Participant 3

SUMMARY KEYWORDS

people, quality, team, developer, mistake, week, agile coaches, client, kanban, work, feel, feedback, crunch, fantastic, back end developers, meetings, feature, produce, created, scrum

SPEAKERS

Researcher
Participant 3

Researcher 00:18
Good morning. Yeah, I can hear you. Good morning. How are you this morning?

Participant 3 00:26
I need to select the different speaker about it. I think I can hear you.

Researcher 00:35
Oh, hello. Yeah, fantastic. Thank you very much. I think we should start because I do have a lot of things to go through. And before I do that, I'd like to thank you for, for participating in the interview. Thank you very much. And I'm looking forward to the discussion.

Participant 3 00:54
We will see each other again. Yeah. Yeah.

Researcher 00:57
Thank you very much. Yeah. Can we start with a brief introduction of yourself, and just briefly, mainly your education and experience?

Participant 3 01:10
So I'm actually educated in economics. So I'm a self taught developer with around 15 years now. reached the level C levels, also technical being working as a Technical Lead and Solution Architect also at this moment. So a lot of lot of things are doing. And of course, I mainly as a back end developer, know, full stack fairy tales. So because personally, I don't believe in those positions, right. That thing is kind of a market demand for this. But so specialised in the back end cloud architectures, and this mine. Today, I work as an architect in a software house.

Researcher 02:06
Yeah. But you've been most of the time as a software engineer, right?

Participant 3 02:10

Of course, yeah. And I actively participate in the development because the architect role is being created in our company. So it's like, slowly transition to it. So it's not like I became the architect the mind when they created this position. It's just like, we are transitioning into this role more. But mostly, I deal with back in development day to day.

Researcher 02:35
Okay, fantastic. Thank you. What type of Agile method you using in your current team?

Participant 3 02:44
Well, we used to use Scrum framework for this. But we had to switch the Kanban. Okay. And there is an explanation also, in one of the examples that I prepared for the questions you asked in the emails. So it will be explained why and how it happened, and so on.

Researcher 03:04
So, so fantastic. No problem. So what is the size of the team?

Participant 3 03:11
This particular project, it's nine people. Okay. Second project that I only oversee is six people. And they're also team extensions. I try to support team extension is just like one person attached to a scrum team or something like that. I'll just stand alone developer. Okay, so yeah.

Researcher 03:40
So is it a cross functional team? or it's only developers? It's only the word. It is only developers or it is a cross functional team?

Participant 3 03:53
A cross functional front and back in the DevOps, and QA is the default setup. We have been working together for almost 2 years.

Researcher 04:01
Yeah. Fantastic. Thank you. What type of software do you develop in this team? And how much self control you have?

Participant 3 04:07
Well, it's web development, right. But I this particular project is very heavy on the backend itself. So it doesn't have a lot of UI. There's a lot of micro services processing the data. A lot of geospatial data science, the there's some AI also in the background. So kind of a challenging mix of all the things. Our management has little control over how we do things.

Researcher 04:35
Okay, fantastic. We will be talking about quality and the rest of the interview. And it has various definitions, and sometimes it can be controversial. So just to make sure that we have the same understanding of what we mean by software quality. I'm going to read to you a definition that we use,

which is the ISO standard Definition and give you an opportunity whether to agree or disagree or comment on it?

Participant 3 05:06
Yeah, ISO status mostly I am not fond of ISO alright. Well, let's read it. Let's see.

Researcher 05:17
Yeah, that would be interesting to see what do you think of it. So the definition says software quality is the degree to which the system satisfies the standard, the stated and implied needs of its various stakeholders and provides value. The ISO model also cover some non functional characteristics like performance, compatibility, usability, reliability, security, maintainability, and portability. So, all these qualities have to be fulfilled in order for a system or a software to have some level of quality. So, do you agree or do you disagree with this definition? Would you like to comment on it?

Participant 3 06:06
Mixed feelings about this, it's a definition for a good sized company, right. I agree with it but does not happen that way all the time. And that exists in the market, but it goes into the brain when you work for a startup that needs to have a time to market as soon as possible. So, procedures, he says, as always procedures, reporting procedures. That's the opposite of what they wrote in the manifesto. So, ISO is good for Microsoft is good for, you know, bigger companies, but for startups, they will adopt it, I think later, it's not something when you start the company.

Researcher 06:51
So how do you define quality in your team?

