[00:39] PARTICIPANT 13:

Can you hear me, Researcher?

[00:42] RESEARCHER:

Yes, I can hear you. Good morning, PARTICIPANT 13.

[00:46] PARTICIPANT 13:

Hi.

[00:48] RESEARCHER:

How are you this morning?

[00:50] PARTICIPANT 13:

I'm okay, and you?

[00:52] RESEARCHER:

I'm really good. Sorry about yesterday. There was a mess-up. I double-booked myself.

[01:44] RESEARCHER:

Alright. Do you have any other questions for me before we start the interview?

[01:49] PARTICIPANT 13:

Ah no.

[01:50] RESEARCHER:

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

[02:05] PARTICIPANT 13:

Okay. So basically, I work with Agile for four and a half, almost five years in software development. I have been into three different companies and their models and the practices that they have were quite different. My first job was as an automation QA for almost three years. What I did, I did automation on .NET. I was part of an Agile team, like altogether team, like developers, QA, like everybody on the same team. After that, I worked in another company as a full-time Scrum Master for six months. They were kind of Agile but with the Waterfall side. Although they were doing sprints and iterations. It wasn't the iterative and incremental development that you are going to expect to come with the Agile. And then the last company that I worked for almost a year and a half. I was there full-time Scrum master. I had four

teams, and each of them had the ability to choose the methodology that they want to work. So, some of them would mix Scrum and some were doing Kanban. Some of them were able to do some kind of mixture. Although with my teams, I had three full Scrum teams and my Kanban team and yeah, I guess that's it.

[03:01] RESEARCHER:

To set the scene, let’s define quality? How do you define software quality in the context of agile?

[03:15] PARTICIPANT 13:

Software quality is both outcome and process. We measure the outcome by product quality; its ability to meet business needs and has no defects. The process is how we create and deliver the software. In agile, this is a continuous improvement process. It gets better by continuously examining the way we work, we learn and change for the better. In agile, we always experiment to come up with better ways.

[03:20] RESEARCHER:

When you say, “it gets better”; do you mean software quality or Scrum?

[03:28] PARTICIPANT 13:

Both. Better Scrum implementation means we get better at software quality.

[04:09] RESEARCHER:

Okay. Fantastic. I know we talking about a specific implementation of agile, which is Scrum. But if we use agile then we mean the Scrum. You worked with a lot of companies with different implementations of Agile. Can you tell us about a good implementation of Agile and how it looks like?

[04:25] PARTICIPANT 13:

Okay, so yeah, I can tell you what I think was working best for me. So the three different models that I had worked on in my first job as I told you there were [inaudible] teams where the developers, the QA, there is a designer available also, there was a mobile developer. Whatever what was needed to of those people were grouped in one team. So, in one team you have let's say back-end developers, frontend developers, QA, and designer if needed. This was very good because at the end of the iteration you can really have a done the product.

[05:09] PARTICIPANT 13:

The core QA tests it and its checked properly and I guess this was a very good model. Maybe for me this was working best from all of the models that I did work with. Second company, there was a different QA testing team. So we have the developers doing kinds of their Scrum in a team and then you have a different team that’s testing and for the developers you have different front-end team and different back-end team and then different testing team. So, the company was [inaudible] which kind of explain why they wanted to have this overhead, but this is not a good implementation of Agile because you were having one sprint for the backend team to complete something, then one sprint for the front-end team to

complete what the backend team worked previously on and then the next sprint for the QA testing team to test it, which was like the work was getting implemented very slowly and especially with having the testers in different team, you have the feedback for what you called in the next sprint.

[06:33] PARTICIPANT 13:

So always there is at least one sprint of delay in what you are doing. And also, at the end of the iteration you can't say that you're truly done because your developer and you expect some bugs to appear. And then they appear and then they have to go to your next iteration or something. It's very small. I guess this was not working at all. It's not really an Agile model. Although they were doing it iteratively. And then the third company that I worked on, what's interesting there, so you have the people in the same team, but what was most specific is that they didn't have any QA or designers or whatever. So, they had software engineers and those people in the team, they are responsible for coding. They are also responsible for testing their own code. They are responsible for doing regression, fixing things and everything. And then the teams, they were all of my teams with four to five people sometimes six, but mostly five people. So, for those five people, they should decide between themself who to do what. If they want to have any person that kind of dedicated to testing or if all of them will be doing coding of the features and then all of them will test their own features. So, from implementation point, this was working okay.

