EPISODE 365

[INTRODUCTION]

[0:00:00.1] JM: Culture fit is a term that is used to describe engineers that have the right personality for a given company. In the hiring process, lack of culture fit is a term that's often used to turn away engineers who are good enough at coding but just don't seem right for the company. As today's guest, Ammon Bartram says, "Lack of culture fit usually means lack of enthusiasm for what a company does."

Ammon is the cofounder of Triplebyte, a company that is debugging the interviewing process. Triplebyte has interviewed thousands of engineers and is discovering which aspects of the current hiring process makes sense and which are based on superstition of just raw tradition. We had a great conversation about what culture really means and how to hire effectively. I really enjoyed this episode with Ammon and I hope you do to.

If you want to find our older episodes, the episodes of Software Engineering Daily that have been really popular, you can find our greatest hits feed in iTunes or wherever you get your podcast. You can also find lots of other topic feeds, like business, and blockchain, and JavaScript, and machine learning. These are specific categorized feeds in case you find our main feed to be too overwhelming with too many different episodes.

You can break up your Software Engineering Daily world into different partitions. You can find more details about that in the show notes today and we'd love to hear your feedback and anything else that we can do to improve Software Engineering Daily. You can email me, jeff@softwareengineeringdaily.com. I hope you like this episode.

[SPONSOR MESSAGE]

[0:01:57.1] JM: Angular, React, View, Knockout, the forecast calls for a flurry of frameworks making it hard to decide which to use, or maybe you already have a preferred JavaScript framework, but you want to try out a new one. Wijmo and GrapeCity bring you the How to Choose the Best JavaScript Framework For Your Team e-book.

In this free e-book, you'll learn about JavaScript frameworks. You'll learn about software design patterns and you'll learn about the advantage of using frameworks with UI libraries, like Wijmo. You'll also learn about the basic history and the purposes of JavaScript's top frameworks. You'll also learn how to integrate a WIjmo UI control in pure JavaScript and in some of the top JavaScript frameworks that we've already were talked about, like Angular, React, View and Knockout.

Wijmo's spec method allows you to determine which framework is best suited to your project based on your priorities. Whatever those priorities and your selections are, you can learn how to migrate to a new framework in this e-book. Best of all, this e-book is free. You can download your copy today to help choose a framework for your work at softwareengineeringdaily.com/grapecity.

Thanks to GrapeCity for being a new sponsor Software Engineering Daily, and you can check out that e-book at softwareengineeringdaily.com/grapecity.

[INTERVIEW]

[0:03:38.8] JM: Ammon Bartram is the cofounder of Triplebyte. He's joining the show for the third time. Ammon, welcome to Software Engineering Daily.

[0:03:47.2] AB: Hello, thank you for having me.

[0:03:48.4] JM: Today we're going to talk about culture fit, which is a topic that has been discussed a lot in software engineering circles. I want to start by explaining a little bit about who you are and what you do around Triplebyte, people who don't know you. They might not know this stuff. Triplebyte is a platform for helping companies do better hiring. Talk a little bit about what Triplebyte does just for people who have not head of the product before and explain how the hiring process is inefficient and what Triplebyte

[0:04:26.2] AB: Sure. Basically, we're a company and we help engineers find jobs at startups, at other companies. What we do is engineers apply to us, we interview them, and then we introduce them to about 200 companies including Apple, Facebook, Dropbox, Stripe, all the way down through Cruise, working on self-driving cars and then to sort of tiny startups that are just a few months old out of Y Combinator.

What we can do is because we do interviewing for all of those companies, we're able to do a much better job matching candidates with companies. What we found is that the applicants that different companies look actually vary pretty significantly, but they don't have a great way of sort of telling applicants this, and so what they have to do is just reject everyone who – They interview who doesn't have these specific skills that they're looking for.

Often, they're not even aware exactly to the extent to which they differ from other companies. Company A might think that we really care about complexity analysis, and anyone who applies who can't talk in-depth about advertised type of complexity, it isn't a good engineer." They don't realize that there are dozens of other companies out there that will very happily hire great people who maybe aren't super solid on complexity analysis.

What we can do is basically match people based on all those different strengths, and so what we see then is that after we do that matching, the offer rate at the companies goes up by about 2-X. Candidates who go through us get offers after about twice as many interviews as general applicants to those same companies.

[0:06:06.7] JM: For the engineers, this is really useful. What you do is you centralize the hiring process. You're a company that focuses on the hiring process, and the advantage of building a company entirely around hiring, whereas in the past, every individual company was doing its hiring alone. You're a cloud services company and you also have to figure out hiring, that has inefficiencies for both the company and for the engineers that are going through the hiring process. For the engineers that are going through the hiring process, if you're applying for jobs, you typically have some set of requirements that you want out of your job.

For most people, it's not like, "I want to work specifically at Stripe," or "I want to work specifically at Amazon. I've got some set of things that I want to be matched with a company. It's like an online dating site almost. You don't go after one – Some people go after one specific person, but you would rather go into this pool and be matched with somebody based on characteristics.

That's great for the engineers. For the companies, what you're saying is because you focus on hiring so much, you have a lot of insight into some of the proclivities towards error that certain companies have. For example, unconsciousness biases. Figuring out what are the unconscious biases that occur widely in the hiring process is something that you're well-suited to because you see so many different points. Overtime, you can grind out these biases that are ever present. What are some of those unconscious biases that affect the hiring process?

