00:02 RESEARCHER:

Good morning, PARTICIPANT 31. Can you hear me?

00:06 PARTICIPANT 31:

Yes, I can hear you.

00:07 RESEARCHER:

Good morning.

00:08 PARTICIPANT 31:

Can you hear me?

00:08 RESEARCHER:

Yeah, I can hear you very clearly. Good morning. How are you?

00:12 PARTICIPANT 31:

I'm fine. Thanks. How are you doing?

00:13 RESEARCHER:

I'm doing well. Where are you from Copenhagen?

00:16 PARTICIPANT 31:

I'm from the western suburbs, Glostrup.

00:19 RESEARCHER:

Okay, yeah, I know where it is. Yeah. As you know I'm at ITU. I work for ITU. I've noticed you've been at ITU.

00:27 PARTICIPANT 31:

Yeah, that was a few years.

00:31 RESEARCHER:

That's great. How about if I start with introducing myself, telling you what I do and the purpose of the interview, and we'll take it from there. The structure is, we will go through some questions. But if you have any things you want to add, feel free. It's an open conversation. Don't feel like you have to answer the question literally, because it's not about literal answer to the questions. So, my name is [Deleted to preserve the participant anonymity] And I do research in software engineering. And in particular, my interest is to understand how software teams manage to achieve quality. The purpose of the interview, currently my

focus is Scrum teams. How does Scrum in particular, help software team to achieve quality. So, I've been talking to a lot of people, but now my focus is software developers, I'd like to understand software developer perspective. So, what we do is we talk to people through this medium of interviews, and we get their perspective. And obviously they share with us their knowledge and experience. And what we do we analyze what they told us, and we try to draw conclusions from that. And those conclusion, we propose them as finding for, ideally, a research report. So, your participation, as I said, is anonymous. Feel free to share with us as much detail as you like, feel free not to mention your employer, but if you do, your name and the name of your employer, won't appear in any documents. And once we get the research report published, we delete this materials. And the materials are within ITU environment, which is GDPR, and privacy approved. So that's the housekeeping and a little bit about me if you have any question about me and about what I do, I'm happy to answer them.

02:47 PARTICIPANT 31:

That's fine. I think it sounds like an interesting project.

02:49 RESEARCHER:

Okay, fantastic. And what I'm doing and once I managed to analyze all the input and draw some conclusion, I'm organizing a workshop with developers to present my findings and get feedback. If you want to participate, please let me know I can invite you to participate.

03:13 PARTICIPANT 31:

Ja, I think that sounds very interesting.

03:14 RESEARCHER:

Okay, I'll just put a note under your name. It may take place in August, and I will try to organize because it will involve few people. So, I'll get in touch. Okay. Thanks, Participant 31. So, we will start with some presentation, if you'd like to present yourself telling us what you do, how long you've been working in the software development industry. And yeah.

03:53 PARTICIPANT 31:

Okay, so my background is I'm coming from geography, interior [inaudible] and bachelor's degree there. After I got that bachelor's degree, I took a master's degree in software engineering at the I.T. University and have been working more or less in the geospatial software development domain since two thousand and five. So, a good sixteen years by now. And I've been working both about half and half I've been working in consultant engineering companies. And the other half, I've been working with the UN, the UN climate change secretariat, both as an internal developer and also as a quality assurance officer there. So, I've been sort of both on the development and the politician side of things and have some perspective from both sides of the fence so to stay. So, both from development and from testing. And I've been working since I guess, around two thousand and seven, two thousand and eight, mostly in [inaudible]. Recently, as you know moved to the UK. I work for software development consultancy.

05:13 RESEARCHER:

Okay, that's a long time. Thirteen years.

05:16 PARTICIPANT 31:

Yeah, so they've been of varying effectiveness and varying degrees of Scrum compliance. So, to say, in my experience, a lot of different ways to interpret Scrum and different ways to implement it more or less effectively. I think, I've seen some common sort of patterns of this functionalities that can occur in Scrum teams.

05:47 RESEARCHER:

Fantastic. We will talk about that. I'll get back to that. Yeah, that's an interesting word. Scrum compliance. I like it.

