Does Stress Impact Technical Interview Performance?

Mahnaz Behroozi
mbehroo@ncsu.edu
NC State University
Raleigh, NC, USA

Shivani Shirolkar
snshirol@ncsu.edu
NC State University
Raleigh, NC, USA

Titus Barik
titus.barik@microsoft.com
Microsoft
Redmond, WA, USA

Chris Parnin
cjparnin@ncsu.edu
NC State University
Raleigh, NC, USA

ABSTRACT

Software engineering candidates commonly participate in whiteboard technical interviews as part of a hiring assessment. During these sessions, candidates write code while thinking aloud as they work towards a solution, under the watchful eye of an interviewer. While technical interviews should allow for an unbiased and inclusive assessment of problem-solving ability, surprisingly, technical interviews may be instead a procedure for identifying candidates who best handle and migrate stress solely caused by being examined by an interviewer (performance anxiety).

To understand if coding interviews—as administered today—can induce stress that significantly hinders performance, we conducted a randomized controlled trial with 48 Computer Science students, comparing them in private and public whiteboard settings. We found that performance is reduced by more than half, by simply being watched by an interviewer. We also observed that stress and cognitive load were significantly higher in a traditional technical interview when compared with our private interview. Consequently, interviewers may be filtering out qualified candidates by confounding assessment of problem-solving ability with unnecessary stress. We propose interview modifications to make problem-solving assessment more equitable and inclusive, such as through private focus sessions and retrospective think-aloud, allowing companies to hire from a larger and diverse pool of talent.

CCS CONCEPTS

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

KEYWORDS

cognitive load, eye-tracking, program comprehension, technical interviews

ACM Reference Format:
Mahnaz Behroozi, Shivani Shirolkar, Titus Barik, and Chris Parnin. 2020. Does Stress Impact Technical Interview Performance?. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3368089.3409712

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11…$15.00
https://doi.org/10.1145/3368089.3409712

1 INTRODUCTION

Most companies in the software industry administer a technical interview as a procedure for hiring a software developer [3, 49]. Companies believe technical interviews offer a "reasonably consistent evaluation of problem-solving ability, communication skills, and preparedness" [75]. Technical interviews can also give visibility to the personality of the candidate, how they interact with their future colleagues and how much they pay attention to details—such as checking all possible test cases—and knowledge of programming languages. Companies also find it important to reduce unwanted stress [8] during a technical interview, as "not everyone does their best work in fast-paced, high-pressure situations" [52]. In principle, companies should expect the technical interview process to be a sound and fair assessment of a candidate's ability, and thus yield a steady stream of qualified candidates for positions that open up in the company.

Technical interviews can also introduce other effects on candidates who report unexpectedly "bombing" [50], "freezing" [67], or "choking" [59] during this critical hiring procedure. Through a happy accident, the software industry has seemingly reinvented a crude yet effective instrument for reliably introducing stress in subjects, which typically manifests as *performance anxiety* [78]. A technical interview has an uncanny resemblance to the *trier social stress test* [41], a procedure used for decades by psychologists and is the best known "gold standard" procedure [1] for the sole purpose of reliably inducing stress. The trier social stress test involves having a subject prepare and then deliver an interview-style presentation and perform mental arithmetic, all in front of an audience. Alone, none of these actions consistently induce stress in a subject; however, the unique combination of cognitive-demanding tasks with a social-evaluative threat (essentially being watched) is consistent and powerful. If a technical interview is essentially a de facto version of a trier social stress test, then the implications can be profound. Rather than measuring the few that answer correctly in a timely manner, companies are most likely measuring the ability of the few who perform well under stress [31]. Rather than measuring explanation skills, companies are most likely measuring the ability of candidates to handle or mitigate stress (e.g. through practice [58]). Finally, rather than avoiding unwanted stress, technical interviews may be inadvertently designed with the sole purpose of inducing it.

To understand how to maintain the desirable goals of a technical interview (e.g. measure time and correctness), while mitigating the undesirable effects of stress, we created a design probe [77] which removed the social-evaluative threat component of a technical interview. To this end, we designed an interview format where participants *privately* solved a technical problem on a whiteboard, without any monitoring or interaction with an interviewer. We evaluated this format in a randomized controlled trial to understand the impact of stress during a technical interview and whether

we could isolate and dampen its influence. We then compared this to a typical interview format, where a candidate used talk-aloud to explain their problem-solving in front of a proctor. Participants wore specialized eye-tracking glasses, which allowed us to obtain measurements associated with high cognitive load and stress. We then compared the correctness, self-reported experiences, and cognitive load and stress levels of participants across the interview settings.

We found that participants reported high levels of stress and often had difficulty coping with someone else being present while they attempted to solve a cognitively-demanding task. The impact on performance was drastic: nearly twice as many participants failed to solve the problem correctly, and the median correctness score was cut more than half, when simply being watched. In contrast, participants in the private setting reported feeling at ease, having time to understand the problem and reflect on their solution. Finally, measures of cognitive load and stress were significantly lower in the private setting, and majority of participants solved the problem correctly. We also observed that no women successfully solved the problem in the public setting, whereas all women solved it correctly in the private setting. Standard NASA-TLX procedure [35] and various cognitive metrics, provided evidence that higher rates of stress could causally explain these differences across groups.

We suggest that private interview settings have considerable advantages for candidates, that both reduce their stress and allow more accurate assessment of their problem-solving abilities. The implications of this work include several guidelines for more effective administrations of coding interviews. By offering the ability to solve a problem in private, even if only for an initial few minutes, it could substantially increase the amount of qualified candidates, particularly in traditionally underrepresented groups, entering the workforce. Furthermore, we can expect these changes to minimize or mitigate other problematic effects experienced by candidates—such as stereotype threat—that impact interview performance but are orthogonal to assessing candidates' actual problem-solving abilities.

2 METHODOLOGY

The goal of our experiment was to understand whether being watched causes stress in participants and affects their problem-solving. We designed a block randomized controlled trial [72] to compare a traditional version of a technical interview (our control) with an experimental condition involving a private problem-solving session (our treatment). Participants were randomly assigned to a condition, performing the same task—a between-subjects design. In this section, we describe our experimental settings and the data we collected to measure the effects of our experiment.

2.1 Participants

We recruited 50 undergraduate and graduate students at NC State University (12 identified as female), who participated in our study. Students were required to have knowledge of Java or any other high-level programming languages and previous course experience in algorithms and data structures. We made sure that students had the typical knowledge required for a technical interview, including constructing data structures, implementing search and sorting algorithms, and characterizing running time and space requirements. Furthermore, students represent potential technical interview candidates, as many were actively engaged in their initial job search prior to graduation. To use head mounted eye-trackers as a part of our study, participants had to have normal or corrected-to-normal vision. They received extra course credit through an experiment pool and could end the experiment at any time. The study was approved by the local ethics board. The first and second authors of the paper conducted the study.

To estimate our population size for our study, we used the results of our pilot study to obtain estimators for effect size. Through a power analysis of 2-sample 1-sided test with 80% power and 5% Type I error [19], we found that approximately 18 participants would be required in each group.

