[01:48] PARTICIPANT 17:
Hi Researcher.

[01:50] RESEARCHER:
Hi, how are you? Yes, I can hear you. How are you today?

[01:56] PARTICIPANT 17:
I'm good.

[01:57] RESEARCHER:
Fantastic.

[01:58] PARTICIPANT 17:
Good weather today.

[02:00] RESEARCHER:
We don't have good weather here in [redacted]. It's been raining unfortunately.

[02:10] PARTICIPANT 17:
Typical.

[02:11] RESEARCHER:
Yes, typical of [redacted].

[02:15] PARTICIPANT 17:
It usually also rains every day. But delays it's acceptable.

[02:19] RESEARCHER:
Okay, that's fantastic. Lucky you. Okay. Do you have any questions for me before we start the interview?

[02:26] PARTICIPANT 17:
No, I don't think so.

[02:27] RESEARCHER:

Let's start with some introduction. Can you introduce yourself and talk to us a little bit about your experience?

[02:37] PARTICIPANT 17:

So, my name is PARTICIPANT 17. [redacted] I'm currently living in the Netherlands, in Amsterdam. And I have a [inaudible] in computers, that's what I studied. So, I did a Bachelors and the Masters in Computer Science. And I started working as a Java developer for like two years. And then I switched to more to be into data, but always connected to the backend and engineering side of things. So that's what I did for some years. And then more recently, when I moved to the Netherlands, which was four years ago, I started to have a lot of contact with Agile and Scrum in particular. When I worked for [redacted], which is the biggest [redacted] in the Netherlands and that's when I started to have more interest in it and how useful it could be. And more recently, for one year, I moved out that company and now in another one, a fashion company. Since January last year, I've been also the Scrum master for the team. Still doing a bit of development. I would say like seventy percent of Scrum master. And thirty percent of development. But with bigger focus on being the Scrum master to drive all the agile the team. And of course, also working together with other Scrum masters and POs in the company to drive the Agile.

[03:14] RESEARCHER:

We will be talking about quality. Hence, it is important to define it first. How do you define software quality in agile software development?

[03:33] PARTICIPANT 17:

I'm glad this is not an exam question! But to narrow it down, I see quality taking place in three levels: product, software code and the development process. I'll explain. The product quality is a functioning product free of defects and conforms with the business needs. Software code quality particularly in agile means clean code and a design that caters for future changes. In agile, we believe in responding to change over following a plan. It is important that we deliver a software that doesn't break each time the business wants a change. The process is important. Chaos doesn't deliver software, but a committed team and a strong process does. Agile believes in continuous learning and improvement. The software process becomes more robust when we learn from our mistakes and improve.

[04:34] RESEARCHER:

Okay, fantastic. Thank you. The next question is, what do you think of agile?

[04:43] PARTICIPANT 17:

Oh, I like it. Obviously, being a scrum master. To give a bit of an overall, I believe it was meant to exist sooner or later. We've seen the world accelerating a lot in the last twenty years, twenty, thirty years with the advent of Internet. To be a successful business, you just cannot work on one big feature over the year. For example, what your clients wanted two years ago, or two years ago when you started doing a project, what they wanted two years ago might not be what they want anymore. So, I think Agile needed to exist because, like, simple solutions or some of the solutions that they bring, it's creating smaller increments of value in very short iterations. I think it's very important. So, I'm totally aligned with Agile thinking.

[06:05] RESEARCHER:

You mentioned something very important, which is acceleration. What does it mean in agile and what's the correlation with Agile?

[06:17] PARTICIPANT 17:

Yeah, what I mean by acceleration is that things can be developed...any feature, any product can be developed much faster now than we had like eight years ago when something will go first, let's check everything that's needed for this product from the beginning to the end. So, if we go a bit into the waterfall model, let's get all the requirements first, everything from one point to the other has to be perfect with what's the client or the customer wants. But I think nowadays the competition is so high and so easier to build. It's easier and faster to build new products and to build new software. Agile is really is a response to that.

