[02:16] RESEARCHER:

Hi, PARTICIPANT 20, how are you?

[02:24] RESEARCHER:

Can you hear me?

[02:29] PARTICIPANT 20

Hi, Researcher.

[02:29] RESEARCHER:

Hi, PARTICIPANT 20, how are you?

[02:31] PARTICIPANT 20

I'm really good. How about you, Researcher?

[02:33] RESEARCHER:

I'm doing very well, thank you. How is [redacted]?

[02:38] PARTICIPANT 20

Oh, my God. [redacted] is doing great in terms of getting back to normal. It's still lockdown. So certainly, a couple of more days or weeks to easily getting back to unlock days.

[02:54] RESEARCHER:

It's winter now, isn't it?

[02:56] PARTICIPANT 20

It's starting to go into winter, yes. So, it's around ten in the night but it's not to be.

[03:05] RESEARCHER:

Okay, PARTICIPANT 20. I'll start by introducing myself and we'll go from there. And my name is [redacted] I'm from the [redacted]. I do my research in software quality. I'm interested in understanding how software teams manage to produce quality. Or how do they achieve quality in their software development processes? I do have an interest in Agile from a research perspective. So, I'm doing my research in Agile to see how Agile software teams achieve quality in their processes and in their final product. That's in a nutshell, what I do. Do you have any questions for me before we start?

[04:06] PARTICIPANT 20:

No. It's so good to listen to you at the moment. The whole of my life was doing the practical part. Just to sum up, the work you're doing is really would be of great quality. So, yeah, not nothing at the moment. I am just curious to know. I just read your PDF which you attached about the kind of the outline of this discussion would be. So just to start with, my question is, have you ever worked in different other models apart from Agile. It would be easy to distinguish the kind of nature of this.

[04:53] RESEARCHER:

I worked in Waterfall for more than fifteen years until Agile came in 2008, I think. Well, Agile came in much earlier. But we start seeing it being adopted in early 2000. I worked in in Waterfall and we delivered big and large projects, very complex projects in Waterfall. And I also worked in Agile. Agile teams I worked for, it wasn't much of a good experience. They struggled in the implementation and the team itself had problems before Agile, and those problems were inherited in the way they implemented Agile. And it didn't work much. After that, I moved out to work for a consulting firm as an I.T. project manager, and I helped a lot of teams implementing Agile. So, I'm very familiar with how it works, etc. So, yeah, I did work in the two worlds very much.

[06:12] PARTICIPANT 20:

So, Researcher, it seems like you have either more experience in terms of delivering products than me. So just as an outline, what do you expect me to, you know. I have worked in Waterfall model. I have worked for V Model as well. Luckily, in the last couple of years since it's all Agile. So, all the clients you work for is looking to adopt these Agile methodologies. So, I got a chance to work in the Agile based model as well.

[06:47] RESEARCHER:

Basically, what I expect from you just to answer my questions, that's all. Nothing else. Answer my questions from your experience perspective. So always bring your experience in answering my question. How about we start by an introduction? Can you introduce yourself and tell us a little bit about your experience?

[07:11] PARTICIPANT 20:

Sure, Researcher. So, my name is [redacted]. I have experience of eight plus years working as a telecom OSS and BSM SME, which is consisting of around five to six years of being a tester. And then going into a part of BA or you can say Scrum master. Most of my work was working for [redacted], as I guess I may have supplied you my resume as well.

[07:42] RESEARCHER:

Yes.

[07:43] PARTICIPANT 20:

Yes. So I do have experience in all kind of testing activities from understanding the requirements, framing the scenarios, reviewing the scenarios, planning that test sprints and making sure the testing execution is done, ensuring bugs are being identified and tracked by

our respective tools, ensuring that we have defect meetings every day or every other day so that we have constant flow on the defects, which are critical. Reporting back to stakeholders on the progress of each day and weekly as well. I have worked with a very big project client, which is Vodafone. So, the team size was really quite high. It's more than hundred in terms of testing activities only. So, it was a big, big project where multiple deliveries were hanging around. So, I got a chance to work right from two sprints to three sprints comprising of one to two weeks, to three to four weeks as well.

