[02:40] RESEARCHER:
Hi PARTICIPANT 11, how are you?

[02:42] PARTICIPANT 11:
I'm doing well. How are you?

[02:44] RESEARCHER:
I'm really well, thank you. And thanks for your time today.

[02:47] PARTICIPANT 11:
No, not all. I think my video and my microphone was disabled by Windows and iBlocker was blocking it and that's why I couldn't hear you. Actually, I could hear you, but you weren't able to hear me.

[03:03] RESEARCHER:
No, I wasn't able to hear you unfortunately, but that's fine. We are here now. Let me start by introducing myself and telling you what I do and why I'm interviewing people. My name is [redacted]. I'm from the [redacted] and I'm a postdoc at the IT university which is based in Denmark. I do research on software quality. I try to understand how software teams and the processes and methodologies achieve software quality. Currently, I'm researching Agile. I'm trying to understand how quality in general is achieved in Agile. We do interviews because we believe that knowledge is embedded in people thoughts and perceptions. So, the interviews is a way of us to extract those thoughts and perceptions and using those thoughts and experiences from practices, we managed to bridge theory and knowledge, theory, and practice. Yeah, so that's why I'm doing interviews. Do you have any questions for me before I start?

[04:52] PARTICIPANT 11:
Let me ask the question, are you trying to focus on the system quality that you are building with the Scrum process? That is what the major takeaway from the interview will be?

[05:09] RESEARCHER:
Yes, correct.

[05:11] PARTICIPANT 11:
Ok. Not the process, but how does the system actually work over the period of time after you practice Scrum?

[05:19] RESEARCHER:
What I'm looking for is how Scrum contributes or how it helps or contributes in achieving software quality, but let’s define it. How do you define software quality in the context of agile software development?

[05:38] PARTICIPANT 11:
OK, great. Software quality is the end result. It has a structural perspective which is the code and the design. It has also an external component which is free of defects and meets the end user requirements.

[05:43] RESEARCHER:
Great, let’s do some introduction. The next question is, would you like to introduce yourself? Talk to us about your experience and what you do.

[05:57 PARTICIPANT 11
Sure, so let's start with Agile. That would be my recent experience. So far, I have six years of experience in Agile, but it won't be like proper Agile because you know how it works in the software industry. It's more the framework modified to needs. So, I do have knowledge about how Agile works because I have management in I.T. I have a degree recently being focused on management in I.T. So, I know how it works. I have learned about Agile practices because I have some certification done. And that's product owner certification being done recently. So technically, I know what the book says and how things are actually. So, I can relate to what you're talking about. It's not how what the books say, you don't go by the book most of the times. You have to manage and have to modify it according to your needs and the organization needs. So, going forward, I have worked with the airline industry, telecom industry before. I do have some freelancing projects done as well. That's a nonprofit organization, but mostly, you don't use proper Agile format in freelancing unless I a proper team set up doing all the work.

[07:26 PARTICIPANT 11
I'm in Canada. So, my focus is mostly into Agile projects on how I can make the processes better. And, with the help of processes makes the system more robust. Agile actually helps you organize things well as you go forward. So that's what I have learned throughout my journey. So, I started with Waterfall, then into a hybrid model. and then went into Agile. I do realize that Agile does make the process much better and easier for the people as well and clients as well.

[08:30] RESEARCHER:
How?

[08:31 PARTICIPANT 11
How? Because it's more organized. You have the time period given to you in order for you to work and for the developers and testers who know what they have to do. It's more transparency that is maintained. And waterfall you are not aware of what you're going to see in the end. I've seen projects going into trash and they don't follow this kind of methodology. There aren't any chances to interact with the customers on a timely basis. You have to have constant communication with the client to know what their needs are. You got to develop something and at the end of the day, you're just saying, OK, we have done this. And the client comes up saying, OK, we don't expect this. What we told you was A and what you have developed is B. So, which doesn't really synchronize.

