[00:11] RESEARCHER:

Hi, PARTICIPANT 22. How are you?

[00:13] PARTICIPANT 22:

Can you hear me?

[00:13] RESEARCHER:

Yes.

[00:16] PARTICIPANT 22:

Been busy. Nearly didn't make it to the meeting.

[00:19] RESEARCHER:

Busy is good. It makes the time go fast.

[00:22] PARTICIPANT 22:

Yes. In these times.

[00:25] RESEARCHER:

Yeah, in this new world. Okay, I'll start the interview by introducing myself, telling you who I am and what I do, and we'll kick off with the interview. My name is [redacted]. I'm a postdoc researcher at the [redacted]. My research interests is basically software quality. I'm interested to understand how software teams produce quality software. I've done [redacted]. And now I'm interested in agile. I'd like to understand how agile teams produce quality software. And what does quality mean to agile teams? In a nutshell, that's me. Do you have any questions for me?

[01:30] PARTICIPANT 22:

Yeah, you want to write a paper about that?

[01:35] RESEARCHER:

Yes, yes. The outcome of these interviews is a paper, an academic paper. If you are interested, I can send you a copy of the paper when it's ready. If you are interested, of course.

[01:52] PARTICIPANT 22:

Yeah, that would be nice.

[01:54] RESEARCHER:

Yeah, it will be ready around August, you will hear from me around August. So, the purpose of the interviews is to gather people's perspective because people perspectives is the practice. And by understanding the practice, we bridge theory and the practice, and we address the gaps in theory. That's why we do interviews basically.

[02:28] PARTICIPANT 22

Okay, about me, I have like, I think eleven years' experience. And I started as a tester. I will just sum up my experience. So, it was just testing content from, let's say, stupid wallpaper to games on, at that time, the iPhone 3G and 3GS. So, all times. And it was basically manual testing. The tool for logging the issues was actually quite simple. It was like a feed we have now in multiple apps or Facebook. So manual testing for mobile devices, that was the first phase. Then again, mobile testing, mobile devices. But it was like four years with [redacted] and other apps like [redacted] after that, again, four years for e-commerce. But like I say, it was the [redacted] I don't know if you heard about them?

[03:46] RESEARCHER:

Mm mm.

[03:47] PARTICIPANT 22:

So, the stores. So, e-commerce meaning that the site itself, but mostly it was what's behind it. Some order management systems and again, manual testing. A lot of business knowledge, a lot of business close customized for each country and so on and so on. So, I forgot to mention that in the first part to it, at the beginning I was a manual tester. I had a small team, after a while, after that with eBay, again, let's say a team of three or four and coordinated another one in Germany and in this one for [redacted] again, a team of two. And we evolved a lot in communication in the agile context with the clients and another team. And we connected things from our office and so on and so on. And in the last two years, two years and a half, something like that, I was in the first six months, like a tester and then somehow most mentoring or let's say having another team. And I didn't test it too much, but I switched, like in the last one and a half years for the position of Scrum master.

[05:16] PARTICIPANT 22:

So, I did it in the past also. But not officially. So, like, many PO for an internship, for a project for a while, as Scrum master. But not officially just being the stand-in and so on and so on. And again, the last two and a half years, again, back to mobile. It's like mobile apps for health insurance from Germany. This is like the context. So, everybody is doing agile, but nobody is doing it as well. Everybody is doing training, but when it comes to money and the deadlines and so on and so on, yeah, everybody's doing agile. Meaning that now I had like six, seven teams of four to five people. And from I don't know, from five or six daily's, on a daily basis, they got to a point that, hey, you have a point. It's no use to use Agile because the project has like a fixed deadline. It was like ten days ago. And the guy from there said, hey, why do we do so many meetings, so many sprints and we set deadlines. We know that we need to finish at a certain date. That's the deadline. Let's do business. I'm sick of all those meetings.

[06:57] PARTICIPANT 22

So, yeah, from so many meetings, they actually switched to many approaches and we switched and frankly, just cutting down meetings like, planning's or groomings or pre planning's, even great roles were cut off because on the planning part we had a lot of dependencies and technical stuff that you can't really predict. So, imagine that developers with twenty-five years of experience couldn't estimate properly the topic. And it wasn't like one, two, three, like five guys like that. So, for talking too much.

