02:16 RESEARCHER:

Hi PARTICIPANT 36. How are you? Can you hear me?

02:21 PARTICIPANT 36:

Hello? I can hear you. Can you hear me?

02:23 RESEARCHER:

Yeah, I can hear you much better. That's how you pronounce your name, PARTICIPANT 36?

02:31 PARTICIPANT 36:

Usually one of our clients like, [redacted], I had a difficult name. So, it's just whatever you like.

02:40 RESEARCHER:

Okay, fine. There is a problem with the audio. Sometimes it's good sometimes it's not good. Can you check it?

02:53 PARTICIPANT 36:

What we could do is, I could try to disable the video. And maybe it improves?

02:58 RESEARCHER:

Yeah, yeah, we can do that. We don't need actually the video.

03:06 PARTICIPANT 36:

Oh, let's check it.

03:08 RESEARCHER:

Alright, it's a little bit better. But if we encounter the problem again, we will, we will try to resolve it. Okay.

03:18 PARTICIPANT 36:

You could try also to disable your video to improve the bandwidth.

03:27 RESEARCHER:

Done.

03:31 PARTICIPANT 36:

Usually, though, the audio is okay without the video.

03:35 RESEARCHER:

Yeah. So, PARTICIPANT 36, I'd like to thank you for the opportunity to talk to you and accepting to do the interview. I really appreciate and I'm looking forward to the conversation.

03:51 PARTICIPANT 36:

I say the same.

03:52 RESEARCHER:

Okay, thank you. I will start by introducing myself, telling you what I do and why I'm doing these interviews. And we'll take it from there. My name is [redacted]. I work for the I.T. University of Copenhagen. One of the universities here in Denmark. I do my research in software quality. And in particular, I'm interested to understand how software teams managed to achieve quality. Currently, I'm running a project where I'm looking at how Scrum teams manage to achieve quality and whether Scrums help them to achieve quality. So, we are doing interviews because we like to understand people experience working with Scrum and whether it made a difference or not. And we capture these, experiencing these into reviews and we analyze it, and we draw some conclusions. And we publish those conclusions, and we disseminate them as well with other activities. So, in a high level, this is who I am. I'll give you the opportunity to ask me question about myself if you have any questions.

05:25 PARTICIPANT 36:

It's not that I'm not interested. But let's just focus on the why we are here.

05:32 RESEARCHER:

Yeah. Okay. I'll start shortly with the interview. And I have prepared some questions. And we will go through those questions, but it is flexible at any time. If you want to add something, please feel free to add it. So how about if we start with a presentation of yourself telling us what you do and a little bit about your experience briefly?

05:59 PARTICIPANT 36:

Briefly?

06:01 RESEARCHER:

Yeah.

06:01 PARTICIPANT 36:

Briefly, I started coding when I was fourteen. And back then there was the DOS era. With sixteen bits operating system hard drive was eight megabytes, or gigabytes. I've been doing coding professionally as an enthusiast. At school, participated in multiple tournaments. Took a break for a while, got back into the programming because it was paying good. Mostly worked at one company. But recently I kept jumping from project to project, did about ten

years of freelancing. That's briefly. That's very briefly, I worked on probably hundreds of projects.

07:00 RESEARCHER:

All right. Great. That's, that's an extensive experience. We'll start with the first question. We are talking about Agile. And in particular, you told me you are using Scrum. So, my first question, what do you think of Agile? What is your opinion of it? What is your experience using Scrum or Agile?