[09:26 PARTICIPANT 11:
And then the customer or the vendor, or the implementation partner who is actually into the project says, OK so this is not what you want then we have to start from scratch. That means no transparency at all. The customers are not able to see the product on the ongoing basis, what is developing doing. How development is processing the requirements. I mean, even getting the requirements properly in the first place. So, there is a lot of confusion. And as a third person, as a developer or business analyst when you sit back and look what the people around are doing from all sides. I need to see they've done something else and you're getting something else. That's where I really think it's very important to clarify things on the foreground. I like how Agile behaves because you can change it on an ongoing basis. You don't have to have the fixed thing at the beginning. Of course, you need some groundwork to be done. Over a period of time, you can change that requirement as it comes because even sometimes if you ask customers what they really want, they could be in the situation that they wouldn't be able to share it in the first place before seeing the POCs. They not clear about the requirements or they don't know the minor details that can be missed or high-end requirements and then the other ones who have to go and do it. Tell them many options and that you have option A, option B, option C which I think is feasible. I mean, the customer always wants the best. You have to see how feasible it is, or how cost effective it is, or how much resources you have for that particular product. So, these things are really important when you try to match it with the client requirements and then get it done?

[11:49] RESEARCHER:
So, is it transparency just on the level of client and the development team or within the team itself?

[12:00] PARTICIPANT 11:
You talking about Agile processes?

[12:02] RESEARCHER:
Yeah.

[12:04] PARTICIPANT 11
It could be actually synced. Usually they make sure that it syncs because of whatever the requirements come and how Agile works is you have...I work with so many projects, how it works. You first get the requirements from the clients. Those are high level requirements. Those are just one liners, or two liners. So, you get them, the product owner, or the business analyst, what he will do is just jot them down. Probably there will be of the views if they can't, they need to go back to the team or the architecture, whoever in charge of architecture on the side of things. They have to go back and check how it's going to be implemented another way. So, they could be high level requirements and they need to come beforehand. Before the sprint starts. I mean, maybe two weeks before. Depending on how much time they need to get the thing started.

[13:09] PARTICIPANT 11:
You have something called a sprint grooming as well. So once the requirement comes in, what we do is, we groom it in a way that can be directly taken into the sprint by the developers going forward into the sprint and then we can start working. Of all the stories you implement in the sprint has to be in such a way, it has to be groomed properly before grooming it, you cannot just pick it up. You have to have its exact application. So, you have to take in the requirements in such a way where you have the prototypes in place. What this thing looks like, what the functionality will be, what will it impact. There are so many teams working on the project, like the Google analytics team, so everybody has to sync. So of course, they will have their own sprint cycle going on. We have our own. Sometimes they don't match. But they do have to match in such a way that some things have to be clarified before the sprint begins. And they could be part of sprint. And it's better actually if everybody's in sync with the sprint. Otherwise, it becomes chaos even requirement that are coming from. What we had in the [inaudible] the product owner was client facing. So, he did the requirements and they used to make sure they want this particular thing done in the release. So, we had releases going on for a month or so.

[14:49] PARTICIPANT 11:
Every month we had a release. So there needs to be new feature or some new upgrades going on toward this release. So beforehand they had to specify that this particular thing has to go into the release. Then people start discussing, OK, so for example a feature has to go in this screen and talk about only this feature. You're not going to go on and on, because we have to get this one done before the month ends. So that's a sprint, basically, but the development team gets

two weeks of time. That means you have two sprints. Each sprint was two weeks. So, we used to have two sprints for each release. Whenever the sprint starts you basically have the prework done beforehand with the product owner and the clients. The requirements used to be gathered in such a way that you know what should go in what release and accordingly you plan it. Am I clear?

[15:56] RESEARCHER:
Yes. I'd like to follow up on transparency. You talked about Scrum or agile making the process transparent. How this helps achieving quality?

[16:09] PARTICIPANT 11:
It does help. For example, I talked about the demos; during these session we show to the customer the features and we get feedback. This feedback helps us to improve the quality and reduce bugs. That's with the customer. Within the team, our work is transparent. In daily bases I know what my peers are doing and I can comment on its quality. Another example, is when a developer push a pull request. Other developers can see it and review the code quality. The review helps improving the code quality. I hope this make sense.

[16:26] RESEARCHER:
Yes. I want to ask some another follow up question. You been talking about syncing in. Is that knowledge sharing? Do you mean that's making sure that you share knowledge across the team and the whole team have the same level of knowledge?