[07:49] RESEARCHER:

No, it's okay, talking too much is good for this. Just to clarify, when you say Agile is means your Scrum experience, right?

[07:53] PARTICIPANT 22:

Yes. All my experience is in Scrum. I do not know Agile other way. In some context, the Agile is not the...

[07:57] RESEARCHER:

So, what happened exactly. Because agile is meant to make people more efficient.

[08:04] PARTICIPANT 22:

I didn't encounter that until now such kinds of projects when you have fixed date. If you made it, you made it. If not, if not, it doesn't matter. So, the client say it doesn't matter how you do it. You actually choose the methodology, how to proceed with it. I just want it to be delivered. So, it wasn't efficient due to the technical aspects. That was the feedback from all the teams that you cannot even estimate. It was let's say, it was a component based on C++ and they couldn't even estimate what they have to do in the next two sprints. Because I was always saying, hey, guys, we have a lot of so-called POs or Dev Leads or so on.

[09:01] PARTICIPANT 22:

Please give me a road map for three sprints. I cannot give you even for two sprints or even this sprint. This one sprint. We know the big lines, but if we dig deeper and deeper onto the topic, then we will encounter more technical difficulties and we have to invest more time. And the cause of this was, I can say the root cause it's in all projects, meaning that if you don't have the requirements or the text specifications well written or defined, you will get in that corner, because the specifications were in German. Let me explain. We had four people, BAs. They were not technical. They were something like copy pasting stuff. And they wrote the stories. But when the technical specialists was diving in, it was like, hey, this is working according to that mechanism. That mechanism is sending me to another document, to another and another. And those were incomplete.

[10:22] PARTICIPANT 22:

And so actually, the root cause was documentation and technical specification. So that's why they were digging deeper and deeper and saying, hey, it's more complex, more complex, more complex. And I'm depending on the other component, on another environment, on another configuration and so on and so on and so on. So, if somebody asked me, I saw your questions that, hey, which is the trick, is agile efficient, isn't it? Yeah, it's efficient when you have the tools to do it and the tools from my side, one big point is the documentation, because you can choose any methodology, any way to do it. If you don't have the documentation and the specifications, you cannot deliver good products, a viable product. And it's not like you have to have them all because it's unreasonable to have every single thing specified. It's normal to dig and to encounter problems. But you really need to have the big lines, the flows, the business flows, and just after that, to put them together.

[11:49] RESEARCHER:

That's a great. Before we go further, let’s define quality, because everything we say here is in the context of quality?

[12:01] PARTICIPANT 22:

Oh! There are hundreds of books and standards written on this topic. But to be honest with you in reality, we don’t know it until we see it and agree this is the quality we want. To narrow it down and for the sake of this conversation, I think software quality is internal and external without ignoring the importance of the process that takes place to produce the software. External means no bugs and adding value to the business. Internal is the quality of the code and the design.

[12:29] RESEARCHER:

Yeah. Fantastic. That's a good. That brings me to ask you, in your personal opinion, what do you think of Agile?

[12:31] PARTICIPANT 22:

Oh Agile. Like I started before, everybody's saying we are doing agile. Everyone wants to transform the company and the processes, and agile is like something you buy at the store. And it's fashionable. This decade we are doing Agile because it sells better. Everybody does. It looks nice on paper and so on. But if you compare it with other methodologies, yes, indeed, you can actually declare that it has some advantages because you can see the problems earlier. And I think the part with having time-boxed intervals to work like two weeks, three or four weeks. And also, there's time-boxed meetings. It's a good part. But on the other hand, you can set the time intervals to work and to deliver in your team or project without saying that you are doing agile. So, you can take some goods parts from Agile and just say, hey, we are self-organizing, but we are not doing agile.