[0:07:52.6] AB: I think the top of that list is just credentials. We view all our assessment background blind, so we don't know what the candidate's background is when we evaluate their skills. What we see is that imbalance sort of credentials do [inaudible 0:08:09.1] what you might expect. People who graduate from MIT as a group are better programmers than people who graduated from an unknown state school.

But vastly, more people who graduated from an unknown state school than graduated from MIT. If you just, say, "Okay, we're going to only talk to people who graduated from MIT," you end up missing, really, the majority of excellent programmers. Yet we see that that bias ends up sort of infecting the process. For a given level of performance, companies are much more likely to hire someone if they have sort of a name brand company or a name brand school on their resume.

I think one of the biggest ones that we specialized in is just finding and helping people who are great programmers who may be have – Because of where they work in the country, maybe they've gone and worked at the fence contractor in visual basics. Believe it or not, there are excellent programmers who work for the fence contracts in visual basic. We can find those people and then sort of give them a leg up over the screening process to an interview at Facebook or Stripe or companies that would not have spoken to them if they didn't have sort our validation.

[0:09:13.1] JM: This blindness that you're talking about, the background blindness, are you saying you don't look at resumes. You don't look at background at all. Is there any screening process for people who come through the door?

[0:09:23.8] AB: Yes, the first step in our process is sort of an automated programming quiz. No. When we do the interview with the candidate, we do not know – We know their name because it pops up in the screen, but we don't know if they've worked for 20 years or there months. We don't know what school they went through. We know none of that.

[0:09:39.3] JM: That wasn't the process at the beginning, was it? Did you realized overtime that, "Oh, actually, looking at a resume – Actually, it causes bias that confounds things," and you actually just want to test people for can they program?

[0:09:54.8] AB: I know. That has been our process from the beginning. If you look at the research out there on the efficacy of resume screens, it's pretty invisible. We had sort of read that going in and we decided to try to come up with a process to directly measure skill. We decided to come up to force ourselves to get better at that to totally cut off access to the candidates backgrounds and say, "If going in we know nothing about a candidate, what's the most accurate way to determine if they are still a programmer?"

[0:10:23.5] JM: You write about these high-signal and low-signal questions. Once somebody is in the door for an interview and you're interviewing them, there are questions that can provide a high-signal as to whether this person would be a good engineering hire. There are also low-signal questions where you're going to ask them a question and whether or not they answer it correctly, it's going to have no impact on whether they'll be a good engineering hire. Can you give an example of each of these types of questions?

[0:10:56.7] AB: Sure. I guess I'll talk in the technical realm for now. We can be talking more about culture fit in a minute. First, it depends a lot on what the company is looking for. Companies do look for very different skills. For example, a company may have thesis that they want to hire people who are very strong in academics. If that's their thesis, asking that kind of question I guess probably makes sense. Whether candidates need to be very academic to be a productive engineer is debatable, but a company may have a reasonable cultural desire to hire academic engineers.

I'm going to sort of try to describe it in a way that doesn't involve the actual content of the question. I think a great sort of rule of thumb is questions that require sort of a single leap of insight are very bad questions, and because that just ends up being extremely noisy. As example of a question like that – This is the real question that you get out there sometimes, the question is; imagine you're walking up a flight of stairs and at every step you – I think a small step up once – Up one step of the stairs, or a large step up two steps, and the question is imagine you're on the nth step of the flight of stairs, how many unique paths could you have taken from the bottom to arrive on the Nth step?

If you think about it, it ends up that the answer is the Fibonacci sequence, and you can prove that to yourself and it's kind of cool observation, but if you ask [inaudible 0:12:22.6] that question, some people are going to know the answer and they might just say – Other people might know the answer and they'll sort of pretend, they'll reason it through and make it look like they're really smart and figured it out. Others might get flustered and feel frustrated and not come up with the answer. That's the interview, there's very little you can do to sort of help. You tell someone, "Hey, it's Fibonacci sequence." Suddenly you've given away and there's no signal there.

A different question that someone might ask is – I don't know. Let's keep it academic here. Let's say matrix multiplication, "Please write a function that measures this and produces the answer." That's a somewhat involved process. Knowing the answers shows slightly familiarity with linear algebra, but it still is fundamentally a series of steps that can be thought through. If a candidate gets stuck on one step, you can get them a little bit of clue to help them move forward and they can still sort of get past that point and demonstrate competence later on.

I think the second question is far better, because you can sort of measure the candidate is moving through the process and give some of the noise people get stuck by giving some clues and not totally invalidate the question. The basic rule of thumb I'd like to think about is questions that can be given away. A questions where you have to – As the interviewer, you have to worry, did someone give away this question to the candidate ahead of time? If that's possible, if there's some single small thing of information, few stenches that could have been told that's going to help them answer the question, I think it's probably a bad question. A better question is a question where the answer to the question is 45 minutes of programming, and so there's very little that could have been done sort of practicing the programming ahead of time to give them an advantage.

[0:14:00.7] JM: I think at this point, most of the listeners understand the process of Triplebyte. A bunch of engineers come through the door, you interview them, you figure out to math them with companies. There's a lot of complexity to this process and we've attacked discussing that in previous episodes where you've been on the show so people can listen back to those.