[07:16] RESEARCHER:

This acceleration, and this fast pace of delivering. Does it create pressure on your team as a scrum master?

[07:30] PARTICIPANT 17:

I think so a bit. So, when I assume that when you say pressure, not the good pressure. I believe the two of them exist? But I think what you meant here is that it's a bad one. It can have, yeah. But from another side, so why think it exists, is that since you can iterate so fast, there's actually pressure to also deliver fast. And sometimes, of course, there are still some things that you cannot deliver that fast. But the solution to that is to just divide or divided the tasks in smaller and smaller increments of value, or smaller tasks, put it this way. But I don't know if I answered your question about the pressure. It does, yeah.

[08:30] RESEARCHER:

It depends. Every team handled it differently. Next question. Can you describe your Scrum environment setup? It could be your current job or any previous job you want to talk about. So, pick up an example and talk about it for us. Please. Thank you.

[08:56] PARTICIPANT 17:

So, my current one, we have a team of six people, including me. It's a cross-functional team so that means that almost everyone can develop almost any story or almost any feature. We kind of push for that as well so we we're not dependent on this specific person to do this or that feature. We want to have...I can work on a product, in more cases, is data warehouses providing data to do multiple stakeholders. We give a lot of importance to every person in the team to be able to develop any feature. And I think that's really important. Then we have me as a scrum master of course, and part of the six. Then we have one PO, so one product owner. We do two week sprints so that's the iteration size that we do. We have two week sprints and of course, we do all the ceremonies. We do stand ups every day. We do refinements when there are new stories that needs to go from the product backlog into the sprints.

[10:24] PARTICIPANT 17:

So, we refinements, we also do retrospectives. So, by the end of each sprint, we look at what went good, what went to bad or what can we improve for the next sprint or sprints. We also have one delivery manager which is not quite part of the scrum, or even of agile, but that's how the company is right now and I think it actually for this case, it takes some responsibilities out of the PO and scrum master. So, in some cases that might be useful. We also have support. We also have the Kanban so for all the support tickets, all the maintenance of the product. There will always be bugs here and there. So, we have the Kanban board where it's first come, first serve. But it also depends on the product, of course. So, it's not scrum, but it's in our team because we are responsible for the product.

[11:39] RESEARCHER:

You didn't talk about QA being part of your team. Do you have QA or who does the testing?

[11:50] PARTICIPANT 17:

In Scrum no defined role for QA. We don't have a dedicated one but what we do is a story, or if I go to the beginning, can I talk how a feature goes from the beginning?

[12:16] RESEARCHER:

Yes. I was going to ask you a question like that, but you can do it now, from the beginning.

[12:21] PARTICIPANT 17:

Yeah, I think that would be useful. So, from the beginning, the PO, the businessperson to go to. The PO is in close contact with the stakeholders. We have different teams that are stakeholders. So, I can say just quickly, finance, buyers, because it's an e-commerce business so we also have buyers that will get the data and try to anticipate what they will need. So, all of these people are stakeholders and they all want different things from our team to deliver. So, the PO, product owner, sits with them, gathers the requirements, gathers their feedback, and tries to come up with new things for the team to develop. What will be the direction of the team, what should we have our focus on, what should we be working on? So then after that, he writes some tickets based on that, based on some loose requirements, some things that he wants to be done. He writes some tickets on it, or some stories, some epics.

[13:30] PARTICIPANT 17:

After that, the development team refines it, so we have a refinement, right? Here is one of the first points where the development team can, and we do challenge with the PO on the business value of these new features. I think this already brings a bit of QA. It's not like a typical QA, but from a business perspective, we are already challenging the need for it. And then, of course, the PO has to explain, and the team will also have to agree that this will be useful. When we are doing the refinements, sometimes not common, but sometimes we already kind of come up with a development plan on how the feature should be there. So that also brings another point of contact when a lot of people, in our case, five or six people, can already give their input. So that already will help the final quality of any feature. Then after the stories are refined, what we do is the PO being, again, the businessperson, he prioritizes or clarifies the tickets that should be in each sprint, according to the business. And then when the story is going into the sprint, and the new sprint starts and we have the stories to develop, then anyone can pick them up, but we pick them up.

