00:03 RESEARCHER:

Hi PARTICIPANT 37, how are you?

00:05 PARTICIPANT 37:

Hi RESEARCHER, fine thanks.

00:06 RESEARCHER:

Good night, or it's a bit early in here. Well, it's midday, it's lunchtime almost. So yeah. Yeah, you are the end of the day. I'd like to start by thanking you for the opportunity to talk to you. I really appreciate and doing this interview. I'm looking forward to talking to you. So, I'll start by introducing myself, telling you what I do and why I'm doing it. And we can go from there. So, my name is [redacted]. I'm a researcher at the I.T. University of Copenhagen, one of the universities here in Copenhagen, Denmark. And my research interest is to understand how software teams achieve or manage to achieve quality. So, my current research project, I'm looking at Scrum, whether it does help to enable software team to achieve quality, and what specifically make Scrum good at helping developers to achieve quality. So, what I'm using, I'm using practitioners experience as a source of knowledge. So, I talk to people like yourself, I capture their perspective. And we use that experience as a source of knowledge to draw conclusions. And once we draw conclusion, we write a research report, and we disseminate the research report. So yeah, briefly, this is who I am and what I'm doing. And if you have any question regarding that, I'm happy to answer.

01:57 PARTICIPANT 37:

No, no. I was born in Denmark, so I'm familiar with Denmark.

02:01 RESEARCHER:

Ah, so you're Danish?

02:03 PARTICIPANT 37:

Yeah, I am yeah. I've been Australia for a long time.

02:08 RESEARCHER:

[Danish language]

02:09 PARTICIPANT 37:

No, no.

02:10 RESEARCHER:

You don't speak Danish, okay.

02:12 PARTICIPANT 37:

No, I've been back to Denmark a couple of times now. Sweden. Sweden a bit more than Denmark.

02:16 RESEARCHER:

Okay. Yeah. Well, Sweden is prettier than Denmark, of course.

02:23 PARTICIPANT 37:

In parts.

02:24 RESEARCHER:

In parts, yeah, it's prettier than Denmark.

02:27 PARTICIPANT 37:

I like the center of Copenhagen.

02:29 RESEARCHER:

Okay, yes it is authentic. So the structure of the interview is I do have questions I like to go through. But it's fluid. We don't have to be rigid and stick to the questions. If you have anything you'd like to add, feel free to add it at any stage. Okay. So, if we can start with a presentation, if you can present yourself talk to us briefly about your experience.

02:59 PARTICIPANT 37:

Yeah, certainly. So, I've been in software development for over twenty-five years now. Started out in uni and sort of straight into finance, developing optimization programs and various other quantitative models and fixed interest in equity and balanced portfolio management. Then moved into one of Reuters companies doing Reuters graphics, their flagship graphics product, and worked there for a while, then went into company doing sort of freelance ecommerce development. We worked on a major project for an operation in Australia dealing with the, all trade and transport coming into and out of Australia. So, I've developed an ecommerce gateway for them with [redacted], a whole invoice billing hub and messaging system, all that sort of thing. That was very successful. Then worked for [redacted] as a senior architect working on ecommerce for their [redacted] for quite a while and set up an office in Atlanta and hired developers over there and all the rest. It was very exciting work too. Where did I go from there, jeez? I worked for one of [redacted] companies for a while in software development. Then finally worked for another couple of banks and then ended up with [redacted], which I've worked with for fifteen years or so. I started off as a senior developer and then became CTO after a couple of months and stayed as the CTO for nearly eight years with them. They were Australia's sort of flagship FinTech provider in the treasury risk management space, did extremely well and destroyed all our competitors in Asia Pacific and started expanding and then the founders sold the company. So, I was CTO there running all the development teams, left there and started my own company in online safety and built up a number of solutions there. That was interesting. And then ended up going back to [redacted] again, as development manager and software development. Later on, about two thousand and fifteen until about two thousand and

nineteen, and now with another company called [redacted] doing development there in the advertising industry.

05:38 RESEARCHER:

That's a long journey.

05:42 PARTICIPANT 37:

Definitely. A lot of different industries. A lot of different companies.

05:43 RESEARCHER:

Yeah, that's great. It's good. I'm looking forward to seeing what your perspective is. So, you've seen different organization's structure, and you've seen different development software development methods. So, what do you think of Agile? And more specifically, what do you think Agile has brought new to software development?

06:07 PARTICIPANT 37:

Very interesting, indeed. I mean, I worked on a lot of Waterfall projects that you can imagine going back twenty-five years, and then sort of move more towards evolutionary prototyping type of SDLC processes. When Agile came along, it was a very sort of radical shift in the way that it was done. And it was sort of thrown all over the companies and became a buzzword very quickly. And companies sort of brought it on without actually understanding what that meant, then tried to fit a process onto their existing business without actually ever understanding it. I think in the early stages, it was quite badly done in the companies I worked for, in terms of again, people trying to figure out how it should work. And sort of trying to set up this massive, sort of rigid framework for Agile, which is not what it's meant to be. They made up with way too many processes and too much control over it. Unfortunately. But when I think in terms of the positive aspects of it, this is now years down the track now, and Agile's evolved somewhat, I think, it definitely helps with respect to Waterfall. And the other sort of processes. Definitely helps in terms of team collaboration, in terms of making the cohesion between teams better, making the interactivity among teams a lot better. It makes work a lot more fluid, I think, is the best description. So, Waterfall, as you know, is a very gated sort of SDLC process. Whereas Agile sort of breaks down those barriers and makes the whole process a lot more fluid.