[09:02] PARTICIPANT 20:

I was also actively involved in test closing activities where you frame all the initial find outs and then approving the scope, your testing outcomes, your bug reporting, and exclusive reporting on exclusions from release. I was also involved in deployment of the product in the go-live environment, making sure that we have a sanity checks done before post, that we identify any live defects. Understanding the root cause of it, making sure that we have an arbitrary approach test set included. Arbitrary is risk-based testing. So, the risk you will develop a different set of important, highly important test scenarios that should be run to ensure that regression is not impacted. So, this is was the whole 360 view of what my work nature was.

[09:13] RESEARCHER:

Let’s start with a definition. We will be talking about quality; we should agree on what we mean. How do you define quality in agile software development?

[09:24] PARTICIPANT 20:

For me, Agile is brings two perspective to quality, the process quality and the structural quality which is the code and the design. Before agile we were mainly concerned by findings bugs. The agile mindset is having a robust process for continuous delivery of valuable software. Agile calls for delivering a working software frequently. This doesn’t happen without a clean code and a sustainable design. Continuous delivery needs a quality process. Agile believes that a process becomes efficient by continuously reflecting on it, tune it and adjust it.

[10:13] RESEARCHER:

Okay, great. Thanks. Thanks for that. Let's start with the first question on Agile. What do you think of Agile? What is your opinion of Agile?

[10:24] PARTICIPANT 20:

For me, Agile is more of a less deliverable model in which you basically work upon the work it takes, ensuring that the amount of deliverable we exchange are less. The number of sprints are low to reduce the expectation of the end product and the delivered product.

Does that make any sense to you?

[10:59] RESEARCHER:

No, it does, yes. So, do you find it good at delivering products?

[11:06] PARTICIPANT 20:

So, Researcher, in my experience, definitely the couple of very major aspects of Agile is the sprint planning and the daily Scrum. So, should I just give you how my whole project used to work?

[11:23] RESEARCHER

Yeah, yeah, you can go ahead.

[11:26] PARTICIPANT 20

It comes with terms, and terminologies. So, as I already told you, Researcher, I work for a very giant domain, telecom domain in UK, which was [redacted], and I have worked with [redacted] as a service provider. So, I got a chance to learn the whole technology process. So, then the Agile model was adopted by [redacted], we have couple of deliverables which were prepared. And the first one was the product backlog. The product backlog is more often user stories or requirements presented by the stakeholder or product owner. So, in these user stories, you have to identify, they should be isolated and automated in nature. The very first step. One user story should only perform one function. For example, [redacted] has introduced a new bundle. So as a customer, I would be able to add a bundle by paying this much amount. And this much would be reflected in my profile. So, it should be very automatic. You have to ensure that all the user stories which are being delivered by the product owner and then we use to discuss those user stories with the stakeholder to ensure that the understanding is not different.

[12:55] PARTICIPANT 20:

So, this is a little different way of treating Agile. Because in case, there has been many occurrences where the requirements, which are framed and the requirement, which are understood are a little different. And at the end, it's always a waste of time and interpretation of work and loads of a lot of energy and human resources. So, my first initial step was to make sure that the product backlog or user stories are very correctly framed and understood. The second step we used to follow was the spring planning. So, the sprint planning was based on the high priority user stories which had been shared by the product owner. So, let's say, Vodafone is introducing five bundles for data where the customers can use unlimited amounts of data for a specific amount of time. So, we have to ensure that all the...so, this was the sprint meeting, which was happening between the dev team and the testing and the Scrum master in that sense.

[14:03] PARTICIPANT 20:

So, they used to play a very popular game, if know Poker. So that Poker is used to understand that first, the estimations are done correctly. Before starting a sprint, we should always have the efforts estimated so that we lose out on time or we over anticipate. This dev team, as well as the test team used to understand the development effort as well as the testing efforts and used to calculate their efforts, which was taken as the amount of days. Like we used to calculate in hours. Okay?