[16:35] PARTICIPANT 11:
There could be situations where the whole team is not that knowledgeable. There could be some developers who are new, or some are highly qualified. It also depends like you have something going on, a release that went before, there was one developer working on that particular release and the other developers don't know anything about it. In a technical Agile world, ideally the developers should know what each one of them is doing. But in the real workplace, I don't think that happens. Most of the time, you're too busy in your own work and then you hardly have any time to notice those things. But the business analyst or the product owner when he sees...the grooming meetings are basically conducted in order to make sure that all their teams knows what needs to be done at least. At least the high level knowledge has to be given. And then are there some developers who will be part of the grooming meetings or the period of time from the beginning to the end. Just like architects or team leads who will actually come up with the questions that are questioning around the functionality. For a feature or a story to be successful, you need to have many things in place. They need to have the designs in place. They need to see what are the attributes that they are sending to the back

end, what they are getting from the back end, and how are you communicating with the others, is there dependency with other teams that needs to be clarified from different perspective? So, they could be chances like even the Google analytics team. So, whenever you click on something, there is online application or something that has some of the elements that you'll see. So those elements are actually connected, the Google analytics, just to see how the user actually behaves with their system. So even that has to be done.

[18:21] PARTICIPANT 11:
So, everything you have to consider in the story and make sure all the things happen meaningfully before the developers can take it out into the sprint. All the groundwork has to be done. So, we need to make sure that in the grooming meeting itself that the developers understood what they have to do and what does need for them to be successful in order to shift the product or the feature. If they have any questions, they can come up. Even if it's as simple, it's more clear but if there is a doubt then it has to be clarified before the thing goes into those sprints. Otherwise, it's not properly clear. Then if it's not groomed, you don't take it. But there are situations where somebody, some developer or new recruitment who doesn't know anything about it who would probably ask highly qualified people who are more experienced on the application just to share some knowledge so there could be knowledge sharing. Functionality wise, how the code works, even the customers or the business analysts won't be able to explain to the junior developer or the person designing. From a sanity point of view, the business analyst or product owner, they go through the application saying that, okay, this is how it works. In the initial stage of things, you can give knowledge but that would be really high level. But most of the time, we do have knowledge sharing when a new feature comes up with something that has to be told to people like this is how it's going to work. So you do have something called as grooming sessions in which the story is mentioned and everything that is related to the story, the functionality, everything is explained to the developers by the business analyst or the product owner so that they get what they have to do. Eventually they would have to point the story and if they don't have knowledge about it, they cannot go and do the pointing.

[20:48] RESEARCHER:
Ok, well, we'll get to some detailed questions. Can you describe your Scrum environment? Choose any of your experience and talk to us about it, how Agile was set up? Who is in the team? How does the team work together, et cetera?

[21:11] PARTICIPANT 11
Let me start with my experience because I have seen literally from scratch how teams work in Scrum. Before we had waterfall and there were one thousand five hundred defects or something that were in place. There was a reason behind it. A lot of the things were changing after every

six months. We have new updates coming up every six months. And it's just for everybody to actually go back and write the code from scratch. So, what we usually do was we just upgrade the system and then they fix the issues related to the upgrades. So, there were so many accessibility defects, there were so many functionality defects, so eventually it was all about the fixing.

[22:09] PARTICIPANT 11:
So, then all the developers were quite frustrated. Development is not only about fixing the defects. You have to come up with some new features that are exciting going on this project. Even the clients were worried about it so they had a time limit that they could actually be able to fix all the defects and then they could start the new work. So eventually it took three months for all the developers. We had around fifty developers. Of course, it was not in one team. They had a print team, so adding boxes like you have a shopping thing, where you make a switch, or something lined up on a page. So those two pages you see like one thing. The other team is a check out team. Then the other team is a payments team. That's how it goes. And the application had about fifty developers. But you have different teams that are working across development. So, we started with this and we were not Agile. They used to call it Agile, but it was not Agile. So, then we joined the team and then we made sure that all the defects needs to be completed. Usually we don't story point the defects. Because what you expected to do before it actually goes as a shipped product, all the defense has to be completed. The story cannot go along with the defects. It's passed and then you do not see any defect in it. So, it's complete only when it's passed otherwise it's not.

[23:59] PARTICIPANT 11:
Going back to the point, so we had three months of time where all the developers are given some time and the business analyst must make sure that, OK, this is the time that you have to fix all the defects and then we would go Agile for new feature development.
It was all chaotic because the board has to be set up. And we did not have anything on the board. So, it was completely empty. So first of all, we had to make sure what are the responsibilities of the core team members - what they need to do. What exactly do we have in place? What work do we have? And what are the responsibilities of each one of them? Like how many QA developers do we need. We already had some basic backlog items or action items so it didn't become chaos from the requirement side, but we had to make sure how the team would work on those. So, tickets went going forward, whatever their new implementation is. How did they take it? So, we actually hired two more developers because it was just about [inaudible] so there was not much work to do. The two developers hired, there was one architect hired because we needed to upgrade it from the lower end system to the higher end. So, we needed an architect who can literally do some kind of work for the thing. We would actually say, okay,

let's look at implementing best practices or put the unit testing in place because we didn't have anything.

