EPISODE 703

[INTRODUCTION]

[00:00:00] JM: Engineers who start companies often find themselves building something that they have no experience building, a hiring process. Hiring engineers today is not as systematic as building software. We don't have lots of data that tells us what makes for an effective programming interview question. The smartest tech companies in the world are still making hiring mistakes often through the false negative of rejecting candidates who did not do well in their interview process, or through the false positive of hiring candidates who did well on the interview, but were not a good fit for the job.

If you are a hiring manager or a company founder, you will eventually have to build a hiring process. If you don't treat that hiring process scientifically, you will likely make some mistakes. Ammon Bartram has conducted more 1,000 interviews with engineers, accumulating a vast amount of data. This data was gathered deliberately and scientifically through closely tracked interview questions and a consistent end-to-end process for the job candidate.

Ammon joins the show to talk about the dataset that he has accumulated. The conclusions from all of these interviews and how engineering organizations can use this data to develop a smart data-driven hiring process. Ammon is cofounder of Triplebyte, a company that helps match engineers and tech companies. Triplebyte also publishes lots of research and blog articles about conducting good interviews, developer salary statistics and bootcamps versus computer science degrees.

Full disclosure; Triplebyte is a sponsor of Software Engineering Daily. However, Ammon has been a guest several times before on the show since before Triplebyte was a sponsor, and I always enjoy getting to talk to him. It's icing on the cake that Triplebyte has become sponsor of Software Engineering Daily because I think the product is really cool and it's something that I would use if I was looking for a job right now as a software engineer.

[SPONSOR MESSAGE]

[00:02:10] JM: Kubernetes can be difficult. Container networking, storage, disaster recovery, these are issues that you would rather not have to figure out alone. Mesosphere's Kubernetes-as-a-service provides single click Kubernetes deployment with simple management, security features and high availability to make your Kubernetes deployments easy. You can find out more about Mesosphere's Kubernetes-as-a-service by going to softwareengineeringdaily.com/mesosphere.

Mesosphere's Kubernetes-as-a-service heals itself when it detects a problem with the state of the cluster. So you don't have to worry about your cluster going down, and they make it easy to install monitoring and logging and other tooling alongside your Kubernetes cluster. With one click install, there's additional tooling like Prometheus, Linkerd, Jenkins and any of the services in the service catalog. Mesosphere is built to make multi-cloud, hybrid-cloud and edge computing easier.

To find out how Mesosphere's Kubernetes-as-a-service can help you easily deploy Kubernetes, you can check out softwareengineeringdaily.com/mesosphere, and it would support Software Engineering Daily as well.

One reason I am a big fan of Mesosphere is that one of the founders, Ben Hindman, is one of the first people I interviewed about software engineering back when I was a host on Software Engineering Radio, and he was so good and so generous with his explanations of various distributed systems concepts, and this was back four or five years ago when some of the applied distributed systems material was a little more scant in the marketplace. It was harder to find information about distributed systems in production, and he was one of the people that was evangelizing it and talking about it and obviously building it in Apache Mesos. So I'm really happy to have Mesosphere as a sponsor, and if you want to check out Mesosphere and support Software Engineering Daily, go to softwareengineeringdaily.com/mesosphere.

[INTERVIEW]

[00:04:29] JM: Ammon Bartram, you are a cofounder at Triplebyte. Welcome back to Software Engineering Daily.

[00:04:34] AB: Thank you. It's good to be here.

[00:04:36] JM: You've done over 1,000 interviews with engineers. How confident are you in your ability to assess an engineer in the limited timeframe of an interview at this point?

[00:04:49] AB: I am confident that I'm getting some real signal. I'm confident that I've improved since we began Triplebyte, and that sort of the lessons we've learned helped sort of our team and me do a better job, but there still is fundamentally a lot of noise in the process. I think it's important to kind of view it through a statistical lens. Somebody interviews and they do poorly or they do well, what that means is, okay, post this interview, there's a higher probability that this person could do the job well or a low probability this person can do the job well. I still very much feel like it's a probabilistic statement and I certainly could be wrong in any one case.

[00:05:28] JM: Across 1,000 interviews and then many more that you did not conduct within the company, you get a pretty detailed dataset, and it's a unique dataset. What kinds of unique data that you feel that you've accumulated from all of these hours of interviews?

[00:05:47] AB: How do I answer that personally? The company as a whole, the really exciting thing that we can do is just keep track of what questions we're asking and then go back after the fact and figure out, "Okay, across all of these thousands of candidates, which questions and which sort of impression of the interviewer are most correlated with candidates doing well at companies?" We can use that to get better overtime.

I guess to answer some of the fundamental, there's a huge amount of debate among even experienced interviewers about what sort of skills matter, what are important questions to ask. This dataset means that we can actually get some empirical answers to some of those questions.

[00:06:27] JM: So you can find out questions that if you ask them, they have a high correlation of the candidates doing well? Is that what you said?

[00:06:38] AB: Yeah. A main thing we do is we interview candidates, then we help them get jobs at companies. So we're able to see the extent to which individuals questions we ask, either correlate or do not correlate with candidates going on to receive job offers or take a job and enjoy that job at a company.