[08:15] PARTICIPANT 13:

Because the team can do everything till done. What wasn't working and I can clearly see because of my prior experience. The software Engineers because most of them were from a development background. They didn't really have any QA professionals. They wanted to have good tests and quality tests, but they were having difficulties with what should be the approach. How should I do it and then we didn't have anything centralized for the company like those are the best practices, those are the test scenarios that you should have. We were having more the approach like each team to decide how they want to go. So, the things that didn't work is that we had many differences in the test from team to team and from some parts of the system to other parts of the system. There were some teams who decided to have a person that was dedicated to testing and I guess this was working quite well for them. My teams, they didn't have anybody with testing knowledge, so they were all software engineers but kind of developers. They were struggling a bit with what should I test, how should I test, what is the right approach, when should I run my test mode of things regarding the test strategy. So, I guess this is working but the software engineers are introduced correctly testing and practices and so on.

[10:01] RESEARCHER:

Okay. Those are interesting stories. Can I ask you a question? What's the impact of a bad implementation of Scrum on software quality?

[10:17] PARTICIPANT 13:

Well, the quality, I mean that's the impact, if you have better implementation of Scrum, it will definitely help to produce better quality. But, it depends you can have a better implementation with Scrum and then still have okay quality. But a lot of the times if you have bad implementation of Scrum and I can speak for the second company that I worked for because this was really bad implementation of Scrum. They were saying, we are Agile, but they weren't trying to be. You have more Waterfall overhead, and your features are getting done much slower than normally. For example, for features, like with the other places I

worked with, we do in a sprint, they were taking two to three sprints and then you never know if it's fully done, if it's fully tested. So, the quality most of the times, it's worse if the implementation of Agile is better. If the team is kind of like, if the company's going on the traditional project management and the Waterfall way and the testing phase I believe that they can have good quality, but just the project won't be completed as quickly as if you have good implementation of Scrum.

[11:48] RESEARCHER: Okay. Fantastic next question. And what do you think of Agile and general? What is your opinion of that?

[11:57] PARTICIPANT 13:

Okay, so I am a Scrum master.

[12:01] RESEARCHER:

So, you're biased.

[12:04] PARTICIPANT 13:

I think it's very useful. It's much better than the traditional project management. The two kinds of things, to have incremental delivery and iterative development. That's helping a lot. Most of the people think about Agile as Scrum but it's not purely Scrum. There is so much more in Agile than Scrum. And yeah, I think if you have the right implementation of Scrum, we will be able to release things very quickly with good quality, hopefully. And then the other thing that's a very big advantage of our Scrum and if you're doing the whole thing correctly is that you will have quick feedback from your customers or your users wherever they are; customers or companies for the features that you're doing. You can easily adapt to their needs and have a successful product.

[13:12] RESEARCHER:

You mentioned something very interesting. You released quickly with good quality. How? Especially how does Scrum achieve good quality.

[13:27] PARTICIPANT 13:

Okay, so when you do your ... generally, Agile caters for technical excellence like having to have good code, having both manual and automation, but mostly automation. So, if you are doing the technical excellence practices of Agile, not just Scrum, then you can have very good quality code and with good quality code, it must release. Also, something that's very part of the Agile culture is to have continuous delivery and continuous integration, at least, if it's not continuous delivery. If you're doing all of those things while coding each iteration not [inaudible], but let's say at some point when you have a releasable product when you have an MVP, you'll be able to release it very quickly.

[14:39] PARTICIPANT 13:

And if you have automation tests, of course, you'll be able to release much quicker than if you have to test everything manual. So, it helps a lot and it helps that you're doing iteration after iteration after iteration.

So, each iteration, you can have something better. Like better automation tests, work through the code in my regression, each iteration you can make tests for the features that you are doing and each iteration, you can improve your continuous integration strategy. So yeah.

[15:23] RESEARCHER:

So, the continuous iterations, it gives a very ongoing and fast loop of feedback and that Loop of feedback helps in improving, right?

[15:40] PARTICIPANT 13:

Sorry, it helps in?

[15:43] RESEARCHER:

It helps in improving the quality.