[25:41] PARTICIPANT 11:
Last year, March, we didn't have anything in place. The goal that wasn't written before was completely messed up. And we had a lot of bugs in it. So, we had to have an architect. So, in my team, we had one architect which would take care of all the high-end. Those defects that were not resolved by the team, he would take care of it or any hand-holding needed or best practices to be followed so in future, you do not have to face such difficulties again. There is no point in creating code that you're going to waste time on fixing the defects. And then we had three developers working under him. One was the junior developer and the two others were senior. They had eight years of experience each. Then it was a business analyst and it was me who was actually responsible for taking things from backlog, the high-end requirements and then split it into smallest features that could be implemented. So, you divided into feature able stuff. We had a designer who could make the modeling or who could come up with some prototypes that could help the team to actually visualize what they were going to develop. Then we have to three testers there were two testers and one tester was a QA lead. We had one scrum master and I think that was it. Of course, there other teams that were working there, but the shopping Agile team - this was the team. We had Google Analytics team that was working in sync with this team, but they were not called part of it. Have you heard of Kanban management systems?

[28:10] RESEARCHER:
Of what, sorry?

[28:12] PARTICIPANT 11:
Kanban management system.

[28:14] RESEARCHER:
No.

[28:16] PARTICIPANT 11:
And there is something called Kanban management system where you can directly update anything and in the system.

[28:27] RESEARCHER:
Yes, I understand.

[28:29] PARTICIPANT 11:
You have to go on every particular page, and you can update it in one system and then it could be published. So, all these different themes needs to be coordinated together but this was the core Agile team that worked on the shopping features. The [inaudible] team, database team, accessibility team, for that team. We had one accessibility QA as well in order to make sure it's accessible by the disable feature and what is visible now.

[29:07] RESEARCHER:
So, OK, I do have a follow up question, but keep going.

[29:14] PARTICIPANT 11:
So, these work like a shared resource because we had so many teams, so these people were a shared resource. These worked for all the teams. So, this was the core Agile team.

[29:36] RESEARCHER:
So, you mentioned that the test team was offshore. So how did you, in an Agile sense, you need to collaborate, and the teams need to be together, at least I think this the scrum stand up and all the ceremonies have to include all the cross-functional teams. So, how were you collaborating with them?

[30:06] PARTICIPANT 11:
The collaboration was through emails that were passed back and forth. Morning standup calls at 7:30 US time. So, it was evening back in India and morning in US. So, whenever we used to have this grooming sessions where we were discussing the features or the acceptance criteria, so we had those QA joining the meetings and made sure it was really early in the mornings in US. So, they are able to attend the calls. The QA lead was actually onshore. So, he was used to the sessions if they are missing anything on it or he used to clarify that doubt. They could email us as well, or the BA or the scrum master if they have any impediments. If they have any questions related to the features, they can come back to me. Or if they have any questions related to the development team, or something is not working with the code that is not running properly, some deployments not done. I don't say that it went really smooth in the first place, but after a certain time, like usually after the weekend, Mondays, we still had weekend on their Mondays, so we have to plan accordingly. So, before the start of the sprint itself, this is what needs to be done and the QA has to come up with the questions beforehand.

[31:56] PARTICIPANT 11:
I know there are some questions that needs to be answered on time but make sure you at least that you're clear with the functionality what needs to be done. And then we used to continuously

make sure that we do have notes attached to every story. If you have any doubts, make sure that you at least address it publicly, within the team, so anyone who knows about it can directly contact you. So, we do have Agile management boards, so we used to use Jira, we used to use VersionOne after some time, so we moved from Jira to VersionOne. So, let the developer know if there is some impediment and say OK, I'm facing this issue, can somebody help. And even when they are updated by the developers or the product owner, it's very important to actually update it with your comments. What happens is that even when the work is completed, why couldn't you go forward or is there anything blocking you. Or do you have any questions - that's why people really need to communicate. At times they wouldn't hundred percent come back to you and it's not like you just have one ticket in place, there are many tickets that need to go into the quality team. QA has a lot of work to do, it's not like they are sitting idle. You should take work and finish it off but sometimes there is a situation where actually it can be blocked. And, then there's no clarification from somebody so you not going to go head. But at least comment on it. Commenting works even in development, even for the management. If the management sees it, they can put it in one queue, and a developer picks it up and keeps it in progress status for long.

