00:08 RESEARCHER:

Good morning, PARTICIPANT 32. How are you? Or [Deleted to preserve the participant anonymity]? How are you?

00:11 PARTICIPANT 32:

Yep, I'm fine. Thank you. I do apologize. My webcam on my computer is not working.

00:18 RESEARCHER:

It's fine the way it is. The most important thing is to hear you and you are able to hear me. So that's the most important thing. Can you hear me well?

00:30 PARTICIPANT 32:

Yes, I can hear you perfectly fine.

00:31 RESEARCHER:

Okay, fantastic. If we can increase a little bit your audio volume would be great.

00:40 PARTICIPANT 32:

Sure. Has that helped?

00:42 RESEARCHER:

Yeah, a little bit. Much better. Thank you.

00:45 PARTICIPANT 32:

Sorry, I just moved close to it. I have a very nice desk mic. So, I just had to move it because I pushed it to one side when I wasn't using it and not pushed it back.

00:59 RESEARCHER:

It's much better now the voice and the audio, the clarity, and the volume is much better. Thank you. I'd like to start by thanking you for the opportunity to talk to you and accepting to do the interview. I really appreciate.

01:15 PARTICIPANT 32:

No, it's fine. Like, I'm actually like a really big fan of talking about Agile as a whole. Mostly because where I used to work until very recently, we were working trying to bring Scrum into that team, as I mentioned to you in my emails. And I won't bore you with the details. But, I mean, you're from Denmark?

01:58 RESEARCHER:

Yes. Correct.

02:00 PARTICIPANT 32:

Yeah. And so over in the UK, my employer, they're quite famous in the news at the moment for, basically, there's a thing, there's a piece of software called Deleted to preserve the participant anonymity].

02:12 RESEARCHER:

Yeah, I know Deleted to preserve the participant anonymity]

02:14 PARTICIPANT 32:

Yes. The Deleted to preserve the participant anonymity]. Yes. So, it's them. So, I've got quite a lot to sort of talk about when it comes to Agile and their attempts to try and do Agile, how we could have done it that would have done things better, and how, you know, essentially, sometimes they can use Agile as like, sort of, almost like a smokescreen for poor working conditions and development standards and things like that. Which is, I don't know how important that is to the sort of research doing, I've got an old like text document here in which I've put together some of the examples I can give.

02:58 RESEARCHER:

Okay, fantastic. I'd like to start by introducing myself, because we didn't do that before and tell you what I'm doing, why we do in the interviews, and explain to you the structure of the interview, and we can take it from there.

03:14 PARTICIPANT 32:

Sure.

03:15 RESEARCHER:

My name is Deleted to preserve the participant anonymity] I'm a researcher at the I.T. University of Copenhagen. And my research interest is mainly how software teams manage to achieve better software quality. And currently, I'm running this project, I'm looking how Agile, or Scrum helps teams to achieve better software quality. So, I'm using an Agile practitioner perspective, people who worked in a Scrum or other teams. I like to hear their perspective. And we use their experience as a source of knowledge. And we capture it through interviews. And once we capture that experience, we analyze it, and we draw conclusions. And once we draw conclusions, we will report them in a form of a research report. And sometimes we make recommendation and implications for other practitioners to learn from our research output.

04:27 PARTICIPANT 32:

Sounds great.

04:25 RESEARCHER:

Yeah. So that's what I'm doing. And yeah, I'm talking currently, I'm talking to a lot of software developers, and I'm really enjoying it. So, they are very passionate about what they do.

04:40 PARTICIPANT 32:

Great to hear that!

04:57 RESEARCHER:

I love it because I'm passionate about my work and I can hear the passion in your voice and the way you talk. Sometimes you have to sense it. It's you, you don't see it, but you can see it in people way of, of reacting to few things. But sometimes you hear it. Yeah. And I do appreciate it. Of course. Before we start, do you have any questions for me?

05:23 PARTICIPANT 32:

Honestly, no, I think you've covered everything.

05:28 RESEARCHER:

Okay, fantastic. So, the structure of the interview is not rigid. I do have questions I like to go through and, but feel free to add it's an open interview. It's rather a discussion guided by a set of questions. So don't feel like you tied to the question I ask. Bring things you want to bring. And, yeah, make it a little bit more fluid. It's up. Let's start with some introduction. Can you introduce yourself telling us what you do and your experience as a software developer?

06:12 PARTICIPANT 32:

Sure. So, I'm Deleted to preserve the participant anonymity] And basically, I've been a software developer for about ten or eleven years now, before that, I was a bit more of a hobbyist, I did a degree in biochemistry followed by a master's in mathematical modeling and biomedical research, specifically into like cell signaling in platelets, and how the blood clots and then modeling now on a computer, so not necessarily originally from a computer science background, but moved into a much later because I kind of realized that systems biology was where my real interest sort of lay, because I wasn't keen on the wet work as they call it in the labs. But I did like the modeling side of things. So, my previous employer, who I have very, very recently partnered with is Deleted to preserve the participant anonymity] which is in the UK in Ireland. And I was working as a software developer in a Scrum team of junior talent, which is basically anyone that is either an apprentice or a graduate. So basically, that's me, and that's who I am in the walls.

08:02 RESEARCHER:

Great! Very interesting journey.

08:04 PARTICIPANT 32:

Yes. It is.

08:14 PARTICIPANT 32:

Now. I’m doing some consulting aside. I've met some very interesting people so far, like, I actually have a recurring contract with a professional Deleted to preserve the participant anonymity] in London and maintaining her website. So, you know, like, you meet all sorts of fascinating people walks of life, and I think that's worth more than money to me.

08:37 RESEARCHER:

Of course. You get a lot of freedom with consulting.

08:37 PARTICIPANT 32:

You know, it's just like, it keeps you open minded, when you just you meet somebody that's, you know, just really out there and different. And when you are, you know, you know, meeting other people that are a bit odd or a bit different is always a bit exciting. You can always pick up something new.

08:58 RESEARCHER:

Okay, let's start with the first question. I mean, you already hinted to your answer to this question, but I'll ask it anyway. What do you think of Agile as a software development methodology?

09:12 PARTICIPANT 32:

I think that Agile when you execute it correctly, is quite literally the best way of working and developing software. You can apply it to not just software development, but you can apply it to government, you could apply it to politics, you can apply it to pretty much anything as like a working method just because it requires like decent general policies and processes in place, and then relying on sort of like the interpersonal interactions, or like, a much lower level, to drive those processes and get to the end point that you and I think there's a lot to be said for the application of Agile to other areas of like working life. But having that sort of structure that is not perceived like chaotic, but it's just sort of like it's an organized chaos. But as I've mentioned, like, particularly in the corporate software development world, I find that often Agile is used as an excuse for like sort of just lazy hands-off management or disorganized, chaotic working environments or poor working conditions. Now, I've seen it in sort of small businesses and things like that, where it's been executed brilliantly, and they've got all the tools in place and things like that. But I think it was a very common problem, which is that, as I said, people often use it to try and get away with things being less sort of productive, I think it's the best way of putting it. So that's, that's my general viewpoint on Agile.

11:07 RESEARCHER:

But overall, what is your experience like with Scrum implementations. Is it positive?