[15:46] PARTICIPANT 13:

Yes. It improves the quality. The things that improve the quality clearly, good code, you should have coding standards. They are set by the developers. They should do code reviews and they should feel kind of responsible for not merging back all the time. Generally, if you think that something a team member had written is bad, you shouldn't have it in your code base. Then after the code, it's like good tests, useful tests because you can test everything, you can automate everything but most of the time you don't even need it. So, it should be done wisely. I used to have a strategy, what to automate, what to test manually, what to run what. And then if you release often there are bugs always so you will have some things to fix for sure. If you're lucky, they will be kind of minor. If you had missed something, it might be some incident that's a bit more impactful. But however, if you can release quickly, you can release quickly and release it again.

[17:11] RESEARCHER:

You mentioned something about code review. Do you do code review?

[17:17] PARTICIPANT 13:

Yes, but as a Scrum Master, no. In my teams yes always. In a team before the developer or software engineer marks their code, they should have a code review from, most of the places that I worked, at least two other people. That's the kind of general rule.

[17:42] RESEARCHER:

Yeah. I understand. Next question. Can you describe your Scrum set up, your current Scrum set up? You can choose any one of your previous projects or experiences and talk to us about it.

[18:01] PARTICIPANT 13:

A Scrum set up in the meaning of teams or how we're working?

[18:06] RESEARCHER:

How do you work and the process? How the process works?

[18:11] PARTICIPANT 13:

Yeah, it's going to be an example from one of my Scrum teams that are doing very good Scrum. Okay so we have software engineers, as I said, like in my last company. Okay, this will take a bit more time than I expected. So, on the higher level we are doing iteratively something like let's say high-level Scrum with all the projects that are in the company. So, we review them each week and then we plan our sprints for the next three months. So, on quarterly basis, the project that is going to be worked on by our teams. And then once we have this plan with the project, each team is assigned to a project that they are going to work. So, once they are assigned to something, the team has a product owner and a project that they are working on. The different projects you can think of them as kind of a feature, different features for one platform.

[19:33] PARTICIPANT 13:

So, once they are assigned to the feature with their product owner...with the teams that will do Scrum. That's most of the cases. It's just we start with planning normally. Prior to the planning, we are doing the backlog refinement or any kind of chartering session, design sessions. Those should be done before starting on your initiative and then once that's all kind of done and in place, normally during the sprint, we do a backlog refinement for the next sprint. So, we go in the backlog, check the stories with the product owner, ask if we have any questions to see if we feel comfortable with the stories or they should be changed. Check if the product owner should give us some more information or if he is not aware to ask somebody. So, normally on the day of the planning, we have our sprint clear, not sprint, I mean our backlog clear. So, the tasks are kind of clear.

[20:49] PARTICIPANT 13:

Some of them might be even estimates from the backlog refinement. Sometimes we just estimate them on the planning on how the team wants. And then on the project, we have it with the product owner, team and product owner agrees how many things they are going to take from the backlog. They estimate it with like our past performance for the past sprints. Just not to over commit or under commit. We check availability on things that are done, estimate things. With some of the teams, we do kind of very short sprint planning to where we have all the stories. We separate them into different subtasks that the engineers can get to know them to make the things quicker than if just one person is getting a story.

[21:49] PARTICIPANT 13:

But that's kind of an optional and then we are done with the sprint planning. During the sprint each day, we have the daily Scrum. In my case always in the morning kind of early our team is in the office. The first thing that you do is check up for a daily Scrum. As I said, we do the backlog refinement session during the sprint. At some point have our backlog clear for the next sprint and then at the end of the

sprint, we do a review with the product owner where the team is showing him the functionalities that they have worked on. At then the product owner is kind of accepting them or he has any questions, asks normally. The product owner is very involved with what the team are doing during the sprint so there aren't any kind of surprises. Like the team has done something and then the product owner says that's not what I wanted. Because most of the times the product owners are also part of the daily Scrum and they know what the teams are working on during the sprint, not just on the sprint review at the end.

[23:11] PARTICIPANT 13:

So yeah, we show to the product owner our work. If it's needed in rare cases will show it to a wider audience, if it's some kind of sprint or final sprint for the project or if there is interest for people that are outside of the team, like let's say our architect or something. We show it to a wider audience. And then after this we do that retrospective. We use different forms for the retrospective every time which is something suitable. It depends on what we want to have our retro focus on. So, is it process, is it like relationships in the team, is it some specific issues that we want to talk about? And yeah after the retro, we are going with the next day planning again, as I explained.

[24:16] RESEARCHER:

Okay, fantastic. Just a follow-up question while describing the process you didn't mention how do you assure quality in this process?