05:58 PARTICIPANT 31:

Yeah. Because it's you know, Scrum is supposed to be this, this lightweight thing, people will process and everything. And still, I think, what I see a lot is that the people, some people have a hard time not having a set process, and they like processes. So, people typically try to impose some sort of process that is not, it's going beyond what is the mission in Scrum. And I think it's especially seems to be the managerial level, which has a hard time with this sort of hands-off approach to developers, and developer self-governance, which can be apparently a hard pill to swallow for some managers. Once I have been in a, what I would call it like one hundred percent Scrum team. It was very focused; it was with full support from business and with very active involvement from business as well to make sure that we did Scrum and self-governing. And this was empowering the developers to do things the Scrum way. And that's also been my best experience with Scrum because it was amazingly effective, compared to other teams I've worked in.

07:26 RESEARCHER:

Okay, I will get back to some of these comments, because they are very interesting. But let's start with some generic question that just to get your perspective, and we will get into the details. Agile as a philosophy, and you've been talking about Scrum, which is a specific implementation of Agile, what do you think of it, in your opinion, and specifically, what has brought new to software development?

07:57 PARTICIPANT 31:

And I think it's compared to I've been doing, I mean, I said, I've been doing Scrum and Agile for a long time, there's been some non-Scrum things in there as well. Because typically, working in consultant engineering, you typically don't have, it's not your company's rules, it's the customer. So, when you're a consultant for another company, it's their processes, and so on. So, some of the things you do in Scrum, and some of the things is in Waterfall. Typically, when you're doing stuff like public, like government IT systems they have, it's more Waterfall, typically, they have a specification, you make a bid, you win the offer, or make the offer, you win the bid. And you do things to this specifications, and that's less Agile. In comparison, what I like about Agile and Scrum in particular is shorter feedback loops. I think it's very important. And very much the ability to change course along the way. So, if you see you're going somewhere that's not so good as you thought or if you have a better idea, you can actually adjust the course. And I think that's a very important component of actually

delivering value to the end user. You're talking about how Scrum helps deliver quality in software engineering and software development. And I think in order for me to answer that, I'll also have to talk a bit about what I think quality is.

09:35 RESEARCHER:

Yes, we can answer that question. Yeah.

09:39 PARTICIPANT 31:

Okay. So, I'll leave that for now. I think, the main benefits of Scrum as I see is short feedback loops and the ability to actually act on that feedback, to change course and you adapt to adapt your solution to feedback from the end user, from the customer.

09:57 RESEARCHER:
Yeah, we will get back to those qualities because that's great. And we will discuss how do they influence quality, but I agree with you that was my next question. How do you define software quality? And specifically in the context of Agile, did Agile bring anything new to this definition?

10:16 PARTICIPANT 31:

So, if you asked me what quality is, I mean, the obvious thing that people think about when they talk about software quality is probably bugs. So, there's something goes wrong computer does something you didn't expect, and so on. That's the obvious, quality mission. But I think in my experience, there's also a more sort of like meta quality, which is the software fit for purpose, does it actually do what it's supposed to do. Is it easy to use, the whole UX user experience thing, can people figure out how to use it? And that's also part of quality. And I think it's especially in those regards, it's not so much in the oh, there's a defect here, we can fix it. That's not an Agile's main strength, in my opinion. The main strength is the ability to get user feedback, and to use that to actually make the software do what it's supposed to do, and to make it function in the way that brings most value to the end user. So, they can say, okay, this is, this is technically correct what you've built, but it doesn't fit our workflow one hundred percent. So could we do these changes, and we'll get a better fit, we'll get less friction in using the system. So, I think that kind of ping pong with the user is very much helping increase quality in the solution in the end.

11:40 RESEARCHER:

But that just one side is fit for purpose. How about the internal quality of the software, you are a software engineer, so I was expecting more definition, because software engineer when I talk to them, they talk more about scalable design, code quality, and less about the user expectation, I'm surprised to see you perhaps because of years of experience, it shows your professional maturity. But what about design and the internal quality, the code itself?