07:27 PARTICIPANT 36:

So, I've been working in Waterfall methodology, for, like fifteen years, I would say I'm an old school guy. So, I'm biased using Waterfall approach. I'm not against Agile? My personal opinion is that software became an industry including teaching, mentoring, learning new languages, creating new languages, methodologies, politics. So, all this teaching is a whole industry. I think, from my side agile is a great opportunity to do software. I could give an example where we had a team of four people, and it got bloated to just because we come over to Agile, we were doing the same amount of work. So, the overhead is, its mind blowing large. Is it a problem? Well, not really. I had seen in healthcare, a project where this was completely acceptable. The project was thirty years old, legacy code, very hard to understand, very big project. The main focus of the project was continuity, predictability, and reliability. And they had statistics that every year, fifty percent of the company, left the company. And they solve this with Agile. People kept coming. They trained for one month, stayed two months and left the company. And this was all predictable. They were productive for two months. And this was solved using Agile where people share their knowledge before leaving. So, there are projects where Agile is good. There are projects where Agile is either inefficient. From my point of view as a coworker, I spent half a day reading emails, attending meetings, everything but working. Is it good? Honestly, I hate it. It's annoying me. I'm not productive. I'm just discussing all the time. But hey, if this is required to hearing to politics, everyone's into politics, discussing. This is my opinion about Agile. As I said, there are good cases and bad cases, every company should know what they want, what they expect from the project from the workers, know what the methodology brings. If they're happy with it, it's not like something everyone should use. It's not a must. It has advantages and disadvantages.

10:44 RESEARCHER:

Of course, like any methodology, and sometimes it's its implementation, that is the problem.

10:52 PARTICIPANT 36:

Yeah, the implementation. Yeah. Well, I don't know what exactly should be kept from Agile. But the whole idea, I mean, let's take my father, for example, who learned for fifty years, years, one single database for GL. This was I don't know, thousands of pages of books. And he was an expert in that field. Agile on the other hand, is the exactly opposite. You know, everything. You can jump from one task to another from one project to another, you keep balancing the lack of resource, which in theory, it's fantastic. It's brings up productivity in practice. It takes me one hour just to focus on the project to see the need to understand what's happening. Of course, it depends on project. But for large projects, this is required for small projects, doesn't matter if you're Agile or not. So where exactly is Agile, good? Well, for companies where people keep coming and leaving, it's great. You are able to replace

resource very fast. But apart from that, personally, I think when there is stability and predictability, agile may not needed.

12:28 RESEARCHER:

I agree. It doesn't help much. Because every day you doing the same thing.

12:31 PARTICIPANT 36:

Yes, it doesn’t. But as a worker, you start to contemplate, okay, so what exactly does the company want me to socialize or to work? Because practically, when you have four hours of meetings a day, and you have thirty-two mails, it does take upon a focus and concentration to switch from a task and read the mail and switch back to the task. So, thirty-two emails, is like, in those hours, eight mails an hour. You are not practically doing anything at that day. And all this Agile stuff is it sounds good. Let's learn about that I have one week training, practical Agile, I'm certified. So, we always need to assess whether we need to be agile or not.

13:38 RESEARCHER:

And you got a certificate as well. So, when does agile or Scrum help?

13:54 PARTICIPANT 36:

Yeah, sure. My positive experience, is collaboration. It makes people work together.

14:26 RESEARCHER:

Okay, from a quality perspective, obviously, your opinion, does it help?

14:36 PARTICIPANT 36:

Yes, I think the code quantity increases. We discussed, we discuss certain tasks so many times that you review it, and many, many times you realize, oh, maybe it's not the perfect code. Maybe I could do it better and you have the time to rewrite it. So, I think the productivity and the quality itself increases.

15:15 RESEARCHER:

So, what makes it increase? So Agile brings some value, then what makes increases, in your opinion, because you have time to do reviews. And that's what you're saying?

15:26 PARTICIPANT 36:

So firstly, I think the most important part is this is Scrum already not Agile?

15:37 RESEARCHER:

Yes. Scrum.

15:38 PARTICIPANT 36:

So, the standups. We programmers are quite anti socials. And the fact that I need to talk about what I'm doing, I'm doing my best to present it competitively to other programmers. And that means I start to focus on for example, code commenting, formatting, testing, all the aspects, I might neglect when I would do it solo. In my corner, where I know, it, only the result that matters. And the fact that I present it, I review it, I reiterate the whole process, the steps and everything. And I think it really improves code quality. Also, there's the code review, the cross checking, the pull requests, the additional steps, the retrospectives, how you imagine you do, and what actually you have done. So, the next time you can do better. So, Scrum itself, I think contains very good steps that improve quality. It brings collaboration which make developer work better and produce better quality.