[00:06:58] JM: I guess there are questions that people could ask, like are manhole covers – Or why are manhole covers round, or if you drop an egg from a floor of an N-story building, how high can you get in the building without the egg breaking. These kinds of questions.

[00:07:15] AB: Yeah. I think the minimum number of eggs it takes to determine the floor in a building, where the highest level that you can drop the egg and the egg won't break.

[00:07:22] JM: Right. So from your data, you can detect that these kinds of questions perhaps are low signal and maybe a question like find all the anagrams in a set of words. Maybe that's a higher signal question. You have described the major challenge in the interview process as inconsistencies. The interview questions can vary across interviews. The interviewers might be having a bad day. The interviewee might be nervous on a particular day. When you look at the interviewing process across the entire industry, what are the inconsistencies that you think cause the most trouble?

[00:08:03] AB: Well, first of all, if you attempt to measure how consistent interview results are, almost everyone who does this is a little bit terrified but what they see. Humu people read a book about this — Laszlo Bock. We have to research on this, and interviews are meaningful. They are important. We don't currently have an alternative. We keep doing them, and yet the actual signal that each individual interview delivers is quite a bit less than we'd like to believe. All of those things you mentioned can contribute to noise.

I think one of the biggest factors is basically what the interviewer thinks is important. Each interviewer has some areas where they themselves are likely strong and they understand, they can deeply feel why it's important to know those things. Yet each interviewer also has areas where they're not strong, things that if you ask them group questions in those areas, they would like less intelligent, let's say. Of course, every interviewer is going to ask about areas where they're strong, and that ends up reducing a lot of noise especially if the company hasn't been careful and rigorous about specifying upfront what skills matter for the job.

If the company decides that it's super important that everyone we hire will be strong in algorithms, then it's totally [inaudible 00:09:14] let's ignore for now whether that's the right or the wrong decision. Let's say it's the right decision for that company. That is totally appropriate for interviewers to come in and push candidates hard on details of a particular algorithmic question. But if a company has not specified whether algorithms are important, you have a stage where half of the interviewers are doing that and the other half are going down hard on to API design, which interviewer you speak to is going to have a huge impact on how well you do, because many of the best candidates who are strong in algorithms are going to do much less well on the API design question and vice versa.

[00:09:45] JM: What you said there about there not being a known alternative to the current interview process – And now I've always been a fan of the contract to hire model, or the trial period model. You and I have talked about this a little bit in the past, but I saw in this talk that you gave that many engineers are actually averse to doing that. Engineers do not want to do this. They would rather go through the interview process for all of the lumps that the interview process has. Good engineers often prefer the interview process to a trial period or a contract to hire period even though they would get paid in a contract to hire period. You come on to a company in a part-time capacity for, let's say, two weeks. You get paid. You get to figure out if you like working on the company. The company gets to figure out if they like you. It seems very win-win. Why don't candidates like that?

[00:10:41] AB: Yeah, first of all, I'm actually very much a fan of that model. Yeah, if both parties are okay with that, it is almost certainly a more fair, more accurate way to judge whether someone's a good fit for your company and let them judge whether you're a good fit for them. The problem as you said is that a large portion of engineers – There are two problems. The first problem actually is that the cost of the company of running a trial period is so high that there's no way that you can really afford to do a trial period for every applicant, everyone who could possibly apply to your company.

This is kind of just kicking the problem of the can down the road. If we're going to use contact to hire as part of our process, it's a very expensive step. We usually have to gate keep that somehow. We have to decide which of all the resume we see goes through to contact the hire and that gets you back in the same messy area of trying to evaluate resumes and trying to conduct interviews to determine who goes forward to the contact to hire. That's the first problem.

The second problem is that – So we did some survey, a survey in this across our candidates. We actually ran a version of our interview, it was a take home project, a very long take home project. So some similarities to a trial period, and what we found is that only about 20% of the engineers on our platform are interested in going that route. 80% – For all of the negativity which you hear about interviews, 80% preferred just the higher stress, but one and done approach. I think if you think about the sort of your friends working in the industry, this actually makes more sense, right? Many people are in the position of currently holding a job. They're working somewhere and they like to find a new job. Juggling, figuring out how to take a week off and spend a week working on a different company is a pretty huge commitment.

Especially, if you want to toe dip a little bit and see what it's like at four or five different companies. There's absolutely no way that if you're currently working fulltime at one company, that you're going to be able to go through four or five trial periods at four or five companies to find the place that's best for you. So many prefer the approach, even though it's a bit higher stress of going in, doing those four or five interviews and then being able to have all the offers on the table and make a decision about where they want to work.

[00:12:42] JM: When you're talking about the traditional interview process, at this point, how many hours does it take with an average candidate to have a strong signal for how good of an engineer they are? Can you typically kick out a lot of people in the first interview part of the funnel or do you really need multiple hours to have a strong signal for whether or not they're worthwhile candidate?

[00:13:09] AB: The more you have measured and tuned your process, the more signals you can get quickly. The trial by interview that candidates do – So candidates going through our process on our platform doing two-hour interview with us. For folks who we're hiring to join our core them, we then do an additional four-hour interview after that. It's a total of six hours people we're hiring. I think that's just actually fairly standard for the industry. I think we're maybe slightly on the long side.

