ($\chi^2 = 6.857, p = 0.0088$), indicating that supervised think-aloud during technical interviews had a major impact on communication ability and perceivable stress levels.

4.2 Impact on Technical Performance

4.2.1 *Problem-Solving Strategies.*

Defining Approach before Coding. We observed 42% (n = 10) of the screencast participants explained their approach before starting to code, while only 25% (n = 6) did so in the whiteboard setting. This indicates, on average, participants completing the technical interviews with private screencasts were able to plan and describe their approach more effectively. While we observed a large effect size, using a chi-squared we found no statistical difference between students defining their approach in the screencast and whiteboard settings ($\chi^2 = 1.5, p = 0.220671$).

Checking Code against Test Cases. Overall, we found 50% (n = 12) of screencast participants tested their solutions against the provided input to validate the correctness of their solution to technical interview problems. However, only five out of the 24 participants (21%) did so in the public whiteboard setting. These results also show that the number of participants who checked their solution was significantly higher in the screencast setting compared to the whiteboard setting ($\chi^2 = 4.463, p = 0.034637$). Thus, interviewees in this setting may be more likely to evaluate their solution against test cases, contributing to higher success rates for participants solving programming questions in this environment.

4.2.2 *Code Quality.*

Correctness. Participants in the whiteboard interview setting received significantly *lower scores* on their solutions. On average, participants in the whiteboard setting covered 52.7%±43% of the test cases, while the asynchronous technical interview participants passed 79.4%±24.3%. Table 3 presents the scoring rate of test cases passed for participants in the asynchronous and synchronous technical interview settings. A Mann–Whitney U test showed the difference to be significant (U=197.5, p=0.025, d=0.748). It is also notable that none of the participants in the asynchronous technical interviews setting failed all of the provided test cases for their task, while 29% of participants in the traditional whiteboard setting failed to pass a single test case for their problem. To that end, we believe asynchronous technical interviews actually improved technical skills for participants without the presence of an interviewer.

Optimality. We found that 75% (n = 18) of the screencast participants provided the most optimal algorithm for solving the assigned coding question. Comparatively, only 13% (n = 3) of participants in the traditional whiteboard interview setting provided the most optimal solution, with the majority of the synchronous interview participants submitting a sub-optimal or invalid solution to the programming problem. The task performance results for optimality of coding solutions for all participants can be seen in Table 3. The difference in solution complexity between asynchronous and synchronous interview participants is statistically significant ($\chi^2 = 19.40476, p < 0.0001$).

5 LIMITATIONS

The primary goal of the study was to evaluate the impact of an intervention on think-aloud. However, there are also several threats to validity that must be considered when interpreting results.

5.1 Construct Validity

Human perception in evaluating think-aloud quality and stress cues can be limited and biased. We mitigated bias by sampling from different parts of sessions, shuffling the voice samples, using independent rating, and choosing raters from differing genders (male and female). Furthermore, while human perception is limited in making these judgements, in practice, these limitations are mirrored by real interviewers who also use human judgement in evaluating candidate's performance. While our task performance measures were drawn from industry settings, our criteria for rating may differ from how raters across different companies may interpret and apply interview criteria [21].

5.2 Internal Validity

The procedure we used to study think-aloud may influence what conclusions we can draw. Our interview settings may have underestimated the amount of pressure candidates face when performing a high-stakes problem-solving task. For example, real whiteboard technical interviews might even have higher stress and lower communication efficacy in practice. In addition, the different nature of the interview settings limits direct comparison on certain measures. For example, the coding problems may have had different levels of difficulty, thus limiting our ability to draw conclusions based on measures such as time or correctness. To address this limitation, we selected other task performance metrics that would generalize across problems, such as using deliberate strategies and test case verification. As a result, we made an explicit trade-off to focus on observing communication ability in a variety of settings at the cost of having a limited ability to compare task performance.