[34:04] PARTICIPANT 11:
Of course, it's going to be bad impression for team as well. And the management wouldn't happen to know that what exactly is going on. It's very important for everybody in the team to participate properly in the sprint and make sure that you go through the comments whatever is provided by each one of us and try to clarify things as much as possible. So, there are no confusions and later point of time there is no blame game, like I told you, but you didn't respond. You can just go back and look at the ticket. He had asked you this question and you were supposed to answer this. So, everybody does get emails, right? So, whenever somebody comments you get an email saying that okay, even if you're tagged suppose, I see under its Researcher and this is my question. So, people do get an email saying that okay, PARTICIPANT 11 has asked this question. Do you want to comment on it? So obviously you check your emails daily so somebody at the time it's not me there is somebody else in team who can really help you out with something. That the easy way to communicate and it does keep records as well. So tomorrow maybe after a month or so, there is something related with the same ticket going on. You can go back and check like what was happening in the ticket. Why was this issue raised or what was the main cause of it? So that helps communication helps. Everything I think should be written. Normally, it's very difficult to even remember or there could be a lot of miscommunication even if people are on floor and I've seen a lot of miscommunication going on.

[35:51] RESEARCHER:
Correct, yeah.

[35:53] PARTICIPANT 11:
So better to write it down. It's not taking a lot of time.

[35:58] RESEARCHER:
So, you already answered some of the next question aspect, but I will still ask this question. Do you think this setup is a good implementation of Scrum and why?

[36:15] PARTICIPANT 11:
I would say if Scrum is followed in a proper way and people do understand the Agile principles pretty well, I think, every software development life cycle would literally go in a proper direction. There are of course some failures toward for example, there is miscommunication at some point of time. You can't just go back when the sprint starts. You certainly can't end it up. So, the foreground work has to be done. The planning work has to be done very properly. It has to be substantial. Without that just jumping into something that into a development phase, it's not going to work. The more grooming you do, the more clarity you get on the features. And one should know what needs to be done in order to be successful at Agile. You should have proper clarity; not only just one person but everybody on the team. It's basically teamwork.

[37:23] PARTICIPANT 11:
Scrum cannot be successful without team collaboration and teamwork, even if one person wants to work in isolation it becomes pretty difficult for that person. Scrum is a very inclusive process. It's more like a friendly way of doing things. Nobody is forced to do anything. It's more like leaders over management style. Everybody gets to speak their own news sprint, groomings, retrospective. The best thing I like about Agile as it's actually gives you a chance to improve every time you go ahead. I know starting with Agile it's very difficult for the team members as well. Sometimes people get irritated with so many meetings going on. And as I told you there is so much of work that needs to be done. In waterfall, the developers just get it. Get the requirement, get it done and they are done with them. They don't have to do all the follow on work and they're pretty cool about but in Agile because of the time boxing that we have, it could be a little bit of more pressure on developers, but that's what it is. Like when you story point it, the developers do have a place where they can really speak it out. But are they okay with implementing it and so much of time or they need some more time with it? Agile is pretty flexible.

[38:54] PARTICIPANT 11:
So, at any point of time or somebody's uncomfortable, they can reach out to people and more information on it. Being more flexible and even the customers are happy with the process. I haven't seen any customer that does not... If you tell them that this is a problem and that's why we cannot do it or okay, we have done it and then you have a shell of a product in front of you, which you actually can demo to the client. The client is more than happy to actually help you with anything and everything that he can. It's good for the client, it's good for the team as well. There is not a lot of pressure on both the ends. So, I think the way you could keep both of them happy... I mean Agile is a good way to do it.

[39:46] RESEARCHER:
You talked a few times about grooming sessions. Can you explain to me how do they work? What happens in this grooming session?

[39:57] PARTICIPANT 11:
So, for example, somebody comes up to you saying that I need to build a website. I need to come up with this system. So, this is just a high level thing. How do you know what exactly needs to be done on the system? So grooming is more like getting to know the requirements in a finer way. So, for example, you have to create a login page. So even the login scenario itself might have so many scenarios in it. For example, then you have the username, you have the password. What are the username fields, what are the password fields? What if both fail? What is the time that you need for a username? Are there any specific characters than needs to go? So, what about if the user enters it three times and still is not successful? What happens after the user is logged in. There are so many use cases that needs to be created. This is just one functionality that I'm talking about, but there could be multiple functions related to it. They could be post conditions like after the login page for example sometimes you're logged out of the page suddenly because of the timestamp and you still have to login. So that doesn't mean that user has to go back to the beginning page again.

