[00:52] PARTICIPANT 3:

Researcher?

[00:53] RESEARCHER:

Good morning. How are you?

[00:55] PARTICIPANT 3:

Hi, good morning to you. I'm sorry I'm late when I joined the call.

[01:02] RESEARCHER:

No, it's fine. Let's do 45 minutes and see how we go.

[01:11] PARTICIPANT 3:

Do you want to do another 20 minutes after your call or something?

[01:16] RESEARCHER:

No, it should be fine. We'll start with the interview and I think we should be fine. We should be able to go through all the questions. How are you this morning?

[01:27] AMANADA

I'm good. Thank you. I've just been for a walk and it turned out to be longer than I expected.

[01:32] RESEARCHER:

No, it's OK. I hope you enjoyed it. Can you start with introducing yourself? Talk to us a little bit about your experience.

[01:43] PARTICIPANT 3

Yes. So, hi, my name is [redacted] and I'm from Singapore, but I used to work for [redacted] for five years. Then I quit two months ago. So, in my role in [redacted], I spent a lot of time doing different things. In my last role, I spent three years in San Francisco being a product manager. So, in my product manager role, I had experience in [redacted]...well, we code in C++ type script and so on. And we use Agile engineering practices. So, the team that I was with was in [redacted] and Artificial Intelligence and I worked on a tool called Visual Studio App Center to help mobile developers build mobile apps. So, there I was the product manager in charge of shipping the new feature where I led about thirty principal and senior software engineers. And the thing we did was whenever we had an idea, we tested it with design. And then we talk to engineering and we just build an MVP. Once we got that out, we shipped every day at noon in Pacific Time.

And once we shipped every day, we hit the pilot meeting, we interview customers and then we got their feedback very rapidly using Agile methodology.

[03:00] RESEARCHER:

So, you were releasing every day?

[03:04] PARTICIPANT 3:

Yes, we shipped our environment in the morning. Everyone has to ship by a certain time to staging and every day at noon we shipped. Our practice was not common in [redacted] because, Microsoft Windows Server, SQL Server, it takes a long time, maybe in the past a year on premise. But our team was newer a team and we were in AI. So, [redacted] was pilot testing this methodology of shipping every day with some teams. And my team was one of the first to be put on this trial run. So, we shipped every day for about two years.

[03:45] RESEARCHER:

Ok. How was it? Did it create pressure on the team?

[03:49] PARTICIPANT 3

Initially, when we first started, but by the time I left, it was very easy. But when it first started, it was hard for the more senior engineers who had been at [redacted] for 20 years. They were steeped in the previous generation. It was harder for them to understand. But the nature of our team, the culture was very innovative because what happened was, they bought several startups like [redacted], and all these got founded together and that was my team. That's why they wanted to test this new culture in the team. At the beginning, it was pretty haphazard. There weren't a lot of engineering practices. So, it was true, like for about six months, a trial run before we finally saw results. When you go to staging, when you want to ship, just make sure all your code's commented, everyone does their review and then we can ship it by a certain time. And I think once the engineers, they were also pretty excited, like I think after two months to see every day their code was shipping, they were all very motivated to comment. So as a product manager, I had to review some of the code and the UI. And I think after a while, I saw a lot more what shifted and that was everyone was writing more documentation. Basically, they were documenting it more.

[04:24] RESEARCHER:

Before we dive onto the subject, Let’s define quality? How do you define quality in the context of agile software development?

[04:35] PARTICIPANT 3:

Good question! From my experience, software quality is subjective, but we always know it when we see it. An attempt to define it would be three attributes: code quality or internal quality, external quality and of course the process quality. Code quality is difficult to define, but in my experience it can be enforced by having standards and guidelines in place plus a peer-review

process for code. The opinion of 2 or 3 reviewers on the quality of the code is always better than the single judgement of the author of the code. I think external quality is a product free of defects and It's the ability to fulfil and meet the requirements of the end user. Obviously, these two attributes do not happen without a robust software development process. Agile is keen in continuous experimentation, learning and improvements. The process is always subject to improvements.