5.3 External Validity

We have only examined this impact in the context of a limited set of coding challenges from participants at one university. While many students participate in technical interviews, they may not represent other developers on the job market [41], who may perform differently in interview settings. Moreover, the measurements of stress related to speech are biased against a variety of participant backgrounds, and we acknowledge our efforts are primarily based on a Western and neurotypical perspective. To mitigate this, we

Table 3: Participants and Performance

Screen Cast					Whiteboard			
ID	Gender	Question	Score	Complexity	ID	Gender	Score	Complexity
P46	F	Q1	1.00	$O(n)$	P22	M	1.00	$O(n)$
P47	M	Q1	1.00	$O(n)$	P25	M	1.00	$O(n)$
P48	M	Q1	1.00	$O(n)$	P31	M	1.00	$O(n)$
P65	F	Q1	1.00	$O(n)$	P24	M	1.00	$O(n^2)$
P55	M	Q2	1.00	$O(n)$	P33	M	1.00	$O(n^2)$
P56	F	Q2	1.00	$O(n)$	P34	M	1.00	$O(n^2)$
P67	F	Q2	1.00	$O(n)$	P40	M	1.00	$O(n^2)$
P68	F	Q2	1.00	$O(n)$	P41	M	1.00	$O(n^2)$
P53	F	Q3	1.00	$O(n^2)$	P45	M	1.00	$O(n^2)$
P51	M	Q4	1.00	$O(n)$	P39	M	1.00	$O(n^3)$
P63	M	Q4	1.00	$O(n)$	P23	M	0.67	–
P69	M	Q4	1.00	$O(n)$	P32	M	0.33	–
P52	M	Q1	0.83	$O(n)$	P42	F	0.33	–
P61	M	Q1	0.83	$O(n)$	P26	M	0.33	–
P59	F	Q3	0.80	$O(n^2)$	P30	F	0.33	–
P49	M	Q2	0.67	$O(n^2)$	P36	M	0.33	–
P54	M	Q3	0.60	$O(nlogn)$	P38	M	0.33	–
P57	M	Q3	0.60	$O(n^2)$	P27	M	0.00	–
P58	F	Q3	0.60	$O(n^2)$	P28	M	0.00	–
P66	F	Q3	0.60	$O(n)$	P29	F	0.00	–
P60	M	Q1	0.50	$O(n)$	P35	M	0.00	–
P62	M	Q1	0.50	$O(n)$	P37	M	0.00	–
P50	M	Q1	0.33	$O(n)$	P43	F	0.00	–
P64	M	Q4	0.20	$O(n)$	P44	F	0.00	–

Score and complexity of solutions for participants in each settings. In the score column, black indicates passed test cases while red indicates a failed test.

attempted to recruit a diverse sample of participants and a diverse team of evaluators. However, our results may not generalize to all candidates completing technical interviews. For future work, a larger collection of studies, including participation from industry professionals and active software engineering job seekers, would be valuable in establishing the generalizability of these results.

6 DISCUSSION

Our findings demonstrate that asynchronous technical interviews have considerable advantages for candidates, including improving their think-aloud quality (Section 4.1.1) and reducing stress (Section 4.1.2), allowing more accurate communication about their problem-solving abilities. Further, we found asynchronicity improved technical performance by helping candidates adopt better problem-solving strategies (section 4.2.1) and produce more correct and optimal programming solutions (section 4.2.2). These results are consistent with findings by Behroozi et al. [6, 10], who show that privacy benefits candidates' performance. In contrast to private interviews, we believe this approach can enhance technical interview processes by allowing employers to have a more accurate evaluation of candidates' thought processes and coding abilities.

However, we recognize naively incorporating asynchronous technical interviews into current software engineering hiring processes is not feasible without further consideration. For instance, contemplating potential communication and ethical concerns as well as other trade-offs to implementing this approach in industry. To that end, we aim to provide guidelines for improving think-aloud quality and while preserving technical performance during technical interviews based on our results.

6.1 Communication Trade-offs

To navigate the tension between facilitating candidate privacy and asynchronicity in our approach while maintaining the necessary interviewer-interviewee interactions during technical interviews, we discuss trade-offs and offer tactics for incorporating these concepts into the communication and technical aspects of software engineering hiring processes.

6.1.1 Follow-up Questions. One potential concern with removing synchronous communication during technical interviews is the loss of dialogue between interviewers and interviewees. In asynchronous technical interviews, candidates submit recordings of problem-solving sessions to be retroactively reviewed by potential employers. While this significantly improves communication and technical aspects of candidates' interview performance, this approach critically removes the opportunity for the interviewer to ask prospective employees follow-up questions during interviews, such as asking about alternative approaches or discussing the motivation behind a particular design decision.