07:54 PARTICIPANT 37:

You end up being able to solve problems a lot faster, especially in stand ups and the various other sort of Scrum type style meetings. Planning is better, collaborative planning is better, you can deliver better to your goals. Waterfall you tend to sort of upfront try and say, this is what we're trying to achieve, but never really collaborating on it. You just set a piece of work that you wanted to achieve, and process has very thumbnail sort of guesses on it. And hope for the best. A lot of the time it fell down because everything got stuck in the various gateways and business planning and analysis, development and testing, etcetera. Whereas Agile tends to remove a lot of that sort of gateway processing, where the testing is done with the developers, the planning is done with the developers and the testers. So, it's a lot more fluid in that whole scheme of things. I think in terms of...I can keep going...

09:02 RESEARCHER:

Yeah, that's the qualities that most people bring about Agile. Currently you using a Scrum environment in your team.

09:11 PARTICIPANT 37:

Yeah, Scrum. Definitely, Scrum.

09:12 RESEARCHER:

Yeah. Can you take me through the process, the development process, and how did you implement Scrum in this particular case?

09:21 PARTICIPANT 37:

Yeah, I think I'll probably go back to my previous company.

09:24 RESEARCHER:

Yeah. Yeah. You could use that.

09:27 PARTICIPANT 37:

So [redacted] had four Scrum teams operating in Sydney and another four in Manila and another five or six over in Chicago. So, the Scrum team was set up in a particular department I was in Five developers, one tester, one product owner and Scrum Master, which was shared across Sydney, among the four different content. So, the process reasonably got I wouldn't say it was a regimented process. It was more designed as we needed the various different recommended processes we adopted. So, we did our daily stand ups every single day to make sure the team was on track with what was done and what was coming up and resolved any problems immediately during the morning. We did our sprint planning with the team using Mastermind, it's like using sticky notepads to estimate things on a whiteboard. As they suggest you do. Then moving the sticky notepads around the planning boards to try and fit it into the two-week time frame for the sprints. The development process itself, again, very collaborative two-week sprints, working towards a monthly release, typically. Retrospective always after the sprints. So, we'd get together as all the four different sprint teams, and go through what each team thought did well, what the whole Sydney team did well. And what we needed to approve and all that sort of thing. Those were the main processes.

11:18 RESEARCHER:

Do you think that was a good implementation of Scrum?

11:22 PARTICIPANT 37:

I think it worked quite well. Yeah, I think in terms of how we had it going, like one of the Scrum teams didn't do stand ups. They used, I think it was Trello, they were using just the manage their particular workflow. And they found that a lot more fluid than standing up every single day and turning their way through it. It was more of a regular collaboration every single second of the day. The stand ups worked best for our team, because we had a

different mix of seniority. And I'd been with the company for a very long time, had a lot of knowledge, and some of the other guys were quite new. So it was, it was easier just to get everybody standing together and talking through things. I'd found the planning sessions were very hard in terms of whiteboarding and trying to figure out what you can do, and what will fit into a sprint. And they seem to take a lot longer than if you were just project managing that sort of thing. Talking people through it.

12:28 RESEARCHER:

Yeah, obviously, because it's collaborative, you bring everybody's opinion. There is not always an agreement, of course, yeah. It will be more consuming to make everyone happy.

12:39 PARTICIPANT 37:

Definitely, yeah.

12:40 RESEARCHER:

Instead of having a hierarchy where the project manager decides.

12:45 PARTICIPANT 37:

Yeah, look, I've been doing project management for twenty years or so. I was sitting in a meeting where people make very elementary sort of decisions, and then talk about it for twenty minutes.

12:57 RESEARCHER:
Yeah, the upcoming questions is about quality, and in particular software quality, but we need to define it before we go ahead. And from your perspective, as a software development manager, how do you describe, how do you define quality, especially in an Agile software development environment?

13:20 PARTICIPANT 37:
Yep. I think there's probably two views of quality. One is the quality that comes from not introducing any more bugs into a system, as well as continually improving your system without introducing any more issues. So that's sort of the quiet code quality SDLC maintenance quality aspects. Then there's the sort of external facing quality, which is more, I suppose, a perceived perception of quality, more so than what's happening inside the code base itself, which could be from anything from UX experience through to performance. How clients actually interact with the system and perceive the system was changing, irrespective of how the development team thinks it's changing. So, two different aspects, I suppose.

14:24 RESEARCHER:
So you talk about two aspect, which is fit for purpose, the user think it fits their purpose. So that's one aspect. And you talked about performance, and which is part of internal quality. How about the code, the design? Do you consider that part of quality?

14:46 PARTICIPANT 37: `
Yes and no. From an overall oversight of code and maintenance, sort of sustainability of the code and the ability to maintain easily to reduce development timeframes. Yes, it definitely helps to have practices in place to peer review code and to have different code quality checking software, whether it's Resharper, or whatever the case may be. That definitely helps. But in terms of quality of software, at least for me, the focus has always been at the end user side of things more so than the maintenance side of things.

15:28 RESEARCHER:

Okay, great. So, you can talk about your current or previous team as you wish. So, in your previous team, for example, what did you do to assure software quality in that Scrum setup?

