A Broken System: Exploring the Failure of Technical Interviews in the Software Engineering Industry Using Technological Momentum

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Quentin Bishop

Spring 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

Introduction: A Broken System

Software engineering is a booming industry that has seen unparalleled growth in the past three decades, growing from 1.8 jobs in 1991 to 4.3 million jobs in 2021 in the USA (U.S. Bureau of Labor Statistics, 2021). During this period, the roles and responsibilities of software engineers have drastically shifted from primarily writing algorithmically efficient code in low-level programming languages such as C and FORTRAN for business and government systems (Chouchanian, 2018) to designing more robust and user-friendly applications utilizing a plethora of newer technologies including the internet, mobile and web applications, machine learning, and object-oriented programming (University of North Florida, 2018). Despite this transition in the skills required for software engineers, the hiring process prevalent throughout the software engineering industry today fails to reflect the technologies of today and is instead reminiscent of the skills and technologies needed in the 90s.

Candidates are selected through a series of technical interviews in which they are tested on their problem-solving ability by being asked to find the most optimal or efficient solution to an algorithmic problem (Google, 2022). Many of these technical interviews test candidates on concepts that are fairly obsolete or completely out of the scope of what the candidate would be expected to do on the job (Kassabian, 2020). While many employers justify these unsuitable interviews as a test of a candidate's problem-solving ability (Google, 2022), technical interviews instead often serve as a test of whether or not the candidate has already encountered and memorized the exact solution from resources in the multi-billion dollar coding-interview prep industry (Behroozi et al., 2020, p. 5). In hopes of landing a job, prospective software engineers don’t experiment with and learn new and useful technologies but instead spend their time and money cramming to memorize algorithmic problems and solutions from prep companies like

Leetcode and books like *Cracking The Coding Interview* knowing that employers often reuse the same technical interview problems year after year (Behroozi et al., 2020, p. 6). As a result, applicants aren't chosen based on their ability to perform the job but rather based on their ability to memorize and regurgitate arbitrary algorithms.

Throughout this paper, I explore the issue of software engineering technical interviews and their effects on the industry. In addition to examining the current state of technical interviews, I explore the origins of the technical interview process and its rise to prominence over other interview styles throughout the industry. Looking at the rise and fall of past interview styles can help shed light on what the future holds for technical interviews. In doing so, I use Hughes' (2009) framework of technological momentum to understand the evolution of the role of technical interviews in the software engineering industry to determine the potential of an improved approach for technical interviews.

Part I: Software Engineering Interviews Have Always Failed To Select The Best Candidates

As previously stated, the current style of technical interviews is fundamentally flawed in the sense that it measures a candidate's memorization of obsolete knowledge. Not only is this detrimental to job candidates, but it also harms employers as they can't determine how well-suited a candidate is to a job. However, these problems aren't unique to the current style of technical interviews - designing interviews to select the best candidates is a problem that has long plagued the software engineering industry.

Technical Interviews Hurt Both Candidates and Employers

Software engineering technical interviews which rely on solving algorithmic problems are almost universal throughout the industry. These interviews are intended to gauge how a candidate approaches solving a new problem. Typically, they are a key stage in a candidate receiving a job offer and without passing the technical interview, a candidate can't move to the next stage in the hiring process. The typical hiring process for software engineering jobs is shown in figure 1.

Figure 1: The Typical Software Engineering Hiring Process (Widmann, 2020)

Despite technical interviews being designed to measure on-the-spot problem solving, nearly all candidates resort to memorizing common technical interview problems and solutions, making these interviews an ineffective way of measuring a candidate's technical ability or potential to succeed on the job. Despite these shortcomings, a study by Xanda Schofield (2015) at Cornell University of software engineering companies reveals that an overwhelming majority of employers consider a candidate's ability to understand and work through technical interview problems the most important factor in evaluating a candidate and deciding whether or not to offer them a job (Schofield, 2015, n.p.). Technical interview performance dwarfed other factors in a candidate's application such as contributions to past projects, educational background, and even knowledge of programming languages needed on the job. Alarmingly, the same study also found that only about 56% of technical interviews are conducted by software engineers, with many

being conducted by individuals without a technical background working in management or human resources departments, who lack the skill set to understand the questions and solutions themselves (Schofield, 2015, n.p.). With technical interviews playing such a significant role in a candidate's application, it's shocking that those without a technical background or understanding of the problems are tasked with determining which candidates have the technical ability to succeed on the job. Even if non-technical interviewers were able to conduct technical interviews flawlessly, there's growing uncertainty as to whether technical interviews measure any technical skills at all.