12:19 PARTICIPANT 31:

It helps with that as well. But I don't, I must admit, I don't see that as the main benefits of Agile. I think you can achieve; you can achieve much of that even if you're working in a Waterfall world, you can still do, you can still do code reviews, you can do peer reviews, you can do program and so on. So, I mean, you can take some of the elements that help improve internal code quality from Agile and implement it as well in a more rigid process. I

think that's sort of like the internal process cycles of development. And then you have the interaction, the larger interaction with the outside world, sort of so to say the world outside the development team, as well. I think it helps quality in both regards. And I think you're right, I possibly didn't even think of it internally as part of Agile because it's just what we do now. And it's so commonplace to do stuff like pull requests and code reviews before we merge a new feature into a product and so on. So, it's, I guess we think tend to forget that that came from Agile.

13:37 RESEARCHER:

Sorry, go ahead.

13:37 PARTICIPANT 31:

It's just a sneeze that got stuck.

13:38 RESEARCHER:

So, what you are saying is the internal quality of software is, but we take it by default. Right?

13:45 PARTICIPANT 31:

I think, at least for me, I think I'm at a place now where I didn't even think of it as a function of Agile. Also, possibly, because there's way less friction, I think, in implementing this internally in a team, because you have your stakeholders, your teammates around the same table and say, okay, what can we do to improve code quality, and you can figure this out, and we implement it. Whereas implementing Agile in the wider context with the other with business units and so on, that takes typically more work because you have to interact with other departments, you have to interact with other managers and so on. So, there's more, perhaps it is also more explicit that that's what we're doing here. You need to have the agreement you need to do this. We do like that. The internal like getting, greasing the wheels internally in the team is much less formal. I think. Typically, you have this set up, your definition of done, sort of lay out the ground rules for how it's going to work. But it's also easier to adjust along the way. When you have it internally, I think that's also a very important thing that tends to be forgotten, in my experience is that all continuous improvement thing. People tend to get complacent. Say that, okay, this is good enough. And sort of just the improvement part becomes lost in the day to day because you also support, and you have whatever. So, taking the time out to do the retrospectives stuff can, in my experience, be hard, organizationally to get it implemented.

15:37 RESEARCHER:

Okay, just to understand a bit the context from a team level, can you describe to me how Scrum in your team’s work? How did you implement it?

15:47 PARTICIPANT 31:

So, with the best implementation, I've tried, we had a team of, I think, six, seven, developers, two of us were external consultants, and my guess five internal developers. And we had a Scrum master, a dedicated Scrum Master for this. It was organized, we had a backlog of stories from business and product owner. And they sort of gave a, they roughly prioritize the major stories. And then we sat down at the beginning of each sprint and planned. And

throughout what we, without deciding input from the program, we as a team decided on what we could get done this sprint and made that commitment clear to the product owner and the Scrum master. So, they could say, okay, it's good that you can do that. I mean, they have the chance to, to object, if you will, to say, okay, you've misunderstood me or something. But typically, they just said, okay, let's in line with my preferences as well, that sounds reasonable. And then we basically got down to this. A little bit of peer programming, but mostly people doing their own thing, because it was a fairly, the team members weren't overlapping very much in their competencies. There were a few, like two or three front end developers. But other than that, the rest of us had specific technical profiles. There wasn't much in the way of overlap with the competencies. So, it was pretty clear who were going to do what. And so, we just basically got on with it, had daily stand ups in the morning, giving feedback on what we would be doing. And from the product owner as well, sometimes. And then just basically, adjusting, if necessary, as we went along. Recording, monitoring the burn-down rate, the task completion, then, with a sprint review for the product owner and any interested stakeholders at the end. And also say in that case, the retrospective was also not a very prioritized item. We mostly just continued for the next sprint as we've done before. So that was pretty much the MO there.

18:55 RESEARCHER:

So, what's made this implementation of Scrum a good implementation of Scrum, is it because it's close to the prescribed method?

19:04 PARTICIPANT 31:

