EPISODE 940

[INTRODUCTION]

[00:00:00] JM: Facebook leadership has a significant amount of engineers in its ranks and engineers understand how to create an environment that appeals to other engineers. Engineers do not like working on projects that they are not interested in. So Facebook optimizes for matching engineers to enjoyable work. Engineers do not like taking orders from managers. So Facebook optimizes for a collaborative self-driven relationship between engineering and management.

Arturo Bejar was an engineering director at Facebook for more than five years. Arturo managed a team that built site integrity tools as well as the product infrastructure team, which created the groundbreaking React and GraphQL open source tools.

Arturo has a deep understanding of what engineers want from a manager, and this is apparent in the results of the teams that he has worked with. Arturo also cares deeply about the human side of technology. When he was at Facebook, Arturo started the Compassion project, which looked at ways that technology could help support people through difficult times and alleviate suicide and bullying. Arturo joins the show for a conversation about engineering management, humanity and the ethos of Facebook.

If you're building a software project, post it on Find Collabs. Find Collabs is the company I'm working on. It's a place to find collaborators for your software projects. We integrate with GitHub and make it easy for you to collaborate with others on your open source projects and find people to work with who have shared interests so that you can actually build software with other people rather than building your software by yourself.

Find Collabs is not only for open source software, it's also a great place to collaborate with other people on low code or no code projects or find a side project if you’re a product manager or somebody who doesn't like to write code. Check it out at findcollabs.com.

[SPONSOR MESSAGE]

[00:02:13] JM: If you're a SaaS of software vendor looking to modernize your application distribution to gain more enterprise adaption, check out replicated.com. Replicated provides tools to delivery your Kubernetes-based application to enterprise customers as a modern on-prem private instance. That means your customers will be able to install and update your application just about anywhere.

Bare metal servers in a cloud VPC, GovCloud and their own Kubernetes cluster, VSphere. This is a secure way that your customers can use your application without ever having to send data outside of their control. Instead of your customer sending their data to you, you send your application to your customer. Now this might sound difficult and maybe you're not used to it because you're a SaaS vendor, you're a software vendor, but Replicated promises that recent advancement from tools like Kubernetes make it far easier than before and the replicated tools can help vendors operationalize and scale this process.

The Replicated tools are already trusted by noteworthy customers like HashiCorp, CircleCI, Sneak and many others. As a result, over 45 of the Fortune 100 already have an application deployed via Replicated in their infrastructure. That's a strong sign of adaption.

Go to replicated.com for a 30-day trial of the full Replicated platform. You can also listen to an interview with Grant Miller, the CEO of replicated that we did a while ago.

Thank you to Replicated for being a sponsor of Software Engineering Daily, and you can check it out for yourself at replicated.com and get a free 30-day trial.

[INTERVIEW]

[00:04:21] JM: Arturo Bejar, welcome to Software Engineering Daily.

[00:04:23] AB: Thank you for having me.

[00:04:25] JM: You joined Facebook in 2009. What had you heard about Facebook that made you want to join?

[00:04:30] AB: I actually had not heard much about it as strange as this may sound. How it ended up happening is that I got a call from somebody that I trusted and said, “Oh, you should really go talk to these people,” and I had been at Yahoo for a long time and then I went in and I began meeting people and I just really liked everybody that I met. Everybody seemed intelligent, grounded, well-intentioned, had great questions and I just loved sort of what I got out of talking to people including like Shroff, my manager, Zuck, engineering folks, Jayro. It’s just like an amazing set of people and that’s what made me want to go there.

[00:05:14] JM: Before Facebook, you had worked at Yahoo for 11 years. What were the most notable ways in which Yahoo engineering differed from Facebook?

[00:05:22] AB: So Yahoo engineering favored a lot of individual independence. So everybody had the service that they were responsible for. They pretty much had their own build, which on the plus side made things very fast as far as getting patches out and doing changes and having agency over your code. But the downside of that was that anything that had to be done broadly was very challenging. Something that I experienced firsthand as my role in security increased overtime when I was working there.

Facebook instead was very disciplined about the way that it did software. So the way it handled code reviews, the way that it handled – Like the infrastructure, everything about it was this whole other level of discipline, which I really appreciated. As well as an approach to security, which was – Well, of course, this is what we have to do from day one.

[00:06:12] JM: Was there something about Facebook that encouraged more cohesion or collaboration among engineers?

[00:06:20] AB: Yeah, I think that the transparency around the way the code was submitted and reviewed helped with that a lot. The fact that you had to have your code reviewed by somebody else and get feedback from it and the quality of the feedback was very high. This was not just in the code. I remember like the first time I did an interview, I went into the interview tool and I put in my feedback for the other person and then I realized you could actually see the feedback that everybody else had written. Something I had not come across in my professional life, and I felt

pretty self-conscious about the quality of my feedback and made me want to up my feedback game. I think this was true of code and all the things that I saw at Facebook.

[00:07:00] JM: Can you go deeper on that? I'd like to know more about how Facebook encouraged collaboration and transparency and trust among engineers.

[00:07:10] AB: I think that it would come down to a few components. The first one was, across the board, if you were a Facebook engineer, you were trusted. You actually didn't have to go through a process through which you had to earn the trust of other people. If you came in and you made it through boot camp, the assumption was that you would have something to add or to contribute. So everybody was very open to feedback or questions.

Then the fact that any piece of code that you wrote would be visible to everybody, and the comments would be visible to everybody, created an environment of transparency, which I think is health, because anytime that you feel that you're doing something that you really don't want other people to see, there's probably not a great reason for that.

The open space, I was definitely one of the people that the open space worked well for me, because I got to approach people and discuss things. Then I think also through the interview process, Facebook did a great job of helping pick people that had kind of social skills and approach that's incohesive with the organization.

I was at Yahoo when it ran very – Grow very fast, and I saw how the culture went downhill, because it's very hard to keep culture as a company grows. The time I was at Facebook, there was a very conscious approach of how do we keep people fit within the culture and how do we onboard them, how do we interview them in order to keep the quality of the culture, and that's something that I respected a lot.