A study by Barik et al. (2020) at North Carolina State University found that many technical interviews in the software engineering industry test whether a candidate has performance anxiety rather than whether a candidate is competent at writing code (n.p.). The study split a group of computer science undergraduates and asked half to solve an algorithmic problem in a traditional technical interview on a whiteboard in front of prospective employers while the other half were asked to solve the same problem on their own in a private room. The study found that students who worked on the problem in a private room performed approximately twice as well as those who worked on the problem during the technical interview. Barik et al. conclude "Our study suggests that a lot of well-qualified job candidates are being eliminated because they're not used to working on a whiteboard in front of an audience … In short, the findings suggest that companies are missing out on really good programmers" (n.p.).

It's difficult to understand how technical interviews that test memorization of archaic concepts, are often administered by non-technical individuals, and measure performance anxiety as opposed to technical ability have risen to prominence in such an innovative industry. In order to do so, it's essential to examine the origin of the current style of technical interviews.

Technical Interviews Have Failed Before

In the early days of the software engineering industry, technical interviews were nonexistent. Instead, interviews mostly consisted of asking candidates about their past experiences, their educational background, and their strengths and weaknesses. Such a style of interview was prevalent throughout the corporate world and wasn't unique to the software engineering industry (Poundstone, 2011). However, this all began to change with the meteoric rise of Microsoft and its founder Bill Gates.

Partly due to a personal obsession with puzzles and partly due to Microsoft's need to find creative problem solvers capable of out-of-the-box thinking, Gates instituted a new unorthodox interview process at Microsoft that consisted of asking candidates logic puzzles, riddles, or even playing logic games in order to assess how a candidate approaches and works through difficult problems (Childers, 2020, p. 6). This style of interview questions, often referred to as brainteasers, was novel at the time as they required candidates to solve problems on the spot as opposed to coming into the interview with a memorized list of talking points. While many of Microsoft's brainteasers seemed eccentric or even random on the surface, the underlying skills required to solve them would be paramount for succeeding on the job. Microsoft's famous brainteasers such as "How many golf balls would fit in a school bus?" required applicants to creatively break down the problem into many simpler problems on the fly (Childers, 2020, p. 21). For illustration, an exaggerated example I've made to demonstrate the thought process of a candidate might start as follows: "Well a school bus is about 4 cars long, my car is about 15 ft long so let's say a school bus is 60 ft long, now a school bus is also pretty tall, let's say it's about 2 cars tall, well I'm 6'2" and when I stand next to my car my head is about a foot over the top, so

let's say a car is 5ft tall, so that means a school bus is about 10 ft tall, now a golf ball is about the size of a Ferrero Rocher and 20 of those fit in a box about the size of a textbook, so …"

This brainteaser interview style proved to be extremely effective at first and resulted in Microsoft selecting software engineering candidates with strong problem-solving abilities and creative thinking skills (Poundstone, 2011). As brainteaser interview questions became the norm at Microsoft, other companies took note and began asking similar brainteasers during interviews. Before long, brainteasers became the norm for software engineering interviews throughout the industry (Highhouse et al., 2018, p. 2). However, as brainteasers became widespread, so did their solutions. As companies continued reusing the same brainteasers during interviews, more and more applicants had already heard the solutions from colleagues or their own past interview experiences, and soon the brainteaser interview only tested whether a candidate had already heard and memorized the answer to that brainteaser before (Highhouse et al., 2018, p. 3). As the popularity of brainteaser interviews began to decline, Google pioneered a new style of interview that tested a candidate's ability to code efficient solutions to complex algorithmic problems - the current technical interview style.