I think because there was, first of all, there was no support tasks being laid onto the team. That's a problem I've seen a lot in other Scrum teams is that along with the whole developing new stuff, we have to dedicate ten to twenty percent of your time to support tickets. And they of course, come in irregularly. You can't plan for it. I mean, you can even set time aside for it, but you can't plan when it comes. So that typically means that you introduce speed bumps, bottlenecks, because somebody just gets taken off what they were doing for a whole day to work on some other issue. And that's sort of distraction wasn't there. And there was also a very clear commitment from the business side, that they were just making a wish list, and the developers had the final say in, in sort of how to go about it. So, there was no attempt to try to manage and intervene in the process along the way. Of course, we asked for feedback when we had questions during the presentation phase with the product owner, that's also an important aspect of it, that they don't just go away, but they're available for feedback. But still accepted that we were planning and committing to delivering on our terms, which in the end meant that we could actually get down to what needs to be done. We had a predictable pool of tasks, and we had very few distractions from outside that needs to be attended to. So, I think a lot of the main factors, I would say were dedication to the actual development process and manager, dedication to trusting that people were doing what needs to be done. And a Scrum master that was a hundred percent a facilitator. And she was not involved in either the project or the development work, she was solely facilitating the process as well. So, I think that was very valuable as well. Also, because this was the first-time doing Scrum for a lot of the people. And the teams were having a dedicated facilitator was very helpful. It's a role that I've seen, given both to developers in the team or to the project manager roles who doubles as a Scrum master suddenly, which gives some conflicts of interests. So having a dedicated Scrum Master with a clear vision of what needs to be done, was very helpful.

22:27 RESEARCHER:

That's fantastic. In this software development process using Scrum, what do you do to assure software quality?

22:39 PARTICIPANT 31:

Typically, we do code reviews. So, when you're done with a feature, or when you have a prototype for a feature, you do a review. You have one or two others; other team members actually review it before you merge this into the code tree. So, you can have someone catch whatever things you've missed. I think that's the main thing I've used from Scrum is the quality. Also, pair programming can be helpful, but that's I think it's not, it's not so much quality, it's mostly for, it mostly helps developing speed and velocity of a feature development. Because you have, if you have a good pair doing pair programming, you get this sort of ping pong effect, which helps get stuff done quicker. I also think in of course, you will catch something the other guy doesn't. And vice versa, but I think it's not so much the quality, rather the velocity that increases there. I think the main thing, looking at the Scrum stuff that helps improve quality is the code review basically. I mean, there's lots of other stuff I usually do to improve code quality, but that's not Scrum specific. It's stuff like static analysis and lenses and culture, code standards and stuff like that.

24:32 RESEARCHER:

So those are software engineering practices that Scrum teams to uses right?

24:38 PARTICIPANT 31:

Yeah.

24:39 RESEARCHER:

So, nothing new that Scrum brought. So, for decades, we've been using the software engineering practices. Yeah. So fantastic. How does this Scrum set up, help the team to produce quality software? We talked about these engineering practices been around for decades. Scrum didn't bring those new, even code review, I wouldn't think Scrum that brought it. But perhaps Scrum brought the loop of feedback and the importance of feedback and the review within the team. But we are not going to go to that argument. But what Scrum, in your experience, how it did help the team to produce better quality?

25:40 PARTICIPANT 31:

I think if you look at its requirements wise, I think the change of focus from the very explicitly specified software into development more by user stories and by less formal methods, where you leave, like you actually think you leave more responsibility to the developers to figure out what needs to be done. I think this in turn forces you to reflect more on what you're doing. And that reflection, in turn, improves the quality of the code you write because you're not just implementing to a recipe, you actually have to sit down and think about what you're doing. To put it a bit on the edge, but I think this increased reflection of not being told what to do. But instead, being told what it needs to be able to do, the software you're writing. It increases your thinking about what you're doing. And that in turn increases the quality of the code you're writing. And then you get this feedback quickly. So, you can actually iterate on the code, which also improves the quality. When you go back and iterate over the same piece of code several times in short span of time frame, I think you tend to pick up more

defects, more problems than you would if you wrote the entire thing at once and had the tester or the end user give feedback on the entire thing at the end. Because then it becomes too big. I think this focus on specific stories on functionality, helps narrow down the focus on this specific thing. And by focusing on one thing at a time, you tend to pick out more of the problems with that specific functionality and the quality gets better.

27:56 RESEARCHER:
That's great. You talked about one quality in particular, which very interesting, and I want you to elaborate more, you talked about self-governance. You talk about the deliver in our terms, which also means self-organizing teams. And you talked about responsibility to the developers, give a responsibility to developers, which you made reference, I think you talk about it enables creativity within the developers. How does this being self-govern and given this level of freedom to the developer helps producing code quality or software quality?