However, I think the most interesting way to answer this question is just to look at sort of the optimization we've done on our two-hour interview. The cool thing we've done there basically have – As our interviewers are conducting interview, we have them at five minute increments throughout each problem. Basically assign a sort of quick on the spot grade. Basically, if I were grading this problem right now, what grade would I give?

What that lets us do them, basically as we ask a problem, we can go in and look over a thousands of candidates. Okay, at what point – This problem currently is an hour long and you have somebody working out. At what point in that problem does the interviewer grading basically lock-in to the final grade?

What we've found there is if we put a lot of work into standardizing and trading interviewers, we are able to compress a lot of the standard interview sections down to a shorter version and still get almost all the signal. For example, we tend to do 30-minute coding sections. We actually started out with a 90-minute coding section, which is a big part of our interview. We kind of believe that a larger practical section was likely to be more predictive, and we found that it was.

Then over the last three years, by measuring at which point in that process we actually learn the signal, we've been able to compress that down to 30 minutes and then free up the time in the interview to add other sections and then get more signal. But I do want to add a little bit of a warning there. So done well, backed up by data, it's possible to compress longer sections down and get a signal in 30 minutes on a certain aspect of the candidate.

But one of the problematic ways that bias can seep into interviews is this tendency of people to make snap judgments, to get into a room and just say, "This person gives me a perception of someone who's a good programmer or this person gives me a perception or someone who's a bad programmer, and I think it's really important to be constantly aware of that and then try to fight back and avoid coming with any kind of preconception. If I was starting a [inaudible 00:15:12] team from scratch and I didn't have access to a large dataset, I would definitely want to start with longer sessions, not shorter sessions.

[00:15:21] JM: You are producing a steady stream of content around the interviewing process where you're essentially reporting on these large scale studies that you're doing as you're interviewing lots of candidates for Triplebyte, and some of the stuff that you're producing is really helpful, because you give a good framework for how tech companies can design their own interview process. I think there's a surprising scarcity of well-reasoned material around how to construct an interview process. It makes sense that each company should have its own interview process, because different companies are hiring different kinds of engineers. You and I did a show about cultural fit, and I think cultural fit is one of these things that pervades an organization. Cultural fit is your fingerprint identity of your organization, and that's going to affect how your organization does engineering. Do they all go out to lunch together? How much pair programming do they do?

Ideally, that kind of stuff should seep into the interview process as well, because if you have an organization that's heavily rooted in pair programming and there's no indication of that pair programming in the interview process, you're going to get some candidates through the door that are going to do really well in the interview process and they're going to be like, "I don't want to do any pair programming. I want to sit in front of my computer and code by myself all day."

So this idea of crafting your own interview process, not copying Google's, or Facebook's, or Microsoft's or any other companies, but really defining your own interview process, I think this is really important. So you've spend a lot of time talking about this. So the interview process can really vary on what the company wants the potential hire to do. Obviously, it also depends on the culture, as I just said, but you and I have talked about this before, where some companies will ask you to code a really hard problem around directed acyclic graphs, for example, something like the Dijkstra's algorithm, some complicated algorithm like that, when they're actually just looking for candidates to maintain their Ruby on Rails application for example. It has nothing to do with directed acyclic graphs. How should a company decide what questions to actually ask their candidates? How can they find questions that map closely to the kinds of work they're actually hiring people for?

[00:17:58] AB: Yeah. There's a few different big topics there. Let me back up and I'll talk about the citing questions in a second about sort of customizing the interview content for the culture. Yeah, that's definitely true. It's especially true from the example you gave from providing the candidate a sense of what's going to be like to work in your company. It's okay to have a sales aspect too, that your goal is to sort of sell yourself or your best foot forward. But it's also important, of course, that candidates who are just not going to be happy and not going to fit in, that they're able to learn that themselves and self-select out, because you of course don't want to hire someone who absolutely hates pair programming and it's going to – Or absolutely hates your code review process or whatever. But I actually think – So that's true. Definitely true. But I don't think that that's one of the major problems affecting interview design. Maybe I'm a little bit biased, but it's funny. When you do a thing at large scale, you can just – It kind of all blend together.

But I speak to so many companies about their hiring process and about their engineering process, and they're not that different. I hate to say it, but the majority of companies – There's obviously are a few buckets, but I think most companies running relatively similar interview processes, if those processes were well-designed and well-thought out would be an improvement to what we have right now. So I don't think that reflecting your specific culture should be a guiding principle when designing an interview.

The second part of your question was sort of how to decide whether algorithms are an important sort of topic for your company.

[00:19:17] JM: That's right.

[00:19:18] AB: Yeah. So unfortunately I don't have a great answer here, because we have – If you look at different companies and different very successful companies, you can find examples of – Google, for example, famously, has made computer science, computer science understanding an important part of what they look for, and they do that in spite of the fact that they hire people for roles where that doesn't — They hire frontend developers to work on product and they want those folks to be strong in computer science.

If you ask them why, they say that they think it shows – It's a measure of rigor. They say that creating super high quality, well-designed frontend code is very important, and they say that they like the fact that they have a single bar for level of CS knowledge that folks can more easily transfer from frontend teams to other teams. That makes perfect sense.