[00:08:39] JM: How else was Facebook able to scale the culture?

[00:08:42] AB: Well, I think one of the big things that we had to work on is that the moment that you start referring to another team as a team name rather than the name of a person within the team, is the moment you start getting into trouble. Because you start building assumptions and

conclusions about the team name that might not apply to the individual. If somebody would come up to me and say, “Oh! You know the platform team did so and so, and I’m pretty upset about it.” I would always be like, “Well, who in the platform team did that?” They were like, “Well, I don’t know.” I’m like, “Who do you know there?” They would be like, “Alice.” I’m like, “Have you talked to Alice about it?” They were like, “No.” I’m like, “Will you talk to Alice about it and then come back?”

Every single time that happened, when people came back, it really changed the nature of discourse with the other team, because most people intend well. If you talk to an individual, they will hear you out and you’ll figure something out. If you start going like, “Well, no, you can’t work with that team.” Then whatever you say is going to feel true to you and it’s going to bias all of your interactions with that team. So we worked pretty hard even as an organization group quite a bit to take team names and transform it into individual names and then work with individuals. That really helps a lot when a company grows.

[00:09:58] JM: That focus on humanizing the other people in your environment, that’s not just been a focus of your engineering work within Facebook, but that’s also something you focused on in terms of the Facebook product. You focused on elements of compassion and how users are treating each other on the platform. It sounds like there is a sense in which you want people to have more human-to-human humanized interactions across the internet both in the workplace and in more social internet functions. Would you say that’s accurate?

[00:10:49] AB: Yeah, I think that’s pretty spot on.

[00:10:51] JM: Do you think more generally the Facebook engineering culture was designed to model the kind of culture that Facebook wanted to see in the world?

[00:11:05] AB: Yeah, I believe so. I think that if you look at the principles of being open and connected, that the approach to transparency, the fact that everybody was approachable, the working, keeping an environment where you could express what’s important for you or ask questions and have access to people that would be difficult to have access to. I think that Facebook did a good job of modeling that in the context.

Now, I do think that at the same time, there are some inherent challenges to keeping that both said organization scales as well as in sort of pretty small scale when we communicate with each other using email or social networks. I mean, there's something inherent to electronic communication that makes it so that it's sometimes tricky to navigate certain disagreements and challenges. I think that's a pretty important work that needs to be done right now. I did some of that work at Facebook, but sort of see the patterns, you can see those patterns across the board be it email, be it texting, about how it's easy to getting to misunderstandings because of the fact that we're not like seeing and talking to each other face-to-face or even using voice.

[00:12:16] JM: This is one thing I like about podcasts, is there is something to the vocal tenor. I can gather certain elements of your personality and how you're feeling about something from the dimensions of where your voice is going that would not be present in text, and you can extrapolate that much further to video and then to real-world three-dimensional interactions and you just can tell that there is so much dimensionality that is being lost in our virtual communications.

[00:12:56] AB: Yeah, correct. Actually, if I may pick on one of the words that you used just now, one of the things that I've seen in the course of my work is sort of the reference to race-to-face as real world. I've been trying to figure out what's the right language for that, because when we did work through all of the issues that people were having through Facebook or when you look at the misunderstandings that happen through our communications tools, what's happening now that is very different is that a significant component of our lives is unfolding through these mediums.

So, if you get angry at a family member, chances are that part of it plays out through texting or part of it plays out through like a social media with an audience or it plays out through different contexts. I think that the real world now encompasses significantly electronic communication and face-to-face communication. But what you said about how – When we're talking right now, you can hear my voice. You could sort of interrupt me gently and you can hear if I'm getting upset. These are all things that are essential for us humans to communicate and helps us adapt to each other and come to a common understanding.

There's I think really good science around like seven signals besides the language that we use to communicate with each other and verbal sort of like facial expressions, query language, and Ben Segal studies this amongst other people. In the absence of many of those signals, we tend to take text very literally and just black and white and we don't know anything about really where the person is coming from in terms of inflection. We like to think that we do, but we really don't. That lends itself to misunderstanding in a sense. So I think it's important to figure out what are the things you can do with voice and video, as you mentioned, that can bring those dimensions back, because I do believe that once you have those back, people figure out whatever the right answer is between them.

[00:14:52] JM: You spent time at Facebook building internal tools. That was part of your work. Internal tools are something that the outside world is not aware of. Facebook doesn't typically even talk about that stuff as much, because it's not like something like product infrastructure, which you also worked on, where it's kind of deeply technical engineering problems getting solved. There are deeply technical engineering problems, but they're more domain-specific to Facebook.

That said, there's so much dispute resolution and safety tooling infrastructure that gets built at Facebook and you contributed to some of those tools. I'd love to get a perspective on how you build these kinds of tools, things like dispute resolution or controlling cyber bullying, these sort of tools that allow moderators and safety staff at Facebook to make the platform healthier.

[00:15:59] AB: Yeah, I think that there're a few parts to that. I think one part is it is very important to work hard at giving the people on the frontlines of reviewing content the right perspective and enough data so they can make the decision they need to make. So, one of the most invaluable things that you can do is to have the engineers that are working on these tools spend time using these tools as if they were agents. Because we'd love to think that we understand and we know these things and we really don't. Every time – Even as much as it's a tool that you built and you think you understand and you think you understand how people are using it. Going in and spending a day being a customer care representative and doing that like at least once a quarter more than one day is one of the most valuable things that you could possibly do.

The second part of this is that when you do that, you understand two things. One of them is engineers are really good at seeing solutions to problems, and in the process of using the tool, you might find that the way that the person was using the tool and what they think is possible is shaped by the way the tool has been up to that point in time.

I remember there was one time where I was working on like – So credit card infrastructure, and there was like a second password to access to access the credit card infrastructure. We went to talk to people in customer care and they were talking about the secret password and they had no idea what they meant by the secret password. I realized that in the process of giving the tools out, I had not given them like a password reset send link that would make everything like a thousand times easier. But they didn't know that it was possible. They knew the tool the way they used it.