2.2 Pilots and Tasks

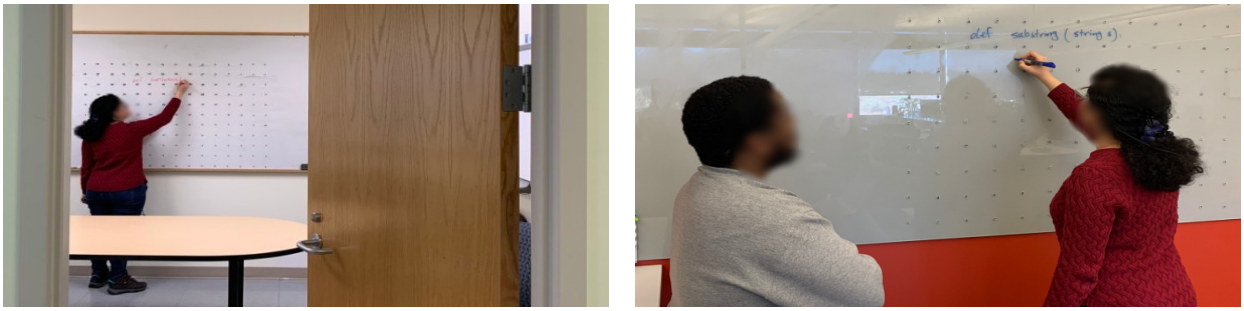
We ran pilot studies at the authors' institutions. During our pilots, we experimented with tasks that would satisfy the following criteria: be solvable within the time limits of the experiment, demonstrate sufficient difficulty such that cognitive load can be induced and is not trivial to solve, and have ecological validity—the problem should be similar to those used in actual interviews. Based on feedback and observations during the pilot studies, we found some tasks were too complex, where participants would quickly give up after a few minutes, and some tasks too ambiguous, which required too many clarification questions. Ultimately, we found the following problem satisfied all our criteria: *Longest Substring Without Repeating Characters*. The problem can be found in "Elements of Programming Interviews in Java (13.9)" [3] and represents a class of problems related to string manipulation, a concept most candidates should be familiar with. For example, we did not want to conflate a lack of familiarity with complex data structures (heaps or red-black trees) or specialized algorithms (dynamic programming or quicksort) with interview performance. Furthermore, the problem itself contains a large solution space with multiple possible approaches, including a brute force solution and other more sophisticated approaches. Finally, we note that—within the past six months—this coding question has been used in technical interviews at well-known software companies such as Amazon, Google, Microsoft, Cisco, Facebook, Adobe, and Uber.[1]

2.3 Procedure

2.3.1 Onboarding. Using a script, the experimenter verbally shared the details of the study with participants. Participants were informed they would be wearing a pair of mobile eye-tracker glasses and recorded audio and video footage would be collected. After consent was obtained, participants filled out a pre-survey to obtain background information.

2.3.2 Interview Formats. We created two interview settings: *public setting* and *private setting* (see Figure 1). In the public setting, wearing an eye-tracker, participants solved the technical interview problem in the presence of an experimenter. The participant was

[1] Based on reports from developers within the LeetCode community, an interview preparation site. The problem can be solved interactively: https://leetcode.com/problems/longest-substring-without-repeating-characters/



(a) Private Interview Setting (b) Public Interview Setting

Figure 1: (a) Solving a problem—alone—in a closed private room. (b) Solving a problem in a presence of an experimenter while thinking aloud.*

* These images are not taken from our actual experimenter and participants.

instructed to talk-aloud while solving the task, which is a standard practice in technical interviews [29, 49]. The lab space contained a whiteboard at the front of the room. An experimenter was situated near the participant for the entire session without interrupting participants' thought process. If a participant asked for clarification, the response was brief. For example, if a participant asked, "Does this look correct", the experimenter was instructed to reply, "Complete to the best of your ability".

In the private setting, participants were provided a private room with a whiteboard to solve the technical interview problem in isolation. Participants were provided a private room with a whiteboard. Participants were informed that the experimenter will step out of the room and will not return while they are solving the problem. Thus, they had to make sure that there is no questions left for them. When the participant was ready to begin the task, they wore the eye-trackers, the door was closed, and the participant worked in privacy, until the task was completed within allotted time. After the experiment, we clarified with participants if they had any uncertainty about the problem they just solved.

2.3.3 Apparatus. Participants wore a head-mounted mobile eye tracking device, SMI Eye Tracking Glasses 2W (SMI-ETG), which participants wore as normal glasses. The SMI-ETG captures both the environment with a high-definition scene camera (960x760 pixel resolution, 30 frames per second), and the participant's eye movements (60 Hz) with binocular inner cameras positioned at the bottom rim of the glasses. The glasses were connected to a mobile recording unit, a customized Samsung Android device. The SMI-ETG is capable of providing metrics related to eye measurements, such as fixations, saccades, and pupil size and projecting the gaze point onto the visual scene and exporting into a video [27].

2.3.4 Eyetracking Data. For each participant and task, we collected screen recordings in video format (at 30 frames per second) and a time-indexed data file containing all eye movements and measurements recorded by the eye-tracking instrument.[2]

2.3.5 Calibration. Calibration improves the accuracy of measurements collected from an eye tracking device. Before calibration, we ensured that the eye-tracker fit comfortably by adjusting the nose bridge. We asked participants to confirm that they do not have difficulty seeing the coding area on the board. To perform the calibration, we initiated the calibration software and then we asked participants to fixate on three markers on the board. Target markers were selected in such a way that they cover different gaze angles.

2.3.6 Experiment. Participants received a printed problem statement. The printout also included three examples, which indicated what the expected output would be when given an input. For example, "Given the input *abcabcbb*, the answer is *abc*, with the length 3." We asked participants to provide a reasonable solution written in a programming language of their choice. We emphasized that their thought process and correctness of the solution were important while efficiency and syntax were secondary. Participants could freely use basic utility functions such as `sort`, if desired.

We asked participants to confirm their understanding of the problem before proceeding. The experiment ended when the participant completed the task or a 30 minute time limit had passed. After the experiment, the participant completed a post-survey and a NASA Task Load Index (TLX) questionnaire [35] to collect information of participants' self-evaluation about their performance. We designed our post-survey to subjectively complement the questions of the NASA-TLX questionnaire.

3 ANALYSIS

We analyzed the data to assess performance on the task, measure the resulting cognitive load, and to identify possible sources of stress or cognitive load.

3.1 Measuring Correctness and Complexity of Solutions

We scored the code against the following test cases, which were also provided to participants:

(1) **Consecutive letters in a substring.** For example, given the input 'abcabcbb', the output is 'abc'.
(2) **Only one letter.** For example, given the input 'bbbbb', the output is 'b'.

[2] Collected eye-tracking data can be found at: https://go.ncsu.edu/technical_interviews

(3) **Non-consecutive letters** in a substring. For example, given the input 'pwwkew', the output is 'wke'.

Passing each test case is worth 1 point. Hence the scores range 0–3. To facilitate evaluation, we went through each solution and manually translated it into executable code, filling in incomplete syntax when necessary. We consider passing all three test cases as indication of a *successful* task, otherwise, the participant has *failed* the task, as done by Wyrich and colleagues [80].

We did not require an optimal solution but we did examine the solution to determine its runtime complexity. For example, a solution using two nested loops would have a $O(n^2)$ runtime complexity.

Finally, we recorded time to complete the experiment. Participants who exceeded the time limit before completing the task were not scored and instead marked with a TIMEOUT.

3.2 Measuring Cognitive Load

We measured cognitive load [73] using two proxy measures well-established in the literature: *fixation duration* and *pupil dilation.*

A fixation is when eyes stop scanning, holding the central foveal vision in place so that detailed information about the stimulus can be caught by the visual system. Fixation duration measures how long the gaze remains engaged in a particular point. A longer fixation is well correlated with task difficulty and indicates increased processing effort [47, 51, 60, 71]. Eye-tracking studies do not traditionally normalize fixations when comparing task difficulty because, in aggregate, they measure effort in a unit of time, and not in a unit of space. It is standard to simply compare lengths of fixations between different text [60], images [16, 82], and code [4, 14, 32] when understanding relative difficulty of tasks.