[13:33] PARTICIPANT 22:

We don't have the so-called meetings. If we are doing a planning or are not doing like two or three hours or daily’s, it's not fifteen minutes. It's what I want it to be so, yeah, it's like, okay, if somebody will ask me, hey, you are working in Agile. Yes, I'm working, but in the end, I managed in the last two months to set some weekly goals. And now we are having, like evening daily wrap ups like a status meeting. But that's not Agile. Yeah, we just customized in a way, Agile, Scrum, just to fit to our project. And it's like customization over eighty percent. And it's not agile. So agile or scrum, however you want to call it, has some good parts because it helps you to deliver maybe better than other methodologies and you can find the problems earlier. But if the management or if the architects or some persons that are actually taking the decisions, are technical or managerial decisions are not how they should be taking them properly, selecting proper technologies and so on.

[15:09] PARTICIPANT 22:

Agile or not, you can fail. So, it's not like agile is the antidote for everything, for bad management or for stuff like that. So, if you have at some point, if you have a mature team, I don't think it's actually matters. It matters that you use Agile or not. Frankly, for two of my teams now, I just do some daily’s and two times a week checking with them because they're mature and they know what they are supposed to do. The only problems are on technical level. And I'm just checking with them from time to time to see, OK, what is your problem today, do you need my help from bureaucratic stuff or even technical or any issue from the role of Scrum master. And from the technical perspective, the Dev Lead is actually taking care of. But in another project, again, mobile, but it's like related projects. In that part I had more power, but I could protect the team more. Meaning that the management and the POs were actually always, and this is happening on our projects, are always pushing the team. Take this, deliver this. Hey, you have to deliver. I don't know at this date, this feature, but nobody is asking, hey, do you have capacity, or they are asking, do you think, okay, I can think, but you didn't give me the whole design. The whole specification. I will say, we'll try.

[17:01] PARTICIPANT 22:

And it's like a mentality problem, even from the team and even from the manager's point of view, because the managers are using their position to impose a point of view and the teams are afraid actually. The teams are afraid, or the people are afraid to say no. I'm always saying to them, hey, say no to your PO or say no to management and raise the risks and say, hey, I don't want to gamble on this part because we can deliver, but it's about quality. And after that, the release manager or somebody will say, hey, why so many bugs? How come? Why? You asked me to develop that and that and we deployed in production without even doing a full regression, like smoke and seeing the happy flows. When I'm seeing happy flows, I'm like, oh, come on, it's not ready for production if it's only happy flows. So, ask your questions because I got somehow around your main topic.

[18:08] RESEARCHER:

No, it's okay. You're already answering a lot of my questions. So, I'd like to follow up on something you stated. You stated that Agile helps you to deliver better. How?

[18:22] PARTICIPANT 22:

So, I can give an example. Somehow my testing point of view is always coming back to me. And as a tester in the past, and even now, because I'm collaborating on other projects as a tester. I'm stumbling on the parts before releasing. Everybody's okay, we want to release but when do we test. So, we had an approach like let's release every three or four sprints. How do we want to proceed? We want more time for testing. When do we test? How do we test and so on? So, we decided to do like two sprints or over two weeks, something like that. And the first sprint everybody is developing, we have the specification. Let's say it's being [inaudible] something like that. You will have problems, but yeah.

[19:24] PARTICIPANT 22:

So, we have a first sprint to develop, test the features and so on and close the sprint. Then we will go to the second sprint. But in the third, only half of it or let's say half of it, it's like stabilization or the whole spring, you can see it as a stabilization sprint, meaning that you don't actually develop big features. In that sprint it's only for bug fixing. And at some point, half of the sprint, you can say something like, hey, we need the cold freeze. We are doing a cold freeze. So, we impose a cold freeze on a branch. And if some developers have time, they can work ahead and develop on another branch. But that branch is dedicated to have a cold freeze on it.

[20:23] PARTICIPANT 22:

And most likely, the testers will start their regression. So, after that regression, the developers and also the managers, we get a report from the testing site. We have a list of issues, the critical, the high ones and so on and so on. So that should be filtered and every fix that's done after that, will be decided if we need another, let's say, not smoke, but a regression on that area. What's the impact if we fix it? If not, we can live with it or not. And to decide if we are going to release in production or to the client. Usually it goes to the client first. And after that, again with him, if it goes in production or not. But this kind of approach which is encountered in agile helped in the last year, those teams to deliver better because, like I said, it's structure. You have like one, two sprints to develop the features according to the plan you made.