[15:10] PARTICIPANT 17:

And then once it's done, the developer tests it because the developer is doing something so of course, he has to test what he did. And then after that, we have a code review made by two people in the team, at least two people in the team. Sometimes if it's a bit more complex feature, we put more eyes on it. But usually we have two people reviewing each small feature that was developed. And then after that or after the approval was done, if everything looks okay according to what was needed, the PO once again, before things go into production, the PO once again looks at what was developed. Checks if it conforms to what he wanted in the first place and then it's deployed. This is to say that through all this process, you can see at least there was like four points, I would say. So, the first point it's more from the business side, then the whole team brainstorms on how to do it. Then we have one other developer doing it and testing it. Then we have two people looking at the code, looking at what was done, checking if it's working as well. And then we have the PO again looking at the feature before it's released.

[16:44] PARTICIPANT 17:

And then after all this, of course, we have some tests in place, some automatic tests that we also develop. So, since we work with data, it's a bit complicated. I don't know how technical I should or could go here, but we just have some tests running on production, on the data to see if it's expected. So, for example, an easy example is, if every day we sell one thousand items, and if one day, one of our tests picks up one million items, something's wrong with it. So that's on part where we have some tests. These tests are all in production, these final ones I'm talking about now. But here also, I think the QA for the quality or quality side of things. Maybe I'm just rambling a bit now.

[17:46] RESEARCHER:

No, no. Keep going.

[17:50] PARTICIPANT 17:

I think on what comes to mind, but I think it also depends on the thoroughness of tests, also depends on the product that you are doing on the software or on the business that you are in. I would say that is if you want to develop, for example health. If it's something related to health, to hospitals that cannot fail at all. Then I would say that you will have to have more tests and more people testing everything than what we have in my team, for example. Because the things that we do, if they fail a bit, I would say that maybe when things go to production, I would say, ninety-five percent of them go well. And we'll have like five percent of the features that have some bugs. For our case, in my opinion, is totally acceptable because it brings us advantages like the speeds and developing things faster and having faster feedback from stakeholders. I think it overwrites the five percent of defects that we find over time. In Scrum, we have this frequent inspection by the business users; the feedback we get early on the process reduces bugs. But of course, this is in our business and in our team. I believe that if it was something crucial that really cannot fail. I believe that we will need to have more and more deep testing than what can we do now.

[19:28] RESEARCHER:

There is something particular about your team is the QA and the developer is the same role.

[19:36] PARTICIPANT 17:

Yeah, yeah. We have no dedicated role.

[19:40] RESEARCHER:

Does this bias the testing a little bit?

[19:46] PARTICIPANT 17:

For sure. Yeah. Especially from the developer. Been a developer for seven years now. I'll say of course. Usually you just test the happy parts. So, the things that go right. I don't know if you've heard this term before.

[20:08] RESEARCHER:

Yes, of course. I use that term myself. Sometimes I use the shiny day or sunny day.

[20:23] PARTICIPANT 17:

Yeah. So being a developer, you usually test does the shiny day, part. But when we put two other people that are also developers and they also understand the product that we have, the feature that we have, the stakeholders that we have to, how can I say, like, we have to be happy about our job in the end. But just to say, when we have two other people looking at it, I think the bias is smaller.

[20:59] RESEARCHER:

And the code review helps as well. Usually code review helps to produce quality code.

[21:07] PARTICIPANT 17:

So yeah, we do have a lot...just off the top of my head, I would say that at least maybe like half of the tickets on code review, they go back. They find something, maybe not half, but like twenty or thirty percent. And it is good. For me, as a scrum master, that's also good. Because it also proves that code reviews are not just clicking a button. And I also think that when you have a dedicated QA, you will be more specialized in bringing all the places where it can go wrong. It will be better for sure. Like I said, it could be better but once again, I think there needs to be a balance between the speed of delivery and the quality of the code. Not that the code quality will be much worse, but I think it will be acceptable considering the speed you are bringing.