Pupil dilation measures the size of the pupil over time. Prior studies have identified a relationship between increases in pupil dilation and the mental demands of a task [38, 69]. That is, when people need to maintain more items in active memory or perform more mental operations, the size of their pupils increase.

While fixation data is readily available from our eye-tracker, pupil dilation is notably noisy and requires more extensive data cleaning. Our eye-tracker provides the pupil diameter (millimeters) over time (approximately every 16.7 milliseconds). We adopted guidelines proposed by Kret and Sjak-Shie [43] for filtering out data that coincided with blinks and any values out of pupil diameter normal range (1.5mm–9mm). We then proceeded with removing disproportionately large absolute pupil size changes sympathetic to luminance changes (2–4mm) [6], which are not related to much smaller information-related dilations (.5mm) [33].

To facilitate comparison across participants and reduce noise, we adopted *temporally course measures* [40] by averaging fixation and pupil diameter data in time windows of 30 seconds. Hence, each time window possesses a mean fixation duration and mean pupil dilation size. Lastly, we used an empirically determined cut-off where we would stop analyzing windows when the number of active participants still engaged in the task dropped below 50%.

3.3 Measuring Factors of Task Difficulty

To identify contributing factors of cognitive load, we first measured six factors obtained from the NASA-TLX [35]: effort, frustration and stress, mental demand, temporal demand, physical demand, and performance, each answered on a scale from 1–21. Each factor was averaged across participants and then compared across groups for differences.

To investigate whether stress played a role in performance differences, we also investigated eye movements related to stress. Stress inhibits attentional processes [11] and as a result, eye movements become more difficult to control and slower [12]. In particular, we used *saccade velocity*, which measures the speed of traveling between areas of interest. Because saccadic velocity is not under voluntary control, numerous studies have used it as an unbiased measure of stress level. For example, participants exhibiting high stress had slower eye movements compared with low stress participants in a gaze targeting task [17]. We also obtained temporally course measures [40] by averaging saccade velocity into 30 second time windows.

4 RESULTS

From the 50 participants we recruited, two participants (both were female) decided to terminate and withdraw from the study. Three were unable to complete the task within the allocated time of the experiment. Finally, one participant had successfully completed the task, but had an incomplete recording due to equipment failure, and four participants did not have pupil dilation data available. As a result, we had 48 participants, 22 in private setting and 26 in public setting, that we could use for complete or partial data analysis (e.g. TLX, correctness of solution, etc.).

4.1 Participant Experiences and Observations

Participants shared general challenges with problem-solving in both interview settings. Participants, typically used to coding on a computer, sometimes struggled with performing the task on a whiteboard. Participants deprived of programming environments, with aids such as syntax highlighting and auto-completion, were burdened with recalling low-level syntax details, such as declaring a function or variable. P24 found it "challenging to remember certain aspects of Java while I was working through the solution." P18 had trouble remembering "function calls specific to java for String Builder", which you could usually "just google really quick." Participants also had trouble thinking of the "appropriate data structures and string operations" needed to solve the problems, and strove to achieve "an optimal solution" (P19), despite our guidance that they focus on completing the task correctly. For example, P34 felt they did poorly because they did not provide an "optimized solution immediately; wasting time of the proctor."

However, each interview setting also brought unique experiences and challenges for participants.

Stress from being watched and think-aloud. Most participants in the public setting expressed concerns related to being watched, time pressure, and multiplexing tasks.

Participants expressed feeling "very nervous," "rushed," "stressed," "monitored," and "unable to concentrate." P43 stated "I was constantly supervised, which made me nervous." P25 felt unnerved that someone was "watching me and they are not commenting on my progress." P22 described being "self-conscious about the whole

Table 1: Participants and Performance

Private[1]					Public[2]				
ID	Gender	Score	Complexity	Time (m:s)	ID	Gender	Score	Complexity	Time (m:s)
P10	M	■■■	$O(n)$	13:55	P22	M	■■■	$O(n)$	09:14
P15	M	■■■	$O(n)$	07:52	P25	M	■■■	$O(n)$	11:38
P17	M	■■■	$O(n)$	16:11	P31	M	■■■	$O(n)$	12:26
P20	F	■■■	$O(n)$	–:–[3]	P24	M	■■■	$O(n^2)$	15:30
P21	F	■■■	$O(n)$	22:43	P33	M	■■■	$O(n^2)$	17:08
P04	M	■■■	$O(n^2)$	20:26	P34	M	■■■	$O(n^2)$	13:34
P05	M	■■■	$O(n^2)$	17:25	P40	M	■■■	$O(n^2)$	25:23
P06	F	■■■	$O(n^2)$	07:40	P41	M	■■■	$O(n^2)$	09:31
P09	M	■■■	$O(n^2)$	13:11	P45	M	■■■	$O(n^2)$	08:59
P11	M	■■■	$O(n^2)$	16:19	P39	M	■■■	$O(n^3)$	16:14
P12	M	■■■	$O(n^2)$	08:16	P23	M	■■□	–	19:08
P13	M	■■■	$O(n^2)$	12:07	P32	M	■□□	–	19:40
P16	M	■■■	$O(n^2)$	13:52	P42	F	■□□	–	08:49
P18	F	■■■	$O(n^2)$	26:15	P26	M	■□□	–	09:29
P02	M	■■□	–	04:47	P30	F	■□□	–	06:12
P07	M	■■□	–	21:33	P36	M	■□□	–	06:35
P19	M	■□□	–	07:39	P38	M	■□□	–	14:20
P08	M	■□□	–	17:34	P27	M	□□□	–	06:42
P01	M	□□□	–	10:03	P28	M	□□□	–	09:55
P14	M	□□□	–	21:07	P29	F	□□□	–	14:17
P03	M	□□□	–	13:31	P35	M	□□□	–	18:23
P46	M	timeout			P37	M	□□□	–	06:01
					P43	F	□□□	–	07:28
					P44	F	□□□	–	04:21
					P47	M	timeout		
					P48	F	timeout		

[1] Participants in private setting.
[2] Participants in public setting.
[3] Participant's time data was lost due to battery failure when recording.

process" because their work was visible. One participant similarly became more deliberate, worrying about "making the mistakes that are usually very obvious" in front of someone—they ultimately did not complete the task within the time limit.

The beginning was especially troublesome. Participants struggled with starting the task: "thinking of a solution immediately" (P27), "organizing my mind in a short time" (P29), and "as soon as I see the problem, I become blank" (P36). For these participants, the beginning was often the most stressful part of the task, especially with someone else evaluating them and "taking pauses might make me seem slow-wired" (P22). Even as this initial wave of stress wore off, participants were still aware of time pressure throughout the task: "I feel like the more time I take, the less chance of me *getting the job*" (P25).

Participants also had difficulty with performing tasks that involved multiple simultaneous actions. Participants felt stressed by having to "talk while trying to write" (P44 and P25), and "think and talk and do code at the same time" (P39). P41 found it difficult to "constantly speak during solving" and "lost breath at a few places during the task."

Time and space to work alone. Participants in the private setting particularly noted a reduction of stress. Solving the problem privately helped participants feel "pretty comfortable" (P14), "less stressed" (P20), or even "not feel any stress" (P15). P18 expressed that solving the problem privately is "better than panel" watching you, otherwise, it would be "difficult to code when someone watches over" (P16).

Participants "did not feel rushed" because they were "working alone" (P2). P6 reported "I liked having space + time to think about the problem without worrying about making sure someone looking over my shoulder could follow what I was doing. I did not feel very rushed since I was the one who determined when I was done."

