Episode 8 | Pair Programming
Guest: Farhan Thawar
11/24/20

Speaker 0 00:00:00 I'm Danielle Royston and this is Telco in 20. Hi, you've joined us in the middle of a great new series about the HR side of managing a big change in your organization. Like say, migrating all your applications to the public cloud in a move to the cloud. You're probably going to have to make big changes to your organization. Maybe hire some new talent or train your existing team. So I wanted to tackle that topic. How do you quickly develop new skills in a large organization? The answer pair programming. I bet you didn't think I was going to say that. Pair programming is this crazy idea that sounds like there's no way it would work, but it totally does. I know because I used it at Optiva to train hundreds of new hires in a matter of months. Our next guest, Farhan Thawar, is probably the biggest evangelist I know out there talking about the huge value of pair programming. I know Farhan from that crazy company in Austin, Trilogy, where we recruited him from Waterloo in Canada, a kickass university, to come join us in 1998. Farhan is totally awesome. Super smart and works crazy hard. And his intellectual curiosity is off the charts. And I can't wait to talk to him. So let's take 20.

Speaker 0 00:01:23 Today we are talking to Farhan Thawar, who's now VP of engineering at that Canadian sensation, Shopify. Welcome Farhan.

Speaker 1 Thanks for having me. I'm psyched. It's going to be super fun.

Speaker 0 I wanted to bring you onto the podcast to talk about this crazy idea that, I mean, I don't know how many people know about pair programming. You talk about it on Twitter all the time. You've written articles about it and you've done talks about it. So what is pair programming and why do you think it's so good?

Speaker 1 00:01:51 So pair programming came about because code reviews are really good as a learning tool and a teaching tool. What would happen if we really turned up the volume on that? So code reviews are something you might do once a month. And if you find them valuable, you have them every two weeks. If you find them more valuable, you do it every week. And then eventually someone said, what if we just wrote the code together? And so pair programming is this idea of having two developers on one computer. So you have two keyboards, two mice, two monitors, but one computer. So actually only one person can type at a time. And what they're doing is, they're writing software together. There's a driver and a navigator. The person whose hands are on the keyboard at the moment is the driver. The navigator is keeping the driver on track, asking questions like, "Why are we doing things this way? "Don't forget. We have to think about this later." "Don't use that variable name because it's not descriptive." And then if the driver gets stuck, the driver can swap, and the navigator becomes the driver. And so what happens is you go back and forth typing and not typing, thinking and talking to build software. It is very much like being in an airplane cockpit because the copilot could say, "Hey, wait a second. I'm seeing something weird on this gauge." And the pilot could say, "okay, take over while I look at that." And that's exactly the and forth you want to have in software. And what we see is again, higher throughput and learning when those engineers are going back and forth, talking things out.

Speaker 0 00:03:07 And I think the other sort of great benefit is just simpler, cleaner code.

Speaker 1 00:03:14 Yeah. What happens in pair programming is that two heads are better than one, because two people are writing it. You get less hacky code; you get more descriptive and more understandable code. You actually get less lines of code because you're actually working together to be like, “what's the right way to build this?”

Speaker 0 00:03:29 That's awesome. And where did you start really using this? How did you build up support and then begin to experiment with it in a significant way?

Speaker 1 00:03:38 It was 2009. When I started at extreme labs in Toronto, I saw what they were able to build using pair programming. I also paired with an intern to build a pretty complicated financial trading system. And I was like, wow, I've never learned anything this fast in my life. And that's when I got really hooked on pair program.

Speaker 0 00:03:55 That's awesome. And then you've rolled it out to pretty large organizations. So how do you use it today in the org that you run?

Speaker 1 00:04:02 Yeah, sure. So at extreme labs, I mean, we hired, uh, like a thousand people in four years and we exclusively did pair programming. You know, when you started a company and they give you like a laptop, we just had desktops and we would only give one desktop to every two people. So there wasn't a choice. Like you didn't have your own computer, you couldn't. So you couldn't even in our job descriptions, they would say pair programming. So people came into the interview and said, “Oh, I'm not interested in pair programming.” I'm like, no problem. See you later, like nice to meet you. But we filtered for that on entrance. We had about at our max, I think about 120, 130 pairs. It was quite a significant team that was a hundred percent pair programming.