15:46 PARTICIPANT 37:

So, we did very regular peer reviews. With every time you're, you were trying to push anything through to the Done stage, you'd have to go through the pull requests inside. From memory, we were using Microsoft DevOps to do that. And all of our team were very good at peer reviewing code, which was good. We had; we didn't have any regimented code standards in place in terms of automated checking. But we definitely had well understood practices around how the existing code was in conformity to that. So that was one part of it. So, we were talking specifically about code?

16:42 RESEARCHER:

Yeah. So, the quality assurance stuff you had in place to assure quality. So, you talk about peer review. Do you use linters for quality checks for code conformance?

16:59 PARTICIPANT 37:

I haven't, no.

16:59 RESEARCHER:

Yeah. Okay. Yeah. So obviously, you had some testing, what type of testing you have?

17:07 PARTICIPANT 37:
Yes, we have, obviously, manual testing for features. Yeah, so any development going through their feature branches would have manual functional testing, at the UX level. They also had API level testing, using Cucumber and most of the tools hitting our API's, as well as automated testing of the actual UX side of things. So, there were quite a lot, quite a lot of automated testing going on, as well as the functional testing. So, the sort of the running plan of every single sprint, was the tester would be doing the functional testing of each of the feature branches. Then, when the code was peer reviewed, and ended up into the main code base, they'd be doing another functional integration testing that code with the rest of the system. Then they'll be updating the all the automation scripts to cover the functional testing they just done. And then by ensuring that was then put in into the mark, as your DevOps, to make sure that it actually it was automated every single time a build was done. As well as all the API scripts, which were predominantly just run by the testing team as a whole on a release basis, as well as the sort of automatic UX test. It was more of the gateway check.

18:38 RESEARCHER:
I'll come back to those later. At the beginning, when you talked about Agile, you talk about few qualities. I'd like to understand if any relationship between these qualities and the ability of a software team to achieve quality. For example, you talk about Agile makes the work environment more collaborative, cohesive team, fluidity. So, in your opinion, and from your experience, that collaboration helps achieving software quality.

19:17 PARTICIPANT 37:

Oh, enormously.

19:19 RESEARCHER:

Yeah, how?

19:20 PARTICIPANT 37:

It's great to have developers sitting there and inside their own little bubble, developing code. If you're not able to talk to peers and get ideas on how to structure your code, and how best to solve problems, one, you can end up developing the wrong thing. You can end up writing code that while it might achieve a goal, it might be totally unmaintainable going forward. You might end up writing very inefficient procedures or stored procedures that are just ridiculous. You might end up spending days on a problem that your teammates could help you solve in an hour. You might, for example, in talking with the team, especially the testers, they might help you pinpoint places in code that you haven't paid attention to. And they need to pay attention to. And even when you're talking about your own development and the sort of thing you're trying to develop. Sorry, I just had to close that. Even when you're talking about your developments in your stand ups, the testing team can also think of other ideas, the things they need to be testing. So, they're not in a bubble, either. I mean, there's all sorts of benefits to collaboration.

20:45 RESEARCHER:
So sharing knowledge and helping each other’s, you become better at writing code. And you acknowledge your mistakes, is that what you're saying?

21:01 PARTICIPANT 37:

Or you definitely become better at writing code. You also become better at helping others write code, and helping, sort of helping the team work better together.

21:16 RESEARCHER:

So yeah, so you talking about...

21:17 PARTICIPANT 37:

I don't know if it's as much about acknowledging mistakes as it is learning from them.

21:25 RESEARCHER:

Yeah, learning from your mistakes. Yeah. Yeah, of course. Yeah. Yeah. So, you're talking about cohesive teams? Can you explain to me what does mean, and how, in your perspective, or from your experience, and how it does help achieving software quality in an Agile or Scrum environment?

21:47 PARTICIPANT 37:
Yep. So, one of the one of the major problems I've had with Scrum, where you have not particularly just the Scrum teams themselves and their pods, they tend to be fit work fairly well, and be fairly cohesive within their own team in terms of collaboration, discussing things and keeping workflows flowing, and all the rest. And even like, ensuring their own little pod is producing high quality output, that tends to work quite well, where you get issues with their jobs is across Scrum teams. Because you're effectively isolating pods of developers with testers in their own little functional areas. And if you're not careful, the different Scrum teams won't end up talking to each other. And you'll end up with cross integration, quality problems across the teams. The hardest problem, I think, to solve with Agile is ensuring that the collaboration works across all these Scrum teams as well, and that somebody is looking after that process. What was the question again?

22:51 RESEARCHER:
Yeah, how does a cohesive team or this Agile quality being cohesive as a team helps the teams to achieve or to deliver better software quality?

23:05 PARTICIPANT 37:
Yeah, I mean, from a cohesion perspective, one getting along with your teammates obviously makes it a lot easier to interact with them. You know, be able to easily go talk to your developers and easily talk to your product owner and product managers and all that sort of thing. And have a good, familiar relation with the team members. Rather than if you go back to the Waterfall, for example, the sort of division between your business analysts, your developers and testers was always very, what's the right word there, sort of a heated sort of friendship, in terms of throwing things over the fence and everybody getting annoyed that various parts weren't considered, and no one ever asked questions and all that sort of thing. Whereas in Agile the sort of cohesion between everybody eliminates all that all that angst, I suppose. As far as the cohesion across the functional teams, or the pods in a Scrum sort of environment, helps keep the quality at an integration level better. I think that perhaps that's one of the things that Waterfall does better than Agile, in my opinion, in terms of Waterfall tends to look after the whole process and the whole solution rather than Agile, which breaks it up into pods. So, you end up with different problems. I suppose when you solve that and get the teams working again, cohesively together it's a good word for it. So yeah, the teams aren't afraid of talking to each other and feels like they actually can. Then again, it helps the quality because everybody's talking to everybody and can help each other with problems and even once you break down the ability to share resources among the pods, which I know, our company was quite resistant to doing that when they first started, it was sort of like you set up a Scrum from pod. And that was your pod. You can't talk, you can't go in and work for a different pod, you just stuck in that little pod, even if the other team is struggling, which is quite stupid. So being able to sort of open that that fluidity between the different pods, helps the whole work better.