Challenges with isolation. However, the private setting also brought its own unique challenges for participants, relating to lack of feedback and difficulty in time management. P2 expressed uncertainty about "the logistics of the question" while being alone in the room. They realized that they were "unsure about the methods" they were writing and needed feedback to proceed. For one participant, the lack of an examiner was problematic because they

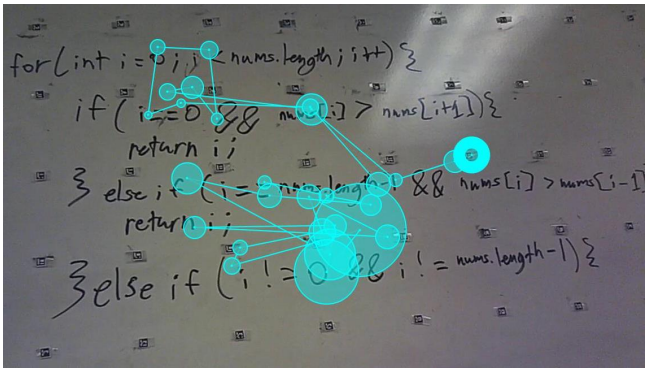


Figure 2: Participants in private settings frequently performed mental execution of test cases to gain confidence in their solution, as indicated by their scan path and utterances from audio recordings.

initially misunderstood the question and had no one available to explain—as a result, they decided to end participation in the study.

Managing time became burdensome for some participants. Participants were not "paying attention to the time during the task" and "remembered it at the very end", resulting in them wondering if they were taking too long (P4). Without having "awareness of time" (P8) at the beginning, when participants realized how much "the time [was] passing", some felt "stressed by a huge margin" (P18).

Observations. Based on replays of video recordings and overlays of eyetracking data, we noted a few observations that helped explain some of the participants' behavior.

Participants had a *higher stability* of eye movement when problem-solving in a private setting versus public setting. That is, participants in public had a more scattered and erratic set of eye movements (characterized by larger visual dispersion and scan paths), whereas in private, the eye movements were more focused and in control. Additionally, participants in private were more likely to perform *mental execution* [81] of their code, that is tracing the behavior of their code under the control of a given test input. For example, in Figure 2, a participant's gaze can be seen following the control flow of the program, while they verbally spoke about the result of each instruction. Many failed participants neglected to perform mental execution of their code when in the public setting—an indication that they may have not felt at liberty to allocate time to it or felt uncomfortable performing mental executions while being observed.

Finally, we also have found some notable instances where participants suddenly *reset* their solution. For example, one participant was more than half way through their solution, when they suddenly erased the board without any declaration or signal. They wanted to start all over, but only five minutes remained in the session, and as a result they could not successfully complete their task.

4.2 Impact on Correctness and Time Performance

Participants in the public setting provided significantly *lower scores.* They also tended to finish their tasks faster than participants in the private setting (on average, 1 minute and 36 seconds sooner); however, not significantly so. Results for all participants can be seen in Table 1. Figure 3 also shows the percentage of the participants in each setting that remained active during each minute of the experiment.

In the public setting 61.5% of the participants failed the task compared with 36.3% in the private setting. The correctness of the solutions in the public setting (excluding timeouts) was 1.58±1.29 with median of 1 passed test case. In the private setting their score was 2.29±1.12 with median of 3 passed test cases. A Mann–Whitney U test showed the difference to be significant (Z=180.5, p=0.038, d=0.57).

Interestingly, a post-hoc analysis revealed that in the public setting, no women (n = 5) successfully solved their task; however, in the private setting, all women (n = 4) successfully solved their task—even providing the most optimal solution in two cases. This may suggest that asking an interview candidate to publicly solve a problem and think-aloud can degrade problem-solving ability.

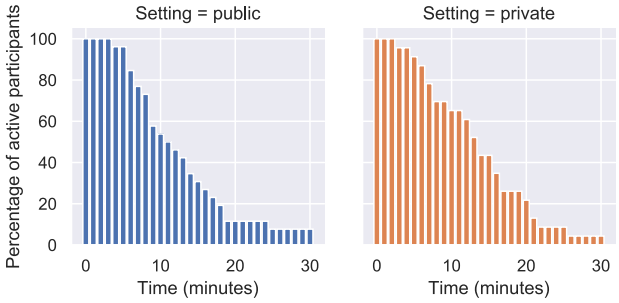


Figure 3: Percentage of participants remaining active during the experiment session. More than 50% of the participants finished their task in less than 10 minutes in the public setting. In the private setting, participants generally took longer.

4.3 Impact on Cognitive Load

Participants in the public setting had a *higher cognitive load.* To enable a better comparison across the settings, we focused on data collected during the first 10 minutes of the programming task, since a majority had already completed the experiment in the public setting by that point (see Figure 3)—i.e., analyzing data after 20 minutes would only include a few participants. Figure 4 shows the mean dilation size changes over time (solid line) and with the colored bands representing 95% confidence intervals. The pupil size remains relatively stabilized and low in the private setting, while it remains high and fluctuates in the public setting.

Participants had significantly longer fixations, a robust measure of cognitive load [47], when problem-solving in a public setting. The mean fixation duration in the public setting (251.48±13.37ms) compared to the private setting (238.99±17.36ms) was significantly different based on a Wilcoxon signed-rank test (Z=25.0, p=0.0028, d=0.78). Participants had significantly larger pupil sizes in the public setting, an indication of elevated cognitive load [69]. A mean dilation size in the public setting (0.13±0.009mm) and in the private

setting (0.12±0.003mm), is significantly higher based on a Wilcoxon signed-rank test between the settings (Z=23.0, p=0.0022, d=1.43).

Returning to Figure 4, interestingly, we observe a decrease typically around five minutes in the public setting. After inspecting participant videos, we hypothesize that this drop is due to participants reaching a partial solution (e.g., their first passing test case). Unfortunately, this respite may be short, as cognitive load continues to rise again as they must address bugs and more difficult test cases. We also note that the two bands do overlap, which indicates that a small subset of candidates in the public setting exhibited the same level of cognitive load as those solving it in private. This represents a wider disparity between participants in the public setting, as some participants may be disproportionately experiencing higher cognitive load.

These results indicate that the public setting increases the *extraneous* cognitive load [73] of a task, which could occur from additional demands from talk-aloud, stress from being watched, or possible changes in problem-solving strategies.

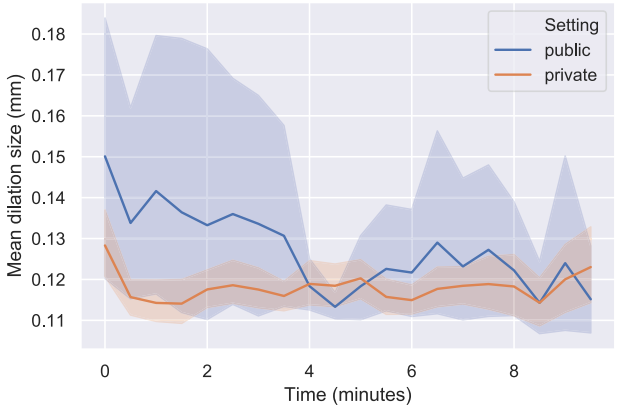


Figure 4: Mean dilation size in first 10 minutes of the experiment across the settings. Participants in the public setting experience larger mean dilation size, indicating a larger cognitive load. Dilation size fluctuates for the public setting participants while it remains stable throughout the experiment in the private setting.

4.4 Influence of Stress on Task Performance

Participants experienced *higher stress* levels in the public setting. A Mann–Whitney U test on median ratings from five NASA-TLX categories did not identify any significance difference in effort, mental demand, physical demand, or performance across settings. However, *frustration and stress* was significantly higher in public (11) vs. in private (7) based on a Mann–Whitney U test (U=184.0, p=0.004, d=0.64).