Guidelines. Our results show that asynchronicity enhanced candidates' communication ability by significantly increasing informativeness and effectiveness while reducing stress. However, additional modifications to asynchronous technical interviews can accommodate further dialogue to gain increased insight into candidates' problem-solving approaches. For example, when reviewing a recording, interviewers can add comments or questions at specific points in the video, which can then be clarified by the author retroactively. Another approach might involve providing additional guidelines to candidates up front with the coding problem about what should be discussed, such as identifying deficiencies with solution or describing alternative approaches. These guidelines could even be interspersed, such that the candidate must submit the recording first, and then submit a second recording that includes answers to the follow-up questions.

6.1.2 Soft Skills and Company Evaluations. Further, the asynchronous nature of our approach prevents interviewers from evaluating candidates' soft skills, or non-technical and interpersonal abilities of potential employees in particular contexts [27], which are crucial for software engineering work [33]. For example, in asynchronous technical interviews employers are unable to inquire to learn more about interviewees' background, experiences, personality, and character to determine their fit for the role. From candidates' perspectives, asynchronous technical interviews prohibits interviewees from asking questions to interviewers to evaluate company culture and expected responsibilities for roles. Similar to how employers evaluate the soft skills of candidates to determine their fit for development teams, these conversations also play a vital role for job seekers to make decisions about culture fit if offered a position.

Guidelines. To incorporate soft skills and company evaluations with asynchronous technical interviews, we suggest employers vary interview types throughout the software engineering hiring pipeline. Think-aloud is often used to gauge candidates' ability to clearly communicate and perform a problem-solving walk-through [21]—however, this is often wrongly conflated with evaluating soft skills. Instead, we argue that an asynchronous approach would allow for assessment of a problem-solving walk-through while separate traditional behavioral interviews or high-level design questions would be more appropriate for assessing soft skills and team fit. For example, at Netflix, some interviewers incorporate a short technical presentation on any topic as part of the interview process. Further, separate sessions could be used for candidates to learn about the culture of organizations and expectations for roles. Thus, by including various types of interview formats in a hiring pipeline, a more complete, accurate, and fair assessment of candidates can be obtained.

6.2 Ethical Trade-offs

Incorporating privacy and asynchronicity into technical interview processes could also introduce ethical dilemmas. We introduce several moral predicaments for both employers and candidates, and provide guidelines to prevent dishonesty and corruption from implementing asynchronous technical interviews in software engineering hiring processes.

6.2.1 Cheating. One complication of integrating asynchronicity into technical interviews is the potential for candidates to cheat. We found that removing evaluator presence significantly improved candidates' technical performance—leading participants to produce more precise and optimal code. However, without synchronous interviewer supervision it would be difficult to prevent interviewees from potentially looking up answers or using external resources while solving programming challenges. Cheating ultimately inhibits employers from getting an accurate evaluation of candidates' knowledge, skills, and abilities to make hiring decisions.

Guidelines. To prevent cheating in asynchronous technical interviews, we suggest formatting hiring evaluations to better reflect realistic job settings. For instance, one software engineer declares "I have a confession to make. I cheat at my job. I cheat all day, every day...I found this website called Stack Overflow that has so many answers to problems I run in to. Sometimes I'll just copy the code directly from the site, without typing it out again myself! Sometimes I even just walk up to colleagues and straight up ask them for help with a problem" [40]. With screencast recordings, employers have the ability to view all resources utilized by interviewees during coding tasks, in addition to observing their problem-solving abilities. Thus, we posit incorporating actual software engineering affordances into technical interviews, including permitting the use of external resources, can mitigate the impact of cheating in technical interviews and enhance how candidates are evaluated and assessed for software engineering positions.

6.2.2 Invasion of Privacy. Another ethical trade-off for asynchronous technical interviews is, despite the increase in privacy for candidates, an invasion of candidate privacy by employers. For instance, with recent shifts to online education, online proctoring systems such as Proctorio[9] have been employed to prevent cheating by monitoring students completing work remotely and without in-person supervision from teachers. However, users frequently question the legality of Proctorio and claim the system violates privacy rights with features such as browser locking and webcam access—leading to outcries from students, instructors, and privacy advocacy groups [4]. Similar practices have been implemented in software engineering hiring processes. For example, Amazon has used ProctorU,[10] another online proctoring platform, during online job assessments to gain access to interviewees' microphone, mouse, webcam, and browser to obstruct online activity, shut down running applications, block screenshots, and track head, eye, and mouth movements [31]. One candidate blogged about their Amazon software development engineer interview experience, noting "the normalization of privacy violation has never felt more real" [39].