11:36 PARTICIPANT 32:

So, I had one manager who was roughly the same age as me, who I was working with who ran a project in like, good, he used to work in a small business. Then he came to [Deleted to preserve the participant anonymity] as an Agile software developer. He is where I learned a lot of my sort of first stuff about Agile from, I then went on to do a lot of training courses, a lot of reading, and all of that sort of stuff, because I got quite hooked on him. But he ran a Scrum team of sort of like me, and like three or four apprentices. And basically, what he did is he took on the burden of a lot of a lot of he took on the role of basically being the technical owner, and being like, the customer interface, and all of that. And also, was doing the code mentoring for the people in that project. So, he cut all of the sort of extra stress out for the apprentices and myself, and basically enabled us to just get on with observing the tasks that we've been step setting, like a story format, and that sort of thing. doing those tasks, completing them and producing an end product. This is possibly one of the best sort of experiences I've had in development, just because this led to it was a decent solution. But I think the outcome could have been a lot better. Because we were all quite new to programming at that point. And this is not like a, I wouldn't say it's a negative, it's just more of a like, if we'd had some senior developers in our team and things like that. Some people to actually look at our code, refactor it, that sort of thing, tell us where we were making mistakes where we were going wrong, it would have been a much better experience, because we would have reached the end of that project, we would have been able to able to identify what we've done wrong in the sort of like classic retrospective fashion. And then we'd be able to improve.

13:51 PARTICIPANT 32:

Instead, there's a recurring theme with working in [Deleted to preserve the participant anonymity] where there wasn't much of the retrospective stuff, where we weren't particularly looking back. And what was done. It was a very sort of fire and forget approach, which I wasn't particularly keen on. That's neither here nor there. But that's the most positive thing that I've experienced with Scrum was just sort of like we could, it could have been done better because we could have, you know, like, had a dedicated technical lead, we could have had dedicated like customer interface and that sort of thing. But, you know, that project ended up coming out and the customer didn't have any complaints about it. You know, it did the job. It was functionally working. And I think, at the end of the day, that's what the goal of Agile development is. And as I mentioned, like, [Deleted to preserve the participant anonymity] was trying to transition from Waterfall to Agile. And I can give you some examples of where that's been poorly implemented as well. But I can also In sort of retrospect, tell you how it could have been done differently and how Agile would have made a difference. Because I've spent long and hard sort of assessing these things. This is one of my positive experiences with Scrum.

15:15 RESEARCHER:

Let's talk use this positive experience. And let's use this project as an example. Okay? Because we are looking for good outcome. So, you describe that the project has produced a product where the customer was happy with. So that's Agile is the end user satisfaction, so can you describe me this Scrum setup, how the process was working, and how the team was working together?

15:42 PARTICIPANT 32:

Sure. So, a lot of this was before COVID. Because once COVID hit, things got a bit more complicated. I don't think anyone really was prepared for the sudden work from home situation. But in essence, we all would sort of sit in the same room. Together, we would all work on our own computers, sometimes there would be an element of pair programming and that sort of thing. We used to have a very sort of like open door policy, which was, you know, just wheel over to the nearest person and peer over their shoulder while they're working on something or just, you know, just call out to the other person just be like, Hey, can I get hammered with this, or I'm stuck in that sort of thing. We had a very rudimentary, like Trello board that my colleague had set up for us, because he was a big fan of Trello. And we didn't really have JIRA, or Trello, readily available to us. So, he set that up. And he did the arduous task of basically preparing all of the requirements that he got from the customer in a sort of, he tried to use Gherkin, where possible. I assume you're familiar with Gherkin, I assume. So, he used Gherkin, or he would use like that as a blank, I want to be able to blank, so that blank. And when it was being done in that fashion, it was brilliant, because, you know, we could get the actual requirements in the functionality captured. And you would have like the tangible end point where you sort of, you know, what you're working towards, and if it produces that endpoint, then you go, excellent. Task complete, done, check that box and move on to the next one. And that really increases like your satisfaction as a worker because nobody likes feeling like they've left a task unfinished.

18:00 PARTICIPANT 32:

And there's some stuff from like psychology to it. Like, it's like the SMART goals. You know that they say that you should make your goals that are like, achievable. And God wants the rest of it. It's like something small, something rather achievable, versus something measurable, achievable. But yeah, either way, I can just, honestly power the internet at my fingertips. I don't even check, but that's it. Yes. specific, measurable, achievable, realistic, and timely, which is like, that's like the principles of setting yourself targets to meet. And Agile, while not necessarily naming it as such, has basically captured that entire concept and put it into this framework that basically allows you to develop things in a fashion that is satisfying, because, you know, many of us have worked on projects that were not Agile, and you are nearly always left with this feeling of what am I doing? Where am I going? What's the next step? You just all these sort of like, self-doubt and confusion feelings that you have from just being quite directionless. And, you know, I think that's, I'm trying to be careful, but I don't repeat myself too much and gush over how much like, Agile just makes you feel more comfortable working. But that's pretty much what it does. It's, it's just it's, what's the word I'm looking for? It's comforting in a way. You always feel like you've achieved something with it even no matter how small it is.

19:56 RESEARCHER:

It gives you a sense of achievement, right?

19:59 PARTICIPANT 32:

Yeah, Plus, if you're looking at breaking the task down into chunks really helps people with their executive function. So, I have been making quite a big point about this in some of the software community is about how Agile is so good for like developer’s satisfaction in general. Because as a hypothetical in a world where sort of, let's say it's at [Deleted to preserve the participant anonymity] where I worked, were sort of, it was like a big, global sort of task board. And people submitted their tasks in like a story format and things like that. But I am of the viewpoint that if you adopted this sort of structured working style, you could create a system in which you can enable people to do those sorts of tasks in a like organized and structured fashion, which makes their life easier. And I can go on for hours about how what you think Agile is just like Tip Top for developers. But that's not the purpose of this. Yeah.

21:36 RESEARCHER:

All right. The next upcoming question is about software quality, and how Scrum or Agile helps software quality. So, in order to have that discussion go when we need to define what quality means in our software development environment. So, what does software quality means to you? How do you define it?

21:55 PARTICIPANT 32:

Okay, so quality to me. The biggest one is that it does what the user wants it to do. And I don't mean this behind the hood, I mean, just sort of like it produces the end result. Now, that is a very weak way to view quality. But it is the biggest important one, if we're moving to looking like behind the hood, the biggest important thing with like quality code to me is making sure number one, that it's readable. Like, if you don't make it readable, it's pretty much worthless, because the number of times that I've been on a project and you've gone in, and you're just presented with like an index dot js, and it's filled with a complete lack of like classes or objects, like anything, it's just long, long lists of functions and that sort of thing. And you sort of take one, look at it and go, good Lord, I have no idea where to even begin with this. We used to have a running joke, which was the nuke from orbit option, which was, I was very famous for the first thing that I would do with an index dot js, which was like that was I would just press Ctrl A and then hit delete and say, right, start over, but then restore it because I can't just do that. But it was just like, it was just a funny joke to sort of represent how I was like, No, we don't do unreadable code around me. And the second one is that don't take shortcuts is that's quality to me. I can be quite rigid about this. But I believe it is fundamentally like one of the biggest things as well. Because, you know, it's the difference between having all your validation in, say, like some encryption software, or like, making sure that your security on the website, like if you are submitting passwords that everyone through sort of lots of different layers of protection, right? It's the difference between that and just creating like an if statement that goes here's the password, is it does it match up with what's in our database?

24:28 PARTICIPANT 32:

If yes, done. And like, you know, there is a clear difference between the two things where you've just got the one where it's just like checking, does it match? Yes, done. Or there's the sort of the extra layering where you're sort of going like, have we received like thirty requests in the past five seconds. If so, that might be a bot. Let's slow it down a bit. And you know, just adding those extra bits and pieces of security. Like that's another really big quality thing to me. What else is there software quality. It's probably worth caveating. I personally like C and Rust as languages. So, my perspective comes from someone that likes low level programming versus high level programming. So, I am acutely aware of the fact that my personal preference for that will put a very heavy bias towards what I view as quality because, you know, in my world, memory management is a very important thing. But in the JavaScript world, memory management is not important at all.