Participants had *slower eye movements*, a marker for high stress [12], when problem-solving in a public setting. We found that the average saccade velocity was slower for the participants in public setting (94.41±45.59°/s) compared to private setting (120.77±39.43°/s). A Wilcoxon signed-rank confirmed this was significantly different between the settings (Z=277, p=0.003, d=0.59). In Figure 5, a boxplot also depicts the distribution of ranges, including median values (public median=84, private median=119.25).

The results of the TLX survey and slower saccade velocity both indicate that stress was higher in the public setting. Increased stress can explain some of the previously observed increase in cognitive load, independent of the *intrinsic* cognitive load [73] of the task itself.

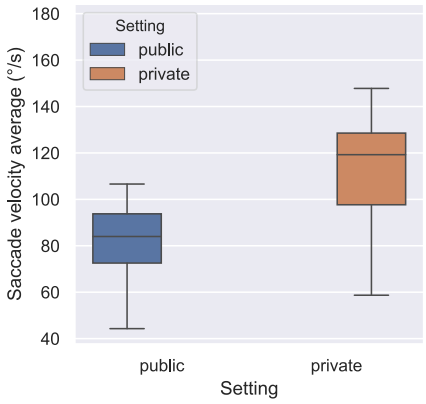


Figure 5: Participants in the public setting had slower eye movements, as indicated by mean saccade velocity, a measure of how fast a person looks away or towards an area of interest. Slow values coincides with high stress, due to inhibited attention processes.

5 LIMITATIONS

Except for one interview, our interviews were all conducted by women. In practice, since most technical interviews are typically conducted by men, additional factors, such as *stereotype threat* [70], may further influence and degrade performance of candidates. A lack of peer parity [30] among underrepresented minorities and hostile or indifferent interviewers [8] could further influence performance.

We have chosen our question to be challenging but solvable. In particular, we asked for candidates to prioritize correctness over runtime optimality. Our reasoning was partly based on results by Wyrich and colleagues [80], where they found that only one participant out of 32 could provide a correct and optimal coding challenge solution. Another reason was that it is a common expectation in interviews for a candidate to first reach a simple solution before engaging in optimization [3, 49]. For our problem, a brute solution $O(n^3)$ is possible [3], yet only one participant used the approach. Participants were also able to obtain the optimal solution $O(n)$ in both settings. Also, participants may have had prior exposure to the problems, though no participant explicitly indicated this.

Some factors of our experiment could affect the generalizability of the results. Our interview settings may be producing less stress than real interviews with real stakes, while adding discomfort from the eye-tracker. Furthermore, our experimenters did not

participate in problem-solving. They did not give feedback if participants were taking a wrong route to the solution nor interrupt with probing questions about the problem-solving process—it is unclear whether an interactive interviewer has an additional *positive* or *negative* effect on performance. Although most students participate in technical interviews, they may not fully represent industry developers [63], who may perform differently in interview settings. Finally, while our technical interview question is used by well-known companies in practice, other coding questions may be easier or more difficult—individual task characteristics [22] may elicit different kinds of mental effort and thus may have different interactions with stress. Similarly, in some cases, small amounts of stress can enhance performance [2]. Hence, we are not claiming that all interview questions will result in the same barriers.

One construct validity issue is the accuracy of the eye tracker. While we used a professional eye tracking instrument, environmental factors can disrupt the accuracy of our measures. For example, our eye tracker was also sensitive to lighting conditions. Furthermore, dynamic and free movement in the environment limited our ability to perform automated analysis of fixed areas of interest, such as fixations on particular words or lines in a coding solution. Some participant data was incomplete, for example, four participants did not have pupil dilation data available, and one participant had no recording due to equipment malfunction during the participant session. We mitigated against this issue by filtering and reducing sources of noise, such as blink events, using temporal coarse values in time windows [40], and using multiple, redundant measures. Future work can consider alternative ways to detect stress [37] and analyze events with normalization [15, 39, 66].

6 RELATED WORK

Despite their importance, technical interviews are understudied in the scientific literature. Ford and colleagues [29] 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. Behroozi and colleagues [8] conducted a qualitative study on comments posted on Hacker News, a social network for software practitioners. 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. Researchers have investigated challenges faced by disadvantaged and low-resource job seekers [5, 54], the effectiveness of resources such as online career mentoring [74], and alternative job seeking interventions, such as *speed dating* [23]. Our study provides empirical evidence validating several of these concerns, and provides additional interventions for job seekers.

Using head-mounted eye trackers, Behroozi and colleagues [7] conducted a preliminary study with 11 participants solving one problem privately on paper and one problem on a whiteboard. They found that the settings significantly differed in metrics associated with cognitive load and stress. The study concludes that "programming is a cognitively intensive task that defies expectations of constant feedback of today's interview processes; however, more studies are needed to better understand what characteristics contribute to high cognitive load." Wyrich and colleagues [80] conducted an exploratory 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. Studies have also characterized the impact of interruptions on programming tasks [20], including more frequent errors [56]. Our study complements this prior work by offering empirical evidence that explains how think-aloud and being watched contribute to lower 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 [44] 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 one technical interview to another 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 [65] investigated a subset of interview data over five years to determine the value of an interviewer's feedback, and found that the four interviews were enough to predict whether someone should be hired at Google with 86% confidence. Unfortunately, this study simply establishes the number of interviews (four) needed to consistently reach a hire/no hire decision, not the accuracy nor validity of the decision. Regardless, our study finds that interview practices may be confounding stress induced from assessment with problem-solving ability.

7 DISCUSSION

Our findings demonstrate that stress impacts technical interview performance; indeed, in our study, participants in a traditional technical interview format

- obtained significantly lower scores (Section 4.2),
- experienced significantly higher extraneous cognitive load (Section 4.3), and
- experienced significantly higher stress levels (Section 4.4).

Furthermore, participants reported (Section 4.1) feeling "very nervous", "rushed", "stressed", "monitored", and "unable to concentrate" as a result of being watched. Participants also reported extraneous cognitive load associated with having to "think and talk and do code at the same time" (P39). When not being monitored, participants were more likely to perform mental execution, allowing them an opportunity to evaluate and build confidence in their solution before having someone examine it. Moreover, our findings suggest that social anxiety may cause the candidates not be capable of communicating their thoughts and solutions, which can add up to their stress and deteriorate their performance.

We have demonstrated how one possible technical interview design, a private interview, significantly reduces the effects of unwanted stress on candidates and results in significant changes in performance, stress, and cognitive load. For companies that want

to avoid conflating assessment of technical and communication skills with handling and mitigating stress, we have provided an experimental framework for designing and evaluating alternative technical interview procedures.

In the remainder of this section, we present *interim* guidance for future directions on technical interviews—consider the recommendations as hypotheses that need further evaluation rather than outright policy.

7.1 Guidance I—Use Retrospective Think-aloud for Accessing Explanation Skills

Although companies want to accurately assess candidates based on their actual skills, they can inadvertently favor a candidate's ability to handle or mitigate stress. In most technical interview formats candidates are asked to think-aloud. Think-aloud protocols are methods to add visibility to cognitive process of a candidate while doing a set of specified tasks rather than only evaluating their final product. There are two main types of think-aloud methods: *concurrent think-aloud* [62, 76] and *retrospective think-aloud* [36, 53].

In concurrent think-aloud, candidates explain their thoughts in tandem with doing the task. Just as subjects given the trier social stress test, candidates who must vocalize their thought-process in real-time risk exposure to a social evaluative threat that can hinder performance in both explanation and problem-solving [21]. Van Den Haak et al. [76] also showed that verbalising thoughts at the time of performing tasks generally increase errors and impede task completion. In retrospective think-aloud, candidates first finish the task and then walk the interviewers through their thought process. That is, candidates can perform the task in their own manner with limited impact on performance [62]. To implement retrospective think-aloud in technical interviews, interviewers can first provide the candidate with a problem description and allow them to privately solve the problem, followed by an explanation of their thought process and a discussion about their solution with the interviewer. This interview format mitigates many of the issues our participants experienced from feeling monitored or supervised, while giving companies an opportunity to still evaluate explanation skills.