[14:48] RESEARCHER:

Yes, just keep going.

[14:50] PARTICIPANT 20:

So, based upon this sprint planning, definitely there would be deliverables shared with the sprint backlog. The sprint backlog contains the sprint planning. How many user stories, which are important between stakeholders and the technical team, are agreed and which will be delivered. So, based upon this, as a test team and a dev team, you have to be in sync. I really feel keeping a note on the progress, and always ensuring there is a common forum where you openly discuss the requirements. As the dev team has to initially start building a model so they always have to understand the requirements with the BA. The BA is the business analyst who has framed the requirement in a technical way. It will always be easy to somehow keep a workshop. That workshop, based on the requirements, as a test team, we start framing the test scenarios or high-level scenarios. On the other hand, the development team has started developing the product.

[16:08] PARTICIPANT 20:

So, it's the best time for the QA or a testing team to have a workshop, a high-level workshop, so all the important scenarios are being discussed with stakeholders. This is a very useful meeting, workshop. We used to say that it’s a change workshop where we used to discuss anything else you need us to test because a part of testing if you have planned our tests accordingly, it will be very easy to execute those tests in that manner. So, that meeting was very fruitful for us where we would take a final agreement in terms of scoping, per test agreement as well as the regression pack in case that has be to be executed. So once this sprint backlog meeting has been completed all the priorities are being aligned with everyone, we start the sprint that generally comprises of two modes. When we are saying a release, a release means a big chunk of sprints that may contain up to three sprints. A normal sprint, which needs a big delivery, that usually comprised of one to two weeks.

[17:32] PARTICIPANT 20:

Out of which development generally takes about four to five days. We used to discuss phased delivery instead of delivering the entire product in one go. If any functionality, which is isolated, and can be tested. You may have understood mob data testing as well as isolated testing. Instead of unit testing from dev team, we also participate in the testing so that initial understanding of the product outcome is there with the test team. If your team or a part of the test, you understand what to test, how to test and what negative test you can definitely put it to ensure that the quality is not impacted. That is the most important thing. With each day, we used to track the development progress as well as the delivery date of that build. The build is the development package. Generally, it depends on project to project if the client or if the stakeholders are quite approachable, we can always build a common environment where development and testing can do sanity.

[19:04] PARTICIPANT 20:

So, let's say if we have a deployment of initial, let's say, a product is divided into three parts. Build A, B, and C. A and B comprises of some unit. A chuck of code. And C comprises of a complete delivery. What we can do with the help of Build A and B, we can do a quick sanity on the development environment, and just mark down or take a screenshot or capture the results for the sanity purpose when it will be delivered to a site. In case we have any issues flagged in times when delivered to a siting, we can quickly check the results been identified on the dev environment or a common environment, as well as the site environment. This helps us to reduce the time to track the bug and find the differences in patching and all that. So, definitely keeping track of which builds are deployed on the same environment is always fruitful.

[20:10] PARTICIPANT 20:

So, let's say, we have Build C deployed, which has complete functionality to test. My first initially analysis is to run those priority scenarios as well as a negative test in the very first days of the test. You have to ensure that nothing is breaking up and all the major functionalities, let's say, five products are being delivered, so adding those five products into a customer account. Sanity customer accounts is always a good test to do. You have to ensure that your planning of scenarios and execution should be spot on. So that you can identify all the major bugs and with those major bugs you can definitely track them back to the Scrum master or the development team ensuring they are being fixed or they are being taken care rightly and timely manner. This whole scenario keeps running for around one to two weeks where you test daily, and you share a report of the critical defects you have seen. And in case you have seen a mismatch where requirement is different and delivery of all is different, and the outcome is different. You have to report that same thing instantly, because I have seen and with my experience, if that has not been handled in the first week, the requirement changes or the code changes there's a repetition of tests to perform. There's a delay of delivery with the build and definitely you have to compromise either with the number of tests or your quality of test.