25:42 RESEARCHER:

Yes, especially when you work in a low-level programming. Yeah, especially I understand your perspective. I just want to follow up in one things before I move to the next question is, when you said avoiding shortcut, do you mean ensuring scalability or avoiding technical debt?

26:05 PARTICIPANT 32:

Both of those would be more than accurate descriptions, quite honest, because technical debt is something that really frustrates me because this is an autism thing. But basically, I'm very good at patterns. And I can often identify a single line of code that looks perfectly innocuous at a glance, but I will look at it and I'll say that is going to cause us a problem. Three weeks, or now, or more from now. Because, you know, you can't justify that sort of thing, because nobody else thinks that way. It's just maybe autistic people. And there was a very famous case of this where we needed to use an adjacency list for program that we were using. And, you know, I was sitting there going, like, we need to use an adjacency list, please help me implement this adjacency list. And nobody was interested to doing it. No, no, it works. But then we got several months down the line of the project. And then we had to redesign the entirety of our database. Because we realized the way that we had implemented it, I am not familiar with the way it was done originally, because I kind of walked away from it sort of like, Look, it needs to be an adjacency list. I don't want to be part of it, if it's going to break in the future.

27:39 PARTICIPANT 32:

And then it did. But this is what I mean is sort of like that is one of the key things that I'd consider to be quality and like technical debt, you know, it's it ties in with the memory management as well, where it's sort of like, people sort of say, like, Oh, it's only microseconds, it's only microseconds and all that sort of thing. And I'm like, Yes, it is only microseconds. But that's on, like, when we're talking about like one part of the program. And it's actually like, if you look at the actual statistics for like bigger applications, all of those microseconds on like, the, like, you've got ten thousand lines of code, and you've got each line of code consuming twenty microseconds more than it should do. And that adds up. And I was like, even that, in itself is a form of technical debt in my eyes. But you know, like, I can have this argument for days about why you should use certain bits of programming for certain things and other bits for other elements is what it's why I'm such a big fan of Rust. Because it basically is the best of all worlds. Like, it's just a very well-designed language. And I get along with it quite well. And yeah. What was the other one? Sorry, it's technical debt and?

29:03 RESEARCHER:

Scalability.

29:04 PARTICIPANT 32:

Yes. And the scalability thing, just quick point on that is just, I often find there's a very sort of, like, frustrating element of people tend to design things for just that one specific task. And again, like you can point out to them say like, Well, look, this is going to break if you know, your work for like ten requests, but we're expecting this to possibly have like a thousand requests. And once it hits a thousand requests like this is going to this is not going to function as well as you want it to. And again, it kind of ties in with the technical debt as well. But most of my perspective comes from like a technical point of view.

29:56 RESEARCHER:

If you fail to produce a scalable design, that's technical debt in itself. Yeah. Okay. We will move to the next question. We'll keep in mind this definition and use the same example, you used before at your project at [Deleted to preserve the participant anonymity] What did you do to assure quality in that project?

30:17 PARTICIPANT 32:

So, the one that went quite well, we had very like rudimentary testing in place. I'll be completely honest that like, the definition of what was good at that point was very different to what my definition of good is now. I can identify now where we could have done better, because, you know, like, we had decent testing in place. That allowed us to make sure that our code was like producing the correct results, it wasn't throwing errors in all the wrong places and stuff like that. I think we also had more time to develop at that point, because we didn't have to get involved with the customer side of things. This later changed, because of just how things worked out. But, you know, our time was basically spent focusing on making the code, changing the code, fixing the code. I think we mostly, we didn't do very good on architecture, because at that time, we didn't really have the skill set for architecture. Since then, I've really heavily got into architecture for software. And I think, again, Agile ties in really neatly with software architecture, because you know, if you are following the whole, sort of like epics, features, stories and tasks element, you can actually map that whole sort of frame to your architecture. And once you've mapped that, you know, it cross references by like story ID, and like, epic ID and feature ID and things like that. And once you've got that sort of cross referencing, you can, you know, like, let's say your epic is I want to create a shop, right?

32:18 PARTICIPANT 32:

So, you've got the epic for the shop functionality of your website, and it breaks down into the features of like, browsing the shop, purchasing the items, and all of that, right, you can then design everything in that modular fashion. And suddenly, you've created like, such, like easily maintainable code, that it's almost like idealistic to think about, but you know, the point is, is that, even if that's not like, a hundred percent obtainable, it is, at least, you know, better than nothing, because project entropy is a real thing. Because, from my science perspective, it's sort of like, entropy is something that you constantly have to battle, like, everything goes from order to chaos, unless you match the amount of energy that's coming out of it, by putting the same or more energy in. And it's like, if you start with like, ten energy in it, rather than a hundred energy. It's only ever going to be like an uphill struggle. And, you know, like, this is where I like the whole Agile framework to architect things, because you know, then you've got like, this really sort of organized structure that's like completely and utterly expandable, however you want it. And it just requires like a bit of flexible thinking with how Agile works. But it's, I'm rambling because like I get really excited about frameworks.

33:50 RESEARCHER:

No, that's fine. We'll get back to that question. Because I didn't get the whole answer. You said you had a lot of testing, which assured the purpose for fit or having a product within the user, the end user expectation, but for code quality and the internal quality of the software, what did you had in place to assure you met some quality standard for the code and the design and the internal software aspect of things?

34:26 PARTICIPANT 32:

Yeah, so [Deleted to preserve the participant anonymity] is not great at that. In all honesty. We had some rudimentary like behavior tests in place just to sort of make sure we were getting the right outputs. Alongside that, there was we didn't really do unit testing at that point. What else did we do to make sure we had quality code.

34:59 RESEARCHER:

Were you reviewing each other's code for example?

35:00 PARTICIPANT 32:

So, we used to do pair programming a lot, and code reviews, and retrospectives was something that we didn't really do at that juncture, we've wanted to do them. Because retrospective is one of the most important things like in my, in my view. I think the problem with the retrospective is that there's a right way to do it, and then there's a wrong way to do it. The wrong way to do it is everything went amazing, and were brilliant, and so on and so forth. The right way to do it is, here's what we did good. Here's what we did bad. Let's improve the bad. But I find that there is a, there's a taboo, when it comes to retrospectives around talking about the bad. And you can find yourself under fire quite a lot for being the person that has the courage to talk about the bad. Because unfortunately, people don't like to hear bad things. But one of the key things that I've learned in life is that talking about the bad things helps you grow and improve. That's the whole point of Agile, you can probably see that I apply Agile to like my own life at this point. Like I'm a bit of a nutter when it comes to that.

36:23 RESEARCHER:

Alright, I hundred percent agree with you is by and Agile, facilitate that by empowering people to talk about their mistakes and improve by looking at their mistake and how to improve them. But for the purpose of this interview, let's move to the next question. Do you think that the Scrum setup you had or from your experience, an Agile setup or a Scrum set up produce software quality and how a Scrum process helped to produce quality software quality?

37:07 PARTICIPANT 32:

Sure, so, I know we want to stick to like positive case, but I'm going to share like one example of like a negative.

37:12 RESEARCHER:

Yeah, sorry, sorry to interrupt, you can contrast between a negative and positive so we can see where the strength of the Scrum method is, you can do that, of course.