So, when you partner very closely with the people that you serve, if you're tools engineering and you spend time with the people that are using your tool, then you can find these things that might be like, "Oh my God! I didn't know," and then it takes you like half a day to write the code for it and then you push it and then you like massively improve the quality of life of the people working on the tools. Because, really, your audience in the case of building toolset, this is a team, the customer care team that builds them. Something you have to watch out for on that, is because the customer care team is working through the tools. They're going to have their list of what they think is the most important things for you to build. You might not agree with that list. I've seen that like three or four different places I've worked.

The process through which you figure out what is a good common list is incredibly important, and shadowing tends to be like the best way to do that. But it's also important too that if you find yourself like authorizing, as we said earlier, where you're going like, "Oh, those people in customer care don't know what they want. They keep asking for certain features." I've seen that happen quite a few times, and it's all a product of the people that work on the frontlines of these tools not knowing that what is possible within the tools, and the moment you kind humanize that relationship, things get better very quickly.

Then when it comes to the domain-specific things of, for example, where you're talking about booming, we realized that context is very important. So one of the biggest insights that I had

during the time at Facebook is we're seeing photo reports go up significantly, and there's something that's called the action rate, which is if you take all of the issues reported on a single queue, what percentage of them violates policy that the queue is for? So we have nudity queue. What percentage of – What was on the queue? Who violate the nudity policy or come close to it? You could even measure that.

It turned out that the percentage was exceptionally low when we began working on this. There was a lot of things in a nudity queue that have nothing to do with nudity. It were just people like hugging and smiling. Now, if you build a tool that only allows you to see the photo grabs, you will never figure out what's going on, because the tool that the team had built, which is wonderful for figuring out what was going on, included a little bit of information on the person who had shared the content, the person who was reporting the content. Through that, we discovered that in most cases, the person reporting the photo was in the photo and it had been shared by somebody who was one of their friends.

Then we're like, "Oh my God! What do you do with that?" Because when you look at the class of issues that unfold on social services, there is a class of things where you expect the service to take care of, like hate speech or nudity, if it's not allowed, and a set of policies where the policies would come in and take away if you had that. But then there's a whole other class of issues that unfolds that really it shouldn't be somebody coming in and making a decision whether that's good or bad. You kind of need to kind of figure out with the other person, and we would have never figured out that whole body of work that came from that insight had the customer care tool not given us enough context to understand what was really going on.

[00:21:02] JM: That's remarkable.

[SPONSOR MESSAGE]

[00:21:10] JM: As a programmer, you think an object. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers and the cloud area. Millions of developers use MongoDB to power the world's most innovative products and services, from crypto currency, to online gaming, IoT and more. Try Mongo DB today with Atlas, the global cloud database service that runs on AWS, Azure and

Google Cloud. Configure, deploy and connect to your database in just a few minutes. Check it out at mongodb.com/atlas. That's mongodb.com/atlas.

Thank you to MongoDB for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:22:06] JM: Coming to another area of the company that you worked on, you started the product infrastructure team, which created both React and GraphQL. Tell me the story about how this team got started.

[00:22:21] AB: So there was a time a few years ago where there was like a need to revamp the privacy settings, and the whole company kind of rallied about doing that, but the way we went about it is it got a very small number of engineers, including Nick Schrock, that you've spoken to and a couple of other people. We got in a room with like a couple of product managers, a couple of engineers, one engineering manager, that would have been me, and we focused on doing some designers for UI, some UI engineers. So it was a very small team, and over a period of like two or three weeks, there was a tremendous amount of work done to make the changes to the infrastructure and the UI, and it got pushed out. It was very well-received at the time.

We realized through doing that and in conversation with the engineers, and this was some of Facebook's – Facebook's people were really good engineers, but I have a lot of love for the people that started product infrastructure with. They had lots of really good ideas about how you could make the software infrastructure much better. So, we decided it was a pretty good idea to give them the space to do so and the support to do so. The initial team was like three people, probably including me, two or three people including me.

The idea was take [inaudible 00:23:48] of the code stack and then build a much better infrastructure approach to doing it. But not just build it, but also go and install it. So it wasn't just like somebody in the mountain handing down a piece of code, which is this shall now be the thing that you do. It was more like you, you know, you're the center, the refrigerator and you roll it into the house and you put it in the kitchen and then you use it for a while and then you realize in the process of doing so what makes a refrigerator good in the house.

So each project that we tackled was handled from that aesthetic of like it's completely driven by the engineers, supported by the managers, by engineers, for engineers, but whatever it is that you build, you have to be able to go and work with a team that's going to be using it until you learn everything that you need to learn about how it's going to be used and the things that you might have gotten wrong in your initial implementation. That formula I think worked very well. I think what the team accomplished with that approach and the engineers who were there was truly remarkable.

[00:24:47] JM: Why did it make sense for GraphQL and React to come out of Facebook? Was there some set of constraints or challenges with web development was Facebook was encountering that forced these technologies to be developed?

[00:25:06] AB: Yeah, I think that for GraphQL and React, they were really born out of a few engineers who had like a burning need. So if you look at the nature of development, once you start building things that scale, you will come across problems that require taking like a step back and saying, "Well, if you had to do that all over again, how would we do it? Now, we've seen that process go wrong in many places where you take a step back and you build something, and that's something that you build, it turns out takes forever and it doesn't really help.

So we decided, and the way that the team handled this was always very pragmatic in terms of what was built hand what got put in, and really before we even thought about and the people who were responsible for this thought about taking it to the outside world, it really had to be a tool that people on the inside got a lot of benefit from. GraphQL was just profoundly transformative in the way that data was acquired and handled, and the same way, I remember sort of talking to the engineers that began all of the React work, and the common thread between those was sort of a very talented engineer seeing what they thought was the problem from an infrastructure. Then as a manager, you basically say, "Okay, you have go for it. When will you have the thing ready to be first put into something? Who is your first client that you're working closely with that?"