[05:14] RESEARCHER:

Fantastic. Let's get some a little bit onto the subject. This is an opinion thing. What do you think of Agile in general?

[05:25] PARTICIPANT 3:

I really like it. I hadn't worked with Waterfall model, but I read about it when I was doing Agile. Because part of my role as a project or Scrum master manager. I haven't had experience at Waterfall, but I find that with the Waterfall method you are spending so much time at the beginning, planning and everything. And even after a year or six months, like a long time after that shipping, I felt that it wasn't a good way to get that customer feedback. I think a lot of times you have an idea, you have a good engineering idea of that problem to solve. But you can only test its durability once you actually ship it out and see how users by using it. Stress testing it. There's no amount of artificial test stress or QA they can do that can actually mimic real life scenarios. So, I really like Agile where you want to build a feature, you put all the parts, design ideas, engineering methods and then shipping and then rapid development. And I think that should be, in my opinion, the future where companies should go. We have to start being more Agile and start working on getting that feedback from our customers.

[06:40] RESEARCHER:

Does it work for everybody?

[06:42] PARTICIPANT 3:

In my experience in [redacted], I think in my development, I think if a big company like [redacted] can do it and, in the beginning, it's going to be hard. Like with any culture changes, with any shipping changes. It's definitely going to be hard. There has been a lot of resistance. I think two years ago, quite a number of people quit, or they moved to another division because they couldn't adapt to the culture. But I think, if you know that the future is going, for example, if you know that the future is going towards like computers, you would not invest in a typewriter because you know where the future is going, you want to spend time and energy. So, I think [redacted] two years ago, the new CEO, [redacted], has been in there for five years. So, I think starting two years ago, they initiated a big change, One Engineering System in [redacted]. So what happened was, it's a bit of Microsoft history, but being a big company, [redacted], each division, Office, Windows or Skype, or even our division, [redacted], I'm in the developer team so my division is [redacted].

[07:55] PARTICIPANT 3:

Everyone has their own coding. Anyone can choose anything that he wants to code in any practices that he wants. So, I think starting about two or three years ago, [redacted] on this new team is an internal team, it's more confidential, but it's called [redacted], where they tried to unify everyone on the same processes. So, let's say, you want to release Windows the same time as SQL, and using the same format, everyone could work more collaboratively. So in the beginning, three years ago, it was a lot of pain and there was a lot of change, but I think if a big company like [redacted] can do it, if you look at we just released the annual earnings, I think two days ago, we actually had been making even in COVID-19, we were real profitable this quarter. So, I think it speaks for itself and a tech company is only as strong as his developers and as good for customers.

[08:45] RESEARCHER:

I do have follow up questions. You mentioned something called Agile culture. What do you mean by that? Can you describe it to me?

[08:54] PARTICIPANT 3:

Agile culture, yes. In my division, since we were one of the teams who started with the big shift, I think the managers got everyone to change their thinking that it doesn't have to be perfect before you launch it. Of course, you don't ship something that has a lot of privacy loopholes, or a lot of bugs, but it doesn't have to be 100 percent perfect before you ship it. It can be maybe 90 percent or as well as it can be to test the idea and then you ship it. And then from there, of course, [redacted] wants you to do a lot of this management before you ship things. If it's 90 percent once, you can ship it. It's okay because then you can tell customers it's in beta. We want to learn from you, communicate that it's not fully fleshed out yet. We want to learn from you and test from there. So, another example is, and I think it's more like a startup mindset. Meaning don't wait for so long before you ship things. So, I think that kind of culture really got people to be more collaborative. They got people to be less afraid of failure. Because if you had a lot of psychological fear of failure, then you can't get true good ideas out there. But if you're like, oh, the only way to ship successful products is if you throw it out there and then you learn from your failures. So, this means shipping and getting it into the hands of the customer and actually seeing how they use it.