You also have a company like Facebook, which has famously taken a sort of a more practical approach, and Facebook tends to be more willing to hire someone who is a very strong frontend hacker and they have zero knowledge of Dijkstra's or perhaps even the concept of complexity analysis. Obviously, both Facebook and Google are immensely successful. So I'm not going to give very specific advice, because there are so many examples of companies that are successful taking different approaches. I've personally taken an approach more skewed toward the practical skills.

I always feel terrible if I reject someone for a job when they could have the underlying work. I'm loathed to look for a skill, let's say, knowledge of algorithms or self-balancing [inaudible 00:20:52] that isn't sort of in use in the role.

[00:20:56] JM: Would you pass judgment on Google there? Is there any chance do you think Google has succeeded in spite of the fact that they are screening people who are just good frontend hackers rather than succeeding because they are screening out those frontend hackers?

[00:21:15] AB: Yes, there's definitely a chance of that. There's definitely a chance of that. The flip side is, you mentioned Dijkstra's as an example of a really hard algorithm. As someone who likes imperial science-like algorithms, that does rough of my feathers slightly just because I can – Dijkstra's is an actually terribly complicated and it's a pretty interesting idea that I do think it's useful to know about.

Part of me does agree with sort of the observation that if you – So if you're a small startup and you're struggling for people to apply, you are absolutely being a fool. You shoot me in the foot if you reject people who can't do your work, because they don't know Dijkstra's algorithm.

But if you're Google and you have access to an extremely ample stream of talent, I do think people who have undertaken the effort to master the theoretical underpinnings of computer science, unbalanced, not all of them, but unbalanced, probably do bring some important rigor and perspective to the company. I'm not necessarily going to criticize a company that has access to enough applicants for screening on those axes.

[SPONSOR MESSAGE]

[00:22:18] JM: Interview Camp is a four-week online boot camp that teaches you techniques for making it through the interview process. If you're trying to land a job at Google, Facebook, or other tech companies, you know that the interview process can be so stressful. Why do I have to learn how to balance a binary tree? Who cares how many of these words are anagrams of each other? Why does it matter that I know how a hashmap works?

Unfortunately, the bizarre hiring process of tech companies is here to stay, whether it makes sense or not. So it's time to learn to play the interview game. Interview Camp takes the stress out of interview preparation. Interview Camp is a four-week online boot camp that teaches you techniques, not just problems.

Their course material is online. They have more than 15 hours of video and more than 100 techniques and curated questions. You have accessed the material for a full year. So you can go at your own pace that suits you.

Interview Camp has weekly live sessions with their students. They answer your questions. They discuss system design and algorithm topics and detail. It's like having a set of peers who are in the same situation discussing topics that you've always had questions about.

Interview Camp is available at interviewcamp.io/p/sedaily. There you can get help with the interview process and support Software Engineering Daily. Again, that's interviewcamp.io/p/sedaily. You might have questions like which company should you apply to. How do you negotiate the best salary offer? Which database should you use for a system design problem? Go to interviewcamp.io/p/sedaily and get help with the interview process.

Thank you to Interview Camp for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:24:30] JM: You emphasized that companies should use a structured interview format, and this is contrast to a freeform intuitive conversational interview. I've had some of these freeform intuitive conversational interviews over the phone with some people, and some people bake it into an interview process where there is a structured interview as part of the interview pipeline, but maybe there's an early conversation where it's — or a late conversation where you just chat with somebody, sort of can you have a half technical, half conversational conversation? But you really emphasized the importance of the structured interview, the come into a room, you're asked a technical question, or two, or three, and you have to answer these and you have to write out code to solve them. Why do you lean on the idea of structured interview so much?

[00:25:23] AB: Yeah. That's one of the results that pretty clear from sort of the research in this area, research into how to reduce bias in interviews, and that's that structure helps. It's a very funny. It's very counterintuitive. If you ask a room full of really experienced excellent engineers who have done many, many interviews about what kind of interview they believe to be most predictive, they almost all tell you that a freeform interview, where they get to sort of ask questions and file their intuition and respond what the candidate says with relevant follow ups. They will almost all say that that is the most accurate way.

You ask a room full of candidates, "What would you prefer?" They almost always also say, "Oh, I think a structured interview – I think a freeform interview is a better way for me to show my skills." But if you look at the actual data on this, and this is true across – This is true in our dataset. This is true across I think nearly every study, engineering and other areas. A structured interview is simply a better predictor of how employees do on the job.

It's more accurate and it's also better at getting over some of the bias that plague us. Some of the sort of preconception people bring in where they judge basically what someone looks like or their gender or their race rather on the skills they see in front of them.

[00:26:37] JM: Perhaps, the intuitive freeform interview feels better, but if we're really talking about a practical interview process that leads to success in the job, structured interviews are better.

[00:26:49] AB: Yes, absolutely. Yeah, actually it goes a little bit more into that. In the best case scenario, structured means that you've done hundreds of interviews for this role and you've standardized all the questions and everything is being asked exactly the same. If you're working in a company that's big enough that you can do that, that's awesome. We do that here. I highly recommend it.