From an engineering manager's perspective, I think my job on that was to make sure that the effort was grounded and it got adapted and whatever was it we learn from adaption happened, but I think that all of the credit on what was accomplished there technically really goes to a set of engineers with a strong vision and a culture that supported the adaption of these technologies and the willingness of an engineering department to invest in things that might not be like an obvious return in the short term, because infrastructure projects require a whole other timeline and approach of doing things. Retrofitting a new way to access the database or changing meaningful components of your UI layer is a really meaningful thing where you're trying to get products done. But I think that Facebook is very good. I want to encourage anybody to make sure that you dedicate part of your engineering budget to that class of issues, because it pays off very handsome in the long-term.

[00:24:41] JM: There's an anecdote, I remember hearing a couple of times around 2015 when React and GraphQL were coming out. I think I first heard it from – It may have been when I attended F8 and I think I heard it from the CTO, Schroepfer, and the quote was, "We're working on virtual reality. We're working on drones that are beaming down internet to people. We're working on this social network that stretches across billions of people, and yet whenever I talk to developers, all they want to talk about is this JavaScript framework that we developed, React, which is hilarious. But actually it makes a lot of sense, because people may not remember, around that time, there was such frustration and confusion around which of these JavaScript frameworks people should be using?

So every frontend developer was like, "Should I be using backbone? Should I be using the flame?" What was it? FlameJS? There's a fire one. BlazeJS. I can't even remember what – I mean, there're a million of them, right? Angular – Then React comes out and for some reason the world consolidates around React, and I was a little – I was doing a bunch of shows about React at the time, and to this day, I've been trying to understand what was so different about React. I mean, I know there were a bunch of different engineering issues that it solved. There're a bunch of little tweaks to it, but it seemed like it was more about the vision that the team had and the standards that we should have around how frontend web development should work. Explain to me your theories on why React was so successful so early and how it's continued to be successful. What did it do so differently?

[00:29:37] AB: So my goal as a manager was to create an environment where people would be doing the best work of their lives. Sometimes when you get very talented engineers, they come with a set of challenging affordances where people sometimes kind of have challenges with communications with other people or they have a very powerful vision and it's kind of hard to bridge that with the environment that they're dealing with.

But ultimately, everybody has a vision of the engineer they would like to be – And the contribution that they would like to make. We can talk about that if you're interested a little bit later about sort of like the management techniques behind this. But I think that the one thing that was true about the product infrastructure team and hopefully other teams that I supported is that you had all of these engineers that had very strong vision about the contribution that they wanted to be able to make to Facebook. It was always in service of other engineers. So that was really clear, by engineers, for engineers was at the heart of everything that we did.

Then when you had somebody that had like a really strong vision about what they had to accomplish, the managing challenge was how do you keep the vision grounded so that cod starts rolling out and gets put into place, and then how do help the person see for themselves what it is that they're doing that might be incoherent with their vision of the kind of engineer they would want to be.

I want to be an engineering leader. I want to create a piece of code that everybody uses. But when you get back to your peer reviews, there're a third of the people that say, "You know that kind of person is kind of hard to talk to."

If you help a person see that and see that from the perspective of the feedback that they're getting, then they realize it doesn't really jive with the vision they have for themselves and they will work very hard and they will only change being a better communicator. I think I can think of everybody in that engineering team that I work with, all of them needed slightly different things in terms of the support that they needed in order to be able to do that work. But we did get to a point where I think that the engineers that were working, they were doing that we're very proud of, that was very adapted by their peers and we created an environment where stuff was getting cranked out and it was very high-quality and it serves some very real needs for people who are their clients. Then it went out to the world.

I left Facebook I think around the time where all that stuff really started going out into the world, and I've been delighted to see what or how far it has been getting. But I think, for me, what I really remember was the time spent with Schrock and with other engineers in the team to work on sort of what makes you do the best work of your life, and that's the thing that I'm most passionate about or interested in as an engineering manager.

[00:32:22] JM: What is the role of engineering leadership at Facebook?

[00:32:28] AB: Supporting the engineers.

[00:32:30] JM: Is there anything that Facebook does uniquely well in terms of engineering management that perhaps you think other management structures would benefit from understanding?

[00:32:47] AB: So, I think that the philosophy that I tried to have as an engineering manager had a few components to it. One component is the language that you use. I think it's very important to say things like the team that I support, versus a team that I manage. I think that the language that we use shapes how it relates to things.

There was somebody, an early Facebook employee who told me this idea that – He didn't believe in like hierarchies, like top-down hierarchies, that felt wrong to her. That really the model for management is one of a tree where the CEO is the base of the tree. Then as you go up, you have branches and your support branches, but your job is to support, to make sure that everybody gets the resources they need. At the same time, to be able to figure out when somebody is not working out and then help them transition successfully.

I think the first element is being really careful about the language that you're using, because if you start talking like my team this and my team that and you start pushing things down, I think you're already kind of like at a disadvantage in terms of what the team is going to be able to accomplish.

The second part is that I think it's very important to educate the individual contributors in the team about the responsibility they share with their manager on being supported successfully. What I mean by that is usually if you do have somebody like the manager, sometimes your individual contributor. Your expectation is you know that person knows everything about my life is making decisions, can solve most of their problems that I have. That's never the case. I mean, every human in the system has things that they're good at and that they're bad at.

So we developed a tool that was used in a number of teams where it's kind of similar like the pulse survey, where you ask questions. This is a tool where you would ask everybody in the team six questions related to their quality of life as an engineer or other role within the team, and it was expressed in terms of their experience. Like, I feel I have enough support to get my work done. I think I can look up the list of the six and tell them to you. I have a note that it's called managing your manager, where all of this stuff is codified, that's public.

In that context, helping people understand what they should be expecting from their managers is really important. So they can know what are the kind of things they can ask for support. You can have some protocols for like conflict resolution, but it can't just be like top-down you do whatever your manager tells you to do. A manager has a service that they have to do to the people that they support. The individual contributors have a responsibility to their manager to make that relationship successful.