Another way to effectively assess explanation skills is to use interview formats that simultaneously reduce the social-evaluative threat and while increasing focus on the explanation component of the assessment. For interviewers who want to assess how the candidate would communicate with other members of the team, the technical problem itself is primarily a shared vehicle through which the interviewer and candidate engage in a conversational dialogue. For example, Microsoft has attempted to reframe technical interviews as as more of a conversation in which both the candidate and the interviewer work together to solve the problem [52]. We recommend starting with *straight-forward problems* that are no more difficult than first-year computer science exercises, and then using these exercises as a way to progressively elicit explanations about their knowledge and experience on different topics. For example, consider the Rainfall Problem[3], a programming task that has been used in a number of studies of programming ability [28].

[3]The basic description of the Rainfall Problem is to write a program that will read in integers and output their average, stopping when the value 9999 is input.

If the candidate does well in their explanation, the interviewer can probe more complex scenarios, such as scaling this problem to a distributed algorithm, or building a user interface for such a system, depending on the expectations of the position. In certain scenarios, it may not even be necessary to fabricate a problem to drive the discussion. At Netflix, some interviewers ask the candidate to "teach something that they know," and the candidate can choose any topic of their interest, job-related or otherwise.[4]

7.2 Guidance II—Evaluate the Kinds of Stress Necessary for Position

Although companies have been mindful to eliminate the influence of stress from their assessment procedures, some view stress as an important characteristic of the job, and should not hire candidates who cannot manage stress: "the real problem is in your head: your anxiety about job interviews is sabotaging something you're otherwise perfectly good at" [8].

In this case, it is important to delineate the kinds of stress a developer would typically encounter in a day-to-day manner. Given important tasks that must be completed in a timely manner, the ability to tolerate stress from *time pressure* is a reasonable consideration [46]. However, time pressure is distinct from other sources of stress [55], which might manifest during an evaluation but not during day-to-day tasks, such as *performance anxiety* [78], commonly called "stage fright," or closely-related *test anxiety* [34]. Therefore, with interview formats explored in our study, it is still possible to assess the ability for a candidate to perform under time-pressure, without evaluating other sources of stress.

If stress is an important consideration for the job, companies should consider ways to help candidates mitigate its effects, such as *stress inoculation* training. As such, interview candidates have been advised to practice (or "grind" [8]) on various problems and solutions in order to become immune to its effects [49]—Mekka Okereke, a senior manager at Google recommends doing at least 40 practice sessions. While stress inoculate training [64] can be effective, developers also note the immense time commitment [8] required to train for technical interviews and the disparity caused by some candidates not having the same available time or resources [54] to train as others. Honeycomb, a women-lead DevOps company, tries to eliminate stress by giving advanced details: "No surprises... unknowns cause anxiety, and people don't perform well when they're anxious. We willingly offer up as much detail in advance about the process as we can, even down to the questions" [45]. Yet, there may be some companies that still need to assess candidates on their stress-tolerance. For them, handling stress may be even more important than candidates' technical skills. In that case, companies should be clear about that requirement. This let candidates to decide whether those companies are suitable for them or that is the culture they want to fit in.

7.3 Guidance III—Provide Accessible Alternatives

We observed several instances where the nature of the whiteboard interview interfered with the candidate's ability to perform their

[4]Personal correspondence with Netflix interviewer.

task. Several changes to the interview procedure can reduce the effects of stress and cognitive load.

7.3.1 Warm-up Interview. The technical interview format is substantially different from how developers do their day-to-day programming activities. To reduce stress, provide the candidate with a warmup exercise that gives them an opportunity to familiarize themselves with the interview setting, experience the interview format, and ask questions about the technical interview process. Ideally, the warmup interview should be conducted by an interviewer who does not score the candidate.

7.3.2 Free Reset. As we found in our study, in stressful situations some participants may forget or "blank" on a particular solution. Interviewers should acknowledge that interviews are stressful situations, and allow the candidate a "free reset" to request another problem or start again on the existing problem without incurring any penalty. Having such a safety net can reduce the candidates' stress and improve their performance in the technical interview. Similarly, dropping the lowest performance interview can reduce noise associated with problematic questions or unfavorable interviewer–candidate interactions.

7.3.3 Partial Program Sketches. If you've ever had to write a paper, you know that starting from an entirely blank page can be both difficult and intimidating. Participants in our study also reported additional stress from having to write a program from scratch, for example, from having to recall the syntax of the language without having any cues. Instead of asking participants to write a program from scratch, interviewers can provide participants with an initial skeleton that contains a partial solution, such as the method signature, a sample invocation, a few input and output examples, and even some sample code snippets for the programming language.

7.3.4 Familiar Affordances. Conducting technical interviews on the whiteboard can also unnecessarily increase the cognitive load for the candidate, especially if the candidate does not routinely write on a whiteboard. To mitigate this, offer the candidate the option to use a laptop, or allow them to solve the technical problems using pencil-and-paper. By doing so, candidates do not have to write code on an unfamiliar medium just for purposes of the interview. Although not well-advertised, interviewers at Google can elect to use pencil-and-paper to conduct their interviews instead of a whiteboard.[5]

7.4 Guidance IV—Consider Impacts on Talent and Diversity

As companies embrace diversity and strive for inclusive hiring practices, companies should evaluate how their technical interviews support or detract from that goal. Hiring procedures that inadvertently exclude large segments of population can contribute to "leaky pipelines" [9], with impact to increased hiring costs [25, 61] and a disproportionate reduction in hiring of minorities and other underrepresented groups [13, 18, 42, 57, 79]. For example, a large population of people are impacted by performance anxiety (estimated 40 percent of all adults in the U.S. [24]). Collectively, otherwise qualified candidates—who happen to perform poorly due to performance anxiety–could be excluded from fair consideration for hiring. Furthermore, scientific evidence finds that women experience disproportionately more negative effects from test and performance anxiety [26, 48, 68], which could explain our observations.

As we observed in our study, candidates who perform in traditional interview setting are more likely to fail, but not necessarily for reasons related to problem-solving ability. As a result, how an individual responses to stress and extraneous cognitive load can be driving hiring decisions instead of ability. For example, if two candidates performed equally well, the job may still go to the candidate who better projects confidence [29]. Beyond gender, these procedures could further impact the performance—and thus exclusion—of other demographics, such as high-anxiety individuals [10] and neurodiverse (e.g., dyslexia, autism, or anxiety) job seekers. Even wider bands of demographics, such as disadvantaged and low-resource job seekers [54] can be impacted by unwanted stress in hiring procedures. For example, Mekka Okereke, a Google senior manager speaking at a recent "Is the Technical Interview Broken?" panel, notes that students at Stanford take courses for passing technical interviews, CS9: Problem-Solving for the CS Technical Interview. Mekka finds that most students typically lack these resources, and specifically runs workshops at HBCUs to provide interview training. He reports that 40 practice interviews are needed to help someone overcome performance anxiety—a time-consuming application of stress inoculation. Companies must decide at what cost they are willing to pay for verifying explanation skills in tandem with problem-solving ability, and what impact that has on their ability to hire diverse and talented candidates.