Participant 3 06:56
Well, first of all, because I also mix everything with motivation. So if you framework yourself to be stiff, in a given a procedure, or have some kind of boundary, it lower motivation, and all the motivation, equality, anything related to the world quality will go down. So how to say that, instead of having procedures, right, we should set some kind of like direction, like, we want to achieve this test coverage, we want to achieve this kind of flow, we want to achieve this kind of velocity. But it doesn't mean you have to do it. Anything can happen, I can go to announce COVID I can go for a week in the hospital and fight for my life is like, who cares about the velocity, my velocity then, right? So. But that's an extreme example, something can go wrong. And we shouldn't have everyone responsible for every single mistake, but we should report and accept and embrace the mistakes and shrug it off and carry on. Now with the reporting, and feed backing up on all the steps. I never, never was fond of it and always excelled with the team, where we got rid of too many procedures, right? That's why but I never worked Microsoft, oh, big, big company. More than I think 4000 employees, I never seen anything like that.

Researcher 08:34
Even big companies now they have similar approaches, because quality is not always achieved by strict guidelines. Yeah. So you need motivated people behind it in order to achieve it. Yeah. Thanks for that. I'd like to ask you what type of processes I know that you don't focus on processes, you prefer that

people are motivated, but you must or you have some practices in place to assure quality, maybe code review unit testing, and you mentioned test coverage. So what do you have in place in order to achieve some level of quality?

Participant 3 09:19
So like you mentioned, on top of that, the code review, right? And we have a Kanban approach. So we have steps of managing those. The flow of tasks write in a much more loose way so you don't commit specifically like I'm gonna do X amount of effort and then X amount of given time. This is what I never liked about the sprints because it puts a much more stress on specifically on the younger developers, right? It is stressing people, it's hardly holding you hostage in that manner. So we picked Kanban for this to allow also client for the client to change the requirements and keep the standard of the delivery of this one single one single feature in a continuous delivery, I believe, over the time continuous delivery. Sorry, that is getting my countdown, continuous delivery will most probably take over because of the dynamics that it allows to with no quality loss, because you can just stop the feature and tell, okay, I'm doing the other feature, and you just leave it down the brand. If you have too many coupled tasks were between each other in a sprint, it's really hard to stop the development and deliver something we have this thing like time to production in a critical situation. Also, we have a procedure that is good. The patchwork is capable of reaching through the CI and through the development deployment within an hour. I believe that is agile way to be flexible in whatever you do at this minimum quality. So the continuous development and CI is also implemented as a quality measure. And to stay on top of the fix. Now all the stuff that was required also by the management, like keep sending the report about quality semaphore upwards. That's that's isa why because we also have this 27,000 signal. So I need to report all this stuff. So I don't focus on that I don't even bother the team to do that to do reporting, which is I just asked for the project managers and all the non delivery personnel to keep doing that. So another thing to keep quality is to make the developers and the testers collaborate as much as possible between each other and be hassle free from all this reporting. And all this stuff that has been from time to time, you know, imposed on them. So like quality semaphore reporting, like, why, why you needed every single week, nothing changed in the project after a week. But you know, things like that happen and it's in the servals. To report every single week, nothing changes in a week, you can do full three tasks. And that's it. Right. So that's the reality from my perspective.

Researcher 12:50
Okay, fantastic. Thank you, we will get to the to the core aspect of the interview, which is based on the on the question I asked on the email. And from the answers I have, I gather that the work environment is highly safe work environment, what do I mean by safe is the work environment provide a sense of security from repercussions. So people feel it's okay to admit mistakes, people feel it's okay to propose initiative and discuss problem. So there is a sense of confidence that the team will not embarrass, reject or punish someone for speaking up. So this confidence comes from the material respect and trust amongst the team. So from your answer, I felt that is highly safe, it fits all this criteria. Do you agree? Or would you like to comment on that?

Participant 3 13:53
I mean, I do agree. I tried to achieve it as much as I can. Because when I was on a C level, I didn't create a safe enough environment in my previous role. And now I see I can even deliver more from one

person. Of course, it's a double everything is a double edge. So confidence and safety does not produce feel of responsibility so much is so you have to agree on the risks that you want to manage. I'm capable of managing the risks of not feeling responsible and delivering to the staging environment, something that wasn't exactly tested, because I put procedures to counter that. Right. So I allow the developers to have high velocity, lower feeling of responsibility because there is something that we'll double check there is high cold beverages. So yeah, it's just a matter of what I accept to be the risk, which is not perfect to have 100% safe environment. It promotes creativity, but the modes, the feeling of responsibility and, and lowers the stress levels to a bare minimum. But I think there are more profits from it for the team than the cost of those risks.

Researcher 15:35
Studies and the research show the opposite. It shows that the higher level of safety, it produces responsibilities and accountability as well.

Participant 3 15:46
That's good. I mean, I, I Oh, so my fear is I only just precocious not even needed to do that if the research shows that people feel more responsible when they feel more safe. I thought maybe they are less alert, like, ah, let's measure it and see what happens. But I guess I was wrong. So.