But if you're smaller and you're interviewing just the first people for a role, you obviously don't have the dataset. You don't have a way to put together this sort of massive standardized process, but there's an important part of structured interview, which we still can do, and that is the part when you decide what you're measuring.

Basically, rather than going into a room with a candidate and saying, "My goal is to determine is this person globally a good candidate for my company?" Rather than doing that, you make a list and you say, "Okay, we want people who are very productive. We want people who are strong in backend architecture and we want people with good communication and soft skills." Okay. Let's do three interview sessions, one focus in each of those areas. Line up your interviewers and tell each of them, "Your goal is to go in this room, interview this person and assess their architecture skill. Your goal is to go in this room, interview the person and assess their productivity." So those interviews, get the scores back and make a decision based on those scores.

The research shows that that ends up being quite a bit less biased. So when interviewers are tasked with making a single global decision, so interviewers is someone who decided if we think this is a person, a good person to hire, their biases creep through much more strongly, than if they go in with a targeted aspect they're trying to evaluate. Going in and judging if this person is productive.

So interestingly, even with standardized interview content, simply by being structured about what it is you're trying to assess and then making a decision on those assessments, you end up with a less biased process.

[00:28:33] JM: You're personally focused on the interviewing process and you are, as I've said, you've written a lot about how companies should construct their own interview process. If a company has decided that they are going to do a structured interview, they're not going to do this freeform intuitive interview. They know what they want to focus on. Maybe they're going to maintain their Rails application, and now they need to define what specific questions they're going to ask somebody. How does a company decide what specific questions to ask? Is there some repository of questions they can choose from? Should they find questions that map closely to internal problems that they're having? How do they define what questions to ask?

[00:29:18] AB: You'll notice that when I write about this I often lead with this – I talk about this after the other things. That's intentional. I think this is actually a less important question. I think once you know what answers you care about, once you have sort of a structure there, questions aren't quite as important. Some key things here first of all is that there is an issue if the candidates know the questions in advance.

So if you use questions from the internet or questions from a repository, that may introduce some noise. So I just recommend sitting down with your engineering team, your interview team, and brainstorming. It's my mental process, kind of like writing a blog post, throw out ideas, workshop them, tweak them. The end goal is to ask every candidate the same question. If you spend a week doing this, you can come up with 20, 30 questions fairly easily. From those, pick the top 10 and just go with those. There are some principles I can recommend sort of when evaluating that list and deciding which ones are sort of best.

[inaudible 00:30:13] that I like is the idea that you want to avoid questions where there's some danger that the question might have been given away to the candidate. If you imagine that, let's say this candidate's friend did the interview last week and they had a two-minute Skype call. What could have the friend have told the candidate without giving them a hugely unfair advantage? Could they have read a Glass Door review of you because that's going to happen? That would have been an unfair advantage.

Some questions – For example, earlier today asked, you've given me examples of some questions. There's like the manhole question, for example. Manhole cover question, right? The answer to that question takes 15 seconds to articulate. So if someone's friend tells them to answer that question, they're going to know it and you're going to get zero signal from it.

Compare that to a question where it's something more like, "Here's the game connect for. Please go implement it and maybe add this feature to it." That question – That interview question, the answer is this process. It's going to un-role over 30 minutes, or 90 minutes and involve hundreds of little small micro steps and decisions made by the candidate. Yeah, obviously, if they're told in advance to expect that, they'll be able to prepare a little bit, but there's no way that the essence of the question can be given away. Does that make sense?

[00:31:28] JM: Yeah, it does. Those are just like trick questions basically, and it's not worth –

[00:31:32] AB: They're not necessarily trick questions, right? There could be a totally reasonable question that is going to give you some signal on whether someone is skilled or not. But if it has this sort of leap of insight quality and it can be given away, what that means is it's going to be more noisy. There will still be some signal. I mean, to be totally frank, even the mental question is going to have. If you ask that question to a thousand people and study the, you know, did some [inaudible] on the results, answering it correctly would be a positive indicator of ability to do probably almost any job actually.

[00:32:02] JM: True.

[00:32:02] AB: But the key thing is that question that have a single leap of insight have much more noise, because under the harsh lights of the interview, the person is very stressed. They may well spin their wheels and just not get that [inaudible 00:32:13] insight, or their friend may have told them in advance. In which case, they have this huge advantage and you don't necessarily know that as an interviewer. So you can't distinguish them from an awesomely brilliant person.

[00:32:21] JM: In a previous episode, you and I talked about this idea of hiring for strengths and not lack of weakness. This is a subtle point. It's one that I think a lot of people miss and it's easy to miss, because there are people who have gaping strengths, but they have really strong – Or gaping weaknesses, but they have really strong strengths. They have kind of a barbell quality to them.

Some people might think that that means that you shouldn't hire them, because they're not well-balanced. Actually, in many kind of work environments, you don't need somebody to be decent at everything. You just need them to be really good at one specific thing. But if you think about that, a lot of interview questions are gauging a more general breadth of skills. So should the interview process try to tailor the questions to a candidate's specific strengths?

[00:33:17] AB: I think, no. I totally agree with you. I think it's a really important point. But I think – So the danger of trying to tailor the questions is I can pull you out of the structured interview into the interview where the interviewer is just guiding and sort of as they see fit. That ends up bringing other noise back in.