So I think when we set that up in the environment, the satisfactions numbers were very high and the productivity was very high, but you have to be very open to feedback and you have to create some processes that help you see when something's wrong for somebody. So this set of six statements I had told you about, we would run them alongside every review cycle. If you got somebody that said, "Well, I disagree that I have enough support to get my work done," then you want to have a conversation with them very quickly that helps figure out what needs to be changed so that it's better. Really, I think the goal of the manager should be that everybody in the organization should agree or strongly agree to all of these questions.

By the way, even the most experienced managers after years of doing it with people they've been working on for years, you run a tool like this and suddenly you get a disagree and you

know you have work to do. It's not something that you figure out and then you're good at it. You have to keep working at getting the people stuff right.

[00:36:30] JM: One of the motivations for doing this series of interviews is that I think people generally don't have a perception of Facebook as a company that does engineering that is on-par, in some cases, better, or highly differentiated in a way that is positive from Google, for example. When Facebook was rising to prominence, Google was the most prominent engineering organization, the one with the best reputation, the one that everybody regarded as representing the highest order computer science principles and software engineering principles. Facebook turned a lot of those things on their head, and it's hard to understand some of those unconventional engineering decisions, like take the scaling of PHP for example in a microcosm. But when you take a step back and you see how the organization fits together, it's quite beautiful.

You see this organization that is able to build internal tools, internal processes, that make engineers incredibly happy and incredibly collaborative. But because it is so hard to explain to the outside world because it's so foreign to the outside world – As I'm talking to these engineers from Facebook, it just feels like a completely different planet in some ways than other organizations that I interview. So I think it's almost like unbelievable to some people, or it's so disjoint from their experience. Does that resonate with you? I mean, is it that foreign of an engineering environment from the things that you've seen in the rest of the industry?

[00:38:30] AB: Yes. To me, in my experience, I think that really interesting thing that I observed within Facebook in contrast to other contexts that I've seen or experienced is that – And this actually goes to something that you mentioned earlier about social, and we talked a little bit about the interview process.

Everybody that Facebook hired had a pretty good level of like social intelligence and communication and openness. So you could see it in lunchtime. You could see it in the way teams relate to each other, and it was just very important because it's very hard to build a social product that's a service if you're not like the social person and kind of like enjoy connecting with other people.

I mean, of course you see everything in terms of doing that, but the norm was, like most of the people that we work with in the kind of an environment that was created, there was a very healthy – In all the time that I was there, there was a very healthy social fabric to it. I think if you take engineering as the union of making sure that you do the highest caliber of computer science based, evidence-based, the metrics will sort of point to the way the clarity of the code, all the principles we hold dear. Then you move them into the context of like but it's to the service of other engineers. It's to the service of the people who are using the tools, and you add kind of the social component to it. It made a huge difference.

I was really surprised when I came over from Yahoo. By the way, I want to say I loved Yahoo. We had a lot of fun. The team there that did all the security work that was called the Paranoid has been one of the great pieces of work I've had to do in my life. But I will say that it was challenging to work through the security stuff, because sometimes you would like had to make a pretty strong case for it, because people were going like, "Well, is there something that's opposition between doing the security work and then doing the engineering or the product work?"

As I kind of came into Facebook expecting to like need to fight those fights, and I never did, because when I came in, thanks to the people who helped start the engineering culture and some of the early engineers, there was like, "Well, of course, we're going to make that secure the way it has to be, right? It's kind of axiomatic," and I was like, "Oh my God!" Then you approach people and people are actually really welcome when you come to see them and had a question to raise or something to work through or some bug that you had found. Then combine that with sort of the love of engineering and infrastructure.

One of the other teams that I helped start was one called security infrastructure. That team originated hack, and it was the same idea of like, "Well, really, we want to build code that makes it harder for engineers to unwittingly introduce security issues in the codebase. At the same time, we want it to be like a good language that's great to use. So, the kind of thought process was very easy to do in the context of Facebook as a company, because of the culture. It was easy to deploy. I mean, there were always challenges. It was not unicorns and rainbows.

But relatively speaking, starting from like IBM, which was my first job, through to Yahoo. Then when I've looked or talked to other organizations, the social fabric of the company, the sense of shared goals and purpose, the fact that you could approach anybody with an issue, and there would be trust, made doing all of these work very attractable. Then you take a talented engineering, you drop that person into that environment, and it does create opportunities to do some really cool work.

[00:42:11] JM: These different legendary companies have strikingly different cultures, and these different cultures work for the different companies. So when we think about the openness within Facebook across the organization, we could easily contrast that with Apple.

At Apple, many engineers – I've heard this from multiple engineers at Apple. You don't know what you're working on. You'll be working on some kind of C++ module that streams data from one place to another and you have no idea what the source of that data is. You have no data what the sync of that data is. You've got a completely firewalled office somewhere in one infinite loop and you don't know where anything else is in the organization, and you might like it. You might like the fact that you're going to go to WWDC and see announcements that are completely unpredictable. You like the mystique. You like being part of that mystique. But it's probably not – I know it's not for everyone.

I saw some of these when I worked at Amazon. There is a very different – The alien organization of Facebook. This would not work at Amazon. You see the pride of the monolith when I'm talking to Nick and talking to Pete Hunt and I'm like, "This wouldn't work at Amazon." I mean, as much I love and respect what you guys are saying, I don't think this would work at Amazon. Maybe that's a product of history. Maybe it's a product of the founders. Maybe it's a product of the business model. But I find it so fascinating that these different organizations have different features and many of these features cannot be borrowed from each other.

But that said, I do want to understand what are the things that more organizations could borrow from Facebook, like principles or are there tools that you think that surprise that have not made their way into the public domain in some form of a product, some kind of internal tooling? What can the rest of the engineering community learn from Facebook and what should they be building to better allow engineers throughout the world to work like Facebook does?

[00:44:42] AB: So, I think that one of the key things, which actually speaks to what you just said, is that it's very important to be very cautious and clear about what kind of culture you want to create in your company and have that be a work in progress. I mean, the moment that company culture is like a whiteboard magnet that you're handing around, that's kind of like game over.