Let's zoom in on culture fit, because this is – For anybody who's done a number of engineering interviews, or a lot of engineering interviews, you've probably been rejected once or twice for something, whether you know it or not, that was internally described with that company that rejected you as cultural fit. What is that term mean or what do companies perceive it to mean? When I say to somebody else, "I didn't hire this person because of cultural fit." What is the unspoken language that's going on there? What is that term actually mean?

[0:14:56.5] AB: That is an excellent question. A thing that we've kind of butted our head with a bunch at Triplebyte, because we do pretty extensive technical screening, and so when candidates who go through our process are rejected, do fail interviews, very often that it's for culture fit reasons. Sort of responding to that, we've went back and interviewed all the companies extensively about what they look for. What we observed is that it's an extremely overloaded term. It doesn't mean any one thing. It means sort of the whole grab bag of nontechnical reasons that a company might want to reject a candidate. Anything from we thought they were arrogant, that they're an asshole, to bias stuff, "We want to hire white male programmers," and this person wasn't a white male." Obviously, no one actually says that. When that kind of stuff comes into the picture, it's under the guise of culture fit.

Two sort of very specific to – This company has – We're a bank and we need programmers who care a lot about security and are very security minded, and this person seem to look like a great programmer, but they weren't – It seems like they're going to move fast and break things rather than think deeply about security.

The interesting thing is that companies themselves aren't very crisp about the differences. It's a catchall phrase that's often pretty poorly defined. I think it's quite useful to dig into it and break the part into its different parts and talk about each of them separately.

[SPONSOR MESSAGE]

[0:16:32.1] JM: Your application sits on layers of dynamic infrastructure and supporting services. Datadog brings you visibility into every part of your infrastructure, plus, APM for monitoring your application's performance. Dashboarding, collaboration tools, and alerts let you develop your own workflow for observability and incident response. Datadog integrates seamlessly with all of your apps and systems; from Slack, to Amazon web services, so you can get visibility in minutes.

Go to softwareengineeringdaily.com/datadog to get started with Datadog and get a free t-shirt. With observability, distributed tracing, and customizable visualizations, Datadog is loved and trusted by thousands of enterprises including Salesforce, PagerDuty, and Zendesk. If you haven't tried Datadog at your company or on your side project, go to softwareengineeringdaily.com/datadog to support Software Engineering Daily and get a free t-shirt.

Our deepest thanks to Datadog for being a new sponsor of Software Engineering Daily, it is only with the help of sponsors like you that this show is successful. Thanks again.

[INTERVIEW CONTINUED]

[0:17:57.9] JM: Right. Triplebyte has a post on the blog, the Triplebyte blog about how to succeed at conventional engineering interviews. In this post you basically say, "Triplebyte, we're trying to change the interviewing process, but in most job interviews you're going to have to deal with this process if you're an engineer. There are some easy tips that you can follow that will make this process a lot easier. Even though it's stupid and stupid to have to adhere to it, that's the way that things work."

The number one piece of advice you have is not around learning any specific algorithms or going on career cup and training yourself, it's actually be enthusiastic. That's the number one piece of advice. You imply that the companies that reject a person for cultural fit, 9 times out of 10, that actually means that person does not have enthusiasm for what the company does. If you want to classify that, it's cultural fit. That almost sounds like a characteristics of humanity as a culture. It's not even like the specific company. It's just like why would you want to hire somebody who's not enthusiastic? You just simply would not want to do that.

[0:19:19.6] AB: Yeah. I think from the point of view of an engineer applying to companies and trying to figure out how you can pass more algebraic interviews, I think the number one piece of advice is definitely be enthusiastic. More precisely, research the company ahead of time and come up with a list of questions, things you are excited about that you can ask about to show your excitement. That screen, it seems a little bit arbitrary, but it actually makes a fair amount of sense. There's a bunch of flaws with it. As a company, you totally do want to hire people who are excited about what you do. I think that's actually sort of a positive aspect of culture fit is that it's recognizing that there are things other than sort of just flashy technical brilliance that matter.

For example, imagine as a hiring manager, you have two candidates. Candidate A is technically brilliant. Any logic problem you give them, they will crush in a second and then they can write the hardest algorithms you can come up with very well, but they're a little lazy and unfriendly and if they join your company, they'll do their job but they'll probably work 9 to 5 and not answer their email off hours and not care about the company. Candidate two is a solid programmer. They're pretty good. There are some math they're not up on that the other candidate is, but they care a lot of what your company does and they find it deeply exciting and they have just this great work ethic. Maybe if they don't know the answer to a problem, they're going to bang their head against it and read books and read Wikipedia and not sleep until they have the answer. Of those two candidates, the second one is almost certainly the better employee at every company. I think anyone who has ever been a hiring manager can identify with that.

Part of it is acknowledging that there are skills outside of technical brilliance that matter. The difficulty then is coming up with how do you actually measure those skills in a way you're not being sort of led astray by confounding factors.

[0:21:21.4] JM: The way that this connects to the ambiguity of the term cultural fit, is cultural fit actually just code for things that are universally true? You universally want somebody who is enthusiastic, but maybe you don't know how to say that after a technical interview. If you have a technical interview and somebody – A candidate has come into your office and they've gone through a technical interview and everybody is like, "Well, they got the question right, but something bugs me. I don't know if he's a good cultural fit or she's a good cultural fit." Is this just code for things that are universally true that we don't know how to articulate?