37:23 PARTICIPANT 32:

Sure, like this is, I think this is a really good one to use. Because basically, this is a much more recent project that we had, and I've got some notes on it just to really sort of like point out how this was an attempt at doing Agile. But it went wrong in like pretty much every way possible. But you can identify those points where things went wrong. And where you're doing the correct retrospective, where we're going, like, here's what went wrong, how do we improve? It would have been a good experience. But the problem was, is that we didn't have a good retrospective, which in itself is a failing of like the whole Agile process. So, you know, like, making that one will. But in essence, we had a customer who, quite literally, by the end of it walked out of the contract, because he was so dissatisfied with essentially just how the project was going. And, like, the issue here was, they would they were developing a tool, and it was adequate, I think is the best way for it. But the problem was, is sort of like so we didn't have designated single points of contact for the customer. So, the customer would reach out to all of the devs at different intervals and add new tasks and change features and that sort of thing right.

38:52 PARTICIPANT 32:

Now, on the one hand, you can see, oh, that's brilliant, because, you know, good customer interaction really captured those requirements. That's perfect. But in reality, what was happening was he was coming to us, he was interrupting the devs from working to get them to answer his questions, rather than going to the single point of contact are sprint goals but keep changing from day to day. So, you know, he would ask for something to be produced. And then halfway through the sprint, he'd be like, scrapped that we don't want that anymore. We want something else. So, then you'd have to ditch all of that work that was done, and then start on something new as well. And then you'd come to the end of the sprint, you'd be like what have you got done, and you would sort of be sat there going. We don't have much because halfway through the sprint you scrapped what you wanted and ask for something new. And of course, customers being customers like they don't see it that way because they don't grasp the nature of how software development works. So, they just get frustrated at the lack of visible progress. Because, you know, people like to see progress, like, that's the whole MVP element of Agile. And, you know, so it's really easy to identify here that like, if we had set a very clear like boundary of like, don't talk to the devs, go to the single point of contact, relay everything that you want through there, and had that single point of contact that was able to sort of say no to the customer. And to say, like, Look, we can't change the sprint goals, right now, the sprint goals have to stay set for the whole sprint, otherwise, you will be disappointed at the end of the sprint. You know, if we'd had that in place, the devs wouldn't have spent so much time being interrupted and not doing things. They would feel sort of more relaxed, because they're not sort of sitting on edge waiting for the next change in the project scope. And they don't have to like sit worrying about what they have to say to the customer either. Because the customer is completely managed by the single point of contact. And that's like one of the key things that should really be sort of done. And similarly, like, we didn't have a tech lead on that project.

41:27 PARTICIPANT 32:

Now, that meant that we had a lot of people just sort of implementing a tech stack. And nobody was signing off on it or authorizing it. And then we get a bit of a mishmash where someone's using, like MongoDB, but someone else's using SQL. And then when we're trying to come together, we're sort of going, Oh, hold on, like, I built my database with Mongo, and you've built it with SQL. This isn't particularly well organized. And really, if we had that technique, like you are meant to happen, like Scrum team, you know, you would go guys, what tech stack are we using? And, you know, totally the day, we're going to go with a man stack for this one. I think just stick with man. If you encounter any problems, come back to me, and we can review it. There may be if you come to some prompts and go oh, yeah, okay, let's switch the lamp then. But, yes, so there's that element that, like I'm trying to, I know, it sounds quite negative when they bring it up. But I'm trying to sort of point out the like, this is the key things that really make Agile, brilliant. Because, you know, Agile boils down to sort of like being good at communicating with the customer and the team. Being like, sort of free to do your job. And also, to like, sort of maintain standards across the whole team. And, you know, back to what I was saying about like retrospectives, like this customer walked out and didn't pay us in the end, the year turned out, there wasn't a contract in place, which is a completely separate issue.

43:23 RESEARCHER:

Wow.

43:23 PARTICIPANT 32:

Yeah, but basically, we got to the end of this project. And the closest thing we had to a retrospective was the manager of the project, telling everyone how big a success it was. But unfortunately, the customer walked out. You know, the developers tried to raise and sort of say, No, well, actually, no, this was not a success. Like, yes, we've learned some new things from it. But what our problem is, is that, you know, we didn't apply the correct procedures in this place in this place in this place. And this got all ignored. And it was just chosen that in favor for the narrative of going like, no, it was all successful. And this is like one of the key things with Agile is sort of like, if you're just painting everything's a success story, which is a very common problem all over the world, particularly in like big companies. But you're not getting that growth ever.

44:23 RESEARCHER:

Yeah, you don't learn from your mistakes.

44:25 PARTICIPANT 32:

Yeah. And you just encourage people to essentially be hacks, because they've realized that if you're not interested in their passion, or their sort of desire to do a good job, then they're not going to be motivated. Agile basically breeds motivation. Because it's about I think that's a really good point actually, just like Agile makes you motivated because of the sort of a The freedom to get on with your job with minimal sort of involvement from other people. And also, as we said before, about the Gold's thing, where it's just satisfying to develop to an Agile framework.

45:09 RESEARCHER:

So, this motivation that Agile brings into the software development team, does it motivates you to write high quality code?

45:17 PARTICIPANT 32:

I would say the code quality is directly related to like mental health, and things like that. So, there's a lot to be said for sort of dopamine, right? With humans, and like, you see, are you familiar with the rat experiment where they had like a dopamine button for the rat, and the rat kept pressing it, pressing it and pressing it to get the dopamine high? Yeah. Yeah. So, humans are different, but not entirely dissimilar from that. And I think it boils back down to like, the goals and tasks element where it's sort of like, Scrum provides you with that structure of like, here's a ticket, solve the ticket tickets done. And because, you know, you've got that visible achievement sense. Because you know, the ticket goes from to do to done. It's like taking a box on, like your own to do list. And that is satisfying, because it's, it's progress, you can, like, it's measurable, you can go well, actually, I sold 20 tickets this week. Like, yeah, that's pretty good. I'm happy with that. And, you know, it's just sort of like, that breeds the motivation, because then you're sort of like, Oh, well, I can turn this into a bit of a game now, like, Can I do twenty-one tickets this week. Now, there is also the pitfall of getting obsessed with that sort of thing, and then beating yourself up personally for not doing enough tickets. But you know, if you're working in a very positive environment, that's encouraging and supportive, then you don't make it about like, how many tickets you could do, but just sort of like just challenge people to do ever so slightly better and stuff. But we're moving into the realms of like psychology and gamification of work, which honestly, has, has a lot to say basically on, like, code, because code quality literally boils down to, are your developers happy? And are they going to put a lot of effort in because they are happy? And if they're not happy? They're almost certainly not going to put the effort in? You know, that's a lot of a lot of everything just boils down to whether people are happy or not.

47:47 RESEARCHER:

So, you talks about this sense of self achievement that a Scrum or Agile brings into the environment? Does it motivate also the developer to keep investing on software quality on writing good code?