8 CONCLUSION

Our study raises key questions about the validity and equity of a core procedure used for making hiring decisions across the software industry. We suggest that private interview settings have considerable advantages for candidates, that both reduce their stress and allow more accurate assessment of their problem-solving abilities and can be easily extended to allow assessment of communication skills through retrospective think-aloud. Although this study is one of the first to provide insights into impacts of stress on technical interviews, we have only examined this effect in the context of one coding challenge, from participants from one University. A larger collection of studies, including active participation by industry to pilot alternative hiring procedures, would be valuable for informing how to create a valid and inclusive hiring process for all.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant number 1755762.

REFERENCES

[1] Andrew P. Allen, Paul J. Kennedy, Samantha Dockray, John F. Cryan, Timothy G. Dinan, and Gerard Clarke. 2017. The Trier Social Stress Test: Principles and practice. *Neurobiology of Stress* 6 (2017), 113–126. https://doi.org/10.1016/j.ynstr.2016.11.001

[2] Kristen Joan Anderson. 1994. Impulsitivity, caffeine, and task difficulty: A within-subjects test of the Yerkes-Dodson law. *Personality and Individual Differences* 16, 6 (1994), 813–829. https://doi.org/10.1016/0191886994902267

[5] Personal correspondence with Google interviewer.

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

[4] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. 2017. Do developers read compiler error messages?. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 575–585. https://doi.org/10.1109/ICSE.2017.59

[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] Jackson Beatty, Brennis Lucero-Wagoner, et al. 2000. The pupillary system. *Handbook of Psychophysiology* 2, 142-162 (2000).

[7] 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

[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] Sian L Beilock and Marci S DeCaro. 2007. From poor performance to success under stress: Working memory, strategy selection, and mathematical problem solving under pressure. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 33, 6 (2007), 983. https://doi.org/10.1037/0278-7393.33.6.983

[11] Sonia J Bishop. 2007. Neurocognitive mechanisms of anxiety: An integrative account. *Trends in Cognitive Sciences* 11, 7 (2007), 307–316. https://doi.org/10.1016/j.tics.2007.05.008

[12] Juliana Bittencourt, Bruna Velasques, Silmar Teixeira, Luis F Basile, JoséInácio Salles, Antonio Egídio Nardi, Henning Budde, Mauricio Cagy, Roberto Piedade, and Pedro Ribeiro. 2013. Saccadic eye movement applications for psychiatric disorders. *Neuropsychiatric Disease and Treatment* 9 (2013), 1393–1409. https://doi.org/10.2147/NDT.S45931

[13] Jacob C. Blickenstaff. 2005. Women and science careers: Leaky pipeline or gender filter? *Gender and Education* 17, 4 (2005), 369–386. https://doi.org/10.1080/09540250500145072

[14] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: Relaxing the linear order. In *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, 255–265. https://doi.org/10.1109/ICPC.2015.36

[15] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. 2014. What influences dwell time during source code reading? Analysis of element type and frequency as factors. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. 335–338.

[16] Monica S Castelhano and John M Henderson. 2008. Stable individual differences across images in human saccadic eye movements. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale* 62, 1 (2008), 1. https://doi.org/10.1037/1196-1961.62.1.1

[17] Nigel T M Chen, Patrick J F Clarke, Tamara L Watson, Colin Macleod, and Adam J Guastella. 2014. Biased saccadic responses to emotional stimuli in anxiety: An antisaccade study. *PloS one* 9, 2 (2014), e86474–e86474. https://doi.org/10.1371/journal.pone.0086474

[18] Naomi C. Chesler, Gilda Barabino, Sangeeta N. Bhatia, and Rebecca Richards-Kortum. 2010. The pipeline still leaks and more than you think: A status report on gender diversity in biomedical engineering. *Annals of Biomedical Engineering* 38, 5 (May 2010), 1928–1935. https://doi.org/10.1007/s10439-010-9958-9

[19] Shein-Chung Chow, Hansheng Wang, and Jun Shao. 2007. *Sample Size Calculations in Clinical Research*. CRC Press. https://doi.org/10.1002/wics.155

[20] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A diary study of task switching and interruptions. In *Conference on Human Factors in Computing Systems (CHI)*. 175–182. https://doi.org/10.1145/985692.985715

[21] Jane Dawson. 2003. Reflectivity, creativity, and the space for silence. *Reflective Practice* 4, 1 (2003), 33–39. https://doi.org/10.1080/1462394032000053512

[22] Brian de Alwis, Gail C. Murphy, and Shawn Minto. 2008. Creating a cognitive metric of programming task difficulty. In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 29–32. https://doi.org/10.1145/1370114.1370122

[23] 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

[24] Vivian Diller. 2013. Performance anxiety. https://www.psychologytoday.com/us/blog/face-it/201304/performance-anxiety.

[25] 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.

[26] Nicole M Else-Quest, Janet Shibley Hyde, and Marcia C Linn. 2010. Cross-national patterns of gender differences in mathematics: A meta-analysis. *Psychological Bulletin* 136, 1 (2010), 103. https://doi.org/10.1037/a0018053

[27] Ralf Engbert, Lars O M Rothkegel, Daniel Backhaus, and Hans Arne Trukenbrod. 2016. *Evaluation of Velocity-based Saccade Detection in the SMI-ETG 2W System*. Technical Report.

[28] Kathi Fisler. 2014. The recurring rainfall problem. In *Conference on International Computing Education Research (ICER)*. 35–42. https://doi.org/10.1145/2632320.2632346

[29] Denae Ford, Titus Barik, 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.

[30] D. Ford, A. Harkins, and C. Parnin. 2017. Someone like me: How does peer parity influence participation of women on stack overflow?. In *Visual Languages and Human-Centric Computing (VL/HCC)*. 239–243. https://doi.org/10.1109/VLHCC.2017.8103473

[31] Chris Fox. 2019. It's time to retire the whiteboard interview. https://hackernoon.com/its-time-to-retire-the-whiteboard-interview-qyr32sd.

[32] Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th international conference on software engineering*. 402–413. https://doi.org/10.1145/2568225.2568266

[33] Bram C Goldwater. 1972. Psychological significance of pupillary movements. *Psychological Bulletin* 77, 5 (1972), 340. https://doi.org/10.1037/0033-2909.124.3.372

[34] Bernard W Harleston. 1962. Test anxiety and performance in problem-solving situations. *Journal of Personality* 30, 4 (1962), 557–573. https://doi.org/10.1111/j.1467-6494.1962.tb01689.x

[35] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in Psychology*. Vol. 52. 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[36] Ron D. Henderson, Mike C. Smith, John Podd, and Hugo Varela-Alvarez. 1995. A comparison of the four prominent user-based methods for evaluating the usability of computer software. *Ergonomics* 38, 10 (1995), 2030–2044. https://doi.org/10.1080/00140139508925248

[37] Javier Hernandez, Pablo Paredes, Asta Roseway, and Mary Czerwinski. 2014. Under pressure: Sensing stress of computer users. In *Conference on Human Factors in Computing Systems (CHI)*. 51–60. https://doi.org/10.1145/2556288.2557165

[38] Eckhard H Hess and James M Polt. 1964. Pupil size in relation to mental activity during simple problem-solving. *Science* 143, 3611 (1964), 1190–1192. https://doi.org/10.1126/science.143.3611.1190

[39] Sebastien Jeanmart, Yann-Gael Gueheneuc, Houari Sahraoui, and Naji Habra. 2009. Impact of the visitor pattern on program comprehension and maintenance. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 69–78.