[21:42] PARTICIPANT 22:

And it has to be made and you have to be in possession of the requirements. And the last sprint for stabilization. And, of course, reserved for a round of full regression of testing. Even if in the first and the second sprint, you tested each story, each feature, but you need like a zero point and say, hey, this is code freeze. Let's test it and then deliver to the client. And after that, in production.

[22:16] RESEARCHER:

Thank you. I'd like to follow up on something else you mentioned. You mentioned that Agile suffers from a mentality problem. What does that mean?

[22:27] PARTICIPANT 22:

Can I sorry...can I just take a call. It will be very short.

[22:34] PARTICIPANT 22:

A mentality problem the development point of view once. [redacted]

[23:15] PARTICIPANT 22:

[redacted]

[23:17] RESEARCHER:

It's okay.

[23:18] PARTICIPANT 22:

From the mentality point of view, he will people being really like oh my God, again, a lot of meetings and especially the guys that are having a lot of experience. Because they are saying, hey, I don't need to, just tell me what I should do. Have a short meeting, not ten. And let me work and when I will have questions, I will come back to you. But just let me do my work and don't annoy me with so many meetings because it's hard for me to focus on the development topic, which I'm really focused on. And then you put a meeting and then I'm working for another hour. And again, other meeting. So, I had a lot of requests that okay, you want meetings, just put them in one day or two. And let me work, so leave me alone. Something like that. This is from the let's say, with my experience, from the new joiners and so on. They actually think it's a lot of theory as a new joiner.

[24:41] PARTICIPANT 22:

And I don't know, you have to say, involved a bit but then, I'm losing my words, one second. So, for the new joiners, I can say it's interesting for them because it's like new for them, but it's hard a bit for them to estimate or adapt to the framework or stuff like that. And coming back to your question again, I want to mention the managers. Maybe it sounds like frustration, but even I had an example. A release manager, which has actually helped us in the past, and he was agreeing with the testing team that, hey, we need time for testing. So, it's not acceptable like two days before finishing the sprint or something like that. You are coming back and saying, hey, the client wants that. And the release manager is saying, yeah, I know it's not the best context, but what can we do? We have to introduce that, and we will assume the risks and take the risks. And it is what it is. And that's it.

[26:20] PARTICIPANT 22:

So, if you go to the managers, upper management and say, hey, we cannot really do this because we are risking the whole release and it will introduce a lot of bugs and so on, they will say the client knows. He will assume the risk. And usually this, even if it's a big thing to introduce or a small thing, every time something comes up and the release manager can say no, or the PO can say no to the upper management. And it's like going down. As a team, we would not say a definite no, like something we will try. And things are losing from their quality. The quality is quite reduced. And then in the end, you will end up in doing a lot of hot fixes, patches, and stuff like that. But on the other hand, each project, I think, has situations like this. It depends how often and how do you handle them.

[27:33] RESEARCHER:

I agree. Let me see, what is the next question. The next question, what do you do to assure software quality in agile setups, in your current setup or in any of your teams or your experiences.

[27:53] PARTICIPANT 22:

On the current project, it has a lot of specifications, meaning technical specifications from the organization who needs the product. So just today, I was actually discussing about user manual, about a team who had a request and actually forgot about it, or just, I don't know, it was missed somehow. At the end, the point was bringing the initial discussions of the project like the quality code, meaning sonar and so on and so on. So, except using tools to improve your quality of the code, except using maybe the four I principle to improve the process of reviewing the code. I mean, like a PR has to be reviewed by two people. Something like that. This is a process to put it in place to assure the quality of the code and of the technical solution. This is what actually we do from this point of view of the code.