25:27 RESEARCHER:
Okay, I will push you to give me an example. You're talking about eliminating barrier and ease to talk into each other's helps quality? Can you give me a concrete example from your own experience?

25:41 PARTICIPANT 37:

Yeah, okay. So, for example, in my last company, I've been there for a very long time. So, I understood perfectly all parts of the system. And I was in a particular team, which dealt with financial instruments, and the mechanics behind financial instruments and cash flows and all that sort of things. You know all the banking sentiment flows and blah, blah, blah. The other team, right next to us, literally five feet away from us, worked on the risk management side of the solution, which I knew very well, because I built it in the older solution. And I suppose, initially, I wasn't able to talk to them at all. Even though they were struggling to try and figure out how to design things, and how to build things, well, and making a lot of silly mistakes. One because the product owner didn't understand risk management. And the developers had absolutely no idea. So, I think from a cohesive perspective, once I sort of decided just to break down that barrier, and just go and talk to them, and help them through things, to understand how they need to develop things. And sort of introduce them to the other parts of the company who previously they weren't allowed to talk to. Once, unfortunately, what happens, I suppose is when their job first gets set up, we get this structure where product owners sort of take control of their pods. And they sort of get a bit defensive of developers and testers going to talk to other parts of the business and other people with knowledge. And they sort of require them to talk to the product owners only which sort of breaks down, knowledge sharing, and all sorts of things. But once you get past that, and you actually open it up, so people can talk to each other again, the whole Agile experience works a lot better. And you end up with people with knowledge open, they help other people who are struggling. And so, the whole quality just slips, so does the output and productivity,

28:01 RESEARCHER:

You talked about this cohesiveness enables sharing knowledge. How does sharing knowledge facilitate better quality?

28:12 PARTICIPANT 37:

Oh, so an example. Okay. So, again, using this same sort of the same two pods, where he was working on risk and working out financial calculations around risk management. So, for example, they went through this period where, for whatever reason, they kept developing the wrong thing and kept getting it wrong for a long time. And it wasn't until I stepped in and helped introduce them to the other quants and the business that we're able to sit down and teach these people in the other pod team, how this stuff was meant to work. Because we've been doing it for ten years. If you didn't have that sort of knowledge sharing, they'd be sitting in a vacuum trying to figure things out, getting things wrong over and over and over again. And clients are just screaming. In the meantime, where we had some absolutely idiotic bugs just because people wouldn't talk to each other. We had one team who developed some stuff in the cash management side, which just to get a basic cash count working, which took close to a year. Because again, they weren't allowed to talk to anybody who actually knew how these things worked. It’s just kind of absurd. If you don't have that sort of...

29:51 RESEARCHER:

Yeah, yeah, go ahead.

29:52 PARTICIPANT 37:

If you don't have that cohesive solution in place, and that sort of ability to collaborate and share knowledge with people that actually know how things are meant to work, you can end up developing the wrong thing. And your quality just follows. Again, it's the difference between code quality versus what client quality expects. It's a very different thing. They could be developing beautiful code. Don't get me wrong. And I've seen some magnificent code that's absolutely wrong, doesn’t deliver the right functionality.

30:27 RESEARCHER:

That doesn't work.

30:29 PARTICIPANT 37:

Yeah, so it's two different perspectives regards to quality.

30:32 RESEARCHER:

They could be writing beautiful code, but because of lack of knowledge, they could be writing something that doesn't work, right.

30:39 PARTICIPANT 37:

That's exactly, yeah. Yeah, exactly.

30:42 RESEARCHER:

Yeah. Okay, you're talked about something very interesting, the fluidity of work that Agile brings to a team. Can you explain me what do you mean by that? And how it helps achieving better quality?

31:01 PARTICIPANT 37:

Sure. So, fluidity, yeah, I mean, I love leading. Typically, because I've been sort of lead developer, I love leading teams that are very fluid in the way things work with very low friction, to get things done. So, in terms of whether its developers being able to work through issues quickly, whether its developers being able to talk to testers easily and resolve testing problems easily. Whether it's working out problems in specifications quickly and easily. Just finding very efficient ways to get through the development process. without friction. Again, it's relative to Waterfall, which was extremely, very, for an SDLC, designed for friction, I think maximum friction, I think is the best way to put that. Whereas Agile is just totally the opposite. If you don't Well, I think the first time we started doing Agile, I suppose the friction was very high, we've got a couple of managers who like to micromanage things. And they sort of held up everything in terms of controlling everything and introduced friction where it didn't need to be. And I think that's a danger for a lot of companies starting Agile is they start with what they know, which is management, which is very hard to let go of. I think once you get that sort of fluid operations working well, fluid development, your whole process

becomes more efficient, your ability to code better becomes more efficient. Your ability to peer review code becomes easier. Even simple things like if you're writing code, and you're trying to figure out a problem, being able to quickly talk to your teammate. and solving that problem in a couple of seconds is a huge, a huge benefit to improving code quality and an output of what you're trying to achieve. Even being able to work with the tester as you're developing things. So, they can tell you problems and you can help them identify problems. It just helps each other enormously, rather than the tester and try to sit in a vacuum and try and figure out okay, well, they're developing this thing that's been written down What do I need to test in a vacuum and then the developer not realizing that this particular thing had to be tested, and they weren't expecting that as it wasn't written in the spec. It solves those sort of problems quite easily.