Speaker 0 00:04:39 And do you have any metrics that prove that this is the right approach?

Speaker 1 00:04:44 Yeah. There's two ways to think about it. The research shows that it's about a 15% overhead to do pair programming -- much smaller than you would imagine, right? With two people working on the same thing, but for that 15%, you get better architecture, better knowledge sharing, happier engineers, um, less bugs, higher quality, faster time to market. Like all those things come along with that. If I went to any manager and said, I can give you all these things for 15%, they would all take it. The other thing we did was just by looking at practice. So extreme labs, we were really focused on building mobile apps. So we had worked with many companies from Uber and Slack to Facebook and Google and Twitter. And one thing that we did keep track of was how often are we building things faster with higher quality, right. As measured by the end-users. And the answer was 100% of the time -- every client we worked with was shocked by how fast and how high quality we were able to build. I would say a more junior team than typically you would imagine, like we had lots of new grads. We have a big intern program. Of course we had senior engineers as well, but it was quite dramatic how much faster we could build using peer programs.

Speaker 0 00:05:53 I have so many questions. I would imagine a junior person being paired with a more senior person working really well because senior person has all this knowledge, but also it could be kind of a drag on the senior person. Cause they're like, my guy doesn't know anything, and you know, this sucks. If I could just be by myself, I would go so much faster. Why do people that are paired, loved this experience?

Speaker 1 00:06:15 Yeah. So you're hitting on something super interesting. Like I mentioned, we hired a lot of people. The number one reason folks were applying to us and wanting to work with us was because of pair programming. So that was a plus across the board, a senior person being paired with somebody competent and smart, but not yet having the experience is actually a wonderful way to learn as well because the senior person has to be the teacher at the beginning of a, an internship, right? We'd hired many interns, like 65 a term, the first two weeks were always draining for the team because the engineers probably didn't know the programming language, didn't know the mobile stack and they always complained, oh my God, I'm going to go so slow. By the end of the term, a hundred percent of the engineers would say, this project would not have happened if it wasn't for this person. And so they were reluctant to let them go back to school. You'd quickly ramp up and quickly become productive, which was one of the main selling points,

Speaker 0 00:07:09 Sort of newbie people obviously are getting like great benefit and the more senior people report that they like it as well.

Speaker 1 00:07:17 A few things. One. They like having another person around to even bounce ideas off of it because they are competent, right? They do have like computer science fundamentals. They may not know the stack you're building or how to work with a client. And so that back and forth is super useful. And what the seniors report is that the junior person does over time become very quickly, quite competent. They don't think that they could actually complete what they're building without the junior person there. And don't forget that junior person can become a senior person quite quickly using this paradigm.

Speaker 0 00:07:48 Yeah. It must be a very common objection from the budget owner of like, wait two heads, one computer, such a waste for someone who's in an IT organization. And, and the other side is like, Hey, I want to experiment with this. What are two or three things that people are going to say no to? And how would they handle those objections?

Speaker 1 00:08:05 Sure. So you're right. The first one seems to be the main one, which is like, how can I have two people on one computer? It must be less efficient. And that's only true if you think the bottleneck to writing great code is like hands on keyboard. Actually, there was a very funny tweet that said, if we have two people on one computer won't they produce half as much code. And the reply is, oh no, no, we hope they produce even less than that. Right? The thinking being that it's not the number of lines of code or a number of characters, it's actually that you want to produce the right code. And so in many cases, pair programming produces even less code than half the code, but it's the right code. Yeah. I would say the other objections are that people feel like the freedom of having your own computer, that how draining it is to pair program.

Speaker 1 00:08:50 Those tend to be the other objections. And so what I say in those cases is if you feel like pairing all the time, eight hours a day for 40 hours a week, you can use it more as a tool in your tool belt, right? So at Shopify, for example, we don't advocate pairing 40 hours a week, you know, eight hours a day. Instead we make it super easy to pair. When we were not in COVID we had pair programming rooms. Today, we have a tool called Tuple. We used to do remote pairing, which is really good. And it's a tool you can use when it makes sense for you. It might be scheduled, like you might say, "Hey, I have twice a week. I'm going to pair." It might be when you run into a problem, but we want to reduce the friction to pairing. And so that's another avenue to try it in an organization.