[09:55] RESEARCHER:

I have a follow up question. When you eliminate the fear of failure, then how this quality helps the team to achieve better software quality?

[10:02] PARTICIPANT 3:

Of course it does! For example, developers can challenge each other on the quality of the code part of the code review without the fear of peer reprisal. A developer can say to another developer, there is a better way to code this without fear. Another example, I saw a whole team taking initiatives to deliver better quality. When there is no fear, developers innovate and invest time to deliver better quality.

[10:23] RESEARCHER:

That's an interesting thing you mentioned is the fear of failure. How does this process eliminate the fear of failure?

[10:33] PARTICIPANT 3:

I think it doesn't eliminate it, but it means you more exposed to it because you ship every day and for example, there's a saying, you put a frog in the pot. If the pot is boiling, the frog will feel it. But if put it in and gradually increase the temperature, he doesn't feel the heat. All I'm saying is that if you keep exposing people to shipping every day, not failing, I don't see it as failing. I see it as getting feedback every day. Getting more feedback and treating it to be more data-driven. I think that is the real benefit of what Agile brings because you don't get customer data unless you ship it.

[11:30] RESEARCHER:

So, you get this fast loop of feedback and you learn from it. And the learning is what incites people to keep going?

[11:42] PARTICIPANT 3:

Right. Because you can only succeed if you keep doing it and you keep getting feedback.

[11:51] RESEARCHER:

Okay, fantastic. Let's move to a more detailed question. Can you describe the Scrum environment in your last job? [redacted], for example, you could use any other example, if you like.

[12:03] PARTICIPANT 3:

The Scrum environment.

[12:07] RESEARCHER:

What I mean by that, is the team, the process, who is working in the team, how the process looks like, etc.

[12:17] PARTICIPANT 3

So, I will start on all divisions. Are you only interested in how engineering works? Are you interested from how product design and engineering works?

[12:27] RESEARCHER:

Both, yes.

[12:30] PARTICIPANT 3

Ok. So, I put everything together. So, what happens is that it depends on what stage of the product it's in. In our stage, we just launched for two years already. So, we were not fully well fleshed like [redacted]. But it was like within two years it was still early phases. So, the process was that whenever the PM had an idea and we wanted to test something you would discuss with design. And immediately within one week, design would get back to us with new designs and we will work with the engineers to build it. While all this is happening, everything is shipping already. But you work with engineering and engineering takes about a week or two to build it. Of course, before you get engineering to work on it, you have to make sure that your leadership approves it. So, every week, once a week on Wednesdays, we had this board called the Change Advisory Board. So, I don't know if you read the book called The Unicorn, by Gene Kim. It's called the Unicorn Project. Basically, it's one of the innovative books for developer tools, the DevOps process, the Agile methodology. The division size was about five hundred people. So, all the engineers were working on different things. So, we had one backlog for all the five hundred engineers working on the same product so that everyone could see what everyone was working on. We had priority slots, so at the beginning of every quarter, or every year, the leadership team would meet and say this is our priority for this year. Say, this year we want to get the best developer tool and we want to shut down [redacted]

[14:19] PARTICIPANT 3:

In order to do that, they assign points per engineering team. We had five teams of engineers and so this team can work on five story points. The next team can work on the ten story points and so on. So, let's say we have thirty points to allocate. Out of these thirty points, five points should be shutting down [redacted]. Ten points should be building GitHub, the next five points should be on GPR and the other five points should be like a surprise bucket. Unexpected things. So, every week when we meet, the stories that we bring, the things we want to work on has to fall in those priorities. So, at the beginning of the year, the leadership sets the direction, like anything that you want to do has to be in that direction. It has to be either helping GitHub, shutting down Visual Studio. If it's not, you can't bring it up because you will get rejected. So, every Wednesday is our chance to bring these items up after you've worked with design. So, in this one hour, all the product managers and engineers can pitch to the leadership team and usually the product managers represent, and usually we argue with each other. A healthy debate. So, a lot of times the PM's will sit there and justify, it's like a court hearing. You have to propose your idea, a factory idea in order for it to be implemented. So once managers choose their list of things to be done, the next day they will prioritize those things. And then after that, once they prioritize those things to be done, they give it off to the engineering teams. The engineering teams takes about two weeks or one week to work on it. And then after they done, they ship it. Is this what you wanted?