48:06 PARTICIPANT 32:

Yes. Again, from like, what I have witnessed with it, and from what I have personally experienced, like I've been through the more Waterfall approach, and it's quite miserable. Like, you know, it's difficult to figure out where you're at what you're doing and what's going on. Whereas in Scrum or the more Agile way, you know, you just go to your manager and say, I've got this problem. It requires like me to take three days to go train, to learn to use this new tech stack. And in a good Agile working environment, they go away, they authorize all of that. And they say, you've got a three-day course to learn that tech stack for the customer, you know. And it will motivate you because you've got the tangible outcome and that like the real sort of like application to it, it's not like just, I'm going to go on this training course, because I need to know AWS, it's more of a sort of like, we need AWS because this customer wants their hosting on AWS, I do not have the skills for AWS. So, I'm going to go and learn it. And it's just about all of that sort of like clear goal setting and just managing people's expectations. Because managing expectations is one of the biggest nightmares of the programming world, like it's the whole sales approach versus the development approach where sales will promise the moon and their development will go. We can't do that. Well, we had a very famous cases where we promised like natural language processing to the customer. And none of us had experience in natural language processing and none of us had any training to do that at any point. In the end, we never delivered it because we didn't have the capability. But yeah, sorry, I digress. But yes, I don't know how else to put to words that I find that Agile in itself is just a very sort of motivating work structure and yeah the more focus I get with my coding the better is the quality.

50:14 RESEARCHER:

I think you covered it very well, I just want to regarding time, we have only ten minutes left, and there are a lot of things I want to go through. Are you happy to keep going for fifteen or ten minutes or twenty minutes?

50:36 PARTICIPANT 32:

Yes. Okay.

50:39 RESEARCHER:

You highlighted some qualities of Scrum or Agile, I list all of them. And I will go one by one. And I will ask some challenging question. You've mentioned that the team has an open-door policy or a flat policy, you talked about the tasks are measurable and achievable. And you also talked about Agile makes you more comfortable working, and a sense of achievement, we talked about the sense of achievement, we're not going to repeat ourselves. And you talked about breaking the task into small chunks. And you talked about also avoiding distraction for the developers. So, I'll start with the first one, open door policy or having a flat structure? How does it help achieving software quality, how this quality of Agile or Scrum enables a developer or software development environment to achieve software quality?

51:49 PARTICIPANT 32:

So, the flat sort of perspective which is talking about like nonhierarchical Yes. That has been one of the most positive working experiences I've ever had in my life. Often, people, like myself, can struggle with hierarchical structures, because often you will encounter that person that has no sort of merit being in the position that they are in. Yeah, this was part of why I left [Deleted to preserve the participant anonymity] because, you know, my team was bringing me all these different sort of issues. And I was trying to bring them to managers to get them resolved. that's neither here nor there, though. What is important, though, is that when I was working under one of my managers, we had a strict like, no hierarchy policy between us, I could come to him. And I could bluntly say anything that I had, on my mind, that was a problem. And he would never question whether or not it was like, an actual problem to me. He'd be like, okay, I've heard you, what can we do to fix it? Or like, have you considered it from this angle? And, you know, we would just have like, a completely open and candid conversation about like, maybe like, our tech choices, or perhaps one member of our team was really struggling with learning to code. And, you know, I would come and say, like, person x is having a really terrible time with this, and they can't coach it, you know, we need to do something about that, because they're not getting the support that they need. You know, and while that is quite a blunt way of putting it, he always knew that I wasn't coming to him to be horrible about this person. I was always going like, this person is struggling, that means we're failing as like, individuals, because we're not supporting them. Because, you know, like, if they're not learning that's just reflecting on us, the more senior people in the team. And this is what I sort of mean. So, what was the question again?

54:14 RESEARCHER:

Yeah, so I'm just going to recap what you said.

54:19 PARTICIPANT 32:

Flat and hierarchical.

54:19 RESEARCHER:

Yes. Flat structure is what you said flat structure enables transparency. That's what they understood anyway.

54:27 PARTICIPANT 32:

Yes. Yeah. It makes it makes everyone feel like they can have a say, and they have a voice. And there is an appropriate way to dismiss a view or to not take on board a view and there is not an appropriate way to do that.

54:43 RESEARCHER:

Yeah, so this quality, how does it help the developer to write high quality code for example?

54:51 PARTICIPANT 32:

So, it gives you the capacity to essentially have no fear of like, repercussion. From your manager, because, again, to draw a comparison, like when I had my good manager in the company, I could say literally anything to him, there would never be a consequence for it, he would recognize that I'm just trying to make the project as good as possible. And I'm trying to help everyone on our team, by all of us improving and growing. However, under the bad manager, I got, you know, I bring him these problems, and I was labeled as negative and complaining. And, you know, that sort of thing. And it's like, the difference here was that my manager viewed us all as equals, in the first case, and in the second case, the manager view disorders subordinates who should do as they're told. And, you know, that is the that is the key difference here is like, when we all felt heard and appreciated, we were driven to improve. Whereas when we felt unheard, and unappreciated, nobody could be bothered to maintain the code quality and things like that you would, you'd have more people just disengaging from work, and sort of like, copy and pasting off of the internet instead of Yeah, we've all been there on Stack Overflow, just grabbing the latest dead. Sea. Yeah, like this is, in my view, why the flat structure is so much better because it just sort of it lets nobody likes to feel like oppressed at work, and nobody likes to feel what I'm heard at work. When we developers feel safe we put extra effort on quality because nobody would tell us why and there are no fear from consequences.

56:32 RESEARCHER:

Fantastic. You answer it very well. I'm happy with that. So, I will move to the next quality you talked about is, is Scrum enables you to have measurable and achievable chunks of work or tasks, how does that help produce in code quality or software quality?

56:55 PARTICIPANT 32:

So, in my view, this is down to I've mentioned modular code before.

57:01 RESEARCHER:

Yes.

57:01 PARTICIPANT 32:

And the reason that I think is really good for that is because it implicitly states that you need to break your projects down into modules. And like, having that element of like, you take one story, and it's got like six tasks under it, which could have like three subtasks each, right? You now know, when you're creating your module, you know, if we're, say, browsing the shop, and the story is, as a user, I want to be able to browse the shop so that I can choose products, right, you can break that you create your first module, which is, you know, essentially a class in a way, but, you know, I won't go into that sort of thing. But like, you create this first module, that is your shopping browse thing, you then take each stories and you create new modules under those, and then you create the functionality in those modules below that, and below that, and through the tasks and stuff. And does that make sense?

58:02 RESEARCHER:

Yes, I understand. I was a software developer myself; I understand the technicality I used to be anyway, years ago,

58:12 PARTICIPANT 32:

I'm used to people not always getting what I'm all about with this, I just like to check to make sure I'm being clear about it. So yeah, like, that's the thing is, it's like it creates this implicit relationship between good readable code and quality code where it's sort of like, you know, saying, like, create this one bit that does this one thing, it is separate from everything else. Because, you know, this was that was the term I'm looking for. It's back to that sort of like the temptation to just lump things all together in the same index file or whatever. And then you end up with like, very messy and chaotic code. Whereas in this approach, you can literally just map it out really neatly. And everyone knows what you're talking about. You can say, oh, story number three zero three zero four, right. Can you go check the module three zero three zero four, and everyone knows you're talking about three zero three zero four. So, they can navigate directly to that bit. Unfortunately, I find a lot of people don't practice that because, you know, it's, well, it's in some circumstances, people do practice it, but it requires everyone to sort of chip in basically. And if we refer back to sort of like that motivation side, if people aren't feeling motivated, then they're not really going to do the maintaining of quality code. I can speak for hours and hours as you're probably guessing.

59:48 RESEARCHER:

Yeah, I'm just going to recap. I'm just going to recap your answer to measurable and achievable and you can comment on that. So, for a developer when he or she is encounter with achievable and measurable tasks, it enable him to be in control of the code. And subsequently, they will write better quality code?