28:43 PARTICIPANT 31:

I think it helps, especially if you have stable teams. Because I've also been in Scrum projects where you had people going off for each sprint, you got one new developer and one other developer, and had this constant flux of people on the team, which doesn't really help. But if you have a stable team, where you get to know and to trust the people who work with, you get to know people's strengths and weaknesses, and you can support mutual each other. And then I think this empowerment and the self-governance in the team very much helps motivate the individual developer to take more responsibility for his own code, but also to have knowledge on the impact that your work has on your teammates. Because when you go from just implementing to a recipe in some sort of Waterfall project, you're just focusing narrowly on this. If instead you have committed to actually doing together with your close colleagues, we are doing this and we have to figure out how it's to be done. And I think if you can get this team spirit and this intra team loyalty. It helps people have focus on each other's strengths and weaknesses. And in the end, that helps build the quality because you also know where you can go to get help. You know that, okay, you can intervene and say, let me take this task, this is my area of expertise, and you have this sense of excellence as well, which helps distribute the tasks, perhaps to the ones who are best suited to implement them.

30:32 PARTICIPANT 31:

But also, this awareness that any, let's say any, it's not that people do errors by mistake, but sometimes we're more diligent than other times. And I think by having this, by improving this feeling of teamwork, you increase people's motivation to be more diligent in the work they do as well, which increases quantity.

31:04 RESEARCHER:

That's great.

31:06 PARTICIPANT 31:

Also, also, because not yet, not just out of altruism. I mean, of course, if I have a good team, I want to help my team members and so on. But it also goes the other way, if I, if I'm in a team of peers, and I deliver something that has way too many defects compared to the others, it reflects poorly on me in the eyes of my peers. Because I think the powerful thing of this self-governing teams is that you are in the typical sort of, let's call it industrial software development setup. I'm responsible to my manager. And if I screw up, my manager comes to me. That's always the classic way, right, but if you have the self-governance team, and

the manifest, just say, build this, and figure out how to do it, then I feel professionally, I have much more skin in the game, because suddenly, it's much more transparent what I'm doing, and it's transparent to my colleagues and teammates as well what I'm doing. So, it gets harder to sort of hide behind and make excuses for bad performance as well, in this case. It quickly becomes more apparent if people in the team aren't pulling their weight, say or if they're doing subpar work. And you can I mean, some people, you can say, okay, you need to improve this. And sometimes you can say, what can I do to help you actually improve on this and see, if you're doing database work, you get more defects than if you're doing front end work. Should we take the tasks off, do you need training, and so on. So, you also have the opportunity to help people because it's, suddenly it's also in my interest that my colleague gets better. Because his performance also reflects on the product that I delivered. So, I think that it goes both ways. And I think it's very powerful, but you're suddenly more, you're not responsible, you're accountable, you're more accountable to your colleagues, rather than to your manager. Because that's the whole technical and reputational thing in it as well.

33:33 RESEARCHER:

There is another very good quality that Scrum brings into the process and the team was you have highlighted at the beginning, which is the short feedback loop. Can you elaborate how the short feedback loop helps to deliver better code and software quality?

33:53 PARTICIPANT 31:

Right. I think it's pretty simple in that it's a pretty well-established fact that the sooner you find a defect, or the sooner you find out that you need to make a change, the less expensive it is. And thus, the quicker you find these things that quickly you can do corrective action and get to where you really want to be. So short feedback loops obviously, decrease the turnaround time on these things. And more importantly, they by extension, increase the number of corrective actions you can do. So, you can do more course corrections and end up I mean, sometimes you over correct, will go too much in the other direction and so on. And you can still catch that. Whereas if you have long feedback loops, if you have the development system and then you get change requests for the next major version next year. It's very expensive to fix things and you get the outcome as you get less capacity for changes. So, apart from the quality side of it, the short feedback loops also give you, it's two sides of the same coin. They also give you increased capacity. You can either change all things, or you can if you have time to implement more things, because it's already pretty good. I think that's a very powerful part of the Scrum and Agile way of working.

35:29 RESEARCHER:

You also mentioned something which I like is Scrum compliance. I'm assuming you mean by that a good implementation of Scrum.

35:37 PARTICIPANT 31:

Yes, yes.

35:38 RESEARCHER:

Yeah. So how does a good implementation of Scrum helps achieve in software quality?