[0:22:03.5] AB: Yeah. I think it's useful to break culture fit down to three parts. Part one I would say is soft skills. It's not technical skills that basically everyone cares about. Communication ability, ability to work on a team, general positive attitude, and excited about what the company does. Those are I think just what you said. Those are skills that everyone — Not technical skills that basically everyone wants and often get bottled under culture fit.

Part two that I think it's useful to separate, part two is specific, sort of intentional choices the company makes about what traits and what types of employees they want to have. This is, for example, a company might decide that it wants to be a very data-driven company. It wants most of its employees to believe in data-driven decision making, and so it's going to set to this sort of filter for who it hires that it wants to sort of hire people who are outliers on believing that the right way to solve a problem is to gather data, look at the data and make a decision as supposed to approaching it from sort of a product angle or sort of a feel angle.

That's an example. If you look at out there at companies, we have Facebook with their famous — I guess they've dropped it now, but their famous early motto of move fast and break things. You can view that as sort of a specific intentional choice that they want to bias towards sort of productive hackers as supposed to other approaches to engineering. Stripe is an example. The little bit less well-known, but Stripe has a particular focus on friendliness to a degree much further than basically any other company. Stripe wants to be an extremely friendly compassionate place and they will reject people who everyone else would hire just because they're only a 6 out of 10 from this rather than an 8 out of 10.

[0:23:55.7] JM: I went to Stripe and I did three interviews there in a day one time, or I guess I did two interviews. I did another one remotely, but I got a good picture of the company. I interfaced with five or six people there in a day. Friendliness is certainly a characteristic that I would use to describe all of them. What I'll say about that, that was an incredibly warm and there's something I can't put my finger on that made me really enjoy the visit to their office. I guess you could encapsulate it as friendliness. It sounds like an underrated/underappreciated element of that company.

[0:24:38.4] AB: Yeah, they're great people.

[0:24:40.7] JM: Yeah, interesting. Certainly, one of the really rising starts, people compare it to Amazon, but that's so fascinating. Okay, but that's – Now you're talking about differentiating characteristics of these big companies. I did a show recently with a guy from Google who had been there from the early days. He was talking about how – I think move fast and break things was not something that was a characteristic of Google. Google is like figure out – The example he gave was if there's a problem with the Linux kernel that's causing your Hadoop job to not run, you don't write a script that makes the Hadoop job run properly. You fix the kernel bug.

At Facebook, it would be like, "Figure out some freaking system that runs correctly. Make sure you can do it in three hours. Break out the duct tape." It is a total cultural difference. There is certain things that are cultural fit that are specific to companies that are not BS.

[0:25:53.0] AB: Yes. The third category of things I think that fall into culture fit is the category that just is BS for lack of a better phrase, and this is the pattern matching. Ignoring, A; the first two categories, thought skills and then specific intentional traits a company wants to have. Just does this candidate feel like a good engineer I've met in the past, or that they feel like someone who they want to be friends with? This is where you see — so Laszlo Bock, the XPP of people at Google has written about sort of research showing that this is actually looking at nontechnical. Interviews for non-engineers, but that the decisions in most nontechnical engineers are made in the first 30 seconds. That's what you're seeing, just, "Does this person gut call feel like —"

[0:26:37.7] JM: He's got data around that?

[0:26:39.1] AB: Does he cite data for that? Yeah, he cites a paper.

[0:26:41.9] JM: Wow!

[0:26:43.0] AB: Yeah. To be clear, it's not looking at programming interviews. It's looking at different interviews. That's sort of the bad pattern matching aspect and that's where sort of the worst reinforcing biases against people who don't look like program we're using in the past creep in.

[0:26:58.3] JM: Do you believe that? Do you think that happens in engineering interviews where most of them are decided in the first 30 seconds?

[0:27:05.0] AB: Yes, in the cultural fit component. I think no in technical components. We're lucky and that most engineering interviewers are actual skill assessments. In skill assessments, there's a really [inaudible 0:27:19.7] the skill being measured for all the flaws that can come into play. You may think someone look stupid but then if they brilliantly solve your problem, that will change your mind. The culture fit section doesn't have hat. There's not really a sort of a tangible output to be measured. In that case, yes, I believe it.

I think if you watch yourself, if you really sort of introspect to look at your own decision making process, when you're trying to do an interview for a sales role or something, I think you'll notice that — Not all the time, but a pretty high percentage of the time, the thing you're thinking 30 seconds in is what you're going to think at the end.

[0:27:58.2] JM: It's funny, I think there have been so many times out of all the engineers I've interacted with where I just think – My first impression of this person, "This person is so dopy and he's like so stupid and he's always eating that stupid sandwich. This person is obviously an idiot," and then I interact with them on the level of code or I like I talk to them about some systems problem and they just like say, "Oh, yeah. It's this. This is how you solve it." I'm like, "Oh, wow! It's actually brilliant." My opinion will totally change on a dime based off of some technical confirmation. I would actually say that most of the engineers that I encounter, the longer the time I spend with them, the more they have a chance to expose that side of themselves where they're really smart about something and my opinion of them improves overtime. I don't know if there's – Do you have any correlation to that in your own experience?