[22:05] PARTICIPANT 20:

So, in this case we used to follow automation test scripts where we used to at least get some of the test, which are likely to happen, get them automated using Selenium, or whichever tool is being used in your organization. Then if everything is cut and clear and you have tested everything clear. definitely you have to write your summary or closing report, which all things can be taken care of in the next sprint. Or we can reduce or share the load with any other team to help get more productivity. At the end of it, we used to have sprint Review meeting, which is more like a show and tell where we used to have a team member displaying all the test scenarios with the evidences, as well as the showcase model of the work which has been done and completed.

[23:15] RESEARCHER:

No, that's great. Thank you. That's very detailed. Thank you very much. Let me ask some follow up questions in describing this process. You said that the product backlog should be correctly framed. Is it your role as a QA to review it or the whole development team does a review of the product backlog? And what does it mean to be correctly framed?

[23:49] PARTICIPANT 20:

So, Researcher, the product backlog is generally meant for both developer and testing team. Lately, I have been into the role of QA lead and a lead should always ensure that whatever deliverable is coming through, should have correct detailing of the product so that the testing activities can commence. So, when the product backlog or user stories or requirements are being framed, you have to ensure that the requirements are correctly mapped. I think you also know there's a requirement ID always. So, in the requirement ID, you can always trace your test scenarios as well. In case you don't understand that requirement, you can... That's why I keep a workshop at the sprint planning time. That workshop is used to elaborate the basic. So, let's say, Researcher, you've been given five requirements, right.

[24:52] PARTICIPANT 20:

So, you, as a lead, you understand the system. How the system works. And this is just an add on to the previous system. A new unit has been added. So, if you understand the system, you understand the flow and, in the flow, you will definitely be able to draft out the high level scenarios, high-level covering all the five requirements. In case your requirements are not correctly mapped with those high level scenarios. Definitely the product owner should give a shout that he's expecting A plus B plus C and you have only framed A and B. Or the requirement is to find the average of some values. And I have to find the other values apart from average. Because it all depends how those requirements are being framed. I guess if you Google out or I can send you a draft, I don't have any project deliverables to share.

[25:58] RESEARCHER:

Yeah, that's okay.

[26:00] PARTICIPANT 20:

But I can always ensure that the requirements are mapped and should be automated in nature. Automate means the requirement should cover that [inaudible] retrospect or you could save to make a view. So, you are a customer, I am a system. And how the customer profile will behave, then the system will do X. So, one action for a customer. One action for system. You cannot have multiple actions for customer and multiple actions for the system. Because the probability to test and the probability to do something wrong in cases where the requirement is not clear.

[26:46] RESEARCHER:

You mentioned something very interesting is that to be in sync, which I think you elaborated a little bit, you said the development team and the QA are in sync. What does it mean?

[27:00] PARTICIPANT 20:

Ok. So, Researcher, so the product backlog or you can say user stories are being shared across development team as well as the testing team as well as a common deliverable, right. Or what you can do is always have a quick check with the development team. What is the understanding of these requirements to them? So as a QA lead or you can say, even as a tester, if you understand the system, you will be able to draft half the scenarios out of the product backlog. But the system behavior should be or, you know, you always have that curiosity to know what would work, what would not work and how it would work. What are they trying to constrain? Is there any special requirements. Most of the examples of my background, so we used to have a lot of value constraints, border analysis, you know, what value should be put in? What is the range of the values? So what I would suggest to you is when you have a check with the development team, just keep good sync in terms of the development mode, how they are developing, not the code, definitely a tester would never able to correctly identify what the code is all about.