[24:28] PARTICIPANT 13:

Okay. So, for my last company, we kind of believe that the engineers are experienced enough to have their code at the right place. As I said, they do code reviews and they are testing. We don't have any formal practice for the testing thing like you should test this or this or this. We kind of believe that they will test whatever is needed. However, before a functionality goes to production, we are doing Scrum release sprints. So, with those sprints, each team has to do regression for a module. They have some kind of stewardship. So, each team is responsible for different parts of the system. The teams are doing a bit more extensive testing with self or regression that should cover the functionalities like everything that is essential and then on higher environment that's needed. So, this is how we assure it.

[25:53] RESEARCHER:

Okay. Fantastic. Can you talk to me about the retrospective what happens in them? And what type of actions you take from these retrospectives. Who implements these actions because that's a very interesting mechanism of improving the processes, is it?

[26:10] PARTICIPANT 13:

Yeah it is. When getting it done correctly. Probably the retrospective starts, at least for me, when I started to work as a Scrum master, I really liked those meetings because you can always experiment with the different format. You can do something different. It can be fun. However, they are very challenging because you should have actionable items not just being so not just complaining so it doesn't become a blame game. People should feel safe and they should feel comfortable in these retrospectives. So, I'm really putting some time to choose the right format depending on the situation. So, we have the discussion set in place and then we're taking the actions, normally the team. But I'm also taking some actions from

the retrospective myself if they're on more kind of a higher level, if the team needs some kind of training, if they have some problems with the environment that they cannot fix themselves.

[27:32] PARTICIPANT 13:

If it's something like on an organizational level and then those things in the team, let's say if they have an issue that they didn't test something properly and then at the end of the iteration, when the code goes into production they were having an incident and they were having issues handling this incident quickly. Any kind of action items that like this for the team, they are just dividing by themselves and take action items and then if there is something super specific it can go to their manager.

[28:12] RESEARCHER:

You mentioned something very interesting, which is people should feel safe. Can you elaborate a little bit on that?

[28:23] PARTICIPANT 13:

Yes, so the retrospective, it's very important that it's not becoming a blame game. It's not becoming some place where the team kind of just joking against each other. It should be a place where they will grow stronger as a team. It should be, so there's any kind of conflict in the team, the retrospective is a good place to manage. But it should be handled in the proper way otherwise the people can have a very bad attitude over the retrospective, and this is not going to give you anything valid. Also if they have some problem with the organization, if they're not happy with something, if they don't feel like let's say the organization's trying to improve them, if they're feeling kind of just used to code and they need something more like a training coordinator or a new project or whatever. This shouldn't be again; they should feel safe to say this and it shouldn't be something that's then used against them. Because the retro should be used as a safe place. And for this, most of the times these are their expectation. Most of the times I don't invite managers of the teams on the retrospective. So, the retrospective is for the team, product owner, Scrum Master, that's it. And then if there are any action items or discussions that should be taken outside. We agree. They are taken, we talk about them. If not, it just stays between us.

[30:33] RESEARCHER:

Another question on the same topic. Do you think Agile facilitates or makes it easy for people to feel safe in their work environment?

[30:50] PARTICIPANT 13:

I don't know. Maybe it helps. But it's not just Agile, you can feel safe even it's a Waterfall company. What helps really is the culture in the company. If the people have this culture, they constantly improving for individuals and doing regular one on one sessions with them, talking to your people, and knowing what's hard for them then they will feel safe. If the company does not, even if they are very good at Agile, or the retrospectives might not make them feel safe.

[31:17] RESEARCHER:

So, how does this feeling safe helps achieving quality?

[31:27] PARTICIPANT 13:

People become comfortable in their own skin. What I observed is developers, QAs and everybody else become fully present, they want the product to succeed. A successful product in software development means at least free of bugs and meets the business expectations.

[31:37] RESEARCHER:

What do you mean by “fully present” and how that helps quality?

[31:47] PARTICIPANT 13:

I mean more engaged and vested in the success of the product and its quality. They care more about quality. They write better code, they test more, motivated to meet customer needs, etc.

[31:57] RESEARCHER:

Ok make sense, let’s move to a different question. Do you think this Scrum setup you been talking about, is it a good implementation of Scrum and why?

[31:58] PARTICIPANT 13:

The last one that I have?

[32:00] RESEARCHER:

Yeah.