[22:22] RESEARCHER:

That brings me to the next question. Do you think this Scrum setup is a good implementation of Scrumcand why?

[22:34] PARTICIPANT 17:

You mean if my Scrum implementation in my team is a good one?

[22:41] RESEARCHER:

Yeah.

[22:43] PARTICIPANT 17:

Yeah, I think so. I would say. So, since I know a bit about Scrum in particular, I would say that the theory is not in the books to have. Because in theory, Scrum teams are self-organizing. They will not need managers per se. But in our case, like I said before, I think it brings a bit more freedom to the team. Like for example, sometimes there might be conflict inside the team on which direction to take. There are different opinions. Sometimes with a manager, that's the person that really will have the final say and everyone agrees that the person will have the final say. So, then I think it is useful to have. So that's on that one. The PO role for me is very important as well. Someone that's just focused on having quick feedback with stakeholders and bringing new features. Someone that is, and I quote, unquote, their only job is bringing business value, which is very important. This is not so easy with team management and taking decisions on development side, because that's for the team. Also having the scrum monster, I also believe it's very important. It's more important for teams that are starting than for teams that are a bit more settled, like mine. So, we've been working together with some small changes, two and a half year. It's kind of been the same team. So, we are very mature already.

[24:46] RESEARCHER:

How long you've been together?

[24:48] PARTICIPANT 17:

Two, or two and a half years without many changes.

[24:56] RESEARCHER:

I do have some follow up questions. The first one. Do you think the size of the team, which is quite small, actually, this is the smallest team I talked to? Yes, it's quite small. Do you think that the size of the team helps you to get Scrum right?

[25:23] PARTICIPANT 17:

First of all, by the scrum standards, they from three to nine people is a good scrum team. So, I think we are in a good spot. Personally, of course, when you're working in a team, you always want more people to help you. To help build the product. For me, if I would have three or four more people, we could do much more. But to answer your question. If it helps. But you mean like compared to having, for example, fifteen people or to having two people?

[25:59] RESEARCHER:

No, compared to having fifteen or twenty people.

[26:05] PARTICIPANT 17:

No, never thought of it. But twenty, it's too much. And fifteen as well. So, I can start by saying that fifteen is too much because we do daily stand ups and in a daily stand up it has to be really fast. I had in the past as well, daily stand ups with twelve, thirteen people, at the time when I was in [redacted]. And I can tell you in my opinion and in that experience, it was too much. Because after two, three, four people talk and say kind of what they are working on and what their challenges are, the other developers will not care that much. And maybe it's just me because I'm also a bit in that situation. If I'm in a team of twelve people, after four or five, I will not listen that much, you know.

[27:02] RESEARCHER:

Yeah, you lose your attention.

[27:04] PARTICIPANT 17:

You lose a bit of attention, right. So, for me, we are now six, it's good. You can still keep because there's a reason why you do the standups. It's to bring everyone together and to raise if there's something that you can raise easily and fast in an everyday meeting. So that's good and that's one reason. And the other is that everyone in the team kind of has an idea on how everyone else is going and what everyone else is working on. You have fifteen people, I think it'll be too much. From what I worked on, it's also too much. In the end actually, I can tell you what was happening in my experience of twelve, thirteen people. In the end, we were just two team. Half was working more on one side and the other half was working more on the other side and we did not have that much of a connection between us.

[28:00] PARTICIPANT 17:

So, in the end, we just had two separate stand ups with the same scrum master because there was only one scrum master at the time. Totally fine, you can have multiple teams, even totally different teams. But I think it was too much. And so, I think now, we this one of six, we also had less people before and now we are six. I would say personally more than like eight, it will be too much. It will be too much confusion. You kind of lose a bit of the group and the team mentality as well.

[28:39] RESEARCHER:

Fantastic. I do have more follow up questions. You talked about the team's maturity. Do you think that the maturity of the team helps you to have a good Scrum implementation?