[0:28:55.3] AB: Yeah, I think there is actually. One observation — Technical interviews are — It's way easier to be the interviewer than to be the interviewee, the candidate. You know the question ahead of time. You've been able to think about it. It's pretty easy to trick yourself into thinking that you would have done better on the question than you actually probably would. The reality is that when the jobs open, people are going to apply and they're going to be bad. It has to happen. You do need to reject people, but it's good to stay humble and realize that there is a super high false negative rate. It is relatively easy to make a person look stupid by asking them a question that they haven't thought about.

One thing that — Experiment we did at Triplebyte was to do take-home projects. Specifically, it was opt-ins. People could do both a regular interview and then also a longer take-home project. Looking at the difference in quality between the work people did during an hour or two when they're on the spot versus when they had their time to think about it, it was really quite enlightening. Just people who we had written off as bad would then turn around and do really impressive professional-level work.

[0:30:07.8] JM: Yeah, I think some people who are on the interviewing side of things, they're interviewing a candidate, there's so much power in that realm. Just like the craving to make somebody slip on the banana peel and just like, "Oops! Oh, well. You couldn't figure out it was a Fibonacci sequence in the stair stepping, and I'm sorry. I know you flew here from Montana, but we're going to have to send you home. I hope you had a good time visiting the Bay Area and come back when you're grown up." People just want to reject.

[0:30:47.0] AB: I think there's sort of one — This goes down to one statement that I think is helpful asking an interviewee is that you're looking for reasons to say yes rather than reasons to say no. You want to try to be on the side of the person and try to find a reason to say they're good rather than a reason to say they're bad.

[0:31:02.9] JM: Debugging this process seems like very much a psychological process. I was looking into the process and you said – There's one quote, you track what the interviewer is thinking every five minutes throughout the interview. What does that mean? How do you do that?

[0:31:22.0] AB: We're doing our interviews over at Google Hangouts, and so we have a customer extension that integrates with Hangout page. We have grading evaluation tools that overlay. Every five minutes, there's just a popup that asks us for our current sentiment. It's actually not used in the search making process. We found that sort of much more concrete measures of progress and a problem and things like that, like sort of what path it took through a problem are much better predictors of how they'll do on other interviews at companies, but kind of to measure — Actually, because the motivation for that interest was the Laszlo Bock data. We're curious to see if it was true that, basically, how our impression was changing. What we found is what I'd said earlier, that the technical — When asking technical questions, the impression actually does change quite a lot overtime. That when asking nontechnical questions, less so.

[0:32:13.3] JM: When you put in a check like that, it seems important because the interviewer realizes they are being examined on a certain axis, and that might cause them to consciously counteract that bias. They might realize, "Oh, I'm being graded, or my sentiment is being measured every five minutes throughout the interview. Maybe I should be more open to changing my sentiment." Maybe there's some Heisenberg uncertainty, like a beneficial Heisenberg uncertainty principle thing that's happening there.

[0:32:45.7] AB: Yeah. I think sort of a similar idea. The best approach to combating bias in interviews appears to be breaking down the skills you're looking for and being clear of what they are giving people a clear sort of — Rather than saying, "In your gut, this person feel good." Saying, "Your goal is to judge algorithmic thinking. Here's the question. Thinking on the scale from one to 10." That appears to produce a much less — By doing that on multiple axes and them summing the result appears to produce a much less bias measure of skill.

This actually gets back to what — I think the solution to the sort of uncertainty and bias, particularly in culture fit screening, is for companies to be very precise about exactly what they're measuring what they mean by a culture fit. For companies that say, "Okay, we care a lot about soft skills." I actually think — It's my personal opinion that soft skills are undervalued relative to the technical brilliance in interviews. That's partly because they're hard to measure crisply, but soft skills matter immensely. Communication ability, team work —

[0:33:48.0] JM: How underrated do you think they are?

[0:33:50.6] AB: I'm not sure what the scale is to answer that question, but I think it's plausible — Companies — Bad hires are terrible. They're very expensive to a company. Companies end up heavily biased in their process toward potentially rejecting people in order to try to avoid hiring bad people. They usually do that mostly on technical axes. I think that's probably right. I think you could probably cut back a good 5, 10 percentage points on technical axes and then sort of make the difference, so you're still passing the same number of people on the soft skill axes and you'll probably get a significantly lower bad hire rate.

[0:34:32.6] JM: If you were Stripe, you could probably just check people for friendliness and then say, "Okay, you've met the friendly quota," and then you bring them in the door and Stripe has so many different roles, whether it's engineering or like technical customer success. So many different roles, you could just do the job matching after they pass the test for friendliness, arguably.

[0:35:00.0] AB: Yeah. I think Stripe is an example of the company that does this very well. I agree a lot with the approach that they take. Also, being precise about — The second category, right? The specific traits that you want to look for at your company. In Stipe's case, that's friendliness. Just being really upfront about that.

A really interesting question, I'm curious to hear your opinion, is to what extent that's a good idea? We have all these examples of successful companies that have sort of thesis about who they want to hire. Clearly, having that thesis is not an obstacle to success. A really interesting question is whether it's actually helpful. I'm not sure what I think. For all the companies we listed, there are plenty more that are very successful that as far as I can tell don't have a specific thesis.