So I think that every place that you have described that has round success in their own set of practices is clear of who they are and they've very upfront about it and then hire the people who like love the mystique of working on something without the knowing the context that it seen or values the way that Google approaches software development and the decision making. They find that very rewarding. If you try to be all things to all people, it gets like pretty messy. So, I think one part of it is just like be really clear about that.

Now, I do think that one of the things that Facebook culturally, which is less on the tool side, is boot camp. So anybody, no matter who you are or how senior you might be, what [inaudible 00:45:24] you come in and no matter if you like wrote 27 PhDs and 15 books on this stuff, you go in and you spend like a number of weeks fixing like CSS bugs is the first – Or like HTML bugs is he first set of bugs that you get. Then you kind of work your way up to more meaningful issues. That is a wonderful way to set the tone of saying no matter who you are, you do the work that the organization needs.

It's also a really good way to get like coding standards and introduction the infrastructure and get people kind of connected with each other. So I thought that the way Facebook get boot camp was one of the best things that I've ever seen, and I could not encourage enough other organizations to have this kind of more deliberate onboarding process that helps somebody go into the organization. Because most people, when they start at any organization, what they're going to try and do is take whatever made them successful in the previous organization and try to apply it to the new organization, and that's just like a recipe for dissonance and issues. Because everybody wants to do a good job, but if everybody has a different idea of what that means, it becomes very difficult to manage and very challenging for a company culture.

Then I think the other thing that is sort of very important is that you really have to do a very good job of cultivating the strongest engineers in the organization in terms of their capacity to get things done that are of service to other people. I think it's important to – If that's your thing. I used to love interviewing engineering managers, and my favorite question to ask in an engineering management interview is you have an engineer that's like 10 times more productive than other engineers, but causes a fair amount of social collateral damage. What do you do?

In some cultures, they'd like that one, so they will just isolate the person, surround then with people that they can work with and they can take advantage of other productivity. But if the organization, you want it to be inherently where all engineers are connected. Then you have to either work through the person so they change their behavior or have the person find a different job, because the cause to the organization of having somebody like that, when the organization relies on the connections between engineers to get work done is I believe too high.

So you really have to sort of watch the culture. I think that the other thing that Facebook did very well is that you came in and you had a lot of potential, and even if you were like straight out of boot camp, you would've given enough problem space to work on to demonstrate that you could deal with it and it was okay to fail.

When I started at IBM, when I was pretty young, it took you years to earn the rights to work on something really interesting. I think some places still think about it that way. But after looking at Facebook and supporting so many people that it's such a great work, I realized that if somebody comes in and shows a potential, it's interesting to make their scope of responsibility be – Because you can reasonably make it fast and then see how they react to learning from that. Then you can get really strong engineers really fast, but it does require a good feedback process.

So I think – Again, most of my lens on these things is through people more than tools. I mean, the teams that I supported that set integrity. We call it the spam prevention technology, which is incredible, security infrastructure, product infrastructure, the tools teams, all these things, really wonderful technical work. But I do believe that at the end of the day, all of these is made possible by the engineer who's writing the code and the responsibility of the manager is to figure

out how to support the person in such a way that they can do that and they become aware of the things that they're doing that are counterproductive. So they can choose to work on them.

Because if you imagine you're telling people to do something and they're not doing what you're telling them to do, that never works. A deal is like a month of change if you're really lucky, one or two months of change if you're lucky. But the only meaningful change that is going to make somebody a better engineer or a better contributor is going to be a complete sense of ownership about what they need to be working on and a shared agreement with their manager that if they do work on these things, what is it that they can expect from that?

[SPONSOR MESSAGE]

[00:50:10] JM: Monday.com is a team management platform that brings all of your work, external tools and communications into one place making cross-team collaboration easy. You can try Monday.com and get a 14-day trial by going to Monday.com/sedaily. If you decide to become a customer, you will get 10% off by coupon code SEDAILY.

What I love most about Monday.com is how fast it is. Many project management tools are hard to use because they take so long to respond, and when you're engaging with project management and communication software, you need it to be fast. You need it to be responsive and you need the UI to be intuitive.

Monday.com has a modern interface that's beautiful to look at. There are lots of ways to use Monday, but it doesn't feel overly opinionated. It's flexible. It can adapt to whatever application you need, dashboards, communication, Kanban boards, issue tracking.

If you're ready to change the way that you work online, give Monday.com a try by going to Monday.com/sedaily an get a free 14-day trial, and you will also get 10% off if you use the discount code SEDAILY.

Monday.com received a Webby award for productivity app of the year, and that's because many teams have used Monday.com to become productive. Companies like WeWork, and Philips and Wix.com. Try out Monday.com today by going to Monday.com/sedaily.

Thank you to Monday.com for being a sponsor of Software Engineering Daily

[INTERVIEW CONTINUED]

[00:52:00] JM: I watched this interview that you did with Philip Glass, and it brought to mind actually another interview or an interview that I did with the Netlify CEO who comes from a musical background. He studied I think like how to be a conductor, basically, which is kind of amazing, because now he's running like an infrastructure company and he's directed groups of musicians before, which is amazing.

I think there's some similarity between managing an artistic group, such as musicians and managing software engineers, because they have this impetuous element. They have this emotional side. They have this highly-creative dimension to them if you're talking about really good engineers.

Then there's also this similarity between music and writing music where you have these patterns and layering elements and these units of abstraction and this necessity to see the bigger picture as well as to see the microcosms. This is evident in Philip Glass' music as much as anyone. He's music is highly patterned and has this programmatic feel to it.

In your conversations with him, did you notice any similarities between how he thinks to how a programmer thinks?

[00:53:23] AB: Oh, absolutely. I've been talking to him and working with him with him for a few years now, and I've observed a couple of things. One of them is – By the way, I can't recommend highly enough his autobiography, Words Without Music. I was very surprised when I read it. Also in my conversations with him, I found somebody who was into their 80s and beyond sort of very open-minded about sort of problem solving and seeing patterns and exploring things. So, letting in different kinds of music, different disciplines, has always been a part of the work that he's done. Somebody who's been very disciplined.