17:03 RESEARCHER:

Can you give me an example how the retrospective ceremony or meeting has helped you to produce better code quality?

17:15 PARTICIPANT 36:

Sometimes, so your large projects, you do testing, and you have certain expectations. And a simple example is multi-threaded, converting single threaded to multi-threaded, and then you pass, the next step would be the testing, the tester cycle goes through it and do a more professional testing, and you obtain some results. And I think retrospective, you contemplate, oh, maybe we should have used a spin lock instead of semaphore, or a thread synchronization. And we weren't expecting it to work like that. But it would have taken too much time. So, you contemplate on what you expected to do. And don't think the results. Maybe next time, you would do better. You don't just deliver results; you also contemplate on providing better results next time.

18:31 RESEARCHER:

Does it mean better results for a developer better code? High quality code, better quality software?

18:39 PARTICIPANT 36:

Of course.

18:44 RESEARCHER:

Okay, I'll proceed with some question just to keep the conversation because it's already interesting. How do you define quality in the context of software development and specifically in the context of Scrum?

19:07 PARTICIPANT 36:

I think quality might mean different things for different projects.

19:16 RESEARCHER:

Correct. Yeah.

19:18 PARTICIPANT 36:

I have worked on projects where quality meant that you squeeze every bit of CPU power for the battery life of a device. I've seen projects were, like, healthcare projects where quality, absolutely zero chance of an error. And you might think, yeah, it's, let's do a test case and provide a hundred percent reliability, but it might not be that, for example, for Google search engine. They have so many computers that the CPU has a very, very small error rate, like, I don't know zero point zero zero zero one percent. But when you have millions of computers processing data, you already have to think about cross checking results. So, you might need the same calculations more than once or more than one PC. And cross check the results. So, this is good quality for them to provide reliable results, not just that your code is good, including the underlying hardware will provide good results. Until this day, NASA, in their mathematical CPUs, they need special hardware to ensure that very high precision, math will provide accurate results, not like oops, there was a zero zero one percent chance for an error and then crashed, you cannot afford this quality results from the project. Quality on long term projects might also mean that you are able to handover your part of the project to someone else in a clean manner. So, the code is commented, the code is beautiful, I would say, this might mean for them quality code. Because if they keep bringing on new programmers, and there's no quality code, after a while the project will crash from start from zero. Code quality might mean testing friendly code. So many cases you expect your code to behave in a certain manner. And why behave many situations. But for maybe even Google, if you want to provide a quality project, you are able, this is why I like functional programming. You can test every part, you have a very good code coverage for retesting, because you know you can test all the inputs for a simple function, and you have certain output ranges. Now if you are coding object oriented and your object can do absolutely everything. That is almost impossible to do code coverage and reliable testing. So might not be quality code for them. Simply because it's unpredictable. What else could mean good quality. Quality code from a UI perspective. We can talk a lot about it. Is it usability? Does the menu, the order of the menus pop up, which are used more frequently?

23:38 PARTICIPANT 36:

Or is it arranged that buttons are hidden; important information is in the center of the screen. When trackers, click counters, so based on performance counter, the UI is what the user expects and not what the programmer thinks is good. ID quality, UI quality project. Quality UI might also be something that is trendy, the colors, rounded edges, the size of the fonts. Do they, are they placed in the current trending style? Or is it for a future trending style? Or is it some retro style? What kind of clients focusing on? These might be quality project just that it actually targets the correct personas. A fictional personas that actually the project management requires the developers to target wanting what the management expects. And what the programmers provide. Quality would also mean that actually meets expectations.