33:48 RESEARCHER:

That brings me to the next quality you talked about, is solving problems. It is a little bit obvious that it helps quality if we talk about functional or errors in the code or errors in the usability etcetera. Can you give me example or elaborate a little bit further how it helps producing better quality?

34:15 PARTICIPANT 37:

By solving problems.

34:17 RESEARCHER:

Yes.

34:19 PARTICIPANT 37:

Yeah, I suppose, simply put is if you are able to identify issues whether from a client facing perspective, they helping you identify problems in your software. And even internally, being able to identify problems and raise them in the issue tracking cycle of things, in the issue planning cycle and things. You can obviously continually improve quality. It's one of the problems that we had early on with a job was that the product owner was sort of planning everything and everything everybody else was sort of subservient to what the product owner had planned without being able to appraise issues as they found them. So, for example, a developer would be working through cash flow calculations or accounting calculations for some particular loan or bond structure and working out effective rates or something like that. And they would find that find a bug in the current version was causing ridiculous results. Because that wasn't on the current trajectory of planning by the PO, they couldn't actually raise it. Nobody would listen to them. So, they just had to sit there knowing there's a bug. And it was at the point where the unit tests were being developed to confirm the bug was correct. Even though rather than trying to fix the bug and write proper unit test to ensure the system work, they'd be writing unit tests, knowing the bug was there and having to pass an incorrect unit test. And that's what ends up happening in that process. To being able to easily raise issues as you find them and get them into the development cycle, as well as clients finding issues and telling you about them, it helps makes the whole system a lot better, very fast. But again, it's breaking down that barrier, and allowing the team to be fluid. Not having these gateways of you know, you can't, because the PO manages that, or you're not allowed to talk to him about this. Yeah. Those sort of issues.

36:36 RESEARCHER:

Okay, fantastic. I like the example. So, you also mentioned that planning is better in Scrum, or Agile. Can you elaborate how it is better?

36:50 PARTICIPANT 37:
Yeah, I have two views on this. One, because I have one, I've done a lot of things. So, I don't mind the Waterfall approach. If it's done well, and you have somebody that understands how to plan well, yeah. And somebody who's able to talk to people and figure things out and, and not to sit in a box and do it themselves. Without anybody's input that can actually work kind of, well, it can speed up their process. What I find that Agile does better though, is having everybody come together. And for a sprint that's planned for two weeks, for example, to actually all agree on what they can achieve collectively, within those two weeks. And while people still get it wrong, because estimations are really a hard thing. And people tend to underestimate a lot, especially developers. What it does do though, is help the predictability of what's going to be delivered on time. So, it helps from an oversight perspective, in terms of release management and expectations of clients and all that side of things, helps improve time frames and helps improve predictability of what's going on. I think, overall, I think the, I don't know if it necessarily helps quality though. But it definitely helps achieve goals better and achieve the planning being undertaken to solve problems, which I suppose is a route to quality improvement.

38:29 RESEARCHER:

Isn't it, I will challenge you in that one, isn't it when your goals are well defined and better evaluated from a time perspective? Or whether we can achieve them or not, doesn't help you to focus and subsequently deliver better quality?

38:50 PARTICIPANT 37:

Yes or no. So, it comes from the sense of helping you achieve what you've agreed you're going to achieve. But what tends to happen I found is that where a team underestimates, or even overestimates, you can spend time achieving what you've set out to achieve and then being sort of stagnant for a while waiting for more work or pushing into other Kanban activities while you're waiting for the next sprint to start. But I think there's less of a driven approach to Agile. So, I think Waterfall does a better in the sense that somebody is overseeing things a lot more stringently, to make sure that quality is being adhered to, to the timeframes that are trying to be achieved by the company, relative to the team setting their own goals, to try and achieve what they think they can achieve, irrespective of what the company is trying to achieve. If you know what I mean it's a very different perspective. So, while the company or the development manager, whoever might say, okay, you've got two weeks, here are the issues. You should theoretically be able to achieve this much. The team might say we don't think we can do that much; we'll do half of it. And there's no, there doesn't seem to be that ability to contend that by the development managers to saying, look, you can do more, what you're saying is rubbish. You know, there has to be that sort of compromise, I suppose, between the teams themselves in the Scrum approach versus the Agile approach versus what oversight needs to achieve. And I think that's where our job breaks down a little bit relative to Waterfall, in my experience anyway.

40:48 RESEARCHER:

So, you mentioned that Agile helps meeting the goals? Can you elaborate a little bit more on that? How does it do that?

40:55 PARTICIPANT 37:

Ah, yeah, definitely, I think from the perspective of being able to communicate goals to the business, it seems to do a lot better in terms of, again, providing predictability around what's going to be delivered. So, it's not so much achieving goals of what the clients need to be delivered, or what the business needs to be delivered from a sales perspective, and those sort of things. But more so predictability of the goals of what the development team is able to deliver. And I think it gets a clearer picture internally, of what is expected to come out of a particular release, and the goals that are being achieved.