The Prep Industry

Google's intent with their new style of algorithm-solving technical interviews was similar to Microsoft's - hire candidates who could demonstrate exceptional problem-solving abilities during an interview (Google, 2022). Like Microsoft's brainteaser interviews, the technical interview soon exploded in popularity and was adopted throughout the software engineering industry. However, it suffered the same fatal flaw - candidates soon became familiar with the types of problems being asked and the interviews soon became a test of whether or not the

candidate had seen the problem and solution beforehand. Instead of giving up on technical interviews, companies responded by creating increasingly complex technical interviews to the point where even senior software engineers with decades of experience would have trouble solving the problems (Behroozi et al., 2020, p. 7).

As problems became increasingly impossible for candidates to solve, a coding interview prep industry began to blossom. Companies such as Leetcode began compiling common technical interview questions and solutions asked by software engineering companies (Chawla, 2019). As a result, candidates turned to memorizing problems and solutions in hopes that the same questions would show up in their technical interviews. As the cycle continued, the coding interview prep industry continued to grow to its current estimated value of approximately five billion dollars. As candidates spend more of their time memorizing arbitrary algorithms for their interviews, they spend less time interacting with more useful technologies that are needed on the job (Behroozi et al., 2020, p. 7). Subsequently, the candidates hired are those that have invested the most time into memorizing problems and solutions for their technical interviews and not the ones that have the skills needed to succeed on the job, leading to a situation where companies reject qualified candidates in favor of less qualified ones. In order to better understand the rise and ultimate failure of the software engineering technical interview, as well as gain insight into its future impact and future changes needed in the industry, I propose using the framework of technological momentum.

Part II: Analyzing Technical Interviews through a Technological Momentum Framework

As the previous section demonstrated, the current system of technical interviews is flawed but well-established and difficult to change. Historian of technology Thomas Hughes

(2009) developed the concept of technological momentum to explain how new systems — including practices like technical interviews–become established, gain momentum, and eventually lose momentum.

Technological Momentum

Given that the current system of technical interviews benefits neither employers nor candidates, one can't help but ask the question as to why the process remains so dominant throughout the software engineering industry. Candidates despise the process as they are forced to memorize endless archaic algorithms to solve very niche problems in hopes of landing a job. Employers are also hurt by the process since they miss out on countless highly-skilled applicants capable of creative thinking and instead opt for candidates who can memorize and regurgitate algorithms of no importance on the job. Such a system that is detrimental to both parties should surely be changed, but efforts on behalf of both employers and candidates have ultimately been unsuccessful in scouring the current technical interview process from the industry.

In order to better understand why technical interviews remain so dominant, I propose using Thomas Hughes' framework of technological momentum. In his writings, Hughes describes technological momentum as a framework that utilizes key ideas from both technological determinism and the social construction of technology while also encompassing the complexity of technological change (Hughes, 2009). Hughes explains that technological determinism is "the belief that technical forces determine social and cultural changes" while social construction is the inverse - "the belief that social and cultural forces determine technical changes." Technological momentum is a more complex concept that posits social development both shapes and is shaped by technology in a time-dependent manner. In essence, technological momentum postulates that technology is initially a result of social and cultural forces like in

social constructionism. However, technological momentum postulates that over time technology develops momentum and thus transitions from being shaped by society to shaping society, like in technological determinism (Hughes, 2009, p. 2 - 3). Figure 2 shows how momentum allows the transition from social constructionism to technological determinism.