[32:11] PARTICIPANT 13:

It's an okay implementation. Kind of on the code start, there are things that might be improved. If I can say for something that having more knowledge regarding the testing and the quality regardless if it’s done by a dedicated person. The engineer's they should have some kind of guidance to do this. So that's one thing. Though it's kind of a good implementation and some other thing that I see myself struggling with and most of the people as I explained they're having their product owner together with the project or the feature that they are working on. So sometimes it happens that the team have worked with different product owners for the past six months, which is very confusing. They don't have the one-to-one relationship product owner team. I would improve it if it was my decision.

[33:19] RESEARCHER:

How would you improve it?

[33:23] PARTICIPANT 13:

How would I improve it, I would [inaudible] the product owner to the team regardless of what they are working on. They will have just a product owner. I guess that's not that easy because we were having a very big platform with different features. Like very big legacy system some of the places for the system, they haven't been touched for years. So, it's also hard to expect from the product owner to know-it-all. In an Agile environment, you have one product owner like one product owner can work with two, three, four teams. Depends. I don't know. However, but one team should always work with one product owner.

[34:20] RESEARCHER:

Yeah. Okay. Fantastic. Let's move to the next question. Do you think this Scrum setup you've been describing, produces quality software and how?

[34:38] PARTICIPANT 13:

Yeah it produces quality software. It can be better because for the things that I said like no common testing practices, no common [inaudible] for the testing. The places that we had good quality, it was because we had very good software engineers and they were really careful with what they're doing, they were really diligent in their work. It wasn't because of our great implementation; it was purely the people. The regression thing and the smoke thing. Again, we didn't have the strategy for having a common approach. We were doing like everybody's doing something as they deemed appropriate. But again, if you need to have transitions or something from one team to another, that's really hard. For the regression part regression should be done before it releases. I guess, there are a lot of opinions about this. But again, that is very specific for each team. Because if one team has to do the regression for other teams, that was just not possible, and you don't want this. You want everybody to be able to run the regression when it's needed.

[36:22] RESEARCHER:

You already shared with us some positive stories. Can you share with us a positive story about Agile and software quality? If you can talk about both, it would be nice.

[36:40] PARTICIPANT 13:

Yes, on my first job where I was the automation QA. We were doing the Agile as I told you, with the team set-up of developers, QA, all together in one team. It was very nice because we were having almost fully automated regression like I guess 95 percent of the regression were automated. The tests that we were doing for the regression that where manual was very little. We released very often. So, our iteration was twice a week most of the times, we released each week. So, two times per iteration we were releasing. It was all because the software engineers, they were doing all of the technical excellence practices. We were having the QA that were very responsible for the quality.

[37:49] PARTICIPANT 13:

Our software was deployed to a higher environment only if it has a QA sign off. I don't know if it's a very common Agile practice, probably not, but it really helped for the quality. And then because we were able to release quickly, when we were having issues on production, which is not uncommon for every project. I mean you always have this. What's important here is how quickly you can react. So, we were able to

react very quickly and like most of the times when we had an issue, we fixed it the same day. Fix it, deploy and then there is no issue. So, if you can do it the same day, it's great. Most of the times we did it and I guess this is a very very positive story.

[38:42] RESEARCHER:

Yeah, it is fantastic. Thank you very much. How about a negative story?

[38:50] PARTICIPANT 13:

A negative story...The company that I worked for where the QA was in different teams, different backend, different frontend. Like almost all of our projects were delayed with six months. Always knowing you will have six months delay, and you kind of do the job and again you have this base where you are just fixing your defects because they are plenty and everything is happening so slowly. You do a feature and then in an almost grand scheme, you will do this picture test it, if you're not happy then you go back to dev and then eventually it will be done. And there you do the feature from the backend, then it's done from the frontend. And then you need to change something for the backend because something won't work with this back end and like the same feature going over and over and over so many times before being completely done. And then you think it's completely done; it goes to the QA and they find some issues.

[40:09] PARTICIPANT 13:

Most of the time don't even fix them because you are late with the other features that have to be done and we're just, yeah, we'll fix this when we have time. And then eventually you need this like six months, now an additional three months, additional phase where we are just fixing the defect. That's not a good story. I can give you another one that's a bit more on the Agile part because this was a bit Agile, but kind of Waterfall just masked as Agile.