41:40 RESEARCHER:

Does this quality help the team to achieve better quality?

41:47 PARTICIPANT 37:

That's an interesting question. That depends a lot on the rest of the process around how the roadmaps being managed, and how the incoming issues are being managed, relative to what's being prioritized, I think it's a bit of a different problem there. Like solving issues around quality has more to do with the various inputs to the process, rather than the process itself. And how those inputs are managed. You'll get, for example, issues around quality coming from the sales teams from demos, you get issues coming from clients, as they're using it, you've had issues coming from developers, as they developing, you get issues coming from testers that haven't been able to be resolved during the sprint, you'll get issues from even people implementing and releasing code for things that don't quite work at the release time. So, they coming from lots of different angles. And it depends how well the product owners does, sort of the simulating all that input into the roadmap, in terms of what's been delivered by the team. And the team, depending who they are, and how they manage things might end up picking the wrong things to be delivered, which can happen I mean, you can get a team, I have been on some teams where some people are more, what's the right word, hardworking and more dedicated to the job and dedicated to delivering high quality than others. And when you end up with teams that aren't that dedicated, they'll tend to pick issues that they're more comfortable with, rather than solving hard problems that need to be solved. And again, if they have the ability to choose that and plan accordingly, without oversight, then you can end up with worse quality. Again, you still need that oversight coming in to make sure that somebody's watching the roadmaps overall, that things aren't just slipping through the cracks because nobody can be bothered to solve them.

44:08 RESEARCHER:

Okay, great. Thank you. You talked about some Scrum ceremonies. You talked about retrospectives. How does, I know what a retrospective is and most people familiar with Scrum, they do know what's retrospective is. How do they help the teams to improve their ability to meet quality expectations?

44:39 PARTICIPANT 37:

Yeah, that's a good question. I think that one obviously if you're going through what you did well helps cement good practices and especially around things like peer reviews or things that work that have been delivered or met expectations. Well, all those sorts of things again, leading to on the quality side of things. In terms of analyzing things that weren't done so well helps the team to continually improve themselves. I'm sure you've heard that sort of thing a thousand times but being able to focus on things you haven't done so well and, and sort of owning up to mistakes, I think it's hard for some people, but as long as you're doing in a

good atmosphere, with the right intentions, it can help the team learn and to help improve practices. And whether it be coding specific coding problems they've had and how to improve their style or different feedback from senior developers at the time of retrospective rather than trying to get in people's faces during development time. It can really set quality somewhat. In terms of expectations, as from the team as a whole, I think the retrospectives do a good job of allowing everybody to have their say, without judgment, I think that's what it does really well. Especially when you're looking across teams, where you'll get the sort of feedback that obviously different teams perform at different speeds and achieve different things quality wise, whether it's coding or output or number of bugs, all that sort of thing. And having other teams challenge other teams helps, you know, just if they are producing a particular feature that is particularly buggy, or took way too long in testing, or whatever the case may be. Being able to challenge other teams can help. You can sort of get collaboration between the senior development leaders to try figure out how to improve that going forward. That's a huge thing that was missing in the whole Waterfall process.

47:03 RESEARCHER:

Fantastic. You talk about the stand up as a quality of Scrum, does it help? How does it bring or how does it enable the team to deliver better quality?

47:17 PARTICIPANT 37:

The daily stand ups?

47:18 RESEARCHER:

Yes.

47:19 PARTICIPANT 37:

I think it's good from a perspective that people struggling with problems around how to deliver what they're trying to deliver. It can get very fast solutions for other people who know how to do things, or I've done it before, or can help them navigate their way away from issues. It can help the testing get obviously better insight into what's happening and have a daily interaction with everybody in terms of what they're seeing versus what the developers are seeing versus what the PO seen as the Sprint's evolve. And as the developments come out, it doesn't let anybody hide, you know what I mean? So, Waterfall lets people hide very easily. I can hide for weeks, without people knowing what's going on. Whereas Agile and the sort of daily meetings as long as it's in good spirit. And people understand it's in good, good, all good intentions. Yeah. It enables everybody to sort of open up the journey a bit and say, you know, this is what I'm seeing. You know, we better fix it.

48:41 RESEARCHER:

So, this transparency that the standup brings?

48:46 PARTICIPANT 37:

That's the right word, yeah.

48:46 RESEARCHER:

Yeah, that's brings to place. How does it help achieving quality? People point out their error with confidence, and others point out other errors? Or how it does do it within a team?

49:07 PARTICIPANT 37:

Yeah, I mean, it's not, from my experience, it's not so many people pointing out other people's errors, as much as people asking for help.

49:16 RESEARCHER:

Okay.

49:17 PARTICIPANT 37:

Yeah, yeah. So, I mean, you'll get occasions where somebody will be, again, you've only got a very short period of time to go through things where you'll get people talking about what they're developing or testing or whatever. And they'll be describing it, and somebody else will just tell them no it's the wrong way to do it. It's very open. It's very frank, it's very quick. And when somebody is challenged or presented with a better way of doing things or sort of redirected to change what they're doing for the better. It allows the team to go off and quickly work that out in a very, very quick period of time, rather than nobody knowing for a couple of weeks.