1:00:14 PARTICIPANT 32:

Yeah, in essence, but not only that, it's to do with like scoping, where it's like, you have a small task. So, your focus is on producing that functionality. And it's, it's almost like unit testing, and things like that, like, you create this small piece of the big picture. It is easy to test because it is small, limited in scope and functionality, you can then test that for his behavior, and everything like that. And then, because it's so small, you can use in your like Git source control. If you've got automated tests and things in place, you can literally just push that into the main code base. And it's, it will function, I will admit, I find this a problem with programming languages that scoping isn't very well managed. Like, you have to do all of the declarations for your scoping. And then that gets quite complicated. I think there is a lot to be said for managing scoping better. But that's a completely different conversation to have about programming in general.

1:01:25 RESEARCHER:

Yeah, and I guess the thanks for recapping that. I will move to the next quality of Scrum or Agile you highlighted. And we will try to make correlation between this quality and this quality of Agile and achieving software quality. So, you said Agile makes you more comfortable at working, or it makes the work environment more motivating or comfortable using your words. How this quality helps the developer to write better quality code and better software quality?

1:02:05 PARTICIPANT 32:

Would you mind just repeating that once more for me?

1:02:05 RESEARCHER:

Yeah, so you said Agile makes you more comfortable working. So how does this quality helps the developer to write better code?

1:02:19 PARTICIPANT 32:

Sure. So, when I'm talking about this, it's if you are feeling comfortable at work, and biologically speaking, human beings need to feel safe. And it's like down to the Maslow's hierarchy of needs and all of that malarkey of if you have your needs met, you're going to feel less stressed and less worried. And there is quite literally scientific evidence, the having your needs, unmet reduces, I'm not a fan of IQ, but like, but it's got a noticeable and observable effect on your IQ to not have your needs met. They did like a study on like plantation owners who spent ninety percent of the year in poverty and like ten percent of the year, the wealth, and for like ninety percent of the year, their IQ dropped. But when they had all the money, their IQ went back up again, which is your essentially your decision making is impaired, if your needs are met as a human being, which we're moving into the realm of like diversity and inclusion, as well, which is amusing. But it's sort of like, if you are feeling comfortable at work, you will make better decisions. Because you know, if you're living in fear of like the manager cracking the whip on you, you're living in fear of the manager cracking the whip on you. And that's not going to encourage you to do your best that's going to encourage you to cut corners including quality.

1:03:49 PARTICIPANT 32:

And that's the thing. It's kind of like dealing with crunch culture, which is a really big problem in big industry programming companies, specifically the video games industry, because, you know, like, I'm going to use a very specific example here, but [Deleted to preserve the participant anonymity] you might know their subsidiary [Deleted to preserve the participant anonymity] in Sweden.

1:04:17 RESEARCHER:

Yeah.

1:04:19 PARTICIPANT 32:

So, [Deleted to preserve the participant anonymity] used to be a fantastic, wonderful company back in the day of like Battlefield two and things like that. They used to produce really like quality games from the heart that were very enjoyable. And then [Deleted to preserve the participant anonymity] started to get more and more sort of like crunch culturally on the muscle like pushing them for deadlines. And the quality of their games visibly went down at that point. actually, noticed several developers that work there, and I think it's improving lately but you know, sort of like this is what I mean by the comfort thing is sort of like if you are forcing you Want to work to this like insane deadline, which is, unfortunately, a product of trying to optimize costs for the customer. And like, you have to play this balancing act here of sort of like, the customer wants to spend this much, what is actually feasible for that much. And also, like, trying to juggle that your business wants to make money, those are two, like goals that are not aligned. And I think that's there's a lot to be said for, like crunch culture just being like a clash of sort of needs between customer and business. But, in essence, like, what I'm trying to get at is that when your developers are feeling comfortable, and that they are relaxed, not to the point of just sort of like, laid back and not doing anything relaxed, but sort of like, just they feel like they can take a step away from their work when they need to. or, like, they feel like they can just reach out to a colleague, who's not too busy to turn around and say, Look, sorry, I'm swamped come back later, you know, like, they just want the environment to work in which is, like, just comfortable. And that's how it brings about the quality because it just enables cooperation better, because you don't get like the snippy dev who sat in the corner carrying the weight of the whole team with his shoulders. Because, you know, he's basically sat there going, I'm keeping this entire project to flow on my own. Because I'm not being assigned the resources. And then everyone goes to him for like, this is a specific individual I used to work with I can think of the he actually was he's brilliant coder, like one of the best JavaScript devs I've met. But unfortunately, that led to everyone coming to him asking him questions. And he was overloaded with work, as well as overloaded with questions. That was not good for him. And it wasn't healthy for him. And, you know, I'm very confident that his code quality decreased, he definitely stopped caring as much about the quality as he got more and more stressed. But yeah, like.

1:07:14 RESEARCHER:

So, I'm just going to recap. You said something very interesting, this safety that Agile brings to the working environment, you said, the developer makes better decision and does his best. So how do you correlate better decision and do your best with producing software, software quality or code quality?

1:07:40 PARTICIPANT 32:

So, it's standard, like, you've got a two-week deadline, if you are juggling trying to get the tests done, trying to fire fight the latest bugs, then you've got the technical debt and all of that, that's just sort of there. And you don't have the sort of processes in place to manage tackle that as it comes up, then you start to feel there's a good metaphor, which is like you're juggling some balls when you're a programmer. And you know, one of them is testing, one of them is code maintainability. One is like, I don't know, something else like architecture or whatever. And if people keep throwing you new balls, eventually, you're going to drop all the balls, because you can only juggle x balls, which is down to the individual of how many balls they can juggle. myself, personally, I could figure it if they only juggle, like three or four balls, any more than that, and I start to get too overwhelmed. But I've met people that can juggle like ten. Figuratively, again. And that is, that is exactly what I'm trying to get out with it. Which is just, you know, it's, it all boils down to basically treating each other like human beings, not like objects to us up and expand. And I think that is actually the crux of why Agile is so good. Because it's not just like, a way of working it's like a culture thing as well. It's, it's just about its competition between each other, but positive competition. So, like, you're not competing to beat each other. You're competing to lift each other up. Because the outcome is all of your outcome versus like the outcome is like the one person that's the sales guy who put that project on, you know, like, so you've got ten devs, none of them will get the credit for the work that they've done. It'll just be the product owner going to his bosses and saying, Look at this amazing work I've produced. He gets all the bonuses and stuff and the devs are left feeling sort of well we did all the actual work here. So, what's the point whereas you know, that just encourages you to compete in a negative fashion where you like push people under the figurative bus to get ahead, whereas the Agile sort of approach is like you know you're all invested in this, you're all pursuing the same goal. The point is to make the product not to make yourself look good to somebody else. Or I mean, you could argue that you're trying to make yourself look good to the customer. But like, the point is, you're working on this one project together to give it to the customer. And the customer is going to go nice. Thank you so much. That was brilliant. So yeah, like, that's my deal now.

1:10:37 RESEARCHER:

Yeah. So, the last quality you mentioned that Agile brings to work or one of the condition perhaps, is avoiding distraction for developer. So don't distract the developer and you cited a negative example. How does this help the developer to produce a quality?