[29:10] PARTICIPANT 22:

Then, of course, you have to have a good architecture and to follow the best practices and to select the proper technologies, because we even had issues with that. Somebody selected the technology and when it was asked, please bring me a solution because I have this and this problem, but I'm not an expert. So, how did you choose the technologies, fix it, you proposed it. This is about the quality of the code and the whole architecture and so on. A lot of problems are coming from this and a lot of bugs are actually introduced due to this and also a lack of knowledge due to the business flow. A lot of the developers are just developing but when I'm asking them, hey, what's behind this, what's the logic and what's the business flow that I should see in the front end or so on. I don't know. I just did what's written in the story so that.

[30:30] PARTICIPANT 22:

This is from the architecture, maybe even the PO sometimes and the code quality, but just cut off the PO. So, from the other part, from the specification point, the PO, the business analysts, and others, which are writing the stories, use cases, how do you want to call them. From there, we can have a lot of quality issues and not lists the testing part. So, the testing part, it's quite important. And it depends how do you approach it? Meaning that manual, automated, so gain time and value and so on, performance or security, you name it. So usually what happens, in the last five years, I don't think the people are, let's say, forgetting about testing. But in the past, it was quite a lot. Just we don't need testers, ah, it's okay. The developer will test it a bit and that's it. So, testing and a part that's nobody likes. The processes. Meaning that in Agile, usually the people are complaining that you have a lot of processes. Processes from mitigating some tickets. Processes from the tickets, or the ticket type that you use in your Jira. For example, we have epics. We have stories. In one project, we have over epics services and now we have something like another ticket type. And from that ticket type, you can have five or four stories being connected to that.

[32:49] PARTICIPANT 22:

So, if I tell a developer, hey, you need to pass that in that state to be able to pass the other ticket and so on and so on, like Jira workflows or internal processes. You are working with other companies and you are getting bugs or other tickets from them. So, another process, how to handle that and how to move your way up until you are able to finish your work or get to review or stuff like that. So, this is like processes inside the project. And when people are complaining about this, oh, come on, another process, I'm losing more time. So many, many things. Now I have to learn this. It's not development. It's just bureaucracy. Just leave me. Something like that. And what can I say?

[34:00] RESEARCHER:

So, let me ask you a question. You at what stage the testers are involved in the process.

[34:08] PARTICIPANT 22:

In the process of release or...?

[34:11] RESEARCHER:

In the process of Scrum in your experience.

[34:15] PARTICIPANT 22:

Actually, you should be involved from... I'm actually reading some stuff in the last months and theory is actually contradicting what's happening in reality. You don't have zero sprinting in Scrum. But actually, you have because in the first sprint you cannot have something to do for the testers. It's a real thing to test or something. So, the testers from the start should get a few stories, get the big features or topics, just study what's planned and then start creating test cases. So, this isn't sprint zero, but if you are thinking a bit ahead on sprint one the maybe they have some knowledge from the wireframes where in the wireframes you have the workflows, the business flows on high level that are scratched on. So as a user, I want to add some documents, delete, rename, stuff like that.

[35:32] PARTICIPANT 22:

So even from sprint zero, but most likely from Sprint one, they all have to do some sort of test preparation, meaning that they will start to write some test cases. They will not be perfect on a manual level. Just understand the flow and think a lot about the scenarios that they can do, that and that and that and so on. So that's the first phase for the testers. And one second, sorry.

[36:37] PARTICIPANT 22:

Sorry, I lost my idea.

[36:42] RESEARCHER:

Yeah, we were talking about testing in Scrum. And at what stage the tester is...

[36:47] PARTICIPANT 22:

So, they are starting with preparation. The test preparation, meaning that they are studying the work flows as much as they can do it. So, yeah, creating test cases. And usually when the testers are doing this, they are coming back with a lot of questions to the PO or to the business analysts and saying, hey, what does it happen in this scenario? Oh, I forgot about this detail. I have to get back to you with small clarifications from the client or from the documentation or I would try to clarify with somebody from the client or I will double check or something like that. So, naturally there is a stream between the testers and the PO or even the client directly. The client, because the client doesn't really know what he wants until you show him these are the possibilities. We think that this brings more value because of that, of that, and that. Or do you want us to do this or that or stuff like that? Actually, the testers are always sending feedback to the business stream and asking them to clarify the specifications. So, I saw a question on your PDF asking about the flow of a story.