Researcher 16:11
So but it is a balance, it is a balance you have every day to become accountable, you need to share the expectations and the responsibility to the whole team. But research shows that the safer the environment is people become more responsible and accountable. We will come to the discussion later on. So But overall, do you feel that your team work environment? Strong? Do you strongly agree it is safe? Or? Or do you agree or disagree or strongly disagree?

Participant 3 16:51
I agree. I'm the shield from the client is a little bit more hostile. But once when he lost, I just I'm just giving an example to once he was even screwing on the meeting, and I took over and I pacify the client. So in a matter that the team witnessed the whole process of pacifying a frustrated patient, but also joined big company with us joining the clients project, right. So he was on a trial period. Now I just call him Oh, again, you turning on paint you Picasso because he likes to draw and paint what he wants to do, instead of using draw, I have, you know, those logic boxes, like an old school guy, a little bit too old school, but I pacify them, and the team was impressed. And now we make jokes with that fine. So So also, you know, the hostilities, they can come from anywhere. It just need to be ready for them. My I just showed a little bit more maturity, because the team is really young, like 10 years younger. And they really, you know, they can do the political meetings, they get defend themselves. So much. So. So

Researcher 18:07
Yeah, fantastic. Thanks for that example. So do you think this, this environment is safe because of your leadership style?

Participant 3 18:20
I don't like giving too much credit to myself, but I can share the feedbacks, also, because you wanted the examples of the feedbacks. And when I read them again, it's like, I need to guess I need to accept some credit for version. So yeah, I think that played at least a significant role in this process. Right. I don't want to take the whole credit.

Researcher 18:44
Okay. That's, that's fantastic. I myself come from a culture that doesn't promote selling yourself or taking credit for yourself all the time. So yeah, I understand that. I never praise myself. I never talk highly about myself.

Participant 3 19:04
Okay, internally, you feel bad.

Researcher 19:07
Yes. You feel it's not a good thing to do. You should expect from others to tell you or give you credit. Yeah. Fantastic. Let's get to the to the question I sent by email. We'll go through them one by one and we'll discuss some examples. Okay. So the first one says, if you make mistakes in your team, it is often held against you. You said no. Usually we assess the velocity versus the plan and we try to pinpoint the root causes of the mistake. Would you like to elaborate on this? How does it work?

Participant 3 19:44
Alright, so I have example was like this month we are deploying, from the dev to staging environment. So we are ready to you know, start demoing what we've done in the past six months and then we are deploying it and the senior backend person admitted to making a mistake of not migrating the data from the client servers. But he was making direct queries to his dummy database. But then I explained to him, it's a backup that we use. So we don't overload the API. So we need to migrate it because you don't, you don't want to, you never want to, from the business perspective, pay a lot of money for an archive. So it's an archive database. That's why it takes like 20 or 30 seconds to do one complicated query. I mean, it's not even complicated because it promotes single simple requests, right for them to create, not create too long chain for it. And as like, and I mentioned that also before to him, like we need to do that, but then he took the risk, like, I guess, it's not needed, right. And he made a decision to not do that. So he was really feeling sorry, when we're deploying and watching how slow it is the first time, and then he just rolled it back. But, you know, I told him, don't worry about this. Nobody held him accountable so much about it. He held himself accountable for it. And I'd say that that's enough. No need to feel shame and feel sorry. I think that's good enough to let him do it. We just wasted one day on this. So he rolled it back in one day. Everyone carried on with their own tasks. And yeah, it's forgiven. Okay.

Researcher 21:45
Yeah, I do have some follow up question. Do you have another example for me? Or you want me to follow up on this example?

Participant 3 21:53
Well, I can have another example.

Researcher 21:57
If you have an example, specific, related to software quality, I'd appreciate.

Participant 3 22:04
Alright. So I don't forget about this. So yeah, the code practices were not, were not implemented fully two months ago by the same person. He admitted to it. And he felt the guilty despite of a slight crunch we had so yeah, he admitted to of not doing the Toko rich for the universe. Also, I allow for this. So I share the responsibilities were crunching. And there was something that we needed a demo on our level, right to demo for the client. And not that them I mentioned the now we're gonna do, which is for high level employees, right? The C level of the client, and the developer is responsible for this. But when we brought it up, it was also caused by me allowing for this and declined putting the pressure. So in the end, I mitigated that everyone felt responsible for this. And we did a feature freeze after this particular crunch was over. So he had time to clean this up. And he was, you know, promised, and he didn't feel so much stress, because we share the responsibility. I told him, we accept this that we are planning that the depth, instead of doing reactive management, I tried to avoid it, because reactive management and code that is just multiplier of each other. So that was the example. So a big crunch and created that situation that not all practices, not Domain Driven Development was done. Also there were cross dependency injections. And the QA was not able to test it all also, and unit tests. So we didn't have if you don't do unit tests, then in your plan of quality sanity tests are not being also covered in the minimum coverage that you want to have. So yeah, that was a period of three weeks. But then, they were given two weeks of feature freeze, do whatever you want to wherever you need and work. So in the end, it resulted in five weeks total work, which we plan anyway. So also the tracking lead, like there's no need for the branch at all.