Speaker 0 00:09:29 Well, I would imagine pair programming 40 hours a week would probably increase the productivity of your organization because you couldn't jack around. You couldn't be like, "Oh, I'm going to cruise over to Amazon and shop for two seconds."

Speaker 1 00:09:43 Right. Like imagine, you know, even just you and I, DR. If you and I sat on a computer for 40 hours a week working on whatever, you know, whether you were pairing with me on Shopify stuff, or I was pairing with you on whatever you're working on now. But yeah, well, well, we would, we would be learning so much from each other. Right. The learning rate would be super high. Everything from like, what keyboard shortcuts do you use and what tools do you use.

Speaker 0 00:10:01 That can be draining and exhausting 40 hours a week./

Speaker 1 00:10:03 Yeah, it is. But what we found is over two or three weeks, you get used to it. Then it sounds funny to work intensely for eight hours a day, but that's literally all you need to actually get significant momentum in engineering.

Speaker 0 00:10:18 So, you know, with COVID, Shopify has announced that Shopify is going a hundred percent remote, I think forever.

Speaker 1 00:10:26 Yep. Right now we're basically leaning into the uncertainty. We're like, cool.

Speaker 0 00:10:30 So now in a remote organization, how has that impacted pair programming and how have you sort of, you know, reacted to the new situation?

Speaker 1 00:10:40 Yeah. So in some ways it's almost easier to track how many people paired, right? Because before, if you went into a pair programming room, it was hard to know

how often Shopify were pairing. Now, because we use Tuple at an aggregate level, it's much easier for me to see, “Hey, this many hours was spent using the tool.” So we do see, you know, over a thousand engineers using a pair programming tool every week.
Speaker 0   00:11:04   I wanted to tell a story here. When I was CEO of Optiva, we made a decision to pivot pretty hard for the public cloud. We had pretty much close to zero cloud capabilities. I knew I was going to have to do significant hiring of a completely different skillset and also train some of the people to learn cloud skills. Farhan was on the board at Optiva. And I talked to him about pair programming. And I was like, I think this is something that we could really use. We hired probably 200 engineers. Now, I was running a hundred percent remote organization. Within six months we had super productive engineers impacting a 20-year-old product and starting to move it to the public cloud. And it sounds insane, but it, it totally worked. It's a tool in the tool kit.

Speaker 1   00:11: It's hard to know when to deploy it. This happens to me all the time, even at Shopify where someone's like, we're writing up a product brief, or we're writing a very important email to the whole company. I'll just say, Hey, can we just jump on a meet and like pair on it together and there's value in doing stuff async but I find that there's also times when you should deploy sync, like you should actually say, Hey, can we do this at 11 o'clock and let's pair on the email together and you'll see how much collaboration and learning happens during those sync versus just async.

Speaker 0   00:12:24   Yeah, no, it's funny because we started to say pair programming, but we were doing it in places like support or managed services, right? It just became PP. For everything where we needed to train people in a really short period of time by putting them together and then working on the same doc. And there's all these great collaboration tools like Google docs. And it doesn't matter what time zone or location, as long as the person has access to the internet.

Speaker 1   00:12:49   Right. It's called pair programming, but there's so many more domains at extreme labs. I also ran finance. And once a month I would sit down with our head of finance and we would pair on doing the invoices for the month and guess what? She would find errors in my calculations. I would find errors in her account, like while we would do it together.

Speaker 0   00:13:05   That's awesome. One thing I think I read on Twitter is you're now extending it into the recruiting process and using it in interviews. And so I used to run recruiting at Trilogy. Trilogy was probably pretty famous for doing really awesome recruiting. And we used to do these grueling eight interviews, brain teasers code on a whiteboard. We were kind of mean, but what I've learned is the best way to really know if they're going to be a match in your organization and they're capable of doing the job is to give them actual real work. You've started to do this with your interviews. And so can you talk about how you're doing that and how to candidates react to that?