[28:55] PARTICIPANT 17:

Definitely. So, when I talk a bit about maturity...so people that are new Agile methodologies, and new to scrum, they don't understand or they don't know much on what to say on standups, on how to approach retrospectives and retrospectives are very important. And, of course, maybe they will be thinking in another framework. So, they will be thinking of...I can give you an example. if you're not that much into Agile, you would be looking into the tickets and say, but what about this, what about that, what if this happens? What about that one? So instead of focusing on developing and losing quick new increments of value, you'll be just thinking about what if, what this, what that. I believe that's not the correct way of thinking, if you work in a scrum team, in agile methodology. But when you have someone, people who are mature...just to comment on that bit as well. In my team, I do have people that embrace more the agile way and some others that are a bit more about checking all the requirements first, or " let me think about this for two weeks before I write the story." So that's the thing with Agile, it's nice.
Of course, in the end, everyone brings their own thing to the team.

[30:46] PARTICIPANT 17:

What I mean by maturity that's good is that when everyone has the same vibe, the same way of thinking, the same way of working, it helps. It brings the Agile way must faster, and better quality.

[31:11] RESEARCHER:

That is another thing I'd like to follow up again is you mentioned that you are self-organizing team. Are your truly a self-organizing team?

[31:23] PARTICIPANT 17:

Yeah. So, by self-organizing, maybe we have different definitions. To me, by self-organizing is we can decide or any person in the team can decide on what feature to work on for the next sprint. This is for me, this is the most important thing to state first, which means that it will not have anyone like a manager telling us, "okay, you work on feature X and the other guy works on feature Y". We all have our skills. We can develop all the stories that come into the sprint. This is another thing we also aim for and also for me as a scrum master. No story can go into a sprint if people cannot develop it. Of course, in the real world, some things slip up a bit but that's why we always trying to improve. But what I mean by self-organizing is that every person basically can pick up any story and we can decide on when to start and which one to start on.

[32:47] RESEARCHER:

So, you have some level of freedom of work?

[32:50] PARTICIPANT 17:

Yeah but this is all on the development team level, so then if you include the PO as well on the self-organizing concept, it means the PO also will not answer, quote, unquote, answer to managers, or management on a broader level. He will only answer to his clients, to the stakeholders because we have different teams that are like our stakeholders. If it includes the development team and the PO, these two components are self-organizing in a way that the PO can kind of decide where the team goes. Of course, in a bigger level, it's still a company, right? We still have to kind of follow the general path of the company, the general goals, the general objectives. But we will not have three or four people, not even like a CTO can come up to our team and say, "hey, guys, for the next two weeks you're going to develop this. We're going to put some new feature, some random one, just because I want to."

[34:04] PARTICIPANT 17:

So, we kind of have the freedom to decide on which direction the teams should go. Then, of course, what I think is also very important to mention here is when you have a self-organizing team and you have self-organizing most people, the people have to be responsible. And people have to deliver because in the end, if you don't deliver quality work, then you're accountable to the team. Sometimes this is a bit hard to have, to make the teams and the people accountable, because me personally, I believe that when you have too much freedom, then it's a bit harder to make people accountable for it for what they deliver. In the good times and the bad times, of course. To be honest, I think that's a bit of a disadvantage in the whole Agile setup. If you don't have responsible people and hardworking people, it can backfire a bit.

[35:18] RESEARCHER:

Okay. Fantastic. I do have follow ups, but I'll do them later on. I'm going to move to the next question, which you already touched on. But I want some more details. What do you do to assure quality or software quality in this Scrum setup?

[35:41] PARTICIPANT 17:

I think I would've said ...

[35:42] RESEARCHER:

Yes. We already touched onto four points. That's great. But I want to understand how agile ceremonies help to assure quality. For example, the backlog grooming. Do you do backlog grooming or whatever you call it.