50:02 RESEARCHER:

Okay, great. We talked about some quality assurance practices that you have used in your previous teams, a peer review, testing automation, this stuff has existed for decades. I mean, it existed in Waterfall. And it has been promoted by software engineering community for years to achieve quality. Did Scrum or Agile bring anything new to these practices or made them better in a software development team?

50:42 PARTICIPANT 37:

Good question. I think in terms of manual feature testing, definitely. In terms of the, again, the transparency and the ability to see what's going on and interact with the team and sort of even confront development on the spot, helps enormously in feature testing cycles. In terms of developing automation, I agree, it's been around forever. But I think what Agile brings is sort of a bit more collaboration between the automation testers who typically are developers and typically struggle with development problems, trying to develop automation software, to be able to get help from the developers who do it for a living very easily, rather than again, sitting in a black box, trying to figure it out themselves. It just opens up those doors to make that easier, I suppose. It also, it also helps from a perspective of it depends how a Waterfall has worked in the past and how the company operates. But it can also help in the developers helping highlight better ways of automating testing. And you know, for example, if they developing a particular API with particular authentication methods, they can help the development team understand how to test that and how to write automation against it. So, it opens up that sort of discussion and dialogue. Again, people have been doing this forever. But you know, it makes it easier.

52:22 RESEARCHER:

I'm going to recap it for you and confirm. So, the collaborative and the transparency environment that Agile brings to the pictures, make these practices more efficient and work better.

52:38 PARTICIPANT 37:

Definitely.

52:39 RESEARCHER:

Okay. Fantastic.

52:41 PARTICIPANT 37:

That's a nice summary.

52:44 RESEARCHER:

Yeah, yeah, yeah. Okay, I'll move to the next question. We've been talking about it, I just want you to summarize it, because you've been a software developer for thirty years. So how does for example, Scrum, or Agile methods, from your past experience, help you to produce high quality code?

53:09 PARTICIPANT 37:

We've talked about a lot of it already.

53:11 RESEARCHER:

Yes. We talked about it a lot but if you can summarize it for me in a good story would be nice.

53:18 PARTICIPANT 37:

Yeah. Give me a sec.

53:20 RESEARCHER:

Take your time.

53:35 RESEARCHER:

I can ask another question, which is much easier. So, this is a better question. I mean, how does a Scrum environment or an Agile environment motivate you to achieve better code quality as a developer?

53:53 PARTICIPANT 37:

Yeah, that's a good question. Definitely. In every instance I've come across, it definitely motivates us to do better for your team. Because you're working interactively, every single day talking, collaborating, discussing what's going on, confronting issues, challenging other people's conceptions, and practices. So, it's a continual challenge to improve yourself. So again, I think the Agile approach of you know, there is no "I" in Agile, it's a team thing in terms of responsibility and accountability. I think that's goes a long way to enforcing that, I suppose.

54:47 RESEARCHER:

Yeah. You mentioned a good word, which other developers and other Agile practitioners mentioned, it makes you accountable to your team. Does making you accountable to your team incite you or motivate you to write better code?

55:05 PARTICIPANT 37:

Me personally, yes. Some people probably not. Yeah, depends on the personality. I'm sure.

55:13 RESEARCHER:

There are some people, whatever you do they never accountable.

55:18 PARTICIPANT 37:

That is very true.

55:19 RESEARCHER:

I worked with those type of people. I mean, whatever you do they never accountable. They never care.

55:27 PARTICIPANT 37:

But again, it depends how cohesive the team is, and the sort of relationships you get between the people. And I think forming the right people in a team in the Scrum environment is vital. If you get the wrong people, it just won't work. But it definitely doesn't make you accountable.

55:48 RESEARCHER:

So, let's, I think this is more granular question. How does for example, Scrum, or Agile in general helps, for example, or facilitate finding bugs better?

56:06 PARTICIPANT 37:

Probably, I would say in terms of more oversight of what's happening, both from a peer perspective, as well as interplay between product owners and teams, and that sort of thing, retrospective wise, and even daily stand-up planning sort of stuff as well helps, again, the

transparency side, and the visibility to what's happened is dramatically improved. That's the main thing.

56:48 RESEARCHER:

I had the last question, but I'm going to change it given your profile. You worked in Waterfall for year and translated to Agile after that. We talked about it is obvious from your answer. You lean towards Agile; it works better, at least from the team level. And you highlighted some example where Waterfall bit better. But from a quality perspective, do you think one method works better than the other method to deliver quality?

57:23 PARTICIPANT 37:

I think Agile works better overall because you end up, the problem I struggled with Agile when I was moving towards it was the desire for lack of requirements specification. That's what I really struggled with. The approach of having minimal specs and talking your way through things as you went. A lot of developers struggle with it, a lot of testers struggle with it. And a lot of development managers struggle with it. Yeah, I mean, if your developers that don't know what they're doing and the PO goes absent for three days, you're in real trouble. So that's what I struggled with initially. But I think in terms of overall quality, and as sort of impacts that sort of ability to have on hand easily somebody who's able to solve problems quickly for you. And not having sort of siloed stages of development, where things can go wrong. Causing enormous problems down the track for release management, definitely improves the whole quality side of things.

58:36 RESEARCHER:

Okay, fantastic. We come to an end. Before we conclude, what we do is we draw conclusion, and we summarize them. And I'm planning to organize workshops with developers, to take them through my conclusions and to get their feedback, just to validate our understanding. And if you like to participate, I can get in touch with you.

59:13 PARTICIPANT 37:

Yeah, sure.