[16:12] RESEARCHER:

Yes. Just listening. I've noticed there is a lot of tight collaboration and transparency in the process. Can you explain to me how transparency and collaboration helps achieve quality in this process?

[16:31] PARTICIPANT 3:

You know, I think it definitely helps a lot because it helps to make everyone feel psychological about these things. It's all very professional and our team is one of the higher functioning performing teams in [redacted]. That's why I stayed for five years. But basically, for example, if I fight for my idea, somehow, I'll back down because I know that it's the best thing that they want to do is better for the organization. So, I think the leadership did very well in telling us that, you know, it's all together as a team and having a transparent backlog where everyone could see what everyone was working on was also really good towards achieving it. Because if things weren't transparent, if you were working on your thing and I was working my thing, you don't know what everyone is working on. And when it comes to promotion season or review season, you don't know if your work has been impactful. But when everything is so transparent between design... basically design, engineering and product has the same backlog. So, you go through design tasks, as you go through engineering tasks...because everyone is so aligned and all the leadership teams, all the PMs and all the designers and engineers attend these meetings.

[17:44] PARTICIPANT 3:

It was very democratic. At the end, leadership decides, but the product managers and engineers and designers at least felt that they could influence to a certain degree. It gives the illusion that we can influence things but actually they decide everything. But at least it gives us more empowerment. So, I found that initially, in the first six months, when we implemented this, people were not happy. They were like, why are we wasting time? You don't trust us to make our own decisions. I think after that, people felt more empowered in agile or the illusion of empowerment. People got more productive. In agile, I personally felt more empowered and more passionate about my job. Seeing that at least I know how decisions are being made. In the past, it was decisions were being made but I didn't know how the leadership was doing it. At least you can put in some inputs, and I think that makes the difference in the way we work together. I found my team very collaborative. It was a great team. Most of [redacted] is not like that.

[18:47] RESEARCHER:

You mentioned a very good concept, which is empowerment. Does empowerment make people more efficient in this Scrum setup you talking about?

[18:56] PARTICIPANT 3:

I don't think they are correlated because you can be empowered but if you're not a really efficient worker, you will use empowerment at the wrong time to do something. But I think just because in my team or the nature of those who [redacted] has hired, people are generally a Type A. They are very driven, very ambitious and very efficient. So, I feel like in the past they were efficient, but because it was not transparent, they were not empowered. So, they didn't want to do a really good job. Or people see [redacted] not really as a place to contribute, but to sit back

and do the least work, to get the most amount of money. Big companies. And generally, I would say is most of [redacted] That has been my experience in parts of [redacted], that I worked in different divisions and I felt that. But I felt like in this group because they meet us, the product owners, every one of the product, it generally created a better sense of empowerment and empowerment given to talented people, and transparency and everything, I think it can be a very powerful force to the potential of every employee working for the team and organization

[20:14] RESEARCHER:

I've sensed from your answer that empowerment plus a high performing team made Agile work better. Am I correct?

[20:25] PARTICIPANT 3:

Yes, but I would say also in a Waterfall model, I would say it's not making Agile better, but I think it's not like the root cause is empowerment and high performing teams is the effect of an Agile environment being better. I would say actually Agile environment creates empowerment. Well, Agile environment lifts an already efficient team. That's the root cause. And then the effect is, creates empowerment, which creates more efficiency. And tapping on the unharnessed potential of the employees. And then therefore it's a better product for the user and in the end drives [redacted] revenue.