[40] Olivia E. Kang, Katherine E. Huffer, and Thalia P. Wheatley. 2014. Pupil dilation dynamics track attention to high-level information. *PloS one* 9, 8 (2014), e102463. https://doi.org/10.1371/journal.pone.0102463

[41] Clemens Kirschbaum, Karl-Martin Pirke, and Dirk H. Hellhammer. 1993. The "Trier Social Stress Test": A tool for investigating psychobiological stress responses in a laboratory setting. *Neuropsychobiology* 28, 1-2 (1993), 76–81. https://doi.org/10.1159/000119004

[42] Freada Kapor Klein and Ana Díaz-Hernández. 2014. Pattern recognition: How hidden bias operates in tech startup culture. *Crossroads (XRDS)* 20, 4 (June 2014), 20–23. https://doi.org/10.1145/2604991

[43] Mariska E. Kret and Elio E. Sjak-Shie. 2019. Preprocessing pupil size data: Guidelines and code. *Behavior Research Methods* 51, 3 (2019), 1336–1342. https://doi.org/10.3758/s13428-018-1075-y

[44] Aline Lerner. 2016. After a lot more data, technical interview performance really is kind of arbitrary. http://blog.interviewing.io/after-a-lot-more-data-technical-interview-performance-really-is-kind-of-arbitrary/

[45] Charity Majors. 2018. Observations on the enterprise of hiring. https://www.honeycomb.io/blog/observations-on-the-enterprise-of-hiring/.

[46] Mika V. Mäntylä, Kai Petersen, Timo O. A. Lehtinen, and Casper Lassenius. 2014. Time pressure: A controlled experiment of test case development and requirements review. In *International Conference on Software Engineering (ICSE)*. 83–94. https://doi.org/10.1145/2568225.2568245

[47] Gerhard Marquart, Christopher Cabrall, and Joost de Winter. 2015. Review of eye-related measures of drivers' mental workload. *Procedia Manufacturing* 3 (2015), 2854–2861. https://doi.org/10.1016/j.promfg.2015.07.783

[48] Julie M McCarthy and Richard D Goffin. 2005. Selection test anxiety: Exploring tension and fear of failure across the sexes in simulated selection scenarios. *International Journal of Selection and Assessment* 13, 4 (2005), 282–295. https://doi.org/10.1111/j.1468-2389.2005.00325.x

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

[50] Peter Miles. 2018. I bombed my first technical interview, and I'm glad that I did. https://codeburst.io/i-bombed-my-first-technical-interview-and-im-glad-that-i-did-2d218b465298.

[51] Toshiaki Miura. 1987. Behavior oriented vision: functional field of view and processing resources. In *Eye movements from physiology to cognition*. Elsevier, 563–572. https://doi.org/10.1016/B978-0-444-70113-8.50080-1

[52] John Montgomery. 2018. Rethinking how we interview in Microsoft's Developer Division. https://blog.usejournal.com/rethinking-how-we-interview-in-microsofts-developer-division-8f404cfd075a.

[53] Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann. https://doi.org/10.1016/C2009-0-21512-1

[54] 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

[55] Shauna Orfus. 2008. The effect test anxiety and time pressure on performance. *The Huron University College Journal of Learning and Motivation* 46, 1 (2008).

[56] Chris Parnin and Robert DeLine. 2010. Evaluating cues for resuming interrupted programming tasks. In *Conference on Human Factors in Computing Systems (CHI)*. 93–102. https://doi.org/10.1145/1753326.1753342

[57] A. N. Pell. 1996. Fixing the leaky pipeline: Women scientists in academia. *Journal of Animal Science* 74, 11 (Nov. 1996), 2843–2848. https://doi.org/10.2527/1996.74112843x

[58] Napala Pratini. 2018. 6 things no one tells you about the whiteboard interview. https://hired.com/blog/candidates/6-things-no-one-tells-whiteboard-interview/.

[59] PTG. 2006. Ever go to an interview and freeze up, bomb the interview? Woah! I just did! https://arstechnica.com/civis/viewtopic.php?f=25&t=321065

[60] Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124, 3 (1998), 372. https://doi.org/10.1037/0033-2909.124.3.372

[61] Allison Reilly. 2013. How to think about the business impact of recruiting. https://blog.rpoassociation.org/blog/bid/320407/How-to-Think-About-the-Business-Impact-of-Recruiting.

[62] J Edward Russo, Eric J Johnson, and Debra L Stephens. 1989. The validity of verbal protocols. *Memory & Cognition* 17, 6 (1989), 759–769. https://doi.org/10.3758/bf03202637

[63] 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

[64] Teri Saunders, James E Driskell, Joan Hall Johnston, and Eduardo Salas. 1996. The effect of stress inoculation training on anxiety and performance. *Journal of Occupational Health Psychology* 1, 2 (1996), 170. https://doi.org/10.1037/1076-8998.1.2.170

[65] Shannon Shaper. 2017. How many interviews does it take to hire a Googler? https://rework.withgoogle.com/blog/google-rule-of-four/

[66] Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. 2020. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering* (2020), 1–47.

[67] P Sheffield. 2017. Freezing up during live coding interviews. https://www.reddit.com/r/cscareerquestions/comments/6dk4ji/freezing_up_during_live_coding_interviews/.

[68] Monika Sieverding. 2009. 'Be Cool!': Emotional costs of hiding feelings in a job interview. *International Journal of Selection and Assessment* 17, 4 (2009), 391–401. https://doi.org/10.1111/j.1468-2389.2009.00481.x

[69] Sylvain Sirois and Julie Brisson. 2014. Pupillometry. *Wiley Interdisciplinary Reviews: Cognitive Science* 5, 6 (2014), 679–692. https://doi.org/10.1002/wcs.1323

[70] Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. 1999. Stereotype threat and women's math performance. *Journal of Experimental Social Psychology* 35, 1 (1999), 4 – 28. https://doi.org/10.1006/jesp.1998.1373

[71] Patrick Suppes. 1990. Eye-movement models for arithmetic and reading performance. *Eye Movements and Their Role in Visual and Cognitive Processes* 4 (1990), 455–477.

[72] K. P. Suresh. 2011. An overview of randomization techniques: an unbiased assessment of outcome in clinical research. *Journal of Human Reproductive Sciences* 4, 1 (2011), 8. https://doi.org/10.4103/0974-1208.82352

[73] John Sweller. 2011. Cognitive load theory. In *Psychology of Learning and Motivation*. Vol. 55. Elsevier, 37–76. https://doi.org/10.1016/B978-0-12-387691-1.00002-8

[74] 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

[75] Lynne Tye. 2018. Engineering whiteboard interviews: Yay or nay? https://dev.to/lynnetye/engineering-whiteboard-interviews-yay-or-nay-3hko.

[76] Maaike Van Den Haak, Menno De Jong, and Peter Jan Schellens. 2003. Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue. *Behaviour & Information Technology* 22, 5 (2003), 339–351. https://doi.org/10.1080/0044929031000

[77] Jayne Wallace, John McCarthy, Peter C. Wright, and Patrick Olivier. 2013. Making design probes work. In *Conference on Human Factors in Computing Systems* (Paris, France) *(CHI)*. 3441–3450. https://doi.org/10.1145/2470654.2466473

[78] 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

[79] Alison T Wynn and Shelley J Correll. 2018. Puncturing the pipeline: Do technology companies alienate women in recruiting sessions? *Social Studies of Science* 48, 1 (2018), 149–164. https://doi.org/10.1177/0306312718756766

[80] Marvin Wyrich, Daniel Graziotin, and Stefan Wagner. 2019. A theory on individual characteristics of successful coding challenge solvers. *PeerJ Computer Science* 5 (2019), e173. https://doi.org/10.7717/peerj-cs.173

[81] Benjamin Xie, Greg L. Nelson, and Andrew J. Ko. 2018. An explicit strategy to scaffold novice program tracing. In *Computer Science Education (SIGCSE)*. 344–349. https://doi.org/10.1145/3159450.3159527

[82] Alfred L Yarbus. 1967. Eye movements during perception of complex objects. In *Eye movements and vision*. Springer, 171–211. https://doi.org/10.1007/978-1-4899-5379-7_8