25:10 RESEARCHER:

Okay, fantastic. Let's move to the next question. That was a good perspective on the definition of quality. Thanks for that. What do you do in a Scrum environment to assure software quality? You talked about code review, what other processes or technique or tools you have in place to assure quality?

25:38 PARTICIPANT 36:

I wouldn't say quality assurance is really Scrum related. We have been doing quality assurance for years before Scrum. For example, in my industry, Health care, you have testing, pros testing, [inaudible] testing, automated testing. So really about code coverage and testing. That is the first step or line of defense expected and provided results. And really when it comes to testing, it's the fact that when you test and retest the same thing over and over again, you start to neglect steps in the testing. So, unless you have automated testing, errors might pop up in the testing steps. And these skip steps, might lead to bad code quality. Really the code quality depends on the quality on the testing. And a lot of projects try to save resources on reducing testing. They usually say the developer will handle everything from planning, implementation, testing, building, launching and releasing everything. One man team army, which is very bad. It comes back to Agile when human resources are allocated to require the lack of resources, and then I read this is very bad from my perspective, because you cannot in this industry do everything. Very simple areas require your multiple years of experience, and you cannot keep in your head, database testing, implementation, environments. Impossible. Whoever expects you this has no experience in project planning. The project is probably going to fail. A large project or if it's a small project, it's probably can be rewritten multiple times. Code quality, also Scrum. So having multiple types of teams in Scrum helps you with the testing. We do have multiple teams, like a team that focuses on people actually look at you, your mental state, try to help you improve your mental state. The actual implementation team, the tester team, the team managers, so it's a much clear separation between roles. I think Scrum helps in this regard the code quality because it helps you focus on your specific code in large projects. In small projects, another story.

29:20 RESEARCHER:

Yeah, that's good. Do you have a positive story about Scrum and how it did help produce in better quality?

29:32 PARTICIPANT 36:

As I told you, I think Scrum in every scenario brought better quality code, in every project I attended. But not every company can afford it. So, this is the negative side of Scrum. To have a good implementation of the methodology, you usually require a larger budget for the project. And this is the drawback. You should expect half of the methodology, half of the resources. In many small projects where the budget is very low, you actually do everything from project planning, task breakdown. You skip the standups, you skip the retrospective, you do the implementation, testing and you just drag and drop the task to the done. So, you've created the task, you close the task and that's it. So, you just keep the Kanban board from the old Scrum, it does help you even in this case, the simple fact that you break down a large task into small doable tasks already helping a lot.

31:02 RESEARCHER:

How' does it, sorry? How does it help chunking or putting a task into small tasks? How does it help to achieve better quality?

31:15 PARTICIPANT 36:

So, when usually, when you finish university, yet, you practice Waterfall had not practiced Scrum. And usually, you have an assignment at university doing this project, and you do the whole project. And you will try to break down the issues into smaller issues. But you will

probably still end up with, for example, object-oriented coding classes that do a lot of stuff. So, you no longer have the projects or simple, how you say, the projects that are separate functionality, modularize, keep it simple. Don't complicate. So, this is already the part of the Scrum when you try to break down tasks. And each task could be one single class. And that class focuses on a single issue. And this modularization will help you in case it is required to replace the feature if the code is well modularized. Let's say you write a module that reads CD ROM drives. And time passes and you realize, oh yeah, the whole project is good. But now we read USB drives. And then you go and switch out that single class and if it is well written, quality code separated. You can do it very fast, very efficiently without a lot of retesting, because it was well testable, small test, small results. Sprint Scrum. So, this is how small tasks help you, start to finish with well separated modules, the code.

33:36 RESEARCHER:

Okay, great. Thank you. I move to the next question. How does for example, Scrum helped you find bugs? If you have examples would be great.