[36:04] PARTICIPANT 17:

Yeah, so, yeah. So, I would say the backlog grooming is an action that's then between the PO and scrum master. Just to have a bit of pre talk about the features that we want to or that the PO wants to bring in for the team to develop. So new things, new features for the product. We do that. But I think the backlog grooming in this case, like I mentioned earlier, the quality here is more of a business needs. And that's also quality. So, the business would not like bugs, right. But maybe it's not that needed or maybe it's needed in some other ways because the development team also has a say in this, because I believe they also bring a lot of value because it's not like you are robots, like code monkeys that just develop what's needed. They also bring their own opinions on what's needed by the business.

[37:13] RESEARCHER:

So how do you involve the developers?

[37:17] PARTICIPANT 17:

So, when we go into refinement, and the PO brings so the stories that he thinks are needed, what business value it has. After a bit of the backlog grooming like you mentioned before, that's been with the scrum master. But then we go into refinements and the whole development team is there and the PO brings the stories that should be almost finished, written. Not developed, but written. And then presents it to the team and the team talks about it and the team challenges the PO. And that's OK. Why is it needed? This if the PO did not say it clearly in the beginning, because it's his responsibility to do that. But if the developers have questions, that's the place where they should ask them. And that's quite needed. And then we have a bit of a quick brainstorm on the complexity of the needed feature and after that a bit of a brainstorm as well on how we should implement it. Me personally, I'm not of the opinion that some teams do that, but not in my team. Some teams in the refinement, when they look at the story for the first time, they kind of try to solve or have a game plan on how to implement it.

[38:56] PARTICIPANT 17:

For me personally, I don't agree with it because I want the developer who picks up the story to also have a bit of a clear mind. So, there's no predefined, preconceived ideas on how to develop it. So, to also give them a bit of freedom. But then, of course, when he starts developing it, he can always ask the peers to help or even sometimes some things are more complex, then that person requests a meeting and we all go into a room and we just brainstorm together for one hour. But in the refinements, there's always a touch point where we all talk about it and we kind of have an idea on how it will work best to get this new feature into what we have right now. And then personally, in my team, we have a lot of, I would say, probably a big project that's been going for five, six years, which means a lot of it code. So, for every new thing that we add, it has to be carefully thought of on how best to do it because some things are easier. In general, all of the features have to be carefully thought on how to implement this. When developers choose their own story to work on, they feel ownership and duty to deliver better work and its quality is far superior. This is from experience.

[40:21] RESEARCHER:

I like to follow up on two things you mentioned. The first one is the ability of the developer to challenge, to challenge the PO and being involved from the start and the requirement discussion. Does it impact or does it change the behavior of the developer? Because this is a good thing. I mean, you're empowering your developers. This empowerment, does is change behavior?

[40:57] PARTICIPANT 17:

I don't know if I understand what you mean by changes in behavior. Having the possibility, you mean to challenge the PO? Is that what you mean?

[41:06] RESEARCHER:

Having the possibility to challenge the PO, for example, being involved from the start. It empowers the developers instead of handing to them the requirement in a thirty-page document or a hundred page document. It makes them involve. When you get involved, I mean, you feel good about yourself.

[41:33] PARTICIPANT 17:

Yeah, a part of it you want to do more. To do better.

[41:37] RESEARCHER:

To do more and to the better. Yes. That's what I was looking for. Because it changes your behavior. That's what I meant. It makes

[41:48] PARTICIPANT 17:

I totally agree. I think it's good to empower developers without feeling fear from the consequences. And also, not only empower the people to change behavior and to voice their opinions. So, this I can also sidestep a bit here. This is also a bit what the scrum master has to do. Because some people will talk less, and some others will talk more. But in this setup, once again, it's nice to have a scrum master that can read the room and I think that person over there has a question that maybe he's not raising it right now. So, it's also good to have people that's has more of the agile mentality, to bring the voice out of people. So, I totally agree. It really brings people closer to what they are developing in the end. In the end, if you're just going to your workplace, not now because we work from home, but you just do your new features and then you turn it off and you really don't know the impact it has then after a while, I think it gets a bit boring. And you just don't care that much about if it will be a good one or a bad one.