So he had said a time of day, every day, where he sits down and composes. When he was a young man, he didn't want to wait for the great moment of inspiration to hit before he did all of the work. He trained himself to be like, "Okay, each day at like 10AM, I'm going to sit down and I'm going to train my body and my mind to be able to consistently write music." I always like really respected that.

Then his music, I think as you find a lot about, like finding patterns and exploring them and putting them into larger structures using, in his case, the language of Western music. But the other thing about him that's really fascinated me is that he puts together – There's this festival that he does in Carmel [inaudible 00:54:58] once a year. Before that, he brings artists from difference disciplines. It's called the Days and Nights Festival, and he brings artists from different disciplines. I watch him do that and he is masterful at giving space to each of the artists to bring in their vision or whatever their ideas are for the music. So he gets like the first time you see gambian core player with a western harp on stage like ever, and he does all these crazy combinations like that that are really interesting.

When he brings these people together, he does a combination of like bringing in people who are pretty good about like taking feedback and doing that, but at the same time have a strong creative vision. But then the other thing is he gives them a ton of space to express and play and explore and he controls the boundaries very gently.

If sometimes an artist starts doing something that really is like we have to work with sort of what everybody is trying to get done. He comes in and very gently goes, "Yeah, you could probably like do that like a little bit less," and he does it with a very light touch. That way he can get like – I've seen this in other concerts, he'll get Patti Smith playing with like – There's like Iggy Pop and Blondie, and then there's like – I forgot all the bands. Name like a weirdest combination you can think of and then get them on stage together playing in a way that's harmonious and everybody gets along and has a fun time doing it even though all of these people are very intelligent and driven and powerful.

I would easily derail the process I think in a different kind of context. So I have a lot of respect for Philip and the way that he works, and I see a lot of parallels working with him from what I've

seen from some of the most effective engineering managers at great teams that get a lot of really wonderful work done.

[00:56:50] JM: All right. Last one or two questions. What would the Facebook product look like in 10 years?

[00:56:56] AB: So, I think that real life is something that plays out across the interactions that we have in-person or face-to-face and all the services that we use. When we are in-person or face-to-face, we have developed a set of social norms and conflict resolution tools, and these are a combination of things that have evolved and things that we sort of learned through the community we're a part of that help us get along.

The recent context in the world where you can just like can just come in and say whatever is on your mind, because it's important to you. I mean, there's a few, but like for the most part, if you're at work, if you're at dinner, if you're hanging out with other people, there's kind of like a social contract that you play by that is not enforced by the police or by other entities, because it's just kind of like the social norms of the groups that you're in.

When I look at services right now, I think that in most of them, not all of them, there's an absence of these kind of social norms in a way that is negatively affecting the nature of discourse. If you look at like how, for example, the Marvel subreddit works, in that context, you see that there's a community, that there's a set of things that the mods enforce, but there's a set of things where like you make a comment and somebody goes, "Yeah, this is really not the place for that," and people go like, "Oh, okay. I didn't mean that." But that's not present in other contexts. I think like the main phase with product or Twitter or other services like that, because they haven't yet kind of built the mechanisms that help us give feedback to each other to create these social norms that make these spaces feel safe to us.

So, I think my hope is that one of the things that I see is that when you're an engineering company and you have a set of issues that you deal with, your first go-to-tool is engineering solutions. So, if I have content that I don't want on the network, my first reaction is going to be, "I'm going to go build a classifier that's going to flag that content and down rank it or disappear it." But when you do that, how do the people that are writing content learn to behave differently?

Now, there's some really interesting work on that front right now. For example, Instagram who they did something that as you're typing a comment, they have a classifier about whether based on their experience, they comment hits kind of agro. Then they nudge you and they go like, "Well, that could be kind of agro. Would you like to rephrase that?" In a way that respects agency of the person writing the comment, and then people would be like, "Oh, wait. I didn't mean to write it that way. Yeah, you're totally right. Thank you." Then you kind of write it a little bit differently.

So, my hope for Facebook 10 years from now and for Twitter 10 years now is that – As much as they're working on what you could think of of technological, straight technological solutions, where it's computer making the decisions about these kinds of things, about what might be harmful content that you stop on the network, that they invest equally in solutions that help sort of humans develop norms with each other in a way that's respectful that makes those environments feel safer.

I say there's a couple of really wonderful examples of that, but I don't think there's really any place in the world other than some of these services where you're just going with a group of friends and you say, "Whatever your political thing is, completely unfiltered," and the way that it unfolds right now is when I was working on the reporting stuff, this was my favorite finding in all of the time that I was at Facebook.

We asked people whose content was reported, what their intention was with sharing it, and we run this study like four or five times using different questions, and every time we run it, it came back as 90% of people had positive intentions sharing whatever it is that they had shared. They thought it was funny. They thought that it was important for their friends to know. All these things would come up, and only 10% would be because they want it to be provocative or like more malicious intensions, which you know humans, happen everywhere.

But that 90%, if what you do is you write something that's important for you because you're upset about it or you're activated, and then the only affordance that you easily have is to like or say you're angry, but in front of all of your other friends or commenting in front of all your other friends, it's really interesting when you take these circumstances and you move them into a

room. Imagine you're in a meeting with a lot of people and somebody says something that's upsetting to you. How will it feel to you in front of everybody else and say, "You know, that actually kind is upsetting to me." But how is that person going to know any better about what to share if you don't give them feedback?

Feedback, I don't think that feedback is a place of the Facebook or the Twitters of the world to give to the person. I mean, I really believe that – I saw this in all the work that I did, that as long as a feedback is presented respectfully, people will go like, "Oh, I didn't realize. Thank you for letting me know." Then both people learn in the context of that interaction. So I would love to see in terms of like conversation health or social norms or even as the election is coming up, what is the work that you do to help people really become aware of the impact they're having on each other so that they can develop the social norms that make the space feel good to be at.