I think my favorite way to approach this problem is to keep in mind. You realize and you give your managers, your people making the decisions that it really is okay to accept, to hire someone who's weak in the area if they're very strong in other areas that you care about. That's one point.

But the second one is that this step of deciding what skills matter for your company and then designing a structure around that ends up going a long way to solving the problem. Because if you decide that it really is vital that all candidates you hire will be strong in this area, then you do want to fail someone who's weak in that area even if they have other strengths, right? That's part of making that decision. A key point here is that it's okay to sort of have more than one – Let me use the term archetype of engineer you're hiring.

For example, you might decide we want to hire engineers who are either super mathy. They can go work on our machine learning models and do this really hard mathy stuff, and we also want to hire people who are super productive, product hackers. We don't care at all if they know any math. Maybe you have one interview process, but you have – In that case you would ask, the process would be structured. So every candidate would be asked a section whether that was evaluating math and hard analytical stuff and every candidate would have a session looking at product sense and productivity. But then when it comes time to make a decision, you would accept people who were strong in one of those areas even if they showed complete massive weakness than the other. Does that make sense?

[00:34:55] JM: Yeah, it does. I agree with that. Now, the overall experience of interviewing and of hiring people is a funnel. So it begins with the process of somebody seeing an ad, or somebody getting reached out to, somebody seeing a LinkedIn post or something like that, and then they come in for their interviews. You've been focused most on this process, the interview process, and then at the end of it, there's the closing process where if the candidate gets approved, then you have a negotiation around salary and whether or not the person wants to work there.

As a company is crafting the overall funnel, the overall hiring funnel, what are the other subjective things that they should be considering along these other axes, like the amount of ads to run, or how to bring in people to the top of funnel, and how to run that closing process?

[00:35:57] AB: Obviously, [inaudible 00:35:57] and then their problems will all be solved. No. So we can't help all of those steps. But in a broader design – I focus mostly on the evaluation. That's to describe my lens, but that does end up impacting all of these. For example, it is important to keep in mind when designing the interview process that if you like someone after the interview, you have to close them. They're not yours yet. You have to convince them to join your company.

If your interview was terribly negative and they were fighting with the interviewers, they're not going to accept the offer. So this is sort of focusing on some of just communication soft skills on the part of the interviewers. Are your interviewers providing a generally positive experience for the candidates? For example, I think the skills required to be an engineer, like really knowing your stuff, that's not enough to be a good interviewer. You need to have those skills and then you also need to probably in the top 50% of all employees in that company on just sort of empathy, communication, ability to just sort of connect with someone. That can go a long way. I think it's also great to actually mix, honestly, some sort of almost sales sessions into the interview. Maybe into the interview, have a session where it's 30 minutes of the candidate, meeting with an engineer talking about things the company is working on currently and maybe unreleased features and demoing that to the candidate and showing them behind the scenes your stats dashboard and things like that. It's not about the evaluation at all. That's purely about bonding with the candidate and setting it up so that if you make them an offer, that they'll be inclined to take it.

[00:37:31] JM: Right. This is another macro suggestion that you give, the idea of focusing on the overall candidate experience throughout the whole process. So you want to be empathetic about how a candidate is experiencing the interview process, and this can include really subtle things, like do you offer them some Barkling water as they come in the door and you put them in a nice little air-conditioned room and so on. What are some other principles around keeping the candidate experience high and making sure that people don't feel uncomfortable?

[00:38:04] AB: A big one is dealing with candidates, dealing with candidates doing poorly on questions or whole sections of the interview. Again, in line with this idea that you're going to be hiring for strengths, you're going to expect that people who you want to hire are going to have individual sessions where they do poorly, and the individual interviewer who's just doing that section might not know that this person absolutely aced the other sections.

This means that interviewers should never write a candidate off and say, "Oh! Well, this person bombed my section. Therefore they don't matter." They should always assume that – Let's assume this person aced everything else and that we're going to really want to hire this person. So that means that when you ask your question and the candidate can answer it, it's really important that you navigate that in a way that doesn't leave them feeling hostile or negative or overly judged.

So some tips there that we find is ending – If you're doing a section and the candidate is not doing well – Candidates don't like being like cutoff and no feedback. They generally like a little bit of closure. So that forms to that are like ending five minutes early and just doing that little bit of just talk with the candidates. You can ask them to talk through, "Okay. I'm sorry, we're out of time, but we have few minutes left. Can you talk through how you would – The steps you would take to finish this process, which you haven't finished during the session?" They'll talk and you might learn something, right? There may be some signal there and the fact that they can actually talk very well about how they would continue it.

Regardless, that will be sort of a better closure for that section than if you had just said, "Oh, sorry. We're done. Now it's over," and that you'd walk out of the room. Other ideas there is making a small talk. Candidates having this section and then making just 30 seconds of small talk with the candidate about something which is not the evaluation so that you end with some little personal relationship with the candidate. The thing we do a lot here is just talking about their tooling. Most engineers don't really like talking about how they configured their editor. You interview someone for 45 minutes and then you talk for five minutes about shortcuts that have added to their editor. They generally like that. It's a way to decompress a bit.