Speaker 1   00:13:44   Sure. So, I mean, I'll start with the caveat of like, interviews are not that great or predictive of performance, right? We know this data and there's research that shows this. And so what we did at extreme, we tried to have a very, very light filter on the way in, and then we used 30, 60 or 90 days of pair programming data, working with somebody 40 hours a week, as a better higher accuracy filter than the interview process. So that's one way to do it. What we do at Shopify is a little bit different in terms of engineering and pair programming. We have a pair programming interview and there's kind of two ways to do it. One is you have a driver and a navigator. The driver is the person who's hands on keyboard. And in our current process, we have the candidate drive and the interviewer navigating -- what do you think we should do next?

Speaker 1   00:14:29   And how about this and how do we solve this problem? And the candidate is typing, but the other way to do it, this is what I learned from pivotal, there is an interview style called the RPI, which stands for Rob pairing interview. He was the CEO of pivotal labs and CEO of pivotal. The candidate is the navigator. So the candidate doesn't even have a keyboard. When we used to do this in person, the candidate had a monitor, but no mouse or keyboard, and they would navigate. It tries to capture the empathy and pushback from a candidate about different ideas around programming. So what would happen is the interviewer is doing all the typing. So the interviewer might sit on, you know, sit their cursor on the line and say, “Hey, we should build a test for this. How should we build the test?” The candidate directs them, but you know, kind of like typing through someone else's hands, you are removing some of the anxiety and syntax of the actual coding part of the interview, and really just getting the ideas and letting the candidate think versus worry about like where the semi-colon should go.

Speaker 1   00:15:27   And so this process I've been experimenting with it at Shopify. And it's quite interesting because you get a different profile from the candidate versus the anxiety inducing “ code in front of me and type” interview.

Speaker 0   00:15:40   And then you said something interesting there where you're like the job interview is like 30, 60, 90 days. And I think this is something that a lot of organizations don't do, it's kind of like, you know, getting married after the first date, you put out an offer to someone and then you sort of feel like you need this, like stay married, you know, in air quotes, stay with this person for like a year. And I think this is a mistake that organizations make all the time where they realize they have a bad hire and they don't do anything about it. And I think Zappos really experimented with this pretty aggressively back in the early two thousands, where they would send you through a bootcamp and they would offer you severance if after six weeks, if you're not into it. It's cheaper for us. If we just give you this severance package and you bail, then staying here for a year and you guys actually do that right. 30, 60 days, you know, 90 days in, it's not working out, are you guys saying goodbye to those people and going to find some new candidates?

Speaker 1   00:16:35   So at Extreme labs, we did it as well. Once you know somebody is not a fit for your organization, it's actually better, not only for the organization, but it's better for the person. In the example I give, I think it's like, Hey, you want to bench press 400 pounds. You don't put somebody in front of a 400 pounds and then say, Hey, let's just try to bench this. And if they can't, you don't just let them keep trying forever. Right? Instead, what you should probably do is say, “Hey, maybe she started like 60 pounds and then 90 pounds.” And over time, maybe I'll never be able to bend 400 pounds. The idea is that people feel like it's just a, not a good experience for the candidate to let them go. And I disagree. I think it's better for both parties to, for people to land in the best spot for them. And I think that is overlooked. And the other thing is that people feel like once you pass the interview, you've got it made and you're there forever. And I don't think that's true at all.

Speaker 0 00:17:26 I totally agree with that. I am pretty aggressive when someone in my organization doesn't look like a fit. On our last podcast, Jim Abolt had this great framework, which is Mr. Spock and Mother Teresa. You need to be super objective and dispassionate about making that decision. If someone's not the right person for the role, if you make the decision to exit them, then that's when you need to have empathy and be Mother Theresa - what's the best way to exit this person? And usually the person also feels that they're not a fit and just eliminated all that stress instead of tiptoeing around this topic.

Speaker 1 00:17:58 I love that analogy, Mr. Spock and Mother Teresa, what a great way to frame it.

Speaker 0 00:18:01 I think you should go listen to the podcast Episode 7.

Speaker 1 00:18:04 You're totally right. And when you're giving somebody that feedback, what you want to hear is “I'm not happy,” but I'm not surprised people realize they're like, you know what, you're right. I'm not a fit, but of course you can be empathetic with severance, help them move, help them find the next thing. Maybe it's a reference letter, whatever it is to help them help me.