Researcher 24:38
Okay, so how did that help quality? I mean, do you think it did improve quality because you're talking to him, and he learned how did it help quality?

Participant 3 24:51
You mean, after we fixed it during the feature freeze?

Researcher 24:55
So the problem yeah, so you fixed the problem together. That's how it helped quality because you collaborated together.

Participant 3 25:07
Oh, so Okay, so first, we delivered the feature, right, it was creating more problems because it wasn't that stable. And, and so the quality fixing the improvement of the quality happened during, well, first out the conversation, the client that shared the responsibility and accepted the feature freeze, right? During the feature freeze, we implemented all the missing parts, we continue to the keys matter, and all the

DDD requirements. We also worked in a mono repo, so we try to not duplicate the code and allow the testers to verify the DOD or the finish of already like, Is it ready or not? So when it was after, during the crunch, when it was delivered, it was wasn't marked, finished. It wasn't marked tested, was just usable, or good enough, like your house made feature. Conscious never produced good software, in my opinion. And the client left it pretty quickly. I hope. I said so.

Researcher 26:20
Okay, fantastic. That's a good example, we will move to the next item which says, member of your team can bring up problem and tough issues you said yes. Commonly do it on the retrospective, sometimes daily, which sometimes can build up a lot of issues just for one meeting. I know what you mean. Do you have an example where a team member brought up a problem related to quality in a retrospective and how the team dealt with it?

Participant 3 26:56
Alright, so in relation to last year's three, I think three week crunch, right. And we still worked in scrum till half of November, then I had enough and I had to switch. That was also the reaction to the retrospective that we had. That was the first good decision, I think that was satisfying the client because you have to keep this also in mind. So first one was constantly changing requirements on a daily basis, we declined. I think our point of contact was not an IT is not a product owner of this idea what we do, it's just a point of contact. So he's relaying what is happening in the internal meetings and just, you know, flow stops stubborn. So we switched from this, right, because people raised it that the constantly changing environments made us not follow the DoD, and do while for all the tasks, right. So and what I wrote this was addressed, which later in a form of feature freeze so that's the follow up that I mentioned also before, and the client was informed, right? So constantly changing requirements. It's not healthy, especially in Scrum. It doesn't work in Scrum. It works in continuous development and Kanban doesn't work in Scrum, of course, that constantly daily changing requirements is also you know, not good alone in anything. I mean, it's like working with your wife in the form of a joke, right. But it's something like that. So you can try to improve the way you work to those things, but you're not gonna reach excellence in constantly changing requests. That was one of the things that was raised by will pretty much every developer in the retrospective. And I had to take the side of the customer and take the side of the developers so meet them halfway, when this produced Kanban and continuous delivery. And also as reporting weekly with the project manager to the client, what you can expect to be done next week. And when it's sent, it's sealed. So it we're not doing sprints, but we are somehow improvising on his tendency to change the requirements to like, Okay, you got this email, you cannot show when anything new during this week. The team has continuous development and Kanban but it behind the scenes it's like sprint like approach to declare it. Okay, you have a new idea next week. Sorry for this. Next week. Next week, we can put it in the backlog and people will just take it and the plan is to do it next week, right? So we have like this on Fridays, because I believe for Fridays, we tend to do minimum stuff. So it's nice to have like, a planning session on Fridays, so they feel relaxed. Those meetings can happen. It can be very long. From my experience, so yeah, let's waste time on planning even two hours, because you're not gonna do anything productive on Fridays, especially afternoons. Nobody does anything on Fridays. At 100% productivity, you will I just embrace the reality. Right. So I know some people would love to make you make the thing work 100% On Fridays, but it doesn't happen. It's just Yeah.

Researcher 30:48
Yeah, the motivation drops. Yeah. Do you think that the shift from Scrum to Kanban, and the developers talking about it and bring it up in the retrospective? They did it because they felt safe to do so?

Participant 3 31:07
Of course. You don't have this framework of time. And a framework of tasks and so it's less stressful, at least right? The velocity improved by 50%? In what 30% In December versus November 50% in January versus November. Right. And I hope to keep it at this 50% I don't know if we're gonna work even better in February. But 50% on anything, if it's a discount? By two for one, everyone will take it. I even told the client in a joking way. Stop complaining 50% Everybody takes it. Everybody thinks free free 50% of anything. Unless it's that sort of.

Researcher 32:04
So that's a great achievement. Do you think this adjustment to the process has also helped the quality? Have you noticed any improvement?