[28:24] PARTICIPANT 20:

But definitely if you keep, let's say, half an hour check point in a day when developers are developing and you are framing those high level scenarios, it will always help you to put extra scenarios or to the point scenarios to ensure that you are not compromising with the

quality or you are able to check the functionality correctly. So, you can have always a quick maybe meeting or daily status where you can just quickly analyze. Because we know developers also do unit testing. And the unit test is based upon what they're developing. I know that is like one third of the testing, which we do as part of test planning or test execution. But those scenarios which are being checked all the time, are being created by development team, is a benefit to us.

[29:24] PARTICIPANT 20:

We can always map our scenarios to ensure that because whenever they do a unit testing, they have the results. So, let's say we have those scenarios incorporated in SIT, we can always compare results and in case of challenges or in case of the results being very different, it's easy to track what is going wrong. And from where it is going wrong. I see a lot of problems when... you understand the UAT team?

[29:57] RESEARCHER:

Yes.

[29:57] PARTICIPANT 20:

The technical architect. So, you know, it's very unlikely to happen that the patching is not done correctly, or the build has failed in sanity and passed in development department. Or, because my project used to have like a daily build because there's a lot of defects coming down. All those daily builds, you need to keep track of what is coming in. And does that build impacting your particular system. Yes or no. If yes, you have to get the same sanity check to ensure that your regular software performance, as well as software functionality, is not being impacted. This was what I was trying to mean, be in sync up with the UAT team as well as the development team, because their unit test scenarios will always be a blessing in disguise.

[30:54] RESEARCHER:

It seems from the description that it is a highly collaborative environment. Does collaboration help to achieve quality in this type of environment?

[31:12] PARTICIPANT 20:

So, Researcher, it generally depends on the rush or what is the size of your deliverable. With reference to my experience, in all my sprints used to be off like three weeks each. And then there was a clean run. You understand the clean run phase?

[31:36] RESEARCHER:

No. Can you explain it a little bit?

[31:39] PARTICIPANT 20:

Ok, sorry I maybe have missed it initially. Let's say you have two sprints and those two sprints will be comprising together to form a release. You should always go back to or you can plan this with your stakeholders to have one week of clean run. Clean run is the phase which is after sprint one and sprint two and should contain all the defects, all the

functionalities of sprint one which is working. All the functionalities of sprint two which is working. Some production level defects which were fixed. You should have a complete build that will go into production a week back in your environment. And you have done all the major scenarios which were failed in sprint one and sprint two. Why I'm saying failed because the chances of those defects, which were again fixed by development teams, can break things again and again. So that one week of the clean run phase will always help ensure that all those broken things is fixed. And we are testing the final build, not patches.

[32:59] PARTICIPANT 20:

The defects have a patch. I raised five defects today, development said I will fix that in three days. So, he might release a single patch in one day, then two fixes, then on fix. So, he's giving you new patches. In the meantime, while fixing the one, he's not going to check in his unit testing, which he might have broken the previous code. So, the chances of having a bug deduction is more when you're fixing a forward fix. Do you understand the forward fix?

[33:34] RESEARCHER:

Yes.

[33:34] PARTICIPANT 20:

When you're backtracking the fixes, you are just going ahead with the new fixes on the previous code. This forward fixing is always deadly. With my experience, the team did the clean run phase, maybe depends upon the size of the deliverable, it may last for two days, but for us, it used to be five days into your daily performance testing. So, the initial three days were for SIT and the last two days were for performance teams to ensure that the performance is up to mark and not causing any problem. You can start to collaborate from day one with the dev team to the day last model is not delivered.

[34:22] RESEARCHER:

Another thing you talked about is these Agile ceremonies. You talked about sprint planning. So how do these Agile ceremonies help in getting the process right and in improving the quality of the final product.

[34:42] PARTICIPANT 20:

Definitely. OK. For one thing, Researcher, what I really believe, and I want to share that with you. Agile is very, you know, modules based delivery model. Like in Waterfall, you understand, there is always five, six deliveries in test planning and then test preparation and then test execution. There is a lot of things we exchange with clients. Client as the stakeholders or you can say product owners. But in terms of Agile, I really feel the number of deliverables are very low. Let's say sprint planning, so once you understand the user requirements or user stories from the product owner, with their priorities, you discuss within your team and understand which priorities can be met within the same time frame. Or your sprint planning should be, OK, so I don't know this is very internal, I'm just sharing, and it might help you. So, it is very iterative and modular. It helps a lot to inspect the features and correct any errors very quickly, because as I said the scope is small and the iterations are fast.