Speaker 0 00:18:22 I work with managers all the time where they don't want to make the call because they're worried about having the conversation. And so I'm always like separating these two things. Be dispassionately objective. But then when it's time to actually communicate, show a lot of empathy. And so I'm like, why don't we just like pay them the rest of the year? It's the same money to the company. And they were like, no, that's crazy. I'm like, okay. Right. Like how about less money? And we usually get to a place where the manager's like, okay, this is actually feels really good. This feels like the right answer for the company. And it feels like I'm treating the person well, and it ends up being a better thing. So, um, do you have any advice? Let's say our telco listeners, our exact listeners are like, this sounds like a great idea. How would you get started using pair programming? Are there like books to read or talks to watch or, or like, do you start with a small team and kind of grow?

Speaker 1 00:19:16 Yeah. So I think there's, there's lots of material online. There's a book called pair programming illuminated, which is quite good. Definitely start small. You find a few internal people who are willing to try it and you get them set up correctly, whether it's remote or in person with the right desktop and monitor and most setup. Um, and let those folks pair program, maybe it's four hours a day, maybe it's eight hours a day for like a few months. And while it may feel unnatural or look dumb, I think that it's important to try it for at least a significant amount of time that you have to get into a rhythm in terms of when you should interrupt somebody. And do you want to catch somebody’s typos or do you want to let them fix it themselves? There's an etiquette to pair programming.

Speaker 1   00:19:56   There's a great analogy around basketball. And I think Malcolm Gladwell did a podcast on this, that the underhanded free throw. It has a higher success rate in basketball than overhand, but it looks dumb. And there's actually like a great article, which Shaquille O’Neil, who is, you know, a hall of famer basketball player, but notoriously bad at free throws. And somebody said to him, “Hey, Shaquille, like, do you know that underhanded free throws have a higher percentage of landing?” And he goes, “I'm never doing that. It looks stupid.” Even though he was paid to make baskets, but he wouldn't do it. And it's because it looked dumb. And so the thing to figure out is, would you rather have higher learning, better design, lower bugs, higher quality software, if it looks dumb, or just not look dumb and do all the other things.

Speaker 0   00:20:39   So many people won't do things cause they, it looks dumb.

Speaker 1   00:20:42   Right? Correct. This is why my wife always cringes. Whenever we go for dinner, because if I overhear an interesting conversation, I will walk over to them and ask them what they do and try to hire them. But again, it looks dumb, right? You're like, why are you going over to that table?

Speaker 0   00:20:56   And that's how you find great people, right? Just go up to them and talk to them. I agree. Um, you were saying, you know, do it for a period of time. What do you think the low end of giving it a try? Is it like six months? Is it three months?

Speaker 1   00:21:08   Three months is a great amount of time to learn about the process and actually have a few pairs delivering software in high quality, learning time. And it would be enough time for other teams to look at them and say, wow, how are they able to deliver so quickly? I would say the bare minimum is probably closer to like four to six weeks, but I really feel like three months is an amazing experiment that you could write up, you can have metrics against and really feel like there's something here or not for your organization.

Speaker 0   00:21:37   That's awesome. Well, I think this was really great. I really appreciate you coming onto the podcast and talking to us about pair programming. I learned a lot. Stick around because we're ending each podcast with a Telco in 20 takeaway. I have 20 seconds to tell you something you need to know .

Speaker 0   00:21:56   When you're the crazy cloud girl in telco, you're kind of just asking for a fight and I'm bringing it. Join me at Telecom TVs’ The Great Telco Debate happening through December 10, I'm going to bring my crazy public cloud self to convince even more people that it's time for telcos to embrace the public cloud. I want to talk about the BFCs. Are they friend or foe? My take? They should totally be more than friends. They should be your BFFs. I'm also psyched to talk to Ray Le Maistre, editorial director of telecom TV. Someone buy that man a great cup of coffee.

A quick note, keep your ears out for a special year end podcast during the last week of December.

I want to give a huge thanks to Farhan and thank you to all of our listeners. Don't forget to hit that subscribe button, share our podcast with your colleagues. And if you liked what you heard, leave us a review. Let's connect on LinkedIn and on Twitter @telcodr. And sign up for our email newsletter at telcodr.com that's T E L C O D R.com.

Later, Nerds!