59:13 RESEARCHER:

All right. I'll get in touch.

59:19 PARTICIPANT 37:

I'll be interested to see what you come up with.

59:19 RESEARCHER:

Yeah. I would be interested to get your feedback after I talked to everybody. So, I'm looking forward to talk to you again. And thanks for your time and good night.

59:32 PARTICIPANT 37:

Can I ask you a question?

59:32 RESEARCHER:

Yeah, yes. Sorry. I didn't give you opportunity to question at the end. Yeah. Do you have any question before I leave, sorry?

59:40 PARTICIPANT 37:

I was just going to ask how many people you're talking to?

59:41 RESEARCHER:

I talked to a lot of people so far. I started with talking to managers and I talked to a few Scrum masters. Not a lot, because their role is to facilitate but they do have a perspective. I talked to a few product owners, and I talked to a few QAs. And now I'm talking to developers. I left the developer at the end. So, because they have the most important say on the process, and I asked for more challenging questions, or sometimes confirming what the Scrum master or what the product owner told me, so I pushed the developer to give me more concrete examples. Because at the end of the day, they are the one who produce what we will consider quality. So, a total so far, I'll give you the total. I talked to thirty-two people.

1:00:53 PARTICIPANT 37:

You talking to testers as well?

1:00:54 RESEARCHER:

I call them QAs because they call themselves most the time QAs. So, I talked to let me tell you, I talked to eight QAs. And some of them are key leads. So, I did get the perspective of, of different levels of testing. And, yeah, so you would participate in the workshop. So, I will summarize everything. So, the perspective of everybody and we'll present it to you.

1:01:34 PARTICIPANT 37:

Yeah. I don't mind chatting. I'm open to chat.

1:01:41 RESEARCHER:

Yeah, yeah. Go ahead.

1:01:42 PARTICIPANT 37:

And one thing I found interesting from a sort of development manager perspective, as well, even in my current job, is that because the test is part of the pods, and I work collaboratively with the developers and the POs, what tends to happen I find is the developers become unsure of what they can challenge from a testing perspective, because they're not a distinct team anymore, is able to say we won't accept this. They tend to be less confident in their ability to challenge. The challenges I've had is as Development Manager every single time is convincing the QA team, the goal, the gatekeepers of the company, and that if they don't

want to let things out the door, they won't go out the door. It's trying to convince them that they have the final call on quality. Yeah, that that tends to be the only problem I've come across with testers.

1:02:45 RESEARCHER:

That's very interesting, because most of the tests that I talked to, they told me that the relationship between them and the developers get better in Agile or Scrum. They felt just, I'm going to summarize what they said most of them, they said that communicating bugs become easier. And developers are not that sensitive or offended by receiving that feedback. Because nobody likes to be told you didn't do your work well. So, they told me that they find that in a Scrum team, it's easier and it's more transparent to communicate bugs than before.

1:03:30 PARTICIPANT 37:

Yeah, yes, it is. Yeah, I've find it different than to the, in relating to Waterfall. I suppose we in Waterfall, you've got the testing team, which sort of acts as a gateway. Whereas in the Agile model, they're given the testing it's collaborative with the developers. And I totally agree that testers find it easier to communicate bugs and easier to resolve bugs in the in that regard. And most developers don't have that pressure. Some of them are, what I find is that they develop sorry, testers lose that independence, which they get with Waterfall, yes. And there's they sort of feel that the commitment is to the team rather than the company. If you know what I mean. It's a different focus. So, they'll tend to do what's best for the team, as opposed to sometimes what's best for the overall product.

1:04:29 RESEARCHER:

Yeah, of course.

1:04:32 PARTICIPANT 37:

You've really got to help them understand that they go on for the product and so on.

1:04:35 RESEARCHER:

That's bring us to the accountability. Yeah, they become more accountable to the team rather than just doing their job.

1:04:48 PARTICIPANT 37:

Yeah, it's a dangerous thing for Agile. But again, if it's managed well, its fine.

1:04:53 RESEARCHER:

I think if this is my perspective, is not I mean, a lot of people told me if there is support from higher management things are easier.

1:05:04 PARTICIPANT 37:

Ah yeah, a hundred percent. It depends a lot on leadership.

1:05:09 RESEARCHER:

Yes, yes. And the ability for the company also and the higher management to let it go, to let go of that control that they had before in a process driven approach. Yeah. I think I get that a lot from Scrum Masters telling me that when higher management let go, we run better. We do it better.

1:05:41 PARTICIPANT 37:

That's interesting. It's not really letting go. What happens is in the Waterfall approach you get more of a management style approach rather than leadership. Whereas in Agile, it's more continual leadership that's going on. But I agree, you got to let go. And you got to let the team perform and try encourage them to perform really well. But it's a continual process from a leadership perspective to encourage everyone to do the best they can achieve. Again. Agile is a lot more interactive all the time. You can't just be a manager and sit back in your desk. You really have to be in there. Finding problems, resolving problems quickly. That's what I had to say.

1:06:35 RESEARCHER:

I enjoyed the conversation, and I learned a lot from you today. I'm looking forward to your participation in the workshop

1:06:44 PARTICIPANT 37:

Me too.

1:06:44 RESEARCHER:

Thanks a lot for your time and I wish you a good night.

1:06:50 PARTICIPANT 37:

Yeah, good night. Thank you very much, I appreciate it.

1:06:52 RESEARCHER:

Bye.