1:11:01 PARTICIPANT 32:

So, from my perspective,I really don't deal well with interruptions. So, I will just caveat here that my opinion on this is going to be slightly different to everyone else's viewpoint on this, which is, you know, for me, if somebody keeps pestering me the things that are actually pestering me that that's how I view it, because I don't like interruptions. But if somebody interrupts me, while I'm in the middle of a task, that is very detrimental for me, because I will struggle to either, I'll be brusque because I want to focus on the task. And if I break my focus, then that's bad. And I'll take forever to get back into it. So, someone might come say, Tom, can you help me, and I'll be like, no, go away, I'm trying to focus on this. I'm not trying to be horrible. It's just, I'm trying to keep my concentration on something. And that consumes all my resources to do so. Because if they come up to me and interrupt me, and I let them interrupt me, it might be three hours when I get back into the task. Now, on the flip side, I have seen similar things where it's like having the customer come to you and asking questions while you're trying to work on something, it muddies the water of the expectations that they have. Because let's say you've set a story, let's go back to the analogy of the shopping basket. And they've asked you to create a shopping basket. And you know, they've given you three stories, one that says you need to be able to check out you need to view the basket, and you need to be able to delete the basket. Yeah, that's, that's three clearly defined goals. But if you've then got the customer messaging you or calling you and going, I've had a really good thought, really, really good thought, what if we had a button on there that let you just like, add everything in the shop into the cart, you now have to take that on board.

1:13:01 PARTICIPANT 32:

Because the trap with Agile is that you're taught like the customer gets what the customer wants, and also interact with the customer to ensure they get the product they want. But that's a trap. Because like you've been given two weeks, you've agreed upon what is plausible within the two weeks. And now they're bringing you something else. And it's so easy to fall into the pitfall of just saying, Yeah, sure, we can get in, that's not too difficult. But of course, you're making a snap judgment, rather than, like a rational assessment of it. And this is what I mean by like, the distractions of bad is sort of like you will make like, you want the person to stop asking you questions. So often, you will just react and be like, yeah, that's fine, we can do that. And that's the, that's the sort of like, the quality problem that I find is sort of like, you end up taking on more and more tasks, particularly if you're like, this is where we move into the role of like personality types and stuff where some people are very, like, people pleasing. Now, that's not a bad thing. But they will say yes to everything. I have a colleague, she was very much like this, at one point, she would say yes to virtually everything. And she ended up having to take time off due to stress. And I took on some mentoring with her and just sort of said, Look, let's talk about when to say no to things, because it's okay to say no. But again, this is all about the quality thing. And it's sort of like by taking on too many things. By being distracted by saying like the customer coming and saying can you do this? Can you do that? Can you do that? Combined with say your manager coming along to you and saying, oh, delegate and then passing you his tasks or her tasks or whatever.

1:14:55 PARTICIPANT 32:

You're just increasing somebodies workload. And as such, you know, you're expanding the scope of their role. And they've only got eight hours in the working day to do that. So, if they if they Our working day is meant to be like do three stories on the task board. If you add more things on to it, they've now got less time to do those three stories. Now, I know it's obviously sort of like, there's some leeway with this, because obviously, those three stories may not take the full day. But that's up to the individual to assess. But the problem is, is like, they need to seek the work themselves. Like if they've, if they finished their tasks of the day, there's always more they can do, they like using an Agile framework, you can say, I've done all the stories that I'm supposed to do for the day, obviously, if there's some tech debt wants to fix, or if there's some bugs that I can repair, and like that sort of thing. And that's why impact on the quality because it's like, if you suddenly expect your developer, to also be, you know, like, the complete team of sales and developer and everything else, all in one go, they're going to get overwhelmed, they're just going to have a hard time with it, because you are asking them to do more, and more and more. And, again, that boils down to the quality again, because you're going to sacrifice quality a lot, because it's like the old thing of like, cost, speed and quality. If you're asking for speed, you're going to sacrifice, like other things, namely, quality. And this is what I mean is like, if you're overwhelming your devs, they're going to have to work faster, not harder, or smarter. And so, the quality decreases.

1:16:49 RESEARCHER:

Fantastic. Just the last question. Before we conclude, how does for example, Scrum, or Agile, from your experience, help finding bugs? If you can give me a good example, how is this facilitate finding bugs, for example?

1:17:11 PARTICIPANT 32:

So, I'm working in a good Scrum team. It's very easy to spot bugs. I can think of a lot of cases of this because I used to be like a programming mentor, for a lot of the junior members of my team. And I had like, again, like my open-door policy was just pop me a message if you're stuck. And, you know, one occasion, I had one of my colleagues come to me and say, I'm really struggling with this SQL database stuff. Could you please help me? I said, Sure. Look in a slot for like later in the day with me. And I'll, I'll sit with you for like half an hour to an hour and sort out, they brought me the code. And I took one look at it and just went, Oh, geez, I was like, You are querying your database for everything. And then you are pulling that into the browser. And then you are sorting that with JavaScript in the browser. And they were like, Yeah, and I was like, right. Okay, so number one red flag here. I was like, SQL is faster than JavaScript, it's sorting your data for you. So, I was like, so you don't want to be selecting all of it. You want to be writing your criteria in SQL, and then injecting that to the database. And then there's like, and then all of this code that you're having a problem with here, I was like, it's got, you don't need that anymore. I was like, like, this is I was trying to be very tactful about it. Because, like, it's very, this is more speaking of my employer and their inability to train people. But, you know, like, I was sort of like, No, no, like, this is how you need to do this. And that would have gone into the project. If we hadn't had like that Agile mentoring sort of thing.

1:19:00 RESEARCHER:

So, what to say. And just to recap, because we have one minute left, just to recap this collaborative and safe environment, create this communication and helping each other's and by collaborating, it's facilitate finding bugs. Like the example you use the right?

1:19:29 PARTICIPANT 32:

Yeah. And this was a very sort of, like common occurrence. Again, like the adjacency list thing I mentioned previously, someone just brought me their database for a project and said, like, Tom, you're pretty good, like, this sort of thing. Could you just take a quick peek at it for us? And you know, I spent five minutes looking at it, I said, this has got a big fatal flaw in it. And I was like, I don't know how to explain it to you. But I'm going to try my best.

1:19:59 RESEARCHER:

And also accepting feedback, accepting that they made an error. That's why they go to their colleagues and try look for help, because this quality is not always in every working environment is another colleague come to you and say, can you look where the error and accepting the mistakes?

1:20:23 PARTICIPANT 32:

Yes, I think is a really key point to draw away from this as well, which is back to those concepts of like safety and things like that is that if you don't have an environment that encourages that feeling of safety, and like inclusivity, and just sort of like, just healthy working environments, you are not going to people will be stressed. And as a result, they will be less open to admitting to mistakes and taking on board feedback. Because when you are stressed out, things that are much more sort of simple in terms of like, someone could say, hey, just, you know, made a mistake with the SQL database, I fixed it. Now, that becomes a personal attack, if someone is feeling really stressed out and unheard at work, because gender and like if they're sad, they're juggling, like all those balls, and I come along and add the feedback ball into that juggling, that might make them drop all of their balls that they're juggling, and then they will lash out at me, because I was the one that basically made them drop the balls. And really, it's not that I've made them drop the balls, it's actually they were juggling too much to begin with. And as a result, dropping the balls was what made them annoyed. And it's not my fault that I've made them annoyed. It's just unfortunate happenstance that I happened to be the was the straw that broke the camel's back, basically. And like, and that's the thing is like, most work conflict tends to boil down. Are your employees feeling relaxed and comfortable and safe to do their job? And are not overloaded? versus, you know, are they feeling like that, and there's a lot, there's a lot to be said, basically for like, the transitionary phase between Agile and Waterfall is particularly tumultuous. Because, you know, you tend to have a lot of older developers that are like, oh, Agile is a buzzword. And then you've got the younger developers that tend to be sort of like, no Agile is really good, we should do more of it. And that people not very good at conflict resolution I find so unfortunately, that doesn't go so great sometimes.