[41:20] PARTICIPANT 11:
Wherever he has landed in the current state of the application. He has to go there. So, all that really matters, so there are so many use cases and you need to take care of even when writing a single login functionality. There could be such hidden functionalities in your application as well. The client actually gives you a higher level requirement, but then you are the person who has to go and do it. For example, what if the functionality fails three times. If I say the log fails three times and then what needs to be done. So, are you going to send them an email, are you going to send the link, or are you going to ask them a security question for the password retrieval, what is the process that needs to be followed? So that could be a high-level

requirements. Then they could be lower level requirements and there could be more granular requirements related to it. Think about all the possibilities that are there that are related to that particular feature.

[42:30] PARTICIPANT 11:
So, grooming what it does is, there are people sitting in one room and actually thinking about the various possibilities that can take place. Is there anything missing that we not thinking about that scenario, are we all groomed with this functionality? Because when I say I'm going to do the login page, all these things come into it. So, is there anything that is left out? So grooming is basically you have to think about the feature, are we missing something, are the screens not ready or are we okay with the backend system. Do we have the backend system in place? How are we going to send it? So, we are going to make a login call for sure. So, the payload has to be username and password, only then it's going to be you know activated. So, for authentication, what do you need? Can you send from the front end to the back end? What is going to come from the backend? What information am I going to get? Because not every user is going to get the same data because depending your password, you're going to get it. So, what goes in, what comes out, everything you have to discussion beforehand. Which teams are you want to connect with? Is this going to impact anywhere else in your system, is there any data sharing going on like? Do you have to share your username and password throughout the application or throughout the team's or it's just going to be this application.

[44:13] PARTICIPANT 11:
So the things that needs to be taken care of, so the architect most of the time you have different architects or team leads that join this team grooming sessions in which the answers over the possibilities that aren't we actually lacking any functionality or we are not thinking about it. So that needs to be going into this.

[44:34] RESEARCHER:
That's bring me to the next question. What do you do to assure software quality in Scrum?

[44:46] PARTICIPANT 11:
First thing is documentation, the acceptance criteria has to be in place. It has to be written and in the right format. Do not miss any of the use cases even if anything is remotely related to it, just jot it down, every use case is important. So, there is something called acceptance criteria and there is something called as definition of done. Acceptance criteria has to be clear enough for the QA team to understand what the process of this is, and this has to be followed. What is the critical path that needs to be followed in order to make this functionality as definition of done? Only when all the acceptance criteria on that you can make sure that the stories actually

done. Even if one acceptance criteria is missing or you know, there's some okay, there could be some cosmetic changes or some cosmetic defects that are related to the features, but those are low level, whether it’s' important or not important. Like the button color is white and even the background is white. That is obviously going to be a high priority issue because you do not see the button anymore. But if it's slight red, not dark red, but a lighter shade of red. So, it's a low priority issue because the functionality still working fine, and the user is still able to access it.

[46:22] PARTICIPANT 11:
So, depends on the product owner actually whether he really wants to make it as a high priority issue or a low priority issue. And then the QA team has to understand all the functionalities. Those people are part of the grooming sessions because they need to know what impact it's going to have elsewhere. So, all the user flows that needs to be taken care of. Suppose if I'm a user and it's a banking application, it's my husband's side, but I'm the one who's going to use it too. So, I have a different password, but we see the same application. So, who is the user; there could be different users? There could be one main user, sub users or maybe an assistant can have administrator or whatever. There could be multiple system users. So, one functionality can be implemented for all the users. So, the QA has to know which functionality is actually attached to which kind of actor or user so that you know, all the use cases are actually jotted down without leaving a single use case in question. So that is one thing.

[47:40] PARTICIPANT 11:
Second thing is if they have anything prior to this implemented similarly in some other application. They can actually have a reference toward them. This is how it was implemented so you can implement it in the same way. For example, we do have sometimes POCs available. So, something that has never been developed before in an organization, we actually start with the POC that is proof of concept. So, the developers actually come up with some proof of concept that you can relate to the actual work. We can even estimate the time taken or what the test cases should be, or some sort of that things, just to get an estimate like how it could look, just a dummy kind of application. So that could be another thing you can attach to the tickets saying that okay, for reference but acceptance criteria are written, and you can't replicate or replace it with anything else. They are important and then it has to be followed. That's actually a checklist for the QA people to understand what actually they need to check in their tickets. So that is one thing.