[35:43] PARTICIPANT 20:

You should always have a formula, or you should always have an estimation of failure as well. That estimation will help to you keep some buffer time. So, let's say you were having a file requirements, which should finish in fourteen days or so, let's say, ten days to be precise, two week delivery in 10 days. You have to test for those five requirements. So, when you get the delivery from the dev team, that is unlikely because we have at least three to four days minimum, at least patches are full build. So, whenever you raise a defect, it may block your one test case or may block your five test cases. And the fix is coming back to you. You have already lost a couple of days because the build was not there, and the defect was kind of blocking those scenarios. So, when we are planning a sprint, you always have to keep a buffer time to make sure that you are not flashing with the timeline at the end.

[36:57] RESEARCHER:

Correct.

[36:58] PARTICIPANT 20:

Because defects will come, and you cannot stop defects from coming. And the integration environment, there are a lot of systems that are working together. Let's say my department had six systems working together. First one was a Siebel, which was used for a kind of creating customers in Oracle. The second one was Boysen, which was kind of supplying all those APIs to all the system. Third one was a Mediation, same as activation, you can make calls. So those calls are being tracked. Fourth one was billing. So, once you made a call, you will be charged accordingly. That was billing. And the fifth one was BI. Once we done with this, we used to create some BI processes and reports. So, you have five systems which even one can break each other. So, you should always play, and it all depends how your project works. How much is the basic estimation, you can do it after one sprint.

[38:06] PARTICIPANT 20:

Let's say after one sprint running forty scenarios, you identified you have run fifteen scenarios extra because of defects. So that means you have concluded twenty-five percent, no, twenty percent at least of excess scenarios with the loss of time because of these defects. So, your sprint planning is a very important phase in terms of evaluating what time we should be doing and what is a priority of those requirements should be in the sprint backlog. In the sprint backlog, you can always prioritize those scenarios which are more likely to fail. Let's say my two requirements are talking about removing a bundle. Let's say you have used something you want to remove it. So those functionalities should be given more attention as well as negatives that should be tested accordingly. So, sprint planning and sprint backlog is the most important phase to plan your testing and you'll ensure that you have sufficient amount of resources. You have sufficient amount of scenarios, testers, and everything ready and somehow reviewed by either product owner in a workshop or in a deliverable. You want to ensure that you are not missing anything.

[39:40] RESEARCHER:

Fantastic. Thank you very much. That's quite detailed. Thank you. I have a follow up question. How this iterative mode of development helps quality?

[39:45] PARTICIPANT 20:

I may not have explain it clearly! It’s the iterative and ongoing inspections. The client inspect frequently and their feedback help us to correct and understand the requirements better. The iterative part is that we fix defects quickly if the pop up after releases. So, the quality at the end of the sprint is better at least less bugs.

[40:02] PARTICIPANT 20:

But I really think Agile helps you to build an efficient product. So, since I have for always products in terms of whatever we use to test and deliver was a part of the product, not the entire software. OK, so Agile is after, let's say we have sprints, so after sprint one, let's say two to three weeks, you have a deficit in terms of the reviews from stakeholders as well as friendly user trials. You understand what is FUT?

[40:46] RESEARCHER:

No.

[40:48] PARTICIPANT 20:

I don't know. I might give you a blast after this call.

[40:55] RESEARCHER:

No. That is great. I like it, yeah.