I think if you ask different people, they tell you that some of this space, "Oh, there's too much politics. It makes me really uncomfortable." Because the one easy tool that you have right now is you can comment in front of your friends or you can turn away. I think that's the solution of that class of problems is essential to that class of applications over the next 10 years, and there's some really wonderful science and other things and other contexts that you can learn from.

Because we as humans have figured out this out with each other and we're still working on it clearly. But there's really wonderful science that you can use about what does work. Once you understand, you'll see how we're in a world that is feeding us things that divides us. Then the communication services, the social services kind of play into that in a way that's not what they intend, but it's kind of a little bit inherent to the tools. But there are solutions to be had, and I hope that they spend time working on that.

[01:03:18] JM: I love so much about what you just said. This is another one of the motivations for doing these interviews, is we have an opportunity to build – Some of it is almost like the Tower of Babel in some ways, or I don't know if that's – Maybe the Tower of Babel, maybe that was in opposition of – I don't remember the biblical history.

Anyway, something that unifies us as a global society and in a very productive way. People want to be unified. There are people that want to be unified. Yes, there are people who want to be in their own sectors, but there are also people who want to be unified. I see this on a daily basis, because engineers who listen to the podcast, whether they are in Africa, or India, or Pakistan, they connect with me and they are interested in communicating.

Sometimes they connect with me on LinkedIn and I see people in Africa, for example, and their names are in all caps. I'm like – This doesn't really bother me, but like I'm just thinking like do you know this is kind of a red flag for people when they see a name that is in all caps, and they probably don't know that. They have no idea.

As we become a more internationalized knowledge workforce, as we should, this is a kind of thing that absolutely matters. It's very important. We have a number of open source projects that we work on, the Software Engineering Daily open source projects. There are many people in India who want to come and contribute to these projects. They want to contribute free code. They may be an engineer with 16 years of experience. They work on virtualization technology in India and they want to come and contribute to the repository, the Software Engineering Daily open source repository, but their conversational norms are not the same as American engineers.

[01:05:12] AB: Totally.

[01:05:15] JM: So they can't cross that barrier and contribute, and this is an environment where we share the same programming languages even. But because we don't share the same human languages, there's a mismatch. So, we have this opportunity with – The Instagram commentary example, where the service might say, "Look, maybe you didn't mean this, but there's a chance that the other people in this comment thread could interpret what you just said as hostile." That's really useful. That could be really, really useful. This is why machine learning is a really big deal. It's really important, because it can add this finesse to our communications in an almost uniformly, productive way. Yes, it can be used like a blunt instrument. Yes, it can be a tool with sharp edges, but there is so much to be gained.

So, I've really enjoyed talking with you about these elements of societal structure and also engineering that there's so much room for improvement. This is all a work in progress. It's not a finished product. Sometimes people criticize Facebook like it's a finished product. This thing is out of control and completely new just like every groundbreaking invention goes through a period of being – There was a time when we had cars without seatbelts. Sorry, we invented the car before the seatbelt. Oops! I guess.

[01:06:55] AB: There is a – And we can go into now. We can go into it later. There's a sort of a deeper conversation to be had about this, because I believe there's a really wonderful set of opportunities right now for people who begin to think about things this way and explore the problem space. Because the patterns have actually been there since textual communication, specially [inaudible 01:07:21] communication became prevalent because of the speed of turnaround.

But one of my default rules is like if you don't come to an agreement in two emails or two comments in a thread, it's not going to happen. It's just going to go wider, especially when there's an audience. There's good science about this. If you say you like a blue wall and I say I like a purple wall and we're aware that there's a set of people around us, rather than you're trying to understand why I don't like the blue wall and me understand why you like the purple wall. Chances are we're just going to start repeating ourselves. The moment that somebody starts repeating their argument, whatever context it's in, it's not going to get resolved.

Now if you are an engineer in an organization and you see this in your code commenting tool or you see it in an email thread and you recognize the pattern, you can just like get up and go be like, "Yeah, you know, can you please help me understand what's going on?" Then you kind of listen to the other person and then you understand what's going on. Once they feel heard, you can tell them what's going on, and that helps.

Now the thing is, is that the human mechanisms which make that possible have really not yet been explored a lot in the context of the tools that we use to do code and the managed repositories and managed open source projects. The understanding of the importance of the language that we use and the differences in language and how people approach problems is something that presents a lot of really wonderful engineering opportunities.

Imagine that it's very easy in like a tool where you have open threads to give feedback on somebody's piece of code. If you're able to easily just open a one-to-one chat that has a context and says, "Hey, just to help me understand without everybody watching little R. Can you please help me know what's happening?"

So there's a range of solutions that begins with interface affordances and understanding of an inherent social dynamics to electronic communication with an audience and the like of sort of like synchronous voice, facial, all the other signals that we share, but you can design for that. Then if you add to that, at certain class of issues, machine learning and you understand that in order to develop a solution, it's not just about the technology that's in front of you, but there's an understanding of what the person is coming from with the things that they're trying to do, and then there's a perception that your client or the people are having that is going to be different. The way I respond to something, you might something that means really well that gets me really upset. If you start from that point from the center, you can then create tools that help people navigate this with ease.

But there's kind of like a social resolution engineering that incorporates all of these elements that is kind of something that we've seen a little bit so far and we see it in Instagram, we see it in the work I got to do, but in as much technology communication, the development of tools and body of knowledge in that context is I think one of the greatest opportunities that is open right now for doing really cool work that helps people work with each other, get along with each other and the solutions and the answer to this do not align necessarily in just the technology that we have, but in looking at a social science research and looking at other bodies of knowledge. Then you go like, "Oh! I can totally see how I can take that principle when we're there and make an implementation of it."

So, I would encourage if anybody is listening going like, "Yeah, that sounds great," to really think about both. You can just apply this with zero tools in the workplace by just observing the way that electronic communication plays out. B, I think it opens up a lot of really wonderful opportunities for things that you could build that help people get along with each other better.

[01:11:48] JM: Arturo Bejar, thanks for coming on the show. It's been really fun talking to you.

[01:11:51] AB: Oh, thank you for having me.

[END OF INTERVIEW]

[01:11:21] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[END]