The starting point being, it's really hard to find good people. I want to hire all the good people that I can find. Anyone who's smart and a good programmer and has reasonable soft skills sounds like a great employee. That actually is the position of the majority of companies we've spoken to. Most companies — Yeah.

[0:36:02.2] JM: Most companies don't know what their own DNA is. They don't know what is a good person at this company.

[0:36:08.6] AB: That's the negative phrasing of it. The negative phrasing is that they don't know their own DNA. The positive phrasing is what they care about is technical ability and soft skills and it's really hard to find good people and so they've decided to not further narrow the field by saying, "We only want people who are extreme outliers and friendly. So we're going to look for people who are extreme outliers on academic ability of whatever it is. I'm actually up in the air about what I think the best strategy is.

[SPONSOR MESSAGE]

[0:36:42.1] JM: For more than 30 years, DNS has been one of the fundamental protocols of the internet. Yet, despite its accepted importance, it has never quite gotten the due that it deserves. Today's dynamic applications, hybrid clouds and volatile internet, demand that you rethink the strategic value and importance of your DNS choices.

Oracle Dyn provides DNS that is as dynamic and intelligent as your applications. Dyn DNS gets your users to the right cloud service, the right CDN, or the right datacenter using intelligent response to steer traffic based on business policies as well as real time internet conditions, like the security and the performance of the network path.

Dyn maps all internet pathways every 24 seconds via more than 500 million traceroutes. This is the equivalent of seven light years of distance, or 1.7 billion times around the circumference of the earth. With over 10 years of experience supporting the likes of Netflix, Twitter, Zappos, Etsy, and Salesforce, Dyn can scale to meet the demand of the largest web applications.

Get started with a free 30-day trial for your application by going to dyn.com/sedaily. After the free trial, Dyn's developer plans start at just $7 a month for world-class DNS. Rethink DNS, go to dyn.com/sedaily to learn more and get your free trial of Dyn DNS.

[INTERVIEW CONTINUED]

[0:38:42.0] JM: My perspective is, basically, if your company has a good product, you should just – People who come through the door that either they have a good resume or they show a lot of enthusiasm, you let them through the door, and this doesn't work for Triplebyte probably, but you let them design their own hiring process and you just say to them, “Hey, tell me how should I rate you right now. Give me –” Some really good hiring advice that I read, this is from Quora, from a guy named Oren Hoffman, who, he writes about startups and hiring and building businesses and he writes a lot of stuff that’s really interesting. One of the things he always says if you hire for strength, not for a lack of weakness. I think a lot of what the engineering hiring processes do is they screen for a weaknesses. They’re looking to detect your weaknesses.

Some people have really weird strengths, and if they were to be able to expose those strengths in the hiring process, you’d be like, “Oh my God! You would be perfect for X-random part of our company that we –” Companies are getting so weird. There are so many niche, companies doing very niched things because of the way that the internet allows for you to scale up even the most niche products because you can find all of your customers for those super niche products.

If you’ve got a niche role in a niche company, then why would you have a general – Or if you’ve got a bunch of niche roles in a niche company, why would you have a hiring process that’s like peanut butter, super generalized? I realized it’s not a super scalable or recipe style way to do hiring, but I think the best way to do hiring is call somebody up, strike up a conversation with them and be like, “Okay, what are you good at? What are you going to come in and do?”

[0:40:36.2] AB: I know. I think that actually is — I think that matches pretty well with what we’re trying to do. That’s pretty much what we’re trying to do is solve that problem by getting better ourselves at identifying all those different niche strengths.

There’s sort of these — If you will, sort of conflicting imperatives here. The problem as a company, the problem with taking that approach and literally saying letting these candidates design their own processes. Then your process becomes totally non-standardized. Let’s say I have a role, I’m looking at three candidates. I ask them all how they want to be evaluated and can they — One is great at computer graphics and they want to produce a demo for me that’s doing some fun thing in the browser. Candidate two is as strong as something else. You get these three totally different things, and you have three different performances on three totally different axes. How do you decide, and how do you decide in a way that’s fair?

Basically, the version of just throw the ball into a bucket and talk about it and decide ends up being a gut call and that gut call ends up being the opportunity a lot of the problematic bias that seep back in. The issue is you want to give each candidate the option to show the thing they're stronger on. I totally agree that strength is what matter not lack of weakness. Problem is how do you do that in a repeatable standardized way that's taking these sort the problematic bias decision.

Also, a second problem with that is that communication ability. You give people a total free form interview, often salesmanship and communication ability on their part ends up dominating the actual technical ability. If you have people talking about work they did in the past, often being able to pitch that work effectively influences your perception more than how technically impressive the work actually was.

[0:42:26.4] JM: Isn't that the ultimate soft skill?

[0:42:29.1] AB: It is. If you're trying to assess soft skills, then sure. If you believe that both things matter, like the hard skill matters as well, then you want to — There's two things. I think the solution is to not have the interviews be totally free form but to have a bunch of different approaches, different tracks. You have a way that you're interviewing people who are very productive product focused and the way you interview people who are very product in mobile developers and the way you interview who are very systemy and then having enough volume on all those tracks that within each of those tracks the process is still standardized. I guess I'm [inaudible 0:43:08.9] to a pitch here, but that could have — Because we're doing hiring for hundreds of companies, we have enough volume that we're able to do that in a way that all but the biggest companies are not able to themselves.