[43:20] RESEARCHER:

You feel like you are at the bottom of the hierarchy and things are coming from the top, down to you. And you are not given a voice. When you are given a voice your behavior changes.

[43:39] PARTICIPANT 17:

Yeah, yeah. True. Also, I think it brings a bit more confidence, right?

[43:43] RESEARCHER:

Yes, of course.

[43:44] PARTICIPANT 17:

I think in the end for me when we talk about this behavior and confidence. In the end, the bottom line will be that people talk more. They will give more opinions and everyone, of course, and some people are better at task X and some other people will be better at task Y. But all of them bring ideas. Maybe it's not the best ideas from everyone at every moment in time, but all of them will bring new ideas. And if you really have this openness, then I believe it will really benefit in the long term.

[44:21] RESEARCHER:

Fantastic. Okay. Can you share with me a positive story about Scrum or agile in general? It did something good, especially if you can link it to software quality.

[44:40] PARTICIPANT 17:

Yeah, so I can give you an example. I don't know how deep I could go technically, but I can give you an example on my team on what we did two years ago. So, since we do data, right, we have a data warehouse and we had all of our data in one database and one technology, and we migrated it to another one. And this is a really big thing because every day we have every day, at every moment in time we have, thirty, forty, fifty people always looking at the database. So, to migrate it was really a big challenge. And I believe that we did it in an agile way. And I will explain it a bit more why. So, if we did this is migration in an old way, like the waterfall way, this is probably how it would have gone. We would have gathered all the people that are using the database. We will have gathered all the jobs that are writing the data, and transforming the data. We would get all timing that these jobs are running, so we'll get all the information, basically we'll gather all the requirements that needs to be up. And then we will start developing on the new one only. After for example, one month, if we were in the middle of replicating a feature in the new database, but a new request would have come to change something that was like a big change, like a big bug or something. We would have to change in the previous one. And also, in the new one that was already developed. But it was just waiting for everything to be done.

[46:56] PARTICIPANT 17:

So, this will be really bad, because I worked in a project like this before. When it was like a two year migration plan of technologies, I left in the middle of it. And I don't think if they ever finished it. It was like six years ago. But in this case, that's how it would have gone as well if we did it in the waterfall way. How we did it: we just gathered the big jobs, the small jobs, the ones that more important, the ones that were less important, the ones that maybe could be down for a while. And then we just started replicating them, migrating them in the new database one by one. But as soon as they were learning the new one, we just deactivated the old one immediately, which means that for like one year, maybe we had both databases running, but we only had to maintain things in one of them.

[48:13] PARTICIPANT 17:

And this way, for people that were using the database, it was smooth. They didn't know if they were accessing the old one or the new one. Because we did it in a way that it was okay, let's just split this into very small parts and just list one after the other. And then sprint into production immediately. So, I think this was a very agile way of just taking chunks and put it in production. Don't wait for everything to be done to then turn off everything in your database and then turn on the new one. That would have been a disaster and that would have taken...maybe even now still trying to do it. And, you know how these things go also in software. The bigger the deployment and the bigger the chunk of work that you want to do in just one go. This will just mean one full weekend or one full week of the developers not sleeping or working fourteen hours a day, because there will always be bugs. Everything that you do or almost everything, there will always be some bugs that you are not expecting, some things that you just didn't...

[49:31] PARTICIPANT 17:

Of course, you can test things forever. It can be six months testing everything to the small detail. But for me, I really live well with let's just put something up and then we'll have really fast feedback and then just fix it. Just look at it and you think, OK, what's the problem, let's fix it instead of just trying to come up with all the possible problems that could come today or in one month or two months. I really believe in just put things there fast in the new one and have people testing. Even the real users, the real stakeholders. Let them also test it because they will be the ones also thinking about things that we did not think before. Because even though a lot of times, even in QA, you can think a lot about what will all the use cases be. In the end, only the customer, only the client will know what he really wants to do. And I know dozens of stories where you do something for six months, for one year. And then in the end, the customer looks at it and says, "but I just wanted to click on this button and do that" and we were doing something that was much more advanced than that for one year anyway. So, I think this was a really good example. of something done in the agile way. Just split it into small parts, put it in production in the new one, and then slowly disable it from the old one.