[00:39:54] JM: I think this part is actually tricky, because in an onsite interview, you're often dealing with maybe five to eight people altogether. You deal with the reception, the recruiters involved somewhere, all the people that interview you. My recollection of this process and people I've talked to, is you always remember the worst part of it. If the recruiter behaves badly for some reason or if one of the people that interviews you seems like they're having a really bad day or they're just like treating you with a sour manner. That is what you remember. It seems pretty hard to craft a holistically positive experience, but it sync the best candidates that end up coming through the door, the ones that you really want to hire. They're going to be remembering the lowest point of the interview. That's what they're going to be anchored to.

[00:40:46] AB: Yeah. That just means that we have to raise our bar as companies and interviewers. So that's a real human – It's how memory works, and we have to know that and up the game sort of across the board. It's not okay for the receptionist to be rude to the candidate, right? It's not okay for people at lunch to be with the candidate. In terms of closing rate, especially, those details all matter. So trying to – Making sure that people come over on our talk, the candidate during lunch are friendly and making sure that they shake their hand and wave goodbye as they walk away. Throughout the entire process, doing this as – They're customers. It's a customer experience and their experience matters. I think it's useful to have someone who's – Again, at a larger company, I think it's useful to have someone, maybe not full-time, but at least when designing the process, totally focused on this. So someone thinking about, "Okay. How do I make this process accurate from their perspective?" Then also someone with equal weight thinking about how do we make this process as positive as possible for the candidate?

[SPONSOR MESSAGE]

[00:41:49] JM: Data holds an incredible amount of value, but extracting value from data is difficult, especially for non-technical non-analyst users. As software builders, you have a unique opportunity to unlock the value of data to users through your product or service. Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love.

Give users intuitive access to data in the ideal place for them to take action within your application. To check out Jaspersoft, go to softwareengineering daily.com/jaspersoft and find out how easy it is to embed reporting and analytics into your application.

Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it is not just your company that wants analytics. It's also your customers that want analytics.

Jaspersoft is made by TIBCO, the software company with two decades of experience in analytics and event processing. In a recent episode of Software Engineering Daily, we discussed the past, present and future of TIBCO as well as the development of Jaspersoft. In the meantime, check out Jaspersoft for yourself at softwareengineeringdaily.com/jaspersoft.

Thanks to Jaspersoft from being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:43:22] JM: Does this stuff, all these stuff that we're talking about, does it apply to both startups and to big companies, and not just big tech companies, but maybe companies that are turning into tech companies, like banks for example. Banks all need software engineers. Is this a uniform advice to constructing a hiring process?

[00:43:44] AB: Yeah, I think it actually probably matters more for banks, because the funny thing about companies that are turning – Banks and even just big – CVS, like all of these companies are now trying to build real engineering teams. They're larger, they're older. They have a culture in place and that culture is often a little bit less, friendly to candidates than engineering culture is. I think that can sometimes make it hard for them to hire, because they have a process in place, they bring candidates in and they use that same process for engineers. Those engineers, comparing that big company to the tech company down the street that was much nicer will often choose the tech company. So I think companies that are trying to hire engineers actually need to focus sort of even more on these things.

Startups gets a little bit of a pass, and that startups are always a little bit of a shit show, and that applies to hiring as well. Candidates do know that. So I think startups should try to hold themselves the same standard, but they are – you know, part of what they like is the like ad hoc, seat of your pants, build the tracks in front of the train aspect of it. So they are given a little bit more of a pass on these issues.

[00:44:53] JM: You and I last spoke about a year ago. How has the business at Triplebyte advanced over the last year?

[00:45:01] AB: Yeah, a big thing is just we've grown a lot. So everything is larger scale. My big focus has shifted from how do you design the optimal assessment process to really want to scaling it. That involves how do we keep this process standard across a growing team of larger and larger numbers of individual interviewers doing the work.

[00:45:20] JM: Has the interviewing process for Triplebyte changed over that period of time?

[00:45:25] AB: For candidates who we're hiring internally?

[00:45:28] JM: No, for candidates who are being hired through the platform.

[00:45:31] AB: It'd say most of our big – It has changed. It's changed a bit more incrementally than a year before really for the same reason. Our priority has shifted from – Our goal is to maintain the same quality and then deal with growth and those things are always a bit of a tradeoff. So it's been a bit more – It'd been kind of, "Okay. How do we scale? How do we – Quality assurance? How do we keep quality high?" It's less on sort of how do we test out totally new classes of evaluation. I think that's – Probably that's just scaling. That's partly just the fact that we've – I guess, product market fit. We've reached the point where the product is working well enough that there's less huge upside. We could probably get an additional 5% upside and predictive power by running some more large tests. Whereas a year ago, we are getting 5, 10% boosts in individual tests.

[00:46:16] JM: How has the strategy for the company changed as you've decided that we have a pretty good product market fit. Do you focus more on marketing or internal tooling? How does that shift? How does that shift affected your strategy?

[00:46:31] AB: To be totally honest right now I'm focused on not being broken due to the volume of companies and candidates on the platform.

[00:46:36] JM: That's great. That's a good place to be.

[00:46:38] AB: Lots of pretty big headache, let's say.