Guidelines. Our recommendation is that employers balance their needs to verify and observe a candidate's knowledge and skills without overreaching. One way forward is to build in trust into the process. For instance, some companies have transitioned to using "mini-workday" interviews, where candidates are given a scoped project that can be accomplished within a given period of time for submission. While these "day in the life" evaluations eliminate think-aloud, they give candidates a sense of common tasks in the target

[9] https://proctorio.com/
[10] https://www.proctoru.com/

role and provide employers with a chance to see how the candidate would perform and if they are a good fit for the position [46]. Another way to allow for interactions without the pressure of direct supervision is by incorporating collaboration in interviews through processes such as pair programming, where candidates and employers work together to complete programming challenge. Pair programming is a common industry practice that provides many benefits to development teams [11], and may also give a better assessment of candidates' communication and technical abilities while providing an idea of how they work in team environments.

6.3 Other Considerations

6.3.1 Time Commitment. Finding qualified candidates is a substantial investment, and hidden costs spent on interviewing and evaluating candidates quickly add up [16]. In communications with CoderPad,[11] a popular online platform for live coding interviews, a common concern with interviewers across companies is the raw time they have to spend with candidates is highly disruptive to the workweek—an engineer can be involved in as many as 3–7 interviews a week. Likewise, for candidates the amount of practice, and sheer number of interviews they must undergo can involve many months of work—leading to biased assessments and frustration for software engineers on the job market [8]. Overall, the traditional interview process requires huge time commitments from both parties, while yielding a low placement rate.

Guidelines. Asynchronous technical interviews offer a unique opportunity to reduce effort and time involved in technical interviews. If support for asynchronous technical interviews is integrated into online platforms, such as CoderPad, then several opportunities for automation and time-saving could be gained. First, automated testing of code could filter out low-performing candidates, allowing for manual effort to be focused on more promising candidates. Second, automated annotation of the recording (e.g., marking audio hotspots) could guide interviewers to salient parts of the interview to review or revisit. Finally, for candidates, a few recorded asynchronous technical interviews in a centralized platform could be used to apply to multiple positions rather than performing each interview manually. Thus, a candidate would be able to maintain a live portfolio of their problem-solving process rather than having to demonstrate these endlessly per interview.

6.3.2 Relationship to Other Assessments. Finally, it is worth considering how asynchronous technical interviews fit in relationship to other common interview formats. In addition to traditional whiteboard interviews, other hiring assessment formats such take-home tests and coding quizzes also exist.

The closest assessments to asynchronous technical interviews are take-home tests, with a few critical differences. In a take-home test, a coding problem is assigned to a candidate to be completed within a certain time frame, and then returned to the interviewer by a deadline [46]. For example, in a recent take-home test provided by GitHub, a candidate was asked to implement a basic REST API for data storage and create a well-formatted pull request within 5 hours of starting the task—the amount of time allocated was not advertised ahead of time. Similarly, AutoIterative[12] offers take-home tests where candidates have two weeks to develop a feature in a production environment, correct against edge cases, and optimize their solution and performance score. Take-home tests are often lauded for their realistic task settings, and companies implementing similar practices are praised for "hiring without whiteboards" [44]; however, common complaints from candidates include the longer time commitment required [48], and interviewers are unable to assess communication skills via think-aloud. Further, quizzes remove interactions between interviewers and interviewees while assessing candidates' knowledge and abilities in unrealistic software engineering environments. Asynchronous technical interviews offer a mechanism for candidates to demonstrate their problem-solving and communication skills within a shorter time period.

7 RELATED WORK

The work by Behroozi et al. [6, 7, 10] is the closest related work in terms of research goals and methodology. Using head-mounted eye trackers, Behroozi and colleagues conducted a controlled study with 48 participants split into two groups, with one group solving one problem on whiteboard attended by a proctor, and another group solving the same problem without a proctor in a private room. The results showed that participants in the private interview had lower cognitive load, lower stress levels, and achieved more optimal solutions and higher problem-solving scores. Our work complements this research by comparing another interview format and considers other criteria essential in a technical interview. Another closely related paper [37] also examined speech and other features, such as facial expressions, during interviews. However, their work was done in the context of predicting candidate performance in behavioral interviews, whereas our work focuses on understanding how the interview setting influences performance.