[38:36] RESEARCHER:

Yes.

[38:37] PARTICIPANT 22:

So actually, we did some Jira flow, the long story short is something like this. You create the story. You are the PO. You create it and it's like, I made this as scrum master, a qualification board. You will put it there and you'll still need more data from the client, from the specifications. You'll just have a sketch. You will define it. And when you consider as a PO that you can move it to ready for dev, sorry, not ready for dev. You are looking on it, you are qualifying it because you think it's ready for dev. And after that, me as a scrum master, I'll say, hey, let's do the, usually we are doing in the past projects some kind of preplanning or grooming. So just send me, you as a PO, the links with the stories. I would send them to the team, meaning developers and testers. They will take a look. They will note down the questions, what's not clear for them.

[39:52] PARTICIPANT 22:

Maybe they will send it's on the channel or on e-mail or something like that. You can double check. And in the next meeting, which is that preplanning or grooming or how do you want to call it, we'll have a discussion. And it's like clarifying all the aspects which are not created regarding that story or use case. And in way, the testers and developers will find out more about the topic, the feature, and most likely they will come up with more questions. So as a team, we will do another round, like the planning itself. And usually during it, you as a PO, will clarify more about the questions that we have. So, after the planning is done, you can say, hey, this story is really ready for dev because you've got the feedback from the developers that, hey, for us, it's clear we know what to do.

[41:08] PARTICIPANT 22:

But other than that, us as a team, the developers, and the testers, we estimated the complexity of the story. And we can say that it's valuing something like five story points. And we have some user stories in the entire sprints in total thirty story points. And we are really confident that based on our experience and the numbers of velocity, and availability and so on, we can deliver those stories or features this sprint. So, these estimations are based on complexity. And every developer and tester should actually say an estimation from their point. The estimation is calculating the complexity from the point of development and also testing. And during the sprint, the testers are actually doing test preparation, meaning creating user stories and so on, but also creating or searching or requesting or manipulating the test data.

[42:27] PARTICIPANT 22:

Usually the test data is a big pain. And the testers should think about from the start to the spring. And for each story and actually each story should have some notes regarding the test data or happy flow at least know how to test it. Because most of the times, there are some POs or not maybe POs, BAs because they are not considering testing during development. And they are not saying this is the use case, just to start the happy flow.

[43:09] RESEARCHER:

Do you think your Scrum implementation helps producing quality software?

[43:15] PARTICIPANT 22:

Quality software compared to what? And what do you mean by quality? Meaning there's not so many bugs.

[43:29] RESEARCHER:

Its quality is quite, it's quite a subjective concept like we talked about earlier. So, every organization may have a different definition of quality. But let's assume for the purpose of this question that we mean by quality is free of bugs.

[43:53] PARTICIPANT 22:

You will not have software free of bugs. And it cannot be bullet proof. This is what I think, it cannot really be bullet proof. But you can affirm that it helps you to deliver more quality, but I will just repeat myself, meaning that, like I said, it helps you detect the problems earlier.

[44:43] RESEARCHER:

Okay.

[44:44] PARTICIPANT 22:

So, I think that's the core of it because every software or every methodology, in the end you will have bugs still existing, but somehow the value brings from the fact that you can detect them more often and more earlier. And actually, I don't know if you can really call it a process, but I think the testing part can be customized more efficiently in Agile. The whole thing with raising risks, meaning that a release process or a Go after the sprint has ended or a feature is completed and so on and so on. For me, agile is more transparent than the others. So, you can somehow see at any point where are we with the status of the feature, also, Sprint and also release. Usually, we are using, I have suggested to have a status page or something, what's the status of the sprint?

[46:20] PARTICIPANT 22:

Where are we with the progress, if it's Green, Red, or Yellow, and another section with, what's the status of the release. Because you can be Green with the sprint, but you realize that in the next sprint, you will have some dependencies related to another team or component or something like that, and you will not have what you need. And it can impact you and so on. If somebody will ask me why waterfall is worse than that Agile, maybe that would be the answer because you can detect stuff earlier. And for me personally, it's more transparent.