[0:43:20.5] JM: You're talking about this idea of like where you, let's say, interview 5 to 10 mobile engineers. If you're a company that's hiring a mobile engineer, the way you're kind of presenting it is; okay, you interview 5 or 10 mobile engineers and then you get in a room and you discuss them and you pick the best one from that group of 5 or 10. My experience is that actually what happens it's more of a serial process and you encounter one person and you interview them and then you say, "Okay, are they better than the average, or are they going to be positive ROI enough?" Then you say, "Man! They're not," and then you move on to the next one. Is it more of a serial process or is it more like what you said; you get a pool of candidates and then you get in a room and you discuss and it's like the apprentice and you just choose the best one.

[0:44:08.2] AB: I think you're totally right. Actually, most other places in the economy hiring is, literally, there's one role interview and people and then pick the best at the end. Software companies tend to have more demand for engineers than they can fill. It is more like interview each candidate and if they're over bar, make the hire decision.

I think that ends up not being conceptually that different. I phrased it the other just because it was a little bit conceptually clearly. You are still fundamentally comparing people. It's fundamentally about, basically people I've seen in the past, "Is this person in the top N% over some bar that we think they're going to be great for our company?"

Even though the decision for each individual person is hire or no hire rather than comparing them to a group, I think it's still best to view the hiring process as sort of a sorting and comparison process on an engineering skill.

[0:45:04.5] JM: Let's get back to cultural fit. What happened with Uber? It seems like their strengths are also their weakness, their ability to ruthlessly grow fast and it seems like they selected for that in what we would classify as the cultural fit. Are you the relentless cutthroat Uber engineer who will get it done at all costs? That's ended up undercutting the company, because perhaps they got too much of a concentration of that. Is there some kind of a lesson here either for engineers or for companies who are trying to build their culture properly?

[0:45:47.4] AB: Yeah. I think this is actually an example — Sort of a really interesting data point in what we're discussing earlier. Whether intentionally — Whether it makes sense to intentionally screen for certain traits. This gets, again, back to the difference between sort of looking for soft skills. I think that definitely matters for everyone versus intentionally screening for a certain trait. A used rule of thumb there is that the difference is that when your intentionally screening, that means you're going to be rejecting people who are very strong. That's sort of saying, "Okay, there's a trait that we're looking for that we care about so much that we're going to reject legitimately great, strong, productive people for not having that trait."

I think Uber is a case of that backfiring. We see that Facebook selected for productive hackers and Stripe select for super friendly people. Uber selected for aggressive-driven, put everything in front of me at all costs to achieve my goal people. That both allowed them to scale, succeed in a really competitive market and overcome local tax regulations and do all those things, but it came back to bite them pretty hard years later.

I think one way to view this is that Uber messed up by focusing too much on culture fit, honestly. If Uber had focused more just on getting a diverse cross — A set of friendly people with good soft skills and not cared so much about selecting for one personality type, one view is that this problem may not have happened.

[0:47:29.4] JM: There's also a distinction between Uber and Facebook because — Facebook, at a certain point, they say, "All right. Will you stop hacking on everything. We got the business. It's working. Let's just chill out, move fast and have stable infrastructure." They changed their cultural messaging.

Uber, I'm not sure if they changed the cultural messaging, telling people, "All right. Listen, sorry guys. Let's drink a little bit less. Let's not be so aggressive." These are perhaps like cultural characteristics that are harder for people to change.

[0:48:07.3] AB: I think they did not, right? They still very explicitly — As of a number of months ago, they still very explicitly sort of tell their employees to be very aggressive. At least until very recently, was still the explicit part of their culture.

[0:48:21.2] JM: Do you think the jury is still out there? Maybe that culture could be useful again if they get into flying cars. There's lot of regulations around what can you do in the sky. Maybe that will be useful again. Personally, I haven't stopped using Uber. Maybe the jury is still out on whether or not this company is really so stricken with issues. Do you think it's — Have we given this case study enough time to bear out whether or not this was a mistake to create this kind of culture?

[0:49:02.6] AB: I believe — There's an article last week. I believe they grew substantially. I believe more than Lyft last quarter. In the short term, they still appear to be winning in their space. I think there's some danger that we're being a little bit too cosmetic and too broad in what the analysis here. These specific issues are sexual harassment and some other specific bad things that executives did.

There's certainly a scenario where Uber have the aggressive culture and just cared more about firing people who were sexual harassers and didn't make a few problematic decisions. In that scenario, maybe they would still have the aggressive culture but have avoided these problems.

[0:49:48.6] JM: Right. Also, actually, framing it that way, maybe Uber can ameliorate the culture just like Facebook ameliorated the culture to stability. Maybe if you fire the bad apples, you create some examples out of people, some scapegoats. You can retain the aggression of the culture, the unique aggression and find a way to hone that aggression without having the costs of sexual harassments.

[0:50:15.3] AB: Yeah. I don't know. If I had to bet, I would. I guess I would still bet in their success.

[0:50:19.6] JM: Yeah, okay. All right. There's a couple of other things I was finding when doing research preparing for this episode. Just thinking more about Triplebyte, this is the third interview I've done with you and I'm still kind of understanding what the company does or what the larger mission is. One thing I'm trying to understand is are you planning to scale to other verticals and do you think that these kinds of hiring biases whether we're talking about culture or we're talking about technical aspects, do these extend to other domains such as law, or accounting, things like that?