Participant 3 32:13
Yeah, it did. First of all, it's like Uncle Bob says I hate that the industry is held hostage by Agile consultants and coaches, I even I will tell someone on the on the meeting in the company. Okay. But please keep in mind, the role of Agile coach is something I disregard as these very negative, but I told him, like, no, never found a useful, Agile coach, if you can, if you want to join us, you need to be useful. Like you're not gonna, you know, be an auditor. Just do something with us. Right? So I'm pretty hostile towards agile coaches, because they always sell they tell you to do everything Oh, it's not Scrum. Can you imagine all the ceremonies? I'm not happy for this right. So, because in the reality I found out that agile trainers I worked with right. So the coach is the active employee and the trainer is the teacher right? That complete different people. Now this teacher he studies and tries to be excellent scram, the coach also, but somehow I found out that those teachers tell me that agile I can implement it, too. The way to reality allows me to write and to make decision which approach is best fit for the team, right? Versus the Agile coaches that always tell me if you don't do all the meetings, and all the ceremonies, you're not agile, it's like, what? I am maybe a hotbed like meeting this type of people, because I met three agile coaches in my life and discriminating the whole industry. I hope, I hope I'm wrong. But yeah.

Researcher 34:12
Yeah, I have the same impression, because most of them have never managed an agiler team. So they tend to tell you what's in the books.

Participant 3 34:29
Yeah, That's actually a big distinction. I am going to review my way of thinking because there must be agile coaches who manage the teams, maybe they are completely different. Or maybe those who teach agile, they manage the teams. That's why those trainers that train, they were more open minded, like, everything is negotiable. Like, you know, like with the guys, I work with the Israeli people, everything is negotiable, according to the culture and Despite of you know, every culture having its flows, I think this is one of the good trades, everything can be negotiable you can, you should be adapt, you should change. So we changed without worried exactly by the framework of Kanban and continuous development, we adjusted it, clients see something in a form of a spring that we plan next week. But we don't commit entirely with all the stress behind it. We just, you know, send him the information that right.

Researcher 35:32
Yeah, fantastic. Thank you, we'll move to the next item, which says, people on your team sometimes reject other for being different. You said, No, I'm fortunate to not have such experience. Yes, what we mean here by rejection is somebody brought a new way of doing things or proposals, or his or her way of the Winton is different. So if I was in your team, or if you have a member in your team who does things differently? So how he would be perceiving the team?

Participant 3 36:14
Oh, this way? So the question is first, brings up a question, does it affect the overall way of how the team works? Right? So that's the first question. If it doesn't, then I just come here, right? I mean, to do it your own way. I mean, for example, I hate working on a Mac, and I'm forced working on a Mac. So, so that gives me empathy to anyone who does differently than I do. Right? And because now I feel it, right. You shouldn't, and I started the topic in the companies like, we have more than 1000 people and, and, okay, you make an operational decision to give a mark to everyone can at least run Linux on it? It's like, you know, trying all windows, I'm not a Mac place. And I don't, you know, disrespect how people will prefer the UX for me, it's like, and, and I feel it. So if the person way of working is not affecting the performance of the team, and it's not affecting the quality, then I don't care, right? You can, you can use WebStorm IntelliJ. All the type of ID all different, you know, test framework, if you can launch it on the code that we wrote, of course, why not. But if it's affecting, right, then I need to hear the arguments behind biggest we should work. I believe. Like if somebody says we want to change from Cyprus, let's say it's a package for testing your website in the browser, right now headless browser, to puppeteer, for example, I indicate the arguments. And I need to allow the majority of the team to switch because for me, it's too low level for me, to just make the decision myself, if it's the lower level, you go down to more resources in your project are being affected all the way. That's from the technical point of view. Now, if somebody accept comes in to our team, and we have a flow, we switch to Kanban, we have the way of our rules, how we deliver the tasks, how we do the code review, and how we do the quality I allow to challenge it, the next challenge will be I think it two weeks for February, because they want to challenge the flow, and in the Kanban. And I allow them to challenge my decisions, right? And to bring up the arguments in a friendly manner. So if you have enough arguments, to convince me and the team also, right? You're free to do it. But if you But in exchange, I asked that. If you fail to convince everyone and you realise that not exactly your way is better. Please be open minded and consider also changing yourself. Right. That's the mentality I tried to solve. So yeah, we are open but in exchange, you have to be also open minded.

Researcher 39:44
Yeah, it's the shared openness. That's what made your work environments safe right. I agree with you. So the next the next item is, it is safe to take risk and initiative in your team, you said yes with promote bold moves ownership and responsibility and do not banish for being brave. That's a beautiful statement. Would you like to comment on it? I like it very much.