[40:58] PARTICIPANT 20:

FUT is called a Friendly User Test. What you can do is you can always because see it is just a prospective. You can always ensure that this is being communicated in a very soft manner. When I am building a requirement, my mind is, if I build something, I always think this is right. And if I give it to you, you will have a different perspective of it. You will think it is seventy-five percent right, but still twenty-five percent, it can be better. In this way, what you can do is always ask your product owner or your stakeholders to have a friendly user test. You are part of a product owner team. Let's say you have four product owners or let's say you have four stakeholders in your company. You can ask a couple of guys off your friends to do that kind of testing who are in the same company.

[41:56] PARTICIPANT 20:

Because if they use things by themselves, they will understand what was missing. What is working fine and what can be improved. So, in terms of sprints, we always have that option to get quick feedback in terms of there are a few things always. Firstly, the functionality of the product, the functionality, which is described, and requirement should be seen. Requirement is the functionality delivered by dev team and testing team. And the requirement which product owner has given should match. Second thing let's say there must be many instances where your requirements was changed at the very last moment.

[42:40] PARTICIPANT 20

At that point of time, you have to somehow compromise with the quality or number of tests or mode of testing. Somehow you have to fit into the shape. You cannot do more than twenty-four hours a day. That's what it is. So, in that way, you will always ensure that in his techniques or his visibility of the requirement is sanely constructed and efficiently conveyed to the other person as well. And when they're doing these few things, they would also find something which is in testing and can be an adapted by a product owner in the future.

[43:24] RESEARCHER:

Thank you very much. So, you made this very interesting statement, Agile helps you deliver an efficient product. Is it of a great quality or only efficient?

[43:40] PARTICIPANT 20:

So, Researcher, if I answer you diplomatically, I will say it's more efficient. Unless you understand the complete history of the product and what features it should include. You can never say it's complete or efficient. What I am prone to adapt to this Agile is, with the complete review and less documentation and more in terms of providing your inputs towards work, you gain to have more success and less bugs in terms of a small requirement and a small feature. And then building up a big, big cruise, I would like to build a boat first, ensure that the basic principles are maintained. If your basic principles are maintained, you will always try to increase on all sides. You don't have to think about let's go left first, let's go right first. Because your principles are aligned, you can go left and right simultaneously. And there is a concept called T-shirt sizing nowadays.

[45:01] RESEARCHER:

Yeah, I've heard of that.

[45:04] PARTICIPANT 20:

Nowadays it's being implemented simultaneously. When somehow a product owner is changing the requirements or manipulating the requirements, you can always opt for t-shirt sizing delivery model, which depends upon how much you are unsure of the final model. We may need to change or update our sprint planning accordingly. And there could be a delay of sometime or miss out of functionality in sprint. So, this risk has always to be shared by both parties. And I really believe this way you can efficiently track your quality as well as quality coming from the product owner or the stakeholders.

[46:04] RESEARCHER:

You've been talking very positively about Agile, which is a good thing.

[46:13] PARTICIPANT 20:

I've seen a lot of faults, not faults, but demerits of having Waterfall and V model. I work for about eight years in Accenture and we were having the approach of doing manual tests. And I have seen from manually tests using Waterfall and V model how inefficient, I'll say. I'll not say they not working models. They are very good work models based upon what kind of delivery you have. But I really feel there is a lot of time that is not efficiently managed in Waterfall and V model but on the other hand, Agile is more about quick work, quick check,

quick feedback, definitely as part of your delivery models. Being overall checked by everyone is first, you have to ensure that whatever you're doing should be done correctly and should be reviewed and checked so that you also get quick feedback.

[47:24] RESEARCHER:

Is this a loop of quick feedback helps improve the quality of the code and the quality of the process?

[47:34] PARTICIPANT 20:

Okay. So, Researcher, this answer is a little bipartite. It's a handshake process. I really feel, if you do your work correctly in terms of identifying all the major things that should be taken care. If you follow your processes right if you ensure that there's always a chance to improve. I have seen a lot of circumstances where things went wrong. At the end when no one is using it, no one is there to take the blame. It's either, the test team or dev team for not producing the things correctly, but when you improvise by yourself, taking a few steps that will always help you to bind things together. The one or two things that I do to have a workshop, not a big meeting, but a workshop to completely tell the client, what are you thinking about and what he's thinking about, not a quick feedback is always great. You always give a thing back to your stakeholders that this is what we are approaching towards. And it may work well, it may not, but you always get to listen and learn.