[51:20] RESEARCHER:

My last question is a little bit provocative, but the purpose is not to upset you, but to get you to talk and give us your opinion. The question is, what do you think of this statement: Agile produces poor software?

[51:38] PARTICIPANT 17:

Yeah, I understand where it might come from. Not having dedicated QA people or teams to look at everything that you put in production or that you deploy. I totally understand and I agree. If you have dedicated QA people, you'll have better quality software. I totally agree with that. But my concern is I think you'll have much better software in a much longer time period. And still, like I was saying just now, still there will be a lot of things that will be missed, not the quality of the software itself, but the features like what people really want. Because the longer that add more steps in the release process of any feature, any new thing that you are working on, the sooner that the client or the user just changes his mind. He'll just want something else. Or maybe he actually wanted that but since there was no feedback along the way for six months, it's not as good as he's expecting in his mind. There is this funny story, with different images, like different squares. And how does a developer see it and how does the QA see it, how does the user want it. And in the end, it's all totally different. The user doesn't want a house, he wants a dolphin or something. It's a funny image.

[53:51] PARTICIPANT 17:

But I totally understand and like I said before, also to touch a bit more on that sentence. Like I mentioned before, in some industries, I believe that it'll still be important to give more importance to having better quality software. Software that really cannot fail like health, like health-related software. But I'm already thinking about what else I could mention.

[54:33] RESEARCHER:

The purpose is counter any misconception about Agile or about the method itself to produce quality software. It is beyond that. It has to do with a lot of things. It has to do with the quality of implementing Agile itself. It has to do with the team, like we discussed earlier, the maturity of the teams, the performance of the team. It has to do with the people, their behavior. It has to do with other organizational variables like in your case, you've been given the freedom of choice. You choose your own task. It has to do with a lot of things. It's very narrow to think that a methodology produces poor software. It's beyond that. That's just what we trying to turn out.

[55:38] PARTICIPANT 17:

Oh, I get it. Yeah. I think when you say poor, it is a very specific word. Right. So, it's not like Agile produces things don't get done or you know what I mean. It's a very specific one. So, when I hear 'poor', there's a bit of emphasis on QA here, I hear poor quality. Like I told you, I do agree to some extent on that, of course. If someone told me Agile produces poor software, I would say no, of course not. It's not because you don't have QA that you will not have a lot of points like what I said, like we do with my team; a lot of points where the quality is assessed like five points, five or six points where the quality is assessed. But in some cases, yes, if you want to deliver fast and you want to deliver small incremental value. Sometimes, yeah. Software could be to have a lower quality. Now, this I agree with, to be honest with you. But for me and in a general approach, I still believe it's a better way of doing things. I think that in the end, you should use more like you recently, give people the freedom to talk, to give their ideas, the power to decide what to work on. And the PO also has a lot of freedom to bring the business value as he sees it. I think in general, you are hearing different opinions. And secondly, you are also taking a bit of the decision from only two or three people. So, I think that's very helpful to bring better quality. It's just delivered faster and for the user to be more user-centric, user stakeholder, or whatever you call it.

[58:01] RESEARCHER:

Fantastic, PARTICIPANT 17. Thank you very much. I'm done with my questions. Do you have any questions for me?

[58:08] PARTICIPANT 17:

No, we have, but I would like to read that.

[58:13] RESEARCHER:

The paper? Yeah, I'll send it to you once it's ready. I just want to check if I have your email. I do have your e-mail which is [redacted]. OK, fantastic. It may take a few months.

[58:33] PARTICIPANT 17:

Okay, yeah, sure.

[58:34] RESEARCHER:

Thank you very much. I enjoyed that very much. Thank you. Bye.