[47:11] PARTICIPANT 22:

And also, one thing, if the PO is experienced, maybe he can really do it like in theory, he can really do tradeoffs. Meaning, you want me to add something to the sprint but please take something else out or use the whole concept of priority, of dependency. Double check if we have backends because multiple times we had started to develop features on the frontend side, but we didn't have a backend.

[48:00] RESEARCHER:

Yeah. Fantastic. Thank you very much. Can you share with me a positive story about Agile and its ability to produce software quality?

[48:12] PARTICIPANT 22:

A positive story?

[48:13] RESEARCHER:

Yeah.

[48:27] PARTICIPANT 22:

I think the teams are really flexible and we had situations when the tester was actually developing, or the developer was actually testing. Sometimes they are really cross-functional like in theory, not multiple times. But can you detail to me a story? Meaning that I don't know because we had an example...

[48:59] RESEARCHER:

An example from practice, an example from your experience. That's what we mean by a story.

[49:06] PARTICIPANT 22:

Yeah, but something like, OK, we used something before and after that we switched to Agile or to a sprint approach. And it worked better.

[49:16] RESEARCHER:

Yes. Or something like that. Yeah.

[49:20] PARTICIPANT 22:

Something like that...

[49:28] PARTICIPANT 22:

Usually it's happened in the past projects the most...The idea is that the impact of Agile on delivery, it was good because you can plan better. I don't think I have a specific story, a detailed story.

[49:58] RESEARCHER:

Yeah, that's fine.

[50:00] PARTICIPANT 22:

It's like this. Because we are going to release, why, that's not ready, why it's not done, why we are waiting about that one and so on. So, when we switched in the past to the approach with the sprints, we can plan better. Like I said before, I will know better that in two sprints I will need that and that and that. And I can be proactive. And actually, this page results in the projects that are dependent on other teams, on other services and so on and so on and it helped planning. This is on the high level. On the low level, meaning inside the team, the people were better and better in estimating their work. So, if you go to a guy and say, hey, please finish this, I need the thing in three months or one month and a half. And we had cases like that when the guy or the other team committed. But she didn't manage to do this. And after he divided the work by using sprints and we were in constant communication with him. It's helped a lot to be more transparent.

[51:28] PARTICIPANT 22:

And this is actually even valid for a team, also for a person, because it will in the end, help him to understand and learn more to estimate his work and give realistic deadlines to management, another team, another project. Hey, I can deliver this and that and that. So, it will help you learn to estimate your work. And regarding the testing, that was a real improvement after we switched to sprints. Due to the part that we somehow know that we need to prepare the testing, we can somehow, based on the stories and the timeline of the sprints, can really say, hey, maybe at that period we can really do a smoke or try a few stories, features or stories, even if they are not finished. And I didn't mention one important point regarding the integration. I mean, integration of a lot of features of your own system with other systems that can reveal got a lot of issues. In the whole idea of the sprint, I think you can focus better on the testing part if you have also a code freeze, which ideally, we will have it in this approach of sprints.

[53:14] PARTICIPANT 22:

And in the last four years, I think only in the last year and a half, I have seen again the part with the cold trees because everybody is like developing, developing, testing, testing, but not having like a zero point called a cold freeze. Stop developing. Let me test them. After that, we will discuss.

[53:41] RESEARCHER:

Fantastic. Thank you. I've run out of questions. Do you have any questions for me or final thoughts before we conclude?

[53:52] PARTICIPANT 22:

No, I'm available for my questions if you want. If you have today more questions, I can help.

[54:04] RESEARCHER:

No, I don't have more questions. Thank you. You answered all my questions. There is just a procedural thing we do for validation. We send the transcript of the interview for the participant to validate and to confirm. So, you will receive from me the transcript of this interview. And if you can have a look and make sure that everything is okay and let me know by email, please.

[54:37] PARTICIPANT 22:

Sure.

[54:38] RESEARCHER:

OK. Thank you very much. If you don't have further questions for me, we can say goodbye to each other.

[54:48] RESEARCHER:

Thank you. Bye.

[54:50] PARTICIPANT 22:

Thank you for your time. Bye.