[20:30] RESEARCHER:

Then, how this empowerment helps achieving software quality?

[20:35] PARTICIPANT 3:

People become more committed and invest more effort in producing quality work including better code, better design and overall the software quality.

[21:09] RESEARCHER:

You describe the team as talented. Can you elaborate a little bit on that?

[21:25] PARTICIPANT 3:

The people that I worked with at [redacted] or the people that I worked with in my last team, which was the [redacted] team. And I was also living in San Francisco. So maybe that's a very selective pool of people. They were all international. My colleagues were from Palestine, and we work across time zones. She was in Palestine, she lived in San Francisco. And another person was working in Brazil and he was Brazilian. Another person was from Czech Republic. He was my principal engineering partner, the engineering manager. He lived in San Francisco. Other people were from very rural areas in America, like Alabama. But it was so smart and different. And I found that one thing that really connects people in terms of talent, isn't really the country they came from, but it's also education. They were all very highly educated from good universities, even though they came from different backgrounds. So even though you spoke to people of different nationalities and everything, that kind of the same level of education, same

frequency was, I think when I say talent. Talent as in globalized, talent that could work from any time zone, talent that had different cultures, different ideas. So, whenever you gave an answer, where like someone was promoting maybe white supremacy and another person would say oh but have, we considered this other thing in this part of the world that we didn't think of. Because we are building products for the world. I think there was a greater representation across the board. That's what I meant by talented. Globalized, diverse and highly educated.

[22:57] RESEARCHER:

Ok. Let's move to the next question. What do you do to assure software quality in this Scrum environment?

[23:06] PARTICIPANT 3:

For us what happened, the software quality because it was shipping every day, we had a lot of automated tests. So, the automated tests was one way of just like, it was stress testing. So, before we shipped anything, so shipping every day doesn't mean that if you submit your code today, it would definitely get shipped today because your code may run into errors. So, for the first test is all the automated tests. It took us quite long, one or two weeks to set up all the automated tests and test security, to test the UI, to test all the things. So that was the first phase. The second phase was human testing. We had a team in China that would do stress testing. So, they would just go clicking on the buttons and testing everything. That was the second phase and they would file bugs. The third phase was actually a bug bash. Where one hour before we ship any feature, all the engineers who worked on it, the product manager, the designer and the engineering manager who is leading the team, would all come together in a room for one hour and everyone was just testing it and finding bugs. So, we had three phases. First is automated testing. The second was a remote team in China. The third thing was the people who actually brought the product. I don't know if that's such a good idea because we're biased by the people who actually built the product were testing it. So, we had two rounds of human intervention before shipping it.

[24:37] RESEARCHER:

Ok. At what stage was the testing happening? Is it part of the sprint or at the end of the sprint? How did you manage testing or the capacity of testing within the sprint? Can you explain to me?

[24:52] PARTICIPANT 3

Yes. It's the end of the sprint. So basically, when development is done, the automated testing happens every time you submit code to GitHub. Every time you submit approve requests. So, there was an automated testing. The human testing only came at the end of the sprint. So, the sprints could be, depending on the size of the feature, it could be one week. It could be one month or two weeks. So human testing only happened at the end. But automated testing happens every time you submit a PR.

[25:27] RESEARCHER:

What do the testers do while we test? Because this is a capacity management thing. For example, there is a sprint; sprint one that is development and testing at the end. After the

developer fully dedicated to testing by the end, managing the defects, etc., do they move to the next sprint.

[25:53] PARTICIPANT 3:

Let's say we have one team and the team has five people, two people would be working on a feature. So, we had a feature and different user stories. So, a particular feature has three user stories. And there are five people, two of them work on one user story. The other two will work on one user story and the last will work on one user story. If the first pair is done, they would test it immediately. And then if the second pair is done, they would test their own thing immediately. The last pair then tests immediately. Once everything is done and put together. Then all of us will come together and test it.