[00:46:40] JM: Yeah. It's good that you did 1,000 interviews yourself, or however many interviews you've done at this point. That's, of course, not exactly scalable, and I know you've hired some people and you've trained some people. What's been the training process for all those interviewers that you now have to hire to help you out through the Triplebyte process?

[00:46:59] AB: Yeah. This has been a big thing I've been focused on. I think there are a few parts to it. The first is that they just have to be very strong in the content. We are fundamentally assessing skills in these areas and people doing the assessing need themselves be top percentile of engineers in their knowledge in these areas, because when doing an interview, candidates will ramble, right? It's really interesting how looking at sort of – When put on the spotlight, how people go about revealing their knowledge. To tell if that candidate is rambling is them going down on adjacent but correct path, versus them stone walling and making things up. That really does require you being pretty darn strong in all the areas.

So first is we look for people who are very strong. I mentioned earlier that soft skills matter a lot. So we end up rejecting actually about as many people who apply for soft skills reasons as we do for technical reasons. It's really important that they can guide the interview process and make it positive. The phrase we use internally is bed side manner. So the bed side manner of the interviewers matters hugely. That's kind of the selection process.

But the biggest thing I focus on is this idea of consistency. Very interestingly, getting a group of people who are really smart, know their stuff, strong soft skills, great communication, great bed side manner is actually not enough to have a good interview team. If you put these people in front the candidates and have them ask the same questions, you will find that they will differ pretty significantly as to how they interpret the answer. There really is significant disagreement about sort of what even constitutes a great answer to the same interview question. So most of our training goes into sort of standardizing that evaluation process.

[00:48:33] JM: One of the reasons I was optimistic about Triplebyte from the earlier days was because I could tell that you were just going to do a bunch of interviews and get your hands dirty and really understand the process, and that's part of what I did when I started Software Engineering Daily. I just figured that if I do tons and tons of interviews, then I can get good at intervening people and make an entertaining product.

One thing I didn't anticipate was the fact that doing all these interviews, maybe about the 200 to 300 interview point, people started telling me that when I was I just in an everyday conversation with them, they felt like they were being interviewed. Did you ever have that issue when you were really going through all these interviews where people started to say, "Hey, Ammon, can you just shut up for a second? You're acting like you're interviewing me."

[00:49:23] AB: No one ever told me that. I notice it in my head. So I would notice when I would fall into a line of reasoning or a line of discussion that I was like – Suddenly, it's this familiar path. I don't think anyone ever – I'm not sure. Maybe my friends just didn't want to break it to me. No one ever gave me the feedback that it sounded like I was falling into interview. I actually get that one thing, which is that when conducting an interview, our goal, the goal of the team is to not seem like we're in an interview mode.

So we want to seem to the candidate that this is the first time we've asked these questions, because that's the best – We want to seem like I'm your friend and I'm asking you these questions, as supposed to I'm an evaluator and I ask this question four times a day.

[00:50:02] JM: Is there anything else that engineers that are looking for a job right now or people that are constructing their hiring pipeline should take away from our conversation?

[00:50:11] AB: Yeah. I think I mentioned in the Y Combinator talk that I gave as well. But a thing that we do to help train the interviewers that I just really love is we have them interview each other. We set up pairs of people and we have them conduct technical interviews where they play the role of the candidate or the interviewer and ask questions. This is an excellent way to highlight how divergent people are what they think great answers are. It’s to get two smart people in the room and have them just conduct sort of gloves off interview of each other.

There’s a particular trick to it that works quite well, which is to – Again, you’re in a room with all of your coworkers and you’re going to play the candidate. They’re going to play the interviewer. They’re going to ask you questions. What you do is you tell them upfront that you’re going to be role playing a candidate that you’re going to intentionally give bad answers to some of the questions. Then they ask you questions, you intentionally give bad answers to them. But others you try to answer really as best as you can.

What that does, when the interview is done, you kind of debrief and your coworker sort of tells you about the mistakes they thought you made. What that does is incentive them to be brutally honest and to like look for flaws in your interview. Because the problems – If you just go and interview your coworker, it’s awkward to say, tell your coworker, “Oh! I think that when you designed that system, it was wrong and stupid.” This sort of decorum. You would never say that.

But by setting up where they know you’re making mistakes, that point, they look stupid. If you say something incorrect and they don’t point it out as a mistake, that’s on them and then they’ll look a bit bad. So what this does is create a situation where they’re incentivized to actually point out all the flaws they saw on your interview.

This is a way for you to basically get an honest assessment of what one of your smart coworker thinks of a lot of your ideas. The key thing there is they’re going to assess both the parts where you were trying to mistakes, but they’re also going to assess the parts where you were trying to give your best answer. This is really eye-opening by the way. [inaudible 00:52:05] bunch and it’s humbling and eye-opening to get smart people to sort of like brutally critique your answers to questions.

[00:52:11] JM: Okay. Well, Ammon Bartram, thank you for coming back on Software Engineering Daily. It’s always a pleasure to talk.

[00:52:16] AB: Thank you.

[END OF INTERVIEW]

[00:52:20] JM: Your audience is most likely global. Your customers are everywhere. They’re in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15 day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[END]