1:22:42 RESEARCHER:

So just to recap, safety creates this safety and collaborative environment creates this feeling of ease to admit your mistakes and look for help. And subsequently, it helps achieving better quality. Right?

1:23:03 PARTICIPANT 32:

Yes. And being able to control your work is also really key to it to feeling sort of comfortable and safe and everything because you know, if you have the choice to go, I am going to do two tasks today. And then you sort of go, No, I can't, I can't think do two. I need to do one today, you know, maybe my mental health is off or something. I don't know, like, whatever my reasoning is, I need the ability to sort of say, No, I'm sorry, I cannot do that now. And, you know, in a less Agile approach, that's not as plausible because, you know, it's, I mean, it's, on one hand, it's easier to get away with doing nothing in a Waterfall approach, because it's harder to track. But, you know, again, it boils down to the intent of why you're tracking things, like if you're tracking someone's progress with the intent to fire them or something, that's bad. But if you're tracking someone's progress, and then you're going to say, hey, you are underperforming at the minute, are you okay? You know, and they go, Well, actually, no, you know, my nan just died and everything like that you get okay. I'm now going to I now see that you are having problems because you know, your problems are visible and transparent to me, because the numbers are slightly down. And I've come to you and asked, like what's up rather than going, you're underperforming, go. You're now just going, Hey, what's up, and it makes less of a sort of like diversity inclusion, nightmare for a company because, you know, if you show that you care about your employees, they're less likely to sue you. Because the law, particularly in the UK is very sort of like clear cut around health and safety and things like that mental health. You know, like, this is just some extra opinions stuff that I find. It's just like, I find that I interlink everything a lot. So, my mind tends to work in like a mosaic fashion, where it's lots of interlinking points all across this thing and you have the unfortunate job of trying to piece together Oh, All the little pieces that I'm presenting to you, but you know.

1:25:18 RESEARCHER:

I know. So, we talked about admitting errors and correcting those errors. So, when you correct errors, it leads to better quality, right?

1:25:28 PARTICIPANT 32:

Yes, it's also like one of the fundamental main ways to learning. Because, like, even I, as a more senior developer, welcome a junior developer to call me out on something that I've done wrong. Because I don't know everything. I know a lot of things. But I do not know everything. And, you know, like, if a junior dev came to me and pointed out something that was, you know, they actually like, you know, you're using like, I don't know, I'll use a really dumb example. But sort of like, oh, you're using for loop here. And that's no longer compatible with, you know, Chrome, version six, or whatever, you know, I'm not going to turn around and go, how dare you, I'm the senior developer go away, I'm going to go, Oh, really sick. Thank you for that. I will try and not use their old for loop anymore. And this is like, that's one of the key problems that we often find is I have a saying that I like to use, which is sort of like, experience is great. But sometimes experience just leads to bias. And you need to be able to reassess your experience through the sort of like retrospective lens of has things changed. And a lot of people in sort of society struggle with this element, because people tend to form their perceptions, and then never really pre evaluate them based on new information. But that's just the sad problem with a lot of sort of people as it is. But yeah, so.

1:27:17 RESEARCHER:

Okay, we have four minutes left. So, can you share with me an example a concrete example where admitting error has led to a better code quality or software quality?

1:27:29 PARTICIPANT 32:

A good example of that. Can I think of a good one? I can think of a halfway decent one, which was a colleague of mine, we were having a discussion about branch programming and branchless programming. And he sort of went to me like, this is all very like, academic who's like, you know, we don't need to think about this sort of thing. And I said, Well, actually, I was like, it's not, because I was like, looking at this way as like, What's an if statement? He was like, would you make I was like, Well, what is an if statement? I was like, it's where you create two possibilities in your code, or more possibilities within your code. And I was like, What? So, what is that? He was like, I'm not sure if I was like, it, that's a branch, you have two possibilities, two branches if you're programming at that point. And I was like, so creating a branch of program is more efficient than having a branch program. And I was like, so you know, by eliminating if statements through, there's some really complex shenanigans that you can do to do this. I've watched a lot of content, that it's interesting and very complicated. Perhaps not entirely practical, but, you know, I sort of saved him just like, you know, so actually, this is a lot more important than you're currently thinking it is. And he actually sat there when he was like, that's it. He was like, when I think about it like that. He was like, that's a really good point. Cause he was like, I was sort of sitting there going, like, who cares about like, branching? Like, but then he was like that. Now, you've pointed out to me, the programming is branched in some way or another. It suddenly becomes a big deal. And I was like, right. And I was like, and that's, I'm, I'd like to think he's taken that on board and possibly learned from it, but I don't know. I think it's just good that he didn't go, No, shut up. You're wrong. Yeah, it's, I think there's a really famous computer saying, which is like there are no problems that you can't solve on a computer without an additional layer of abstraction or something like that. And I feel like that ties in quite nicely because you're presented with a problem to solve, and you just need to approach it from a different angle to always solve it with a computer.

1:30:11 RESEARCHER:

So, he took your feedback on board. And subsequently it's a better quality, right?

1:30:17 PARTICIPANT 32:

And my hope is, is that he would go forward and possibly consider the, like, overuse of if statements that we may have been doing in the beginning. And the overall reliance on def statements, I'm hoping that he went forward and thought, Hey, you know what, maybe I could cut this down a bit. Maybe I could actually use a switch instead. Because that would be a more effective method if they still branched, but like, I could use a switch instead. Because that's a more effective way of handling cases. You know, like, the hope is, is that basically, it's created that thought, and he will now go away and think about that, and possibly learn from it in some ways. I can't, I can't speak for him what he took from it, you can take the complete opposite from it.

1:31:07 RESEARCHER:

Okay, fantastic. I think we used enough time; I think we use an extra thirty minutes. And thank you for your extra time, I really appreciate. do you have any questions for me before we finished?

1:31:25 PARTICIPANT 32:

No. I'm just if you need any, like further questions.

1:31:29 RESEARCHER:

Yes, sure. I was going to ask you, because we draw conclusions from these experiences you shared with us. So, we like to validate them to workshops. It's, I will organize workshop with other developers, and they will present the findings and get your feedback on those findings to just to confirm your understanding, and we capture everything. So, it will be less talking. You won't be doing all the talking. You will be asked to provide your feedback in some statement. So, would you like to participate?

1:32:08 PARTICIPANT 32:

Yeah, I'd be more than happy talking to. Are you publishing a paper?

1:32:11 RESEARCHER:

Yes, yes. If you'd like a copy of the paper.

1:32:16 PARTICIPANT 32:

Is it in Danish or English?

1:32:17 RESEARCHER:

No, no English of course, we will write our result in English.

1:32:22 PARTICIPANT 32:

I am like moderately capable of Swedish. But mostly written, not conversational.

1:32:32 RESEARCHER:

No, we would like in English so everybody can understand and this we can disseminate the knowledge. In Danish, the audience is very small. So, I will get in touch once I organize the workshop and I will be happy to talk to you again. And it was a very interesting conversation. I learned a lot from you. Thank you very much.

1:32:56 PARTICIPANT 32:

No, thank you very much. It's been an absolute pleasure to have a chat with someone about Agile on like a more sort of not like conversational level but just as a sort of a...

1:33:09 RESEARCHER:

Intellectual level, stimulating your intellect. Yeah. And stimulate your thinking of how you see it. I'd like to thank you again, and I wish you a good day.

1:33:20 PARTICIPANT 32:

Yes. And to you too.

1:33:20 RESEARCHER:

Bye.