[26:31] RESEARCHER:

I see. Okay, fantastic. Do you think this Scrum setup produces quality software and how?

[26:44] PARTICIPANT 3:

The answer is yes, I do think Scrum and agile, in general, produces quality software, but I would also say Waterfall can also produce quality software. I don't think there's correlation between quality software and the Agile, Waterfall methodology. Because I think to produce quality software, basically you have to have good engineers writing elegant code or efficient code, having the time to make code efficient and clean and then having a lot of testing to be done on it, whether it's automated or human testing before you ship it. Yeah, that's my honest answer.

[27:36] RESEARCHER:

Do you imply that agile doesn’t make a difference when it comes to achieving software quality?

[28:11] PARTICIPANT 3:

Don’t get me wrong! It does. Yeah. For example, people collaborate more; so, everyone understands the requirements better to achieve a software with fewer bugs. As I told you before, people feel safe and empowered to invest more effort on quality. Agile also very compatible with practice like pair programming.

[28:36] RESEARCHER

What is it? What type of programming?

[28:39] PARTICIPANT 3:

Pair programming.

[28:41] RESEARCHER:

Ah, pair programming. So, you were using pair programming.

[28:47] PARTICIPANT 3:

Some engineers, the most senior ones, they didn't like it. So sometimes they didn't do it. But generally, we encourage pair programming. I think the Agile environment helps in that way because everyone was more transparent. Everyone was less protective over their code. We also had one product backlog, one repository, everyone could see anyone's code or change anyone's code. That also helped with quality. Let's say I code something, and my code has mistakes or whatever, anyone was empowered to comment on my code and help on my code and submit approve request to better it. Of course, the person who wrote it would have to approve it but generally it would be approved.

[29:11] RESEARCHER:

You made reference to transparency. Can you explain how it does help quality?

[28:17] PARTICIPANT 3:

Sure! The code or other artifacts are not hidden or a secret. The opposite, any team member is welcome to scrutinize their quality and as I explained they feel empowered to do so without fear. The result is improved quality. In Scrum, this happens in the daily stand-up. People inform the whole team of their achievements and in retrospectives and reviews; we talk about our weaknesses and we point out areas of improvements.

[29:30] RESEARCHER:

So, you would use pull request based development, right?

[29:33] PARTICIPANT 3

Yes.

[29:35] RESEARCHER:

So, a part of approve request based development is doing code review. Were you doing the code review?

[29:42] PARTICIPANT 3

Yes, I wasn't doing it, but the engineering manager did. Before [redacted], well now it's called [redacted], but [redacted], and everyone had to review each person's code. The person who submitted approve request cannot approve their own code. You need two other approvals from different ... and they have to be managers to approve. And then the person who merged the approve

request cannot be either of those people. So, it's SOC. We had to do some GDPR thing, so we had things to follow.

[30:29] RESEARCHER:

Ok. Fantastic. Can you share with me a positive story about Scrum that has to do with quality?

[30:47] PARTICIPANT 3

Yes. Let me think about it.

[30:57] RESEARCHER:

Okay, take your time. Want the best story.

[31:13] PARTICIPANT 3:

Okay, so this happened once when we were trying to launch a product, a new service called [redacted] in six months' time. So, I think towards the end because there was once an engineer had to do it towards the end, when it was crunch time, it was close to public launch. An engineer, once they shipped the product on Wednesday, they made a PR. I remember towards the end because everyone was so passionate about it. Once he submitted the PR, in the past, he would have asked people to review it. But because we put in this thing called, I think Slack integration, so whenever someone made a PR, you would see a PR in, and all people will be commenting on it. And because of that, the code reviews happened very quickly. Maybe in the past, they would take a day. He would have to ping people, can review this? Can you review this? But in the end, everyone was just so committed to shipping on the dateline that everyone was reviewing it. I think more than five people, a lot of people, so the code review went a lot faster. They gave really great feedback and comments and that enabled that engineer to improve and modify their code. And then they were able to ship it.