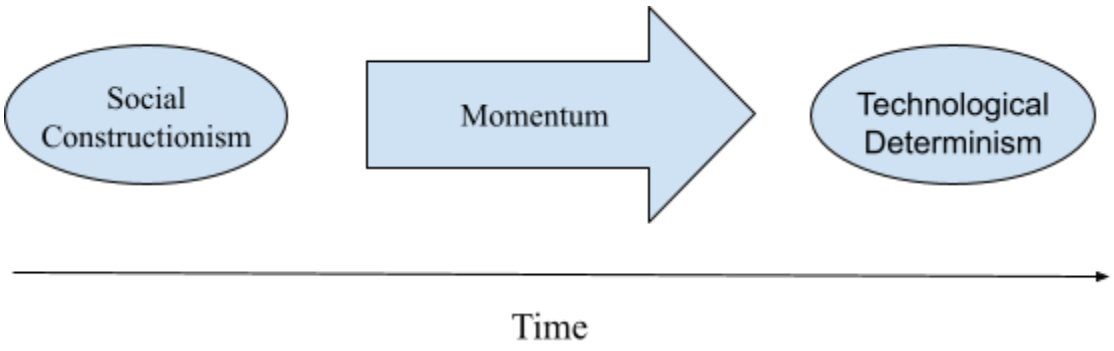

Figure 2: Technological Momentum: The Transition From Social Constructionism to Technological Determinism

Once a technology develops sufficient momentum, it becomes increasingly difficult to contain or mitigate its effects. However, Hughes also notes that technological systems with great momentum are ultimately able to change direction when "a variety of its components are subjected to the forces of change." (p. 149)

Social media platforms like Facebook serve as a textbook example of technological momentum - Facebook was initially developed and succeeded because of its ability to meet people's desire to catch up and keep in touch with old friends, acquaintances, colleagues, and family. However as Facebook continued growing in popularity over time, it became a deeply embedded component of our culture and began to shape the way people interact with each other. Eventually, Facebook developed enough momentum to deeply impact our politics, elections, and society. Only under increasing government regulations and increasing public discontent were

enough of Facebook's components subjected to the forces of change that Facebook needed to introduce new policies regarding privacy, hate speech, and misinformation.

The same framework of technological momentum can be applied to the software engineering technical interview process in order to better understand the issues that plague the system and to find better alternatives.

Part III: Results

Analyzing both current technical interviews as well as brainteaser interviews through the lens of technological momentum can provide invaluable insight into the rise and fall of interviews styles in software engineering. I do this by first looking at how an interview style develops to meet the needs of society through social constructionism. I then analyze the various ways in which these interview styles gain momentum over time. Finally, I explore how a variety of their components are subjected to the forces of change, ultimately leading to a loss of momentum. I believe that understanding the various factors that led to the brainteaser interview's loss of momentum will be instrumental in determining the future of the current style of technical interviews.

Technical Interviews Start Gaining Momentum

In previous sections, I've established that the current technical interview process consisting of writing code to solve algorithmic problems was developed by Google in the early 2010s as a new way to find proficient and qualified software engineers during an era where brainteaser-style interviews were the norm. Throughout the software engineering industry, the brainteaser interview was overused and candidates had resorted to memorizing the answer to as

many brainteasers as possible in hope that they would know the solution to one that would come up in an interview (Highhouse et al., 2018, p. 25). As brainteaser interviews began yielding poorer results and dissatisfaction from both employers and candidates, the momentum brainteaser interviews had developed floundered. Looking through the lens of technological momentum, a variety of its components were subjected to the forces of change, ultimately leading to the decline of brainteaser interviews. In an effort to find highly-skilled candidates capable of critical thinking instead of candidates who simply memorized brainteaser answers, Google's new technical interview required candidates to creatively come up with algorithms on the spot using various data structures. As a result, Google was able to hire many of the top applicants who fueled the company's explosive growth throughout the 2010s (Behroozi et al., 2020, p. 4). In many ways, Google's development of a new interview style was a prime example of social constructionism - Google engineers designed the interview to benefit all involved parties - managers could use the interview to help them select the best candidates, and candidates no longer needed to memorize countless brainteasers.