Participant 3 40:15
I have a perfect example. Yeah. Before mentioned, senior developer, he and two other back end developers came to me. In short, on September last year, I totally created the architecture requirements and all the, you know, my microservice structure for the starting project, right? It consisted of one worker instance, I mean, the code base for it, right. So we've had a lot of different applications, but this specific worker was one, they wanted to, after a few months, they wanted to split it into three types of workers, right. So they came up with this initiative. So each worker would be responsible for different cue. But then I knew is going to create balancing overhead. Because we will be it will be hard for DevOps to balance inside the cluster, how many pods you should have for each worker to maximise the processing of, of the data. And we talked about millions of records. So but I allowed it, because I know we had quite high test coverage. And in case of merging those services, it will be pretty easy going back, I allow for this mistake, to let them realise themselves. Even totally to the management, but after because in the beginning, they might some people might not even allow this, right, like micromanage people, like, so I let them do the mistake. I didn't see it as an example mistake, because I mean, potential mistake, because I always leave this 10% uncertainty as a default value in my head, because I may not be right. And I think just before Christmas, they approached me can and this is it, this split is kind of problematic. And also the upset that like he doesn't know how to balance it. And since it's a synchronous, and you can scale it horizontally as a one worker, you don't have to care take care about consuming three different jobs right. Between them. So he said, I think we will go back to your, to your way. It's like really, but I think we have, we have created another thing because of it. Because they allow allowed them to make this mistake, we created another micro service that was needed, which was we kind of needed it because before every single micro service was talking the database itself. Because of this scaling, of number of microservices, we needed an API for the database itself, and not let any micro service talk directly to the database. And I missed that. So I logged on to make a mistake. And in the end, we produce something in a higher quality, kept the motivation up, get the safety up, because I tell him, you allowed to make this mistake, I knew it's gonna be like that, but But I hoped that I was either wrong, or we're gonna produce something more out of it. So the experiment kind of a world and when the management was asking us about this will happen because the project manager, you know, kind of has to report on faith. I told him, I deliberately allowed this mistake. Why? For motivation purpose. It's like with your children. I have two kids now. I had one when we met last year.

Researcher 44:18
Congratulations!

Participant 3 44:22
Thanks. In a controlled manner, right? That's like, you know, you let people go wild, because, you know, there is a tenuous difference between the oldest developer having the team and me so so I kind of, you know, they kind of need to listen sometimes, and they do, but sometimes I need to let them you know, do those mistakes. So, so, it's not gonna be why you're doing it. That way, not the other way. Because poetry says so. Like, no. You have to realise why because we don't have time to explain all the possible scenarios and transfer my whole experience. So yeah, and it wasted only two days on this throwback, because you just manage those files and deploy one Microsoft. That's it. Plus, but it was invaluable experience. Yeah.

Researcher 45:15
Do you think about allowing them to do mistakes? Sorry, they learned. And they also become confident at what they do.

Participant 3 45:28
Yeah, because in the end, I told them, I told this, the most senior developer that I consider him back at least now, starting from January, because, and then I explained publicly to everyone, like, I know, it was a potential mistake. But he and then he came back to me with the team to go back to it. And we had this data hub idea, right. And in order to be a great developer and a great leader, you need to be able to see our mistakes, and not hide them. And bring them back again, you know, to the surface, and willing to admit your mistake. And then I told publicly to them, this is the moment when you become a leader and a wealthy person that I don't have to even look at your hands. And when whenever I'm offline, you can ask Mattel's does his name, too? Tell you what to do. Because even if you might have mistake, he will come to me telling me my mistake. Trust is higher than talent. I didn't say that. Because that will take this like, Oh, my skill level is long. This one I didn't say that. That's just my comment, right? Like people being trustworthy are more valuable than the talented ones. I mean, it's nice to have talent, you know, super talented super trust to someone. But this is how Navy SEALs also choose the leaders in the elite teams, they have this trust and talent, trust and skill matrix, right? That they promote, and they always promote people who they trust. That's pretty much it.

Researcher 47:23
Okay, thank you. That was a good example. That's bring us to the next item, which is about helping each other and the team. Yes, the statement states, it is difficult to ask other members of the team to help and you said now, we have a culture that is that if you can have a quick Google slack called to help someone than just do it. It is the most rewarding trait in your company. I agree with you. And do you have an example you'd like to share where helping each other has improved the quality of people's work?