[32:40] RESEARCHER:

Fantastic. Obviously not always things goes well in Scrum. I'm sure you have a negative story, would you like to share it with us?

[32:54] PARTICIPANT 3:

Yes. OK, let me think about it.

[32:56] RESEARCHER:

Yeah. Take your time.

[33:05] PARTICIPANT 3:

I think one negative thing, and that is honestly one of the downsides that we tried to mitigate, is that because you ship so much, a lot of things break. So basically, the good thing about Agile is you get a lot of customer feedback, you ship very rapidly, you learn very quickly. But the bad thing is sometimes you don't ship it when it's a hundred percent complete. Sometimes you ship it too early. Most times you ship it ninety percent but sometimes you ship it when it's seventy percent or eighty percent. It depends on the integrity of the product manager and the engineering manager. Sometimes you just have to hit milestones for the leadership. But sometimes you have to be a strong person enough to pull back and say it's not ready to ship yet. So sometimes the people who are leading, like the product manager, he's not strong enough or whatever to acknowledge that. And then things break. And if things break, you have to have other systems in place to find out how fast can you realize that things break and how fast can you pull back. And sometimes I think at the beginning, we didn't handle it very well. So, customers were complaining that, hey, this is not working. We didn't really have good communication channels. So then after that, we set up a state dispatcher that could see our services and everything.

[34:30] PARTICIPANT 3:

I was also an incident manager, an ICM, and I got caught a couple of times over the weekends. A lot of times, especially in the beginning because people were shipping so much. And engineers on a Sunday, they had to be on page, and they had to call the PM to ask what's happening? And we realized that it was because they didn't run enough tests, or it was because something broke and we didn't test it so much. So, I think after we found it out, we start to have more guidelines in place. Only do this, make sure you do this, test everything before you ship, make sure you test in staging, make sure fully done everything. And then these incidents become less and less. But this did happen more frequently. Whereas I would think that in Waterfall because you spend so much time making sure that each part is good before you move onto the next step, things are less prone to breakage. If I think back on my [redacted] history like when did on premises, we ship Windows Server once every two years, once every year, because you had a year long cycle, you had a lot of time to really plan and everything. Whereas if your ship every day, things break. So, I would say that is the biggest negative thing that I can think of. And that leaves a bad impression. And sometimes reputation is very hard to get back.

[35:52] RESEARCHER:

Yeah, true. I would like to go back to some of your statements and do some follow up questions. One of them, you said that transparency and collaboration makes people feel psychologically safe. Can you elaborate a little bit on that?

[36:16] PARTICIPANT 3:

Yeah, I think because when you foster a culture of transparency, where what you say is really what you mean. And also, because in America, in the Western world, it's more direct. So, when people see say stuff, sometimes they do genuinely say very rude things, but they do generally mean it. And when you foster a lot of collaboration, it's also good because you got to understand the person. So, for example, once I was working with a person who had different political views from me, I'm not a Trump supporter. The person was a Trump supporter and he's one of them that I just had to work with. We were all professional and we worked well together,

but there was a project where we had to work so close. And when you work so close with a person all the time, sometimes he says things and I would say, how can you say that women. And then sometimes he said, I'm sure I say things and it's not in line in my view or something. He was Christian and all, it's not to attack him, he was a great person, but I think initially we were working well together, but we were all still very polite. So, there was a divide. Even though things were so transparent but because we had to collaborate so much together, and we had healthy debates to prove our points.

[37:35] PARTICIPANT 3:

Him and I were very aggressive in our points because of that, I think there was a time when he called my manager and said why don't you talk to her. So, my manager talked to me. So, then I just had a talk with him for about two hours, just hashing over things. But we realized that the root cause was because of something else. So, because you collaborate so much together, you are forced to work together, you are supposed to understand that person. And after a while now, now he's one of the best engineering managers I worked with. Just because once we got an understanding of each other, I understand your views, you understand my views but we're not bad people. We are just working together to create something. And we found some common ground. We found that we love Indian music even though he was fifty years old. I'm like twenty-eight and we found different things. And because of that, we were forced to collaborate together, we understand each other more. And it led to greater innovation and to us feeling like psychologically safe.