33:52 PARTICIPANT 36:

Most my experiences in C, C++, and their memory, corruption related problems are almost always present. And the truth is, usually it's a hit and miss. You try something. Well, it wasn't that. Let's try something else. And sometimes you're not even the debugger helps. Not even testing helps. In these cases, Scrum, when you do the stand up and you talk about your life problems, some guy might pop up and say, hey, let's try to disable half of the program and see if that issue still happens. And the guy might be right. Let's try that I wasn't thinking of. I haven't thought about that solution. So, the Scrum helps you with views of the issue that you have not thought about yet. And it might speed up the process of debugging. Guys with more experience might share their experiences regarding your situation. So, this is practically helping you find bugs, fix bugs, avoid bugs. The standups help a lot.

35:27 RESEARCHER:

Great. The next question we touched a little bit onto it, but ideally an example would be great. How does for example, your Scrum setup help producing high quality code?

35:46 PARTICIPANT 36:

Yeah, I have to admit that I lacked for quite a while the practical usage of [inaudible]. I haven't finished computer development at university. So, I was lacking the practical and Scrum practically helped me because I had to present my code. For example, use camel case, or code commenting or a function descriptions that help Doxygen generate better documentation. Breaking up large functions into smaller functions, modularizing code into multiple files, or DLLs, so a lot of self-control, self-supervising. Scrum helped me focus on myself to be able to present myself competitively to the team.

37:06 RESEARCHER:

So, presenting yourself competitively to the team motivates you to write better code. That's what you say.

37:17 PARTICIPANT 36:

Yes.

37:18 RESEARCHER:

Okay, fantastic. That's bring us to the next question. I wish you answered to some degree, but I will. I'll ask it anyway. How does Scrum environment or set up motivate you to achieve code quality? Does it bring anything to the motivate, to motivate you to write better code?

37:45 PARTICIPANT 36:

Yes, it does [Scrum motivate me to write better code]. Personal pride for the retrospective and you present your results. And we are actually quite proud of your code, checks all the boxes. It's beautiful. It runs, it provides results. Because it was a simple task, you have a high chance of success to check all these checkboxes. You estimated enough time to perform all the required steps for the formatting, the commenting, the testing, the rechecking, and you actually are proud of your code. And that brings value your personal value, increases your morale, which helps you actually evolve and do better and better code all the time.

38:52 RESEARCHER:

That's a great statement. Thank you. You worked in in Waterfall. You've been around for a while, just like me. I worked in Waterfall, and I started writing COBOL code years ago.

39:12 PARTICIPANT 36:

I'm not going to envy you.

39:13 RESEARCHER:

Yeah, I don't think it's existed anymore. But still, there are few mainframes with COBOL. So, you looked at Waterfall, you worked in Waterfall and now you work in Scrum. Do you think Scrum brings value to produce better software quality from your experience?

39:34 PARTICIPANT 36:

So, it's hard to compare because time past, there are larger and larger and complex projects. And let's just compare the DOS era where in a couple of floppy disks you fit all the operating system. These days you require multiple DVDs, or cloud or whatever, SB drives. So, it's very hard to compare small projects with very simplistic hardware handling, very simplistic user presentation. Current projects which it's mind-blowing complexity. So, at Zeemas where I work, we have twenty thousand developers working on the same project, I would consider is a decent project. So, you cannot expect, it was really turning pro Waterfall and it was a successful project, implemented successfully. But in modern era, produce better project. Waterfall is actually I mean, Scrum sorry, Scrum is helping. Waterfall is good. Scrum is better. Because currently, we are lead developers. So, every random guy is getting hired. I kid you not we hire truck drivers and medical staff to be programmers and having very, very small tasks, supervised tasks, actually, how do you deliver acceptable results. This was impossible with Waterfall, when you were hidden in your corner doing your stuff, delivering results. And when you had to integrate it into a larger project, the project has a large chance

of [inaudible]. And these days, good methodologies are required to use any resource or project.

42:11 RESEARCHER:

You mentioned something good which Scrum brings them to the team. You are not silo; you are not alone. So, you're part of a team. Does it motivate you to write better code?