[49:11] RESEARCHER:
Do you think this Scrum setup, the one you've been talking about produce quality software?

[49:21] PARTICIPANT 11:
No, your voice is breaking.

[49:23] RESEARCHER:
Okay, I'll repeat the question. Can you hear me now?

[49:27] PARTICIPANT 11:
Yeah, actually the video is broke but that's okay. I can still hear you. Go ahead.

[49:32] PARTICIPANT 11:
All right, I'll drop of the video so it can help a little bit the Voice. All right. Is it better now? Okay, fantastic. So, the question is do you think this Scrum setup, the one you've been talking about, produce quality software.

[49:55] PARTICIPANT 11:
I would say so, a hundred percent because at the end when the features are done, they have to go back to the customer, and they have to be checked. So, checked as in the customer actually runs it through a UAT testing, that is user acceptance. User acceptance when they perform it or even when the demo sessions go on, so technically the developers when they develop something, they go back to the customer and they have to demo it saying are they okay with this? The initial stage itself, the client can actually see how the product looks like and they actually can rectify the things that they really want to, so it becomes out of every initial stage. So, I'm actually going forward in the future as well because this continues on transparency that is maintained between the developers and the clients.

[51:03] PARTICIPANT 11:
It becomes very easy for even them to visualize things and to rectify things that are very early. It actually improves the software quality a lot because if there is anything wrong, we are not going to continue work, we have to stop then and there. So there are a lot of issues that have been introduced, we had a lot of errors that were seen before but going forward, we seen that happening and I would really give it a thumbs up but saying okay you had issues before because you didn't know what you were working on people just keep working on things before even you know, going back and checking are they okay or are they not?

[51:51] PARTICIPANT 11:
The QA might mess up some things, but there are so many people that are actually testing the thing thoroughly in terms of usability testing or integration testing and everything is done in a proper way and you come to know your errors or mistakes beforehand. I'm going forward of

course hundred percent it helps to make the software much better and I have seen that happening. We actually, and the end of 2020 coming to Canada and leaving [redacted], I literally saw a lot of automation test cases running on and actually it was a hundred percent error free. So, I would say that that was a great amazing achievement for all of us, going from eight hundred errors to zero errors. And even the development team over the period of time, they learned a lot from being agile and just because I think they were not mature when they started but they eventually at the end of the product they saw there were literally no errors. There could be some cosmetic errors also, but in the end when we saw this many drops in the number of errors and errors we are facing we actually thought this is a great idea going forward and make the system that robust and it was a big thing. It was a big achievement and when we started, we had so many errors, so that was major reason for me going into Agile because one should know you can't just take up things unnecessarily and start working on it. It's a waste of time for the client, for all the developers as well.

[53:47] PARTICIPANT 11:
Everything was planned because of the planning we grouped all the errors that were related to each other because there were many pages that were somehow related, and somebody used to fix something fix something somewhere and then it affects something somewhere else. It was very difficult for somebody, for example, on day one somebody fixes something on page one and it's somehow related to something on page two, so it actually stops it on page 2. And later is comes up again and again. We literally used to feel is this ever go to end or not. But with Agile, we actually grouped together, then we made sure this particular release, we will fix all of them. And that's how we got all of the defects that are related to each other fixed at a time, which actually did not impact the other ones or did not create another issues. We made sure that whatever is in place has to be fixed before we go to something else something new. So yeah quality wise a hundred percent. Agile does help to produce software quality.

[55:09] RESEARCHER:
Okay, the last question. We have five minutes left. Can you share with me a negative story about Agile?

[55:18 PARTICIPANT 11
A negative story about agile. Let me see. First of all, I know it takes a lot of time for developers, for a business analyst it's still ok. But when we have something to implement and still, we have to attend those meetings and we don't have much time, it becomes a little bit of problem. That's one thing from the developer’s point of view. Sometimes story pointing, people are not that good with story pointing and that's where the problem is. Most of the time we try to actually explain to the developers, this is what needs to be done in this story, but sometimes I think that is some

miscommunication or something going on by which they cannot estimate it properly and then they have a problem. And we do see sometimes, some stories that are done ahead of time which shouldn't be that's not a problem though, but sometimes it takes more than they have estimated and that's where the problem begins. I think even the developers get good at it after some period of time.
But that's another issue.