[40:43] PARTICIPANT 13:

In my first job very when I started, we had a six-month project. It wasn't something very long-term for one client and we were doing the Agile, we were doing every iteration, whatever and then when it comes to the release, the client was like, yeah, but I really wanted this and this and this things. So, if the people that work with Agile, especially your client and your stakeholders, if they're not familiar with the concept, if they don't understand that once you show them something in the sprint or core review, if they accept it then this feature, we consider it done. We don't go over and over and over again to it. If a client is very indecisive and they are not like...if we didn't have the product owner, but the product owner was our client, but the client was very indecisive, and they didn't really know what they wanted. But what was the problem was that they didn't want to release.

[42:04] PARTICIPANT 13:

It's more like they wanted to have everything altogether. And this way you're just going back in circles because they don't want to release something small, just to test to see how their users will interact. They want they want all of the features. They are not sure what they want to do and going back and forth and back and forth. And then you are nowhere.

[42:32] RESEARCHER:

These dysfunctional teams, this functionality. What's the source of this dysfunctionality? Is it the implementation of Scrum or it is inherent to the legacy of the team, or where does it come from in your opinion?

[42:53] PARTICIPANT 13:

I guess it wasn't from clear strategy and kind of lacking on vision because the team was very experienced. But as I said, we as a team because I was QA back then. We didn't have a product owner and this part is very important for Scrum. But when you don't have this person that can take responsibility for what should be done, it's very hard if you are just taking requirements from somebody who is not familiar with the Agile practices and principles. So, this was the problem. Generally, a lack of strategy and that we didn't have the product owner, but we were using the client as kind of a direct product owner, but the client wasn't very familiar with the Agile practices.

[43:53] RESEARCHER:

Okay. Fantastic. Thank you very much. I have the last question. It is a little bit probably provocative and we trying to get your opinion, not to upset you. So, what do you think of this statement: Agile produces poor quality software?

[44:16] PARTICIPANT 13:

I can't agree with this for sure. And I can tell you why. This is the Agile thing, not purely Scrum or not tied to any methodology is to be able to react quickly to changes to release the software. To get feedback often and to inspect and adapt your work as it's deemed necessary. So, if you do this even with the quality, let's say release often and get feedback from your customers and then kind of do the things that are required to test your software. If you do this properly, you can. A lot of people think that Agile produces poor software because especially I see this in some of my teams with developers that have worked ten years as a software engineers or fifteen that have a lot of experience from the past. And they're very used to wanting to see the requirements, they want to have it all written down. They say, I want to kind of sign with blood that this will be and when I code it, it won't change. Even if you do Waterfall, again you would at some point have to change. With Agile, you really want to be happy with being given feedback and just be able to address this feedback very quickly. And then a lot of people also think that when you're doing iterations or something, you're not really going to test it. It's not going to be great. No, that's not the case actually.

[46:18] PARTICIPANT 13:

If you release often, if you test often and if you release at the end of each iteration, you test at the end of each iteration, you would've done much more tests than if you are waiting for a traditional project management. And let's say, you code six months with no testing and then you do three months of testing. For sure, the Agile way is working better. It's not reducing the quality software. But a poor implementation of Agile, really may produce poor quality software.

[46:56] RESEARCHER:

I agree with that statement. Yeah. I have a strong feeling about that statement. Poor implementation of Agile creates poor software. That's how it should be.

[47:07] PARTICIPANT 13:

Yeah, it's not the Agile itself and even Agile, there are so many methodologies and principles like you can go Scrum or lean or extreme programming or Kanban. Like Scrum is not their universal truth for all teams. Some teams and projects are more comfortable with something else. There is a lot to experiment. But if you are just following the practices to get feedback to do things simple and with technical excellence you will have good quality with any methodology that you choose to implement.

[47:47] RESEARCHER:

Fantastic. It was a great interview. Thank you. I've run out of questions. Do you have any questions for me?

[47:54] PARTICIPANT 13:

Well, just one question. I saw that you can send me the result when it's done?

[48:04] RESEARCHER:

Yes, that's fine. I have your email [redacted].

[48:18] PARTICIPANT 13:

Yes.

[48:18] RESEARCHER:

I will send it to that email once it's ready.

[48:22] PARTICIPANT 13:

Yeah, it will be very interesting.

[48:24] RESEARCHER:

It will take a few months.

[48:28] PARTICIPANT 13:

Okay.

[48:30] RESEARCHER:

Thank you very much. Have a good day. Bye.