[38:33] PARTICIPANT 3:

The part of the psychologically safe I wouldn't attribute it to directly to the Agile methodology or a team or like collaboration and transparency. I would also say it's because our leadership team focused a lot on that. One of the values that we had was psychological safety which complement the agile process. and once every three months when we had all those all hands. But because they would keep saying it and because they said it so much, all of us started to sort of practice it. So, whenever we went into these disputes, they taught us that when you have a discussion with someone, don't attack their character. Say that, oh I know you did this, and this is how I felt at the time - was it what you really intended and hash it out. Because we went through all the training and models and [redacted] in general, it's moving towards that culture. Because they did that as a whole, it also helped and the leadership wanted that and also with greater transparency and collaboration, you just have a better and more innovative culture.

[39:35] RESEARCHER:

Fantastic. That brings me to the last question. It is a little bit provocative, but the purpose is not to create controversy but to get to you to talk and give us your opinion. What do you think of this statement: Agile produces poor software?

[40:00] PARTICIPANT 3:

I don't think it's true. I don't think Agile produces poor software. I think Waterfall and any other environment can also produce poor software. I think the software quality is based on the engineering team, the environment, the culture that engineering team is. I think Agile fosters greater transparency, greater learning curve, and greater collaboration and in turn, makes the

engineer or the product team feel that they can see the achievements more rapidly and that helps to foster greater empowerment, ownership, and passion for the job. And it exposes more of the potential of the employee. So, I think if anything, that helps to produce better software and a better product overall rather than poor software.

[40:50] RESEARCHER:

Fantastic. We managed to do it within forty minutes, which is great. Thank you very much. Do you have any questions for me?

[41:08] PARTICIPANT 3:

Yeah. In the two minutes, what is your project about?

[41:11] RESEARCHER:

I'm doing the research in software quality. I'm more interested to understand how Agile produces quality software. Because if we look at the Agile manifesto, it doesn't make a direct call for quality. It makes a direct call for efficiency and excellence. And we may assume that excellence is quality, but sometimes it's not. You can be excellent at what you do, but you may not produce quality. So, we do believe that ideologically Agile has a gap. It has a gap in its guideline about quality. However, in practice, people like yourself, your interview is a good testimony for that, is that people don't ignore quality in their work. The element of quality and Agile itself has some attributes that enable qualities. So, we're bridging that gap in this research. What we're trying to say, we say, yes, the manifesto or the Scrum guide doesn't make direct mention to quality. But this is what people do, they are empowered. This feeling of empowerment gives them what you said or what you described, for example, feeling psychologically safe.

[42:59] RESEARCHER:

The qualities of transparency and collaboration make people feel happy and passionate about what they do, etc. These enablers or these qualities and Agile environments help to achieve quality and Agile teams don't ignore quality. For example, you thoroughly test and thoroughly review codes. I was surprised because it's rare in closed software environment where we see code review. It's fantastic that Microsoft does code review, for example. So, this is my project, it is a research project. So, we try to understand and explore reality and try to understand how the practice works and telling people this is how the practice works. The manifesto doesn't call directly for quality, but this is how people do it. Because I've seen Agile teams produce really poor software. I've seen it really. But like you said yourself, it doesn't have to do with Agile. It has to do with the performance of the team. It has to do with the quality of the team. So, all these elements are very important to achieve quality and irrespective of whether you are in an Agile or whether you are in a Waterfall environment.

[44:55] RESEARCHER:

I'd like to thank you, PARTICIPANT 3, a lot. I have to jump to the next interview. Thank you very much. Please stay in touch. Thank you. Have a good day.