35:43 PARTICIPANT 31:

I think by nailing a lot of the things we've talked about already, the feedback loops, the teamwork, the team spirit, and the empowerment for the, for the individual developer, and for the team as a whole. This empowerment is to actually decide what needs to be done how it needs to be done, I find that to be extremely motivating. Or rather, if I don't have that kind of empowerment, I'm demotivated. Because by now, I've become accustomed to actually having this. So, it's become the default way of working. And having that taken from me, I lose interest quickly. If I'm just, being I mean, I have fifteen, sixteen years of experience, I know what I'm doing, I don't need to be told how to do it. Just tell me what you need. Right? So, it's, it's a way of keeping me engaged, actually. And keeping the velocity and the quality of the work high, basically, because it keeps my job interesting that I have to think about all these things.

37:03 RESEARCHER:

This this quality that Scrum or Agile advocates, which is empowerment, you mentioned that it motivates you. Does it also motivate the developer to achieve or to write better quality code? And consequently, better software?

37:21 PARTICIPANT 31:

I think so. Yes. Because it's if you have very specific requirements, which you typically don't have in Scrum, but in Waterfall world, you have everything specified. The temptation is very much there to do exactly what is written, and nothing else. This is good enough; it ticks all the boxes. Next task. And that means you don't think about all the edge cases, perhaps you don't think about, okay, what if? And what's the interaction with other stuff? If you just simply implement to a specific requirement, check, next. And then when you get it back, and it doesn't work, it basically turns into an argument, sort of Project Manager argument of whether the requirements are met or not. And if you look at it in Scrum way, it's much easier to say, well, does it actually do it do the thing it was supposed to do? Is that good enough? It also forces me as I said before, it forces me earlier in the process to think about interactions with other pieces of the code. And its cases and other stuff that typically gets left out of a detailed specification. And so, I think this is very, a very core part of improving quality is getting the developer to think more about the consequences of what you're implementing.

38:59 RESEARCHER:

I'll ask some more important question. And I'll go back to other questions. How does for example, Scrum as a process, helps or facilitate finding bugs, for example?

39:21 PARTICIPANT 31:

I think maybe by having your focus on stuff like user stories, which isn't Scrum specific, but it's pretty Agile-wide. It also helps focus on specific pieces of functionality, which makes it easier to test rather than if you're doing, I mean, you will still do of course, ad hoc testing where people just click around in your interface and see what breaks but I think the fact that you have described already specific pieces of functionality and what they're supposed to do helps you ensure that all of the different moving parts work. Because it's, it's easy if you get the whole system to test, and you're probably going to miss some sort of workflow and some sort of process through the software that was supposed to work but didn't. So having it broken down by stories, or whatever you call it in your specific setup, is helpful to focus testing and narrowing down. Did we actually get everything, did we get it all implemented?

40:46 RESEARCHER:

Is this new to software development, or something we have been doing for years?

40:54 PARTICIPANT 31:

We've been doing it for years but it's specific to user stories, I would say, I'm not that savvy on the story of requirements, specifications and so on. I did have a very interesting course with [inaudible] at one point at ITU on requirements specifications. But I'd say he also had some quite interesting ideas and templates from that stuff. But it's, I would say it's to the extent that user stories that philosophy during requirements is Agile, then I'd say that's very much what my view drives quality through feedback. Of course, still, the short timelines are still always important. In that, the quicker you get feedback, the closer you are to the scene of the crime. So, to say. The closer you are to having actually written the code, they say you can identify the problems and fix it. So, the temporal aspects of the feedback, the fact that you're getting it fairly quickly after you wrote the code is important as well. Because if you take a year and a half to write entire system, and then you get feedback on it, you can't remember what you were thinking when you were writing this in the beginning.

42:25 RESEARCHER:

So, it's too late, I guess. The next question, you have already answered part of it. We discussed that empowerment motivates developers to write better code. And, and we also discussed transparency and being self-govern, etcetera. My next question, you partly answered it, but I'll ask it anyway, if you have something you want to add. So how does for example, this Scrum setup you have, help producing high quality code?