[0:51:01.2] AB: Yeah, 100%. I don't have much experience with law or accounting specifically, but I think that these sort of problems and biases are actually worst in most other fields. We're lucky and that programming interviews are still three-quarters skills assessments. There's a chance for someone to come in and sort of blow away on the skill assessment and overrule a lot of the other stuff.

I have some friends who are teachers. If you look at how teachers are hired, it's 80% culture fit, 80% communication ability, 80% credentials. 80% is some those things. Yeah, I think there is actually a huge need for just careful data-driven improvement of the hiring process across the entire company. We're pretty laser-focused on engineering right now because we are engineers and we understand it and it's a big enough market and a big enough problem that can keep us busy for a great while, but if in some hypothetical extreme success case, yeah, I would love to sort of expand this to everything.

[0:52:12.2] JM: So much of these data gathering seems like you're figuring out a way to provide structure to unstructured data, because I walk into an interview at a company — Like Google. I did an interview with Google three or four years ago and Google, as sophisticated as it gets when it comes to hiring, and their idea of a data-driven hiring process is like at the end of the hiring, the person like writes down some stuff on paper or maybe they're typing throughout the interview. Even then, it seems like the data is not very well-structured. How do you provide structure — It seems like the best way, and this is like pretty far-flung, would be you record every element of the interview and then you put it into a neural net and have the neural net spit something out if it can. That's sounds overwhelming to some people, but I would actually be like, "Let's do that."

[0:53:10.5] AB: That's our goal basically. I don't actually think Google quite deserves the reputation incidentally. One thing they have not done, at least they have not discussed publicly any way is false negative studies. It's like the careful analysis of people who are not hired and maybe some study if they could have been good. Google definitely moved the bar, but they still — They have a semi-standardized process.

Individual engineers, interviewers, can still come up with their own questions and they still have a culture where they want — I'm using the world culture. They still have a system where they want the majority of engineers to be interviewing very part-time. In that scenario, in that setup, it's just nearly impossible to really be rigorous about making sure that everyone is being asked the same question and rigorous about experimenting with the data recording.

What we're doing here is we have a small group of people who basically — I until recently was one of them. People who basically interview full-time. What that means is that we have weekly meetings where we discuss what we're evaluating and discuss making changes and just basically test, "Okay, we're going to evaluate this, this, this, we're going to record this trait for the next three months and then go back and see if it correlates with the outcomes of the companies."

Yeah, it's really hard. It's this extremely messy complicated process. People have all kinds of different strengths and weaknesses and skills. We spend a lot of energy sort of experimenting with what are the concrete things we can record that are predictive of who will do well at companies.

[0:54:42.1] JM: We're drawing to a close, but what are some of those things. What are some of the states or the data points that you've started to track, things that you can eventually plug in to the neural net?

[0:54:54.0] AB: The simplest ones. We've actually found these to be some of the most predictive is just concrete measures of progress. Give a person a problem and — Per our earlier discussion, making it a problem that's more sort of a series of steps rather than one leap of insight. Spell it out clearly and then just measure how far they get during a period of time. There are obvious flaws to that. There are some people who go slower and are like more careful. We need to have some sort of — We like having like paired metrics. One metric to that is a raw measure of how far they got. The second measure, that's a measure of, for example, process quality.

We also like recording subjective numbers and objective numbers. An objective measure of quality will be; did they write tests? Did they create constants to — Record these values. Did they generalize the problem such that this aspect can be changed and it will still work? A long list of things like that we record yes, no, for these things. Did they do these things?

We're expressing some judgments, because there are some great programmers who might not do some of those things. In balance, we find that people who run a better process get more check marks than people who don't. That's these sort of the objective measure of code quality. The objective measure of progress is just how many of these tests passed within the time that they're working.

Then we have also, to compare, we have a subjective measure of quality. Do I, the interviewer, feel like they ran a good professional process after I watched them program for an hour? Do I, the interviewer, feel like they're productive after I watched the program for an hour? We have those four different numbers that we can plug in to. For now, we don't have — Our data is laboriously gathered one interview at a time and so we're not using big complicated, multilayer neural nets. We're using simple linear models, but we can still plug this in and say, "Okay, do these things correlate with a candidate is doing well?"

What we found actually is that, so far, the objective measures outperforms objective new universally, which is pretty interesting.

[0:56:49.3] JM: That's really promising.

[0:56:51.3] AB: Yeah, it is.

[0:56:51.9] JM: Okay. I'm excited as always to talk to you and I got my eye on Triplebyte. I look forward to doing more shows as you guys discover more. It's so fun to watch, because you're attacking this area that's like basically entirely superstition and just like gradually dissecting out what is BS and it's just so fun to watch.

[0:57:19.6] AB: It's usually fun to do, except for the countless hours I spent in interviewers. Other than that, it's very fun to do.

[0:57:25.7] JM: Yeah. I'm sure you probably learned a lot about human psychology in those interview.

[0:57:31.1] AB: Yeah. People are so diverse.

[0:57:34.8] JM: Yeah. Okay. All right, cool. Thanks, Ammon. Thanks for coming on the show.

[0:57:38.2] AB: Thank you, Jeff.

[END OF INTERVIEW]

[0:57:41.1] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily.

Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]