Participant 3 48:06
Alright, so why don't one of the regular developers after the tracks that I mentioned last year, he experienced a burnout and he wanted out from this project, right? We kind of respect that and we rotated him for a junior developer. Solo low level. So we spent a lot of hours on November and December to bring this junior on, on board on this project. Very coding many hours giving some instructions on and answering in real time. Right and explaining him what his keys what is YANI

specifically YANI with which is you ain't gonna need it is really important for for juniors because they tend to just comment out big piles of code and leave it for later. And just to explain why you don't need and how and why is also good practice like okay, you wrote six slides of code two weeks ago. And then you want to uncomment it there's no valuators just write it again. It is good practice right? Nevertheless, you ain't gonna need it now. And DDD right, tried to teach DDD and all those abstractions and keys and all the methods so I had to write the functions for him and measure the time of like, how a simple function executes something within the given time versus complicated the highest skill level is now actually the company coordinate Guru is considering this, those examples as as a keys. Confluence page was like, okay, just it was for one person if you want to follow the genius why not? And yet, we spent a lot of effort I had two apiece to reply them, yet the velocities still higher than the last month. So we kept the velocity higher. And we work to the junior. So he became the regular developer in January, he, I told the other developer that state, we need to bring him up because he has the skill, but he doesn't have the experience, even in our project and practices that we really wanted to implement. So I was like, even 10 hours a week, just with the call to that person, and we are ready to do that. And I promote this. Sometimes he somebody sends a message, do you have a minute and I just click the huddle and call the person that's the straightaway if I have a minute. So I like created this culture and promote this content people inside this project theory like it, because they don't need to schedule and look at your schedules. You just they just ping you and Are you free now. And if you're lucky enough, you can call straight away with the need occurs. So

Researcher 51:24
Okay, that was a good example, I just want to explore and dig a little bit in that example. So you help this junior developer to catch up or to come to the same level of standard to the rest of the team? How did I understand implicitly it helps quality, obviously, but concretely, how it did help the quality or at least delivering his work become within the same quality standard of the rest of the team?

Participant 3 52:04
How did help? Yes. All right, well, faster onboarding. In the end, it does increase quality, it allows you to catch up quickly with level of the quality of your new team, right, so it's just a matter of stretching the time. So but in the end, he also started to have his own ideas. He's producing around 10 ideas, there's at least one good one, right. So I'm pretty exhausted with diplomatic, you know. But, but still, he produces some of the content is, it allowed to understand quicker the expectations of the team on quality. Because when somebody suddenly rotates out of your team, you think ahead, because if it's one out of three back end developers, that's 30% of your capacity and knowledge going away. So that's not good, right. So the new developer work quality, this particular example, it just allows him to catch up with our standards for quality. But if you allow for quicker collaboration, and being able to, you know, call anyone, at any given time, if there are three of us who have to write this method, then it promotes collaboration, the quality of your collaboration and teamwork produces, I cannot measure that I don't have any KPI for this. But I always knew at my heart level, and other results I've been producing that this must be related that I promote collaboration and communication between the people.

Researcher 53:53
So how this collaboration you call it or helping each other helps people to improve the quality of their work.

Participant 3 54:03
First of all, now we all work people, right? So I told the management within our waste a little bit more time in meetings, why you need to let people speak. They are being isolated. They say that psychology that the whole thing. What happens in the head of the developer is affecting the quality. When people don't feel good about themselves, so about, you know, the team, they are in the relationship inside the team, right? They don't produce the good quality, because they reach the goal when they see one example of a one working example of the future and they ship it and that is that like they were distressed when they're frustrated when they're not happy. And when they sell any kind of negative, you know, feeling thing that they experiencing now they are delivering the features, the moment where, when they see one of the requirements are met, or most of the requirements are met. So untested features, and lower quality of code, we create overhead testing. And going back here with the same feature. And yeah, potentially use time, a lot of the precious time in the project. So this helps specifically all, especially for remote teams. To even feel better at the word like, like, I'm in this project, and I feel comfortable. Like, I can talk with people, I'm not like, I don't like hard stops for the meetings. Like, okay, it's 3pm, I'm out, never choose this never make room between the meetings. For those people, maybe they have they need. It's tiring. It's exhausting. It takes a toll on mine. And right, because I need to sometimes, you know, listen to all the all the needs and all the all the stuff, they want to talk about all the pieces of the Golden want to show. But who said that a leader job is supposed to be easy. So just shove it in. And the result of the team is the result of your work. So safety gives maybe not tremendous, but significant increase in the quality. People learn faster and share and help in the right direction that improve quality.

Researcher 56:34
Okay, fantastic. Thank you. We do have four minutes, and I'd like to spend them on the last item because your answer was very interesting. I'm just going to stay the item just to remind you, it says working with a member of my team, my unique skills and talents are values and utilised. So you said yes, we conduct a transparent approach to feedback culture. Sorry, so everyone periodically gets some feedback on their work. So this ongoing feedback do you think people perceive it constructively because of the safety work environment you made?