Other researchers have studied technical interviews more broadly. Matturro et al. [33] reported 30 skills essential to software engineering positions. According to them, the top-five soft skills are: *communication skills*, *teamwork*, *analytical and problem-solving skills*, *organizational skills*, and *interpersonal skills*. Ford and colleagues [21] conducted a study from the perspective of hiring managers and university students participating in mock technical interviews. The study identified a mismatch between candidates' expectations of what interviewers assess and what they actually look for in a candidate. Behroozi and colleagues [8] conducted a qualitative study on comments posted on Hacker News,[13] a social network for software practitioners, to derive concerns about hiring processes. In a closely related study, they investigated software engineering candidates' experiences in the hiring pipeline through qualitative analysis of posts on Glassdoor[14]—a job review website [9]. Their findings pinpoint poor practices in the hiring processes of software companies. Researchers have also investigated challenges faced by disadvantaged and low-resource job seekers [5, 38], the effectiveness of resources such as online career mentoring [47], and alternative job seeking interventions, such as *speed dating* [15].

[11] https://coderpad.io/

[12] https://autoiterative.com/

[13] https://news.ycombinator.com/

[14] https://www.glassdoor.com/

8 CONCLUSION

Supervised think-aloud in technical interviews unnaturally synchronizes problem-solving with communication. The result is a forced and stressful interaction where less information is conveyed and with more harm inflicted on technical solving ability. Our paper describes an alternative method for evaluating the communication ability of a candidate in a technical interview—asynchronous technical interviews. Our results show that an asynchronous interview setting has considerable advantages for candidates, including improving their think-aloud quality, reducing stress, and allowing a more accurate assessment of their coding and problem-solving abilities. To that end, we posit several trade-offs and guidelines for incorporating asychronicity in technical interviews.

REFERENCES

[1] Arman Atoofi and Hammad M Sadiq. 2018. An Investigation of Linguistic Complexity by Sex and Minority Status Under Stress. (2018).

[2] Adnan Aziz, Tsung-Hsein Lee, and Amit Prakash. 2015. *Elements of Programming Interviews in Java: The Insiders' Guide* (2 ed.).

[3] Dallin J Bailey and Christopher Dromey. 2015. Bidirectional interference between speech and nonspeech tasks in younger, middle-aged, and older adults. *Journal of Speech, Language, and Hearing Research* 58, 6 (2015), 1637–1653.

[4] David G. Balash, Dongkun Kim, Darika Shaibekova, Rahel A. Fainchtein, Micah Sherr, and Adam J. Aviv. 2021. Examining the Examiners: Students' Privacy and Security Perceptions of Online Proctoring Services. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. USENIX Association, 633–652. https://www.usenix.org/conference/soups2021/presentation/balash

[5] Colin Barnes. 2000. A working social model? Disability, work and disability politics in the 21st century. *Critical Social Policy* 20, 4 (2000), 441–457. https://doi.org/10.1177/026101830002000402

[6] Mahnaz Behroozi, Alison Lui, Ian Moore, Denae Ford, and Chris Parnin. 2018. Dazed: Measuring the cognitive load of solving technical interview problems at the whiteboard. In *International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE NIER)*. 93–96. https://doi.org/10.1145/3183399.3183415

[7] Mahnaz Behroozi and Chris Parnin. 2018. Can we predict stressful technical interview settings through eye-tracking?. In *Proceedings of the Workshop on Eye Movements in Programming*. 1–5.

[8] Mahnaz Behroozi, Chris Parnin, and Titus Barik. 2019. Hiring is broken: What do developers say about technical interviews?. In *Visual Languages & Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VLHCC.2019.8818836

[9] Mahnaz Behroozi, Shivani Shirolkar, Titus Barik, and Chris Parnin. 2020. Debugging hiring: What went right and what went wrong in the technical interview process. In *International Conference on Software Engineering: Software Engineering in Society (ICSE SEIS)*. https://doi.org/10.1145/3377815.3381372

[10] Mahnaz Behroozi, Shivani Shirolkar, Titus Barik, and Chris Parnin. 2020. Does stress impact technical interview performance?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 481–492.

[11] Tanja Bipp, Andreas Lepper, and Doris Schmedding. 2008. Pair programming in software development teams–An empirical study of its benefits. *Information and Software Technology* 50, 3 (2008), 231–240.

[12] Bjorn. 2016. Is thinking out loud during an interview really the best strategy? https://softwareengineering.stackexchange.com/questions/102381/is-thinking-out-loud-during-an-interview-really-the-best-strategy.