As Google quickly grew to become one of the biggest, most powerful, and cutting-edge tech companies of the early 21st century, smaller tech companies hoping to recreate Google's unprecedented success began taking note of Google's hiring practices. Before long, companies throughout the software engineering industry started noticing Google's successful new interview style and wanted to use it in order to hire similarly talented software engineers that were responsible for much of Google's growth (Behroozi et al., 2020, p. 6). Hoping to mimic Google's success, other tech giants including Microsoft and Facebook began adopting similar algorithmic technical interviews in lieu of the once-popular brainteaser interviews. Hughes would classify this increasing popularity of technical interviews throughout the industry as the technical

interview gaining momentum and beginning the transition from social constructionism to technological determinism. For the most part, these new styles of interview proved successful throughout the industry and began attracting top talents to companies that utilized Google's new interview style. As most of the technical interview questions were new, candidates had no choice but to come up with creative solutions on the spot which allowed employers to assess candidates' problem-solving abilities. Google's interview style had become the gold standard in the industry as their unique interview style selected top talent. Before long, every company in the software engineering industry wanted their share of talented software engineers, and so more and more companies continued to adopt Google's algorithmic technical interview process, even if the company wasn't entirely in the software engineering industry.

Technical Interviews Achieve Technological Determinism

Technical interviews soon expanded out of the software engineering industry and swept through the entire tech sector. Eventually, technical interviews became so omnipresent that they began to shape the tech sector as a whole - employers in the tech sector consider performance on technical interviews as one of the key factors in deciding whether or not to offer a candidate a job (Schofield, 2015, n.p.). At this point, technical interviews had gained enough momentum that they entered the phase of technological determinism. With technical interviews playing a dominant role in the tech sector, new industries including technical interview preparation began to flourish by taking advantage of the immense significance these new interviews played in candidates finding jobs. These new industries, like the coding interview prep industry discussed in previous sections, only further fuel the momentum of technical interviews and lead to an endless positive feedback loop.

In its current state, the technical interview seems similar to the late stages of the brainteaser interview - unpopular to candidates and employers alike as well as causing active detriment to the reputation and progress of the software engineering industry. It's clear that growing discontent with the current style of technical interviews is the beginning of the forces of change. Looking at the fall of the brainteaser interview can help shed light on the future of technical interviews. In the case of brainteaser interviews, it ultimately took a new and improved style of interviews to demolish the already weakened momentum. Looking at technical interviews in this context suggests that a new interview style that is more capable of effectively selecting qualified applicants will bring about the ultimate demise of technical interviews. However, replacing technical interviews with a new form of interview begs the question as to whether the entire process is cyclical. The new replacement interview might follow the path of brainteaser interviews and technical interviews: gaining momentum, dominating the industry, becoming widespread enough that interviews merely become a matter of memorization instead of a test of skills, slowing the overall progress of the industry, and ultimately being replaced by an even newer and improved style of interview.

Fixing The Problem Requires Companies To Think For Themselves

It may be inevitable that any interview style widespread enough will eventually succumb to becoming a matter of memorization, suggesting that the best path forward is for companies to develop their own interview style that finds candidates best suited for a specific job instead of opting for whatever interview style is trending at the time. To stop future interview styles from achieving similar states of technological determinism, the software engineering industry should focus on the gaining momentum phase; brainteaser interviews stopped working effectively when

everybody copied Microsoft and technical interviews stopped working when everybody copied Google. In order to select the best candidate, companies need to be creative in developing interviews specific to their line of work instead of attempting to replicate interview styles that work well for other companies. As software engineering continues to grow rapidly and become more nuanced, a candidate with a background suited for working at Google may not have the background needed for software engineering at other companies.

Conclusion

Technical interviews in the software engineering industry have entered the technological determinism era of technological momentum, meaning that the interviews themselves are affecting the shape of the entire industry instead of being a tool to measure a candidate's technical and problem-solving ability. Furthermore, technical interviews are causing active detriment to the industry by slowing the pace of innovation and acting as a roadblock to many qualified applicants - the fault is largely on employers opting to use trendy interview styles that don't select the best candidates for the job. Looking through the lens of technological momentum, technical interviews are merely another phase in an endless cycle of interview styles working through the various stages of gaining and losing momentum. As long as employers copy the interview styles of others, the endless rise and fall cycle of new interview styles will continue unchecked. With the software engineering industry continuing to grow, this "one size fits all" approach to interviews will be detrimental. Even if every single company in the industry gave up on the current style of technical interviews today and moved on to a more modern and improved interview style, the new interview would soon gain momentum: a new multi-billion prep industry would replace the old multi-billion prep industry and candidates would continue studying for the