43:05 PARTICIPANT 31:

As I've said, it's a combination of these things. It's the short feedback loops and the closeness to the customer to the end user, the availability. And if at the moment I'm doing internal developments, so we're developing an application for internal use. It's easier to get hold of the end user. I've been working other places where we are doing product development, and that means that you're typically talking to a sales type, who is supposed to know what the customer wants, but isn't the actual customer and that's harder. It's because they have to make guesses on behalf of customer feedback. So, it's my experience is that the feedback impact on quality and on features is greater when you have access to the actual customer rather than someone who is simply the actual user, compared to someone who is the just the owner of the product. I think that's, that's very helpful if you can get access to the actual user and get quick feedback, loops, quick feedback cycles. So that definitely helps the product quality, because they also, always find something that you're rarely the main expert in whatever the application is supposed to be doing. If you're working on the same thing over many years, you may become one and get the sense of what the user needs. But typically, you need to have an actual user who has domain knowledge to point out the things you missed, which is also easier to do the sooner you get the feedback. This feedback helps to improve quality, errors are identified sooner then corrected. And then I also think this is interesting. As I said, the motivation and the interaction with your colleagues and teammates is very helpful in keeping quality up because you get this, in the best cases, you get this we're in this together, and you're mutually supportive. And help each other with the code when you need it.

45:41 RESEARCHER:

Okay, great, PARTICIPANT 31. Thank you. We coming to the last question. Can you share with me an example or a good story where Scrum as a method has helped to better the software quality?

45:57 PARTICIPANT 31:

Well, I think it's, it's something I've experienced a couple of times in a few different teams is the realization by having these the Scrum, the feedback and the daily meetings, that people were actually working on like the same thing in different ways. So, by having this sort of close collaboration, you can cut down on waste as well and prevent problems from people actually overlapping in their work. So, this combination of having multiple developers and the product owner saying after the standup, what you say you're doing, let's just talk because I think there's, there's something here that we're missing something. So that's it's really a communication, the communication aspect of it. Where you have the daily stand ups, like the visibility of tasks on the Scrum board, or whatever the awareness of what your colleagues are doing helps to identify or to pinpoint very early on if there’s people doing stuff, if you've misunderstood something, or if people are working on the same thing. So that's, I think that's something I've tried a couple of times I think really improves intra team communication. And through that helps increase the quality of the work we are doing.

48:10 RESEARCHER:

So, the transparency that Scrum or the process bring in creates this ongoing feedback with intra teams, right. Yeah. Fantastic. Now, it doesn't work all the time. I'm sure yourself says that. You haven't seen very good implementation of Scrum. Can you share with me a negative story where it didn't go well, and it didn't help achieving quality?

48:40 PARTICIPANT 31:

Yeah, I think one of the big danger points that I've seen is the Scrum master having other hats. Because the Scrum Master is supposed to be just a facilitator. And for instance, in one instance, the Scrum master was also the project manager for the product that we were developing. And that means that when the Scrum Master is supposed to simply facilitate the self-governing team. So, we had to figure out what are we going to do for the next sprint, of course, the project manager was very interested in trying to influence what we were taking on. So, the agency of picking out what we're going to do and how we're going to do it, it was very hard for this guy to simply act as a facilitator, and then not to interject with opinions as a project manager on what we're supposed to be doing. But of course, that's not a problem with Scrum. That's a problem with a misapplication of Scrum. I think my main problem with Scrum itself is that it doesn't, it's hard for me or it's hard for me to see how to reconcile having support tasks, which are unpredictable, and handle those at the same time that you're trying to commit to deliver on a sprint. And that's a situation that you're very often in as developers, it's rare that you don't have some sort of second or third level support tasks that are coming to you. And they can really throw a spanner into the, into the commitments if you take on a support task. And it turns out to be much worse than we thought. And suddenly, you can't meet your commitments for the sprint. And the problem with that is, of course, that it erodes trust, both within your team but also within your managerial level, if you can't deliver what you're actually committed to do. Of course, you may, you may have a good explanation it was because we had to restore the database due to some error out of our control. But still, if it happens, enough time it in the end, it doesn't matter that it's not your fault. You're still the one that keeps coming back and say, oh, we didn't, we only implemented eighty percent of what we committed to the sprint as well. And I think that

reality of having to balance firefighting with development is really very hard. And really requires, especially management, I think, to be aware of this, this problem and to accommodate and not to simply accept that it's it happens.