[13] Laurence Bradford. 2020. Technical Interviewing 101: Ultimate Guide to Acing Your Tech Interview in 2021. https://learntocodewith.me/posts/technical-interview/.

[14] Tony W Buchanan, Jacqueline S Laures-Gore, and Melissa C Duff. 2014. Acute stress reduces speech fluency. *Biological psychology* 97 (2014), 60–66.

[15] Tawanna R. Dillahunt, Jason Lam, Alex Lu, and Earnest Wheeler. 2018. Designing future employment applications for underserved job seekers: A Speed Dating Study. In *Designing Interactive Systems (DIS)*. 33–44. https://doi.org/10.1145/3196709.3196770

[16] Nathan Doctor. 2016. The hidden cost of hiring software engineers—$22,750/hire. https://www.qualified.io/blog/posts/the-hidden-cost-of-hiring-software-engineers.

[17] Christopher Dromey and Emily Bates. 2005. Speech interactions with linguistic, cognitive, and visuomotor tasks. (2005).

[18] Christopher Dromey and April Benson. 2003. Effects of concurrent motor, linguistic, or cognitive tasks on speech motor performance. (2003).

[19] Christopher Dromey and Erin Shim. 2008. The effects of divided attention on speech motor, verbal fluency, and manual task performance. *Journal of Speech, Language, and Hearing Research* (2008).

[20] Denae Ford, Titus , Leslie Rand-Pickett, and Chris Parnin. 2017. The Tech-talk balance: What Technical Interviewers Expect from Technical Candidates. In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 43–48. https://doi.org/10.1109/CHASE.2017.8

[21] Denae Ford, Titus Barik, Leslie Rand-Pickett, and Chris Parnin. 2017. 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)*. 43–48. https://doi.org/10.1109/CHASE.2017.8

[22] Dennis B Fry. 1958. Experiments in the perception of stress. *Language and speech* 1, 2 (1958), 126–152.

[23] Elisabeth Gareis. 2006. Guidelines for public speaking. *Department of Communication Studies, Baruch College/CUNY, New York* (2006).

[24] Cengiz Gunay. 2019. Google Interview Prep Guide Software Engineer - University Graduate. https://soft-eng-practicum.github.io/assets/pdfs/Google%20Interview%20Prep%20Guide%20SWE%20.pdf.

[25] Scott Hanselman. 2014. How do you deal with anxiety when Live Coding in Technical Interviews? http://www.hanselman.com/blog/how-do-you-deal-with-anxiety-when-live-coding-in-technical-interviews.

[26] John HL Hansen and Sanjay Patil. 2007. Speech under stress: Analysis, modeling and recognition. In *Speaker classification I*. Springer, 108–137.

[27] Scott A Hurrell, Dora Scholarios, and Paul Thompson. 2013. More than a 'humpty dumpty'term: Strengthening the conceptualization of soft skills. *Economic and Industrial Democracy* 34, 1 (2013), 161–182.

[28] Indeed. 2020. How to Prepare for a Technical Interview. https://www.indeed.com/career-advice/interviewing/technical-interview-preparation.

[29] IBRAHIM IRFAN. 2020. 6 Steps to Acing the Coding Interview. https://www.ibrahimirfan.com/6-steps-to-acing-the-coding-interview/.

[30] isuckatcslol. 2016. Doing bad in technical interviews. I can't talk and code at the same time. https://www.reddit.com/r/cscareerquestions/comments/3rgjwy/doing_bad_in_technical_interviews_i_cant_talk_and/.

[31] Eugene Kim. 2016. Amazon's online job exam takes control of your laptop and tracks things like mouth movement. https://www.businessinsider.com/amazons-tough-online-job-interview-process-2016-12

[32] Charlyn M Laserna, Yi-Tai Seih, and James W Pennebaker. 2014. Um... who like says you know: Filler word use as a function of age, gender, and personality. *Journal of Language and Social Psychology* 33, 3 (2014), 328–338.

[33] Gerardo Matturro, Florencia Raschetti, and Carina Fontán. 2019. A systematic mapping study on soft skills in software engineering. *JUCS-Journal of Universal Computer Science* 25 (2019), 16.

[34] Anthony D. Mays. 2017. Interviewing at Google? Here's 6 Things You Absolutely Need To Do. https://www.linkedin.com/pulse/interviewing-google-heres-6-things-you-absolutely-need-anthony-mays/.