interview instead of for the skills they need on the job. In many ways, it is the act of gaining momentum that leads to the downfall of technical interview styles. This idea could be expanded upon further by exploring Hughes' concept of reverse salients, which are components in a technological system that impede a system from reaching its development goals. Future research could examine technical interviews as a reverse salient in the software engineering industry. As it stands now, breaking the current technical interview process would require a new form of interview that can't be prepared for and measures an applicant's abilities to solve problems on the spot. The best path forward would be to transition away from the universal technical interview and for each company to develop its own type of interview that accurately reflects the duties and candidate will be expected to perform on the job.

References

Barik et. al. (2020, November 11). *Tech sector job interviews assess anxiety, not software skills*. North Carolina State University. Retrieved February 17, 2022, from https://www.engr.ncsu.edu/news/2020/11/11/tech-sector-job-interviews-assess-anxiety-not-software-skills-2/

Behroozi, M., Parnin, C., & Barik, T. (2020). *Hiring is Broken: What Do Developers Say About Technical Interviews?* University of Michigan. Retrieved February 15, 2022, from https://web.eecs.umich.edu/~weimerw/2020-481F/readings/hiring-is-broken.pdf

Chawla, K. (2019, December 19). *Interview Guide - UMD Department of Computer Science*. University of Maryland. Retrieved February 16, 2022, from https://www.cs.umd.edu/~nelson/documents/InterviewGuide.pdf

Childers, M. (2020, December). *Investigating the Validity of Brainteaser Interview Questions*. Ohiolink. Retrieved February 15, 2022, from https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=bgsu160320496813 8574&disposition=inline

Chouchanian, V. (2018). Programming languages. Retrieved February 18, 2022, from http://www.csun.edu/~vgc30838/Projecth.html

Google. (2022). *Google Technical Interviews*. Google Tech Dev Guide. Retrieved February 22, 2022, from https://techdevguide.withgoogle.com/paths/interview/

Highhouse, S., Nye, C. D., & Zhang, D. C. (2018). *Dark Motives and Elective use of Brainteaser Interview Questions*. Applied Psychology. Retrieved February 19, 2022, from https://sites01.lsu.edu/faculty/zhanglab/wp-content/uploads/sites/158/2017/05/apps_1216 3_Rev.pdf

Hughes, T. P. (2009). Technological momentum. In Johnson, D. G. & Wetmore, J. M. (Eds.), Technology and Society: Building our Sociotechnical Future, 141-149.

Kassabian , S. (2020, March 19). *The trouble with technical interviews? they aren't like the job you're interviewing for*. GitLab. Retrieved February 16, 2022, from https://about.gitlab.com/blog/2020/03/19/the-trouble-with-technical-interviews/

Poundstone, W. (2011). *How would you move mount fuji?: Microsoft's cult of the puzzle: How the World's smartest companies select the most creative thinkers*. Little, Brown and Co.

Schofield, X. (2015, April). *What your interviewer wants: Mastering the software engineering interview*. What Your Interviewer Wants: Mastering the Software Engineering Interview - Xanda Schofield. Retrieved February 20, 2022, from https://www.cs.cornell.edu/~xanda/interviews.html

University of North Florida. *What do software engineers do? job types, training, and salary*. University of North Florida. (2018). Retrieved February 16, 2022, from https://bootcamp.unf.edu/blog/what-do-software-engineers-do-job-types-training-and-salary

U.S. Bureau of Labor Statistics. (2021, September 8). *Software developers, Quality Assurance Analysts, and testers : Occupational outlook handbook*. U.S. Bureau of Labor Statistics. Retrieved February 16, 2022, from https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm#tab-1

Widmann, L. (2020, July 13). *Navigating the recruitment process at Simscale: Candidate advice*. SimScale. Retrieved March 3, 2022, from https://www.simscale.com/blog/2020/04/recruitment-process-candidate-advice/