Participant 3 57:23
Um, it's, I think it's more like a question to the whole company, because we have a feedback culture. Which means that every three months, everyone is now not every six months, sorry, if it's after a month, when you join the company, after three months, that and then after six months, when you continuously two times a year, you're supposed to gather feedback about yourself. From what you don't see, unless you know, people click they want to share it, right. So people leave the comments. And they're supposed to be considered constructive, if they give an example of what happened that is the most highly valuable feedback and all maybe some detail details and some opinions about the trades and what they feel good about your different bad about you. There's also start stop continue methods that they use, you can have, you know, as general feedback, what is good, what is bad, or the style

start, stop, continue. It is something new to me, right? Because it creates this situation that you can see what people tell about you signed by the name on assignment and feel, you know, positive, negative, you know something more about yourself. And I have two feedbacks that I received from end of November. I can face them if you want. Do you want to this examples of the feedback or quality of his work? Now, it didn't affect the quality of my work so much. But when I go back to the times when I was in my early 20s I had no idea if I'm doing well. What should I change? I was even eager for someone to tell me. What should I change in the way I work. And I was lacking that. Now those kids in in our company, they get this every six months from all the peers. That's amazing as it's like, that's what I missed the most because I was lost when I was when I was starting it I was completely lost before I was there. I was clueless and they also clueless and they did this So I believe he has huge impact on shaping the type of people you want in your company, and the standard and the direction that they're supposed to take. And in general, if six people tell you, you do something wrong, you're doing it wrong. If people tell you to go all the way to three people tell you to go this direction is probably the best direction you should take. And I'm a. I'm a fan of Stephen Dubner and Steven J. Lewis from the economics when they say the worst advice to business advice is given by your family,

Participant 3 1:00:44
And the family never gives you good business advice. So ambition advice, like so, yeah, but you know, you don't know that when you're 20. Unless you read the book, I was lucky enough to read the book. So when you finish review, and tell you which direction you should take, it makes the life easier. Now, nowadays, the youngsters and also what I was gonna say was tough. It's not like, you know, the American Boomer, you know, people born in the 60s, right? They didn't have it in the side, but when you are in your 20s, nobody wants to hire you. properly, once you face pain has like, what 60% unemployed people under the age of 24. That's like, that's huge that, you know, that keeps a toll on your own self esteem. So there's a bunch of low self esteem people that, for example, and also it's happening in Poland, probably in Denmark, where you are, is it, it has a big number, right? So they don't want you. And when they take you, they don't tell you what you doing wrong. That is the most unjust way of hiring young people and directing them. Now in the company I work in, at least when they wanted, and they are on boarded. We tell them what they doing wrong. In most. I think it's a chapter where if six people tell you doing something wrong, there's no way you can be pissed off about it. You just admit, you're right. But those are gentle ways, you know? So I have, but those feedbacks are like, they're too good, too. I mean, they wrote like, 20. No, they're like, there's like, yeah, eight good things. I'm opening, like, two bad things. Those are not exactly bad things. So I know you want me to copy and paste for you?

Researcher 1:03:11
No, no, I understand. Don't worry about that. Now I understand. So how did it make you feel having that balance feedback?

Participant 3 1:03:23
This feedback was not balanced, right? It means that team in end of November, I think they were charmed by the situation that I solved the issue decline those complete chaos. The scrum did not work. Well. Of course, the scrum will never work if the reality doesn't allow you. The they didn't want Kanban also, but what they saw the Kanban after the first two weeks, right, let me think didn't know come back,

right. Everybody is like sprint scrum sprint Scrum, like, Do you guys know Kanban and continuous development? Like maybe 20% of industry knows what it is. So they are because product owners believe that Scrum is a way to force you to deliver something that was not deliverable in those two weeks because something random happens. Yeah, it gives if you're not going to deliver because something happened in those two weeks or this one way. We move it to another sprint you will show it to the whole team that it was delivered. And we it's like a culture of shaming developers who somehow did not perform in the previous sprint so much. It says it's whipping people publicly. So I don't like

Participant 3 1:04:45
Nevertheless. Well was a sort of love too much. So the feedback is about the charm.

Participant 3 1:04:59
For my no feedback, which is happening just now, and yeah, and so I guess it will be constructed. So I think what they wrote in here is like, it looks like, Alright, I'm a superhero. Okay, guys, we just dealt with a problem that you thought is not delectable. And, you know, they need to learn this, they will see how it's handled. When 10 years ago, I had no idea how to handle this. It's just, you know, take a chance. I'm waiting for you. Two months, and he's been quiet with the customer. So let's see how it goes. Okay, Peter. Fantastic.

Researcher 1:05:40
I enjoyed the conversation very much. We consumed one hour. Thank you a lot. Thanks for the good examples. I enjoyed the conversation, and I hope you did. We will stay in touch. I'll get in touch if ever I need you. Okay. All right. Cool.

Participant 3 1:05:59
By the way. Are you publishing this work?

Researcher 1:06:06
Yes, I will send the paper to you. I will circulate an email with all those who wanted to read it.

Participant 3 1:06:16
Great! Thanks.

Researcher 1:06:18
No problem. All right. Thank you very much. Have a good day. Thanks for everything. Bye.