[35] Gayle Laakmann McDowell. 2019. *Cracking the Coding Interview: 189 Programming Questions and Solutions*. CareerCup.

[36] Microsoft. 2020. Interview tips for all roles. https://careers.microsoft.com/us/en/interviewtips.

[37] Iftekhar Naim, M. Iftekhar Tanveer, Daniel Gildea, and Mohammed Ehsan Hoque. 2015. Automated prediction and analysis of job interview performance: The role of what you say and how you say it. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, Vol. 1. 1–6. https://doi.org/10.1109/FG.2015.7163127

[38] Ihudiya Finda Ogbonnaya-Ogburu, Kentaro Toyama, and Tawanna R. Dillahunt. 2019. Towards an effective digital literacy intervention to assist returning citizens with job search. In *Conference on Human Factors in Computing Systems (CHI)*. Article 85, 85:1–85:12 pages. https://doi.org/10.1145/3290605.3300315

[39] Shivan Kaul Sahib. 2016. 'Clean your desk' : My Amazon interview experience. https://shivankaul.com/blog/clean-your-desk-yet-another-amazon-interview-experience.

[40] Peter Salhofer. 2017. Analysing student behavior in CS courses: A case-study on detecting and preventing cheating. In *2017 IEEE Global Engineering Education Conference (EDUCON)*. 1426–1431. https://doi.org/10.1109/EDUCON.2017.7943035

[41] I. Salman, A. T. Misirli, and N. Juristo. 2015. Are students representatives of professionals in software engineering experiments?. In *International Conference on Software Engineering (ICSE, Vol. 1)*. 666–676. https://doi.org/10.1109/ICSE.2015.82

[42] Abhimanyu Saxena. 2020. What do top tech companies (Google - Amazon - Facebook) seek when hiring? https://www.scaler.com/blog/what-do-top-tech-companies-google-amazon-facebook-seek-when-hiring/.

[43] Jesse Squires. 2021. My worst tech interview experience. https://www.jessesquires.com/blog/2021/12/01/my-worst-tech-interview-experience/

[44] Lauren Tan. [n. d.]. Hiring Without Whiteboards. https://github.com/poteto/hiring-without-whiteboards.

[45] Yangshun Tay. 2017. The 30-minute guide to rocking your next coding interview. https://www.freecodecamp.org/news/coding-interviews-for-dummies-5e048933b82b/.

[46] Glassdoor Team. 2018. The Surprising Ways Companies Assess Job Applicants). https://www.glassdoor.com/blog/ways-companies-assess-job-applicants/.
[47] Maria Tomprou, Laura Dabbish, Robert E. Kraut, and Fannie Liu. 2019. Career mentoring in online communities: Seeking and receiving advice from an online community. In *Conference on Human Factors in Computing Systems (CHI)*. Article 653, 12 pages. https://doi.org/10.1145/3290605.3300883
[48] Cole Turner. 2021. Why I Don't Like Take-Home Challenges. https://cole.codes/posts/why-i-dont-like-take-home-challenges
[49] Jason A Whitfield, Zoe Kriegel, Adam M Fullenkamp, and Daryush D Mehta. 2019. Effects of concurrent manual task performance on connected speech acoustics in individuals with Parkinson disease. *Journal of Speech, Language, and Hearing Research* 62, 7 (2019), 2099–2117.
[50] J. A. Whittaker. 2000. What is software testing? And why is it so hard? *IEEE Software* 17, 1 (2000), 70–79.
[51] Glenn D Wilson and David Roland. 2002. Performance anxiety. *The Science and Psychology of Music Performance: Creative Strategies for Teaching and Learning* (2002), 47–61. https://doi.org/10.1093/acprof:oso/9780195138108.003.0004
[52] David Yang. 2020. Whiteboard Coding Interviews: A 6 Step Process to Solve Any Problem. https://www.fullstackacademy.com/blog/whiteboard-coding-interviews-a-6-step-process-to-solve-any-problem.
[53] Antonio Waldo Zuardi, José Alexandre de Souza Crippa, Jaime Eduardo Cecílio Hallak, and Ricardo Gorayeb. 2013. Human experimental anxiety: actual public speaking induces more intense physiological responses than simulated public speaking. *Brazilian Journal of Psychiatry* 35, 3 (2013), 248–253.