[49:02] RESEARCHER:

And those lessons learned are fed back into the process and the way of working.

[49:07] PARTICIPANT 20:

Actually, every feedback cannot be accomplished in the same day. You have to give your feedback back to the stakeholders saying that, you know, I have learned this, and we will try to improve, but we need your support and we need these things from you. So, if I say I got the requirements wrong, I understood the requirements wrong, I may have done better if the requirements were, let's say the requirements were shared in an Excel sheet or board columns. The requirement ID, the requirement action owner, and the requirement was too big. Even it was to make too big, and the action on this as a customer, as a system and as a machine was not clean. So, what I can always do is go back to the stakeholders, tell them to populate these pages. Or if he is not populating, I can populate and give him the cross-check. So, if I am getting the process right, I will do the process right. If there's a barrier in terms of his communication and my understanding or my communication and his understanding, so what best case, use the same deliverable which he's using to cross-check by himself so that, you know, see, I'm just this service provider. I just [inaudible] and this is the money he's investing in other deliverable margin to make sure that he's getting the end result. So, you know, as we say, the customer is always gold. So, in this case, he's a customer, I'm just giving him the service so that ways we can ensure that he is also taking things positively with little improvisations, every time you learn anything from the last sprint.

[51:01] RESEARCHER:

The last question, it's a little bit provocative, but the purpose is to get your opinion, not to create provocation. What do you think of the statement, Agile produces poor quality software.

[51:18] PARTICIPANT 20:

Even though I'm giving all the credit to Agile, I really feel at some point time, it does have its drawbacks. Because clients, customers, the product owners, all stakeholders can any time change your mind. You cannot freeze anything in the last moment. So definitely this is not a good practice and it always definitely impacts, proportional to the quality and the features that you're building and testing. There's always be a time to hash it because of the bugs you are finding, and the timelines are so less. So, I will say this, this is like seventy percent, I'm with Agile, and thirty percent with not. It is there to build a perfect model for software. I'll say yes, because of the more efficient deliverables are more efficient manner being used. It is there to improvise, and to be dealt with in the best methods. But on the other hand, there are some methods like getting requirements incorrect. Scrum masters not playing the rules well, a wrong estimation at sprint planning can definitely produce a fata result.

[52:55] RESEARCHER:

Well said. Thank you very much.

[52:59] PARTICIPANT 20:

I was talking too much, I guess.

[53:02] RESEARCHER:

It is good because we want you to talk. Your perspective is very important to us. That's why we are here actually is for you to do the talking, not for us to do the talking. So, we want your knowledge. We want you to share with us your knowledge. Thank you very much. Do you have any questions for me before we conclude?

[53:26] PARTICIPANT 20:

Yes. So, what will you do Researcher, now since I have too much encouraged you to do Agile?

[53:34] RESEARCHER:

So, this is a research project. I work for a university, so I do research. So, what we do is we analyze the interviews and we make conclusions and we will write a research paper. If you are interested in receiving the research paper copy, I can send it to you. It will be available in August. So that's what we do. We hoping to bridge the practice with theory and your experience and other participants, in the interviews, experience hopefully it will provide insight for us to bridge between theory and practice. So that's the whole idea behind these interviews.

[54:32] PARTICIPANT 20:

Yeah, sure. Researcher, I guess it was a lovely talking to you. I would love to receive that paper when its published in August. So please.

[54:42] RESEARCHER:

Okay, fantastic. Yeah. I do have your email. It's [redacted], right?

[54:54] PARTICIPANT 20:

Yeah, exactly.

[54:55] RESEARCHER:

Fantastic. So, I wish you good night because it's getting late down there in Melbourne. Thank you very much. Have a good night. Bye.