51:43 RESEARCHER:

So that's brings us back to what you mentioned earlier, which is less distraction, and less distraction enables more dedication to the development process.

51:57 PARTICIPANT 31:

Yeah.

51:57 RESEARCHER:

Yeah. Okay, fantastic. I think you already answered this question, but I'm going to ask it anyway and push you a little bit to tell me more. What would be your reaction or opinion in this statement: Agile or Scrum, which is an implementation of other Agile produces poor quality software?

52:31 PARTICIPANT 31:

Poor quality software?

52:32 RESEARCHER:

Yeah.

52:32 PARTICIPANT 31:

I think my reaction would be to ask for supporting statements. That is, that is not my experience. But my experience is that I think Agile often gets a bad reputation. Because people think they're doing Agile, and it doesn't work for them. I've seen a lot of people going, Agile is the new thing. We need to do Agile. And then they do something, which well, the names are the right names and stuff, but they're not doing the actual things that needs to be done. Rather than it's, it's fairly common in my experience to see managers participate in daily stand ups and demanding status reports. And why have you been spending your time on this and that it's kind of, it seems to be an immaturity on top of the management team to let go. They're not able to let go and trust the developers to do stuff. And I don't know, perhaps this last year and a half where people have been largely working from home and without immediate oversight has actually helped in this regard. I think maybe more people are ready to accept that many software developers are actually able to do their job without you answering them all the time. And I mean, if you have commitment to deliverables then it very quickly becomes apparent anyway. So, I don't really see this whole obsession with management interference, that's maybe just me. I simply don't see, because the problem is that when you begin to interfere like that, you create incentives other than producing quality code, you create incentives through being accountable for my time, instead of my deliverables and stuff like that, which distracts from actually writing the code that needs to be done.

54:55 RESEARCHER:

Why is that important to the developer to have this self-governance environment. And why is it important to the developer? For us academics, we've been giving this for years and to be honest, I don't care about my boss at all. And he knows that. I want to know, because for us...

55:24 PARTICIPANT 31:

The problem is not you don't care about your boss, the problem is whether your boss cares about you and what you doing.

55:28 RESEARCHER:

Yes.

55:30 PARTICIPANT 31:

And that's, I don't care specifically what my boss but when he begins to ask me why are doing this, why are you doing that, that's annoying. I'm a trained professional. I have a long education, I have massive experience, why don't you trust me to do my job? I've been in a place where I didn't work very long, because my boss insisted on telling me what should be done, and he then told me how to do it. I mean that's completely the wrong way around in my view. But he was very detailed oriented. I had to figure out what the requirements were and then he sort of laid out the technical solution for it and I just had to implement it. What is this? I was like why did you hire me, I'm expensive, I'm experienced. What you need or what you want apparently is someone fresh out of university because he wanted to tell people how to do it anyway and you don't need to pay for having a lot of experience and skills. But anyway, I think as I've said earlier, it's demotivating if I don't get to have this sort of influence on my own work.

56:46 RESEARCHER:
Yeah, I understand. It doesn't enable you to create, it doesn't enable you to grow.

56:56 PARTICIPANT 31:

I mean for software engineering, it's more in my view, it's more like a craft. It's a creative pursuit. A lot of it of course is about how you have to do this, or I need to parse an XML file, I need to do whatever but some of it when you need to figure out okay, how do I apply the right algorithm and so on. That's not just typing all the time, that's thinking and reflecting and so on. It's hard to just put out a formula and say, do this, do that and if I need to be creative to do my work, I need to have the freedom to think and act on that.

57:37 RESEARCHER:

Fantastic PARTICIPANT 31. I really enjoyed the conversation. It really interesting and nice, thank you. And thanks for taking part in this interview.

57:47 PARTICIPANT 31:

Sure.

57:47 RESEARCHER:

I will get in touch once I organized the workshop and I'll send you a work request in Upwork. Thank you very much. Do you have any questions for me before we conclude?

58:03 PARTICIPANT 31:

No but I will also say thank you for a very interesting conversation.

58:07 RESEARCHER:

Okay, fantastic. Hi and good day, okay.

58:10 PARTICIPANT 31:

[Danish language]

58:12 RESEARCHER:

[Danish language] Bye.

58:13 PARTICIPANT 31:

Bye.