[56:42] PARTICIPANT 11:
We have to educate developers how and know what they should estimate and what way they should think about it. Give them a clear picture. Train them on estimations and stuff like that so there is no time crunch at the end. And the major impact actually it mostly impacts because of this estimation, and actually impacts the quality. The QA people.

[57:19] RESEARCHER:
Yes.

[57:20] PARTICIPANT 11:
There is always a crunch and then at the end of the day, the sprint's last day and then the developers actually give the story to the QA and the QA like we don't have much time to do this is and the estimations are done in such a way that developers and the QA both put in time and when you estimate a story it's not just the developers time, but both the time of the queue as well as they developer. So, when the developer mostly give it at the end of the day, then QA is pissed off but that kind of behavior which I agree totally because their work is underestimated or something. So, no not sure what this is but it's a regular thing that has been seen throughout my career. So, we actually tell the developers this is the story status, do you think you'll be able to finish it now? Or do you need help from any other developer for you to finish it in time? Because you have three days for this this particular story, and you have already taken two days. You have just one day left. That means the QA will not get enough time to test it. And if the QA doesn't get enough time to test it, we are not going to go forward with the story. We're not going to mark it as done. So, hat is a very common thing seen but other ways to figure things out like you can ask the developers to pitch in if they are done with their work. The only problem comes when other developers are not done with their work and there's nobody to help but most of the time somebody has bandwidth they can just share the load and get it done so that the QA gets enough thing to run all the test cases and mark the story as done.

[59:27] RESEARCHER:
Okay. Thank you very much PARTICIPANT 11. It was really lovely talking to you, and it was very interesting to hear it. I really enjoyed it. Thank you very much. Do you have any questions for me before we end the call?

[59:45] PARTICIPANT 11:
Oh sure. Actually. I'm most interested to know that you're doing a postdoc related to Agile. Like how did you get the idea of doing it, what motivated you because I love agile. I frankly love Agile and I would really go forward. I was planning to do Agile certification practitioner program or something. I mean the PMT and you might have heard of the PMICP? What motivated you to do that?

[1:00:26] RESEARCHER:
There are two motivation sources for this. The first one is academic and theoretical. Agile manifesto doesn't make reference to quality as guidelines. But Scrum is not made for software development, it's made for product development. It doesn't make reference to quality. Theoretically, we question how in practice people achieve quality. We need to understand this gap. What mechanisms are in place to achieve quality. We need to understand that. There is a gap that we want to understand. That's we do, as researchers we look for gaps, we look for misunderstandings and try to fill them and understand them. The second aspect, I used to be a business analyst like yourself and I worked as a project manager and I've seen a lot of chaos in Agile. I've seen a lot of chaos, I've seen it produce poor software but most of the time, it didn’t have to do with Agile itself, but it has to do with the performance of the team and the people. So, teams and people make Agile work. I wanted to understand whether across the globe, people do the same, is it really about the team and people performance that makes Agile work.

[1:02:33] PARTICIPANT 11:
It’s the people for sure that actually make Scrum successful of fail. You have to have that team bonding and even the skill set for a Scrum team should match everybody, all over the team. Where somebody is really experienced and somebody is not, if nobody has any knowledge about anything, it needs good management. He makes sure that everything on the board is taken care of, the responsibilities and the personalities should be matching, but people need some handholding, I would say. Train them properly from the beginning, these are the rules that need to be followed. It actually gets better as you go ahead with it. At the beginning, it's like one or two sprints are almost failure. But once you go on with it, people get used to it and work in the same direction altogether. Basically, they come to know what the goal is. You have to setup a goal for the team, this is what it is, you all have to actually work towards the goal and not go

over the place. I think they just need to be trained and once you do that and they know their responsibilities, it's all good. Agile is good.

[1:04:13] RESEARCHER:
Thank you very much PARTICIPANT 11. Do you have any other questions for me?

[1:04:19] PARTICIPANT 11:
No, it's cool. As of now, nothing.

[1:04:25] RESEARCHER:
OK thank you very much. There is just one item. We do have quality practices in the research as well. What we do is we transcribe the interviews and we send it for the participant to validate the content is okay, we have put anything wrong in there. If you don't mind, I will email it to you in a couple of weeks and if you can have a look and just tell me if everything is okay.
[1:04:56] PARTICIPANT 11:
OK, not a problem. Any time. Any kind of help you need. Sure.

[1:05:01] RESEARCHER:
Thank you very much. I wish you a good day.