42:26 PARTICIPANT 36:

Yes. As I said, competitive, I don't know if everybody has this. I do literally know people sleep on the job and doing absolutely nothing, just waiting for time to pass. But yeah, this is present in Waterfall also. So, let's focus on the good cases where people actually are motivated to be competitive, try to socially bring themselves up to get noticed, actually brings them social value. They provide better code quality. So, it's, it's a win win situation.

43:13 RESEARCHER:

Fantastic. Thank you very much, PARTICIPANT 36. I really liked the conversation. That's a very good perspective and very rich experience. I've run out of questions. Do you have any questions for me before we conclude?

43:33 PARTICIPANT 36:

What do you think is planned for the future generation of programmers?

43:45 RESEARCHER:

I'm like you I think, Agile has been commercialize, I think the intention behind that was noble. But it has been commercialized, and a lot of people are making money out of it. I think we were overwhelmed and loaded with methodologies. I don't think we need more methodologies. I think we need to focus on people and what make people work more efficiently, what makes them work happier, what make them produce quality code, and what makes them motivated to keep working and delivering quality software. So that's what we should be focusing on. I mean, personally, I'm tired of methodologies. There are so many of them. And even Agile has been broken to, I don't know, a hundred methodologies. You have Scrum, which is the most popular, you have Kanban, and you have, and they even start combining. Now you have Scrum-Kanban. I even met someone who talked to me about Scrum fall, which combining Scrum with Waterfall and crystal, yeah, I think there are so many methodology. I think it's enough. We should be focusing on people and what makes them more efficient, what makes them happier to produce quality code. I'm against more methodologies to be honest with you. And if we produce methodologies, someone will try to make money out of it. Either training or consulting or whatever. Yeah, so I wouldn't promote or thinks that we need more quality, not more methodologies.

45:46 PARTICIPANT 36:

A hundred percent agree with you.

45:49 RESEARCHER:

I think we need to focus on educating more developers, like you said, there is big shortages, especially in Europe, there is a big shortage. I know in Denmark, there's a big shortage. And we think also in educating them, showing them how to work in a team, not only working at teams, how to learn from their colleagues, how to accept criticism from the colleague, and how to evolve. You use a good word, it's keep evolving. Once you keep evolving you, you become better at what you do. I think you did put it in a nice way. So, I think that what we should be focusing on, not new methodologies. I think I'm personally tired of new methodologies and I left the industry. I worked in the industry like you almost twenty years. And I left the industry mainly because people misuse this methodologies or these practices. I've seen so dysfunctional implementation of Agile and like you said yourself, is the quality of implementation makes a difference. I think we need to teach people whatever works for them, it should be high quality implementations, whatever they choose to do, whatever they choose work for them, and be less dysfunctional. I've seen organization bring in Agile, just for the sake of, of calling themselves Agile, or because it's the trend. I worked for a large bank, one of the largest in the world. And Agile doesn't work for them, they are very controlling, they very risk mitigation oriented, they, they don't let go the control. It was a disaster. To be honest with you, it was a real disaster because the organization historically, it's a legacy for control and risk mitigation. And Agile as a mindset, or as a philosophy doesn't work for them. But just they adopted because they believe it deliver better, it delivers quicker, which is true, but it's for other organizations, which they have different culture. It does. But for you. It wouldn't you, you need to know yourself and check what works for you. I'm a big proponent, that software teams need to adjust to what's worked for them as a team.

48:41 PARTICIPANT 36:

I agree but this whole industry. Let's start with they chop down, the make trends for you to keep learning.

50:54 RESEARCHER:

Yeah, fantastic. That's a good way of putting it. I'd like to thank you again for your time and interesting conversation. And thanks again. And I wish you a good day. Thanks for the interview. I appreciate.

55:11 PARTICIPANT 36:

Thank you, to you.

55:10 RESEARCHER:

Okay, bye bye.

55:13 PARTICIPANT 36:

Goodluck with your project. Bye.