[00:58] PARTICIPANT 10:

Hello?

[00:58] RESEARCHER:

Hello, how are you?

[01:00] PARTICIPANT 10:

I'm well, thank you.

[01:03] RESEARCHER:

Okay, fantastic. Do you have any questions for me before we start?

[01:10] PARTICIPANT 10:

No, I went through the documents that you provided me.

[01:17] RESEARCHER:

Okay. Fantastic. Let's start with the questions. Can you please introduce yourself and talk about your experience?

[01:27] PARTICIPANT 10:

My name is [Deleted to preserve the participant anonymity] I'm a business analyst, product owner, project manager, UX person, a lot of things. So, I can give you a very brief description of my background. I am not, you know, trained in I.T. in any way. By education, philologist, which is the study of languages and literature. [Deleted to preserve the participant anonymity]

[02:09] RESEARCHER:

[Deleted to preserve the participant anonymity]

[02:12] PARTICIPANT 10:

[Deleted to preserve the participant anonymity] But I forgot everything. It was a really long time ago. Last time [Deleted to preserve the participant anonymity] was sixteen, seventeen years ago. [Deleted to preserve the participant anonymity] Anyway, so after university, I found a job as a technical writer at a company called [Deleted to preserve the participant anonymity]. It's a big American company that does boxed products. It's very traditional. They do like Active Directory, Exchange for big corporations. So that was sort of my introduction in

the I.T. world. Though I was a technical writer, so I wasn't involved in the requirements or anything like that.

[03:09] PARTICIPANT 10:

It still gave me a glimpse of how it operates in a more traditional environment where you have big releases, you have a whole bunch of requirements. And then you have huge teams doing a lot of stuff. Then I moved to a [Deleted to preserve the participant anonymity] they had a branch, a service branch where I started doing business analysis and requirements and those type of things. Ten, no nine years ago, I left Russia and started to freelance. So, I started working in the form of different projects and combining a lot of different roles because, you know, when you put in small teams, you kind of have to. And mostly, you know, the teams would be remote then I would have clients mostly in the US. But they ranged from Europe to Africa. So, yeah, for the past nine years, I've been working with enterprise clients, small clients, bigger clients, startups (absolute nightmare). So, this is in general my background experience. So, unless you want me to go into detail?

[04:06] RESEARCHER:

Before we go into the details, how do you define quality in the context of agile software development?

[04:17] PARTICIPANT 10:

It is difficult to define and measure. That’s for sure. I think before agile, there was a great focus on product level quality. Mainly free of defects and meets customer’s needs. I saw in agile a shift toward code quality and a design that responds to change. There is also a continuous improvement of the process effort.

[04:46] RESEARCHER:

We'll go to the detail later on in further questions. Yeah, we do have detailed questions.

That was fantastic. Thank you. The next question, you may be biased in answering this question, but a bias is OK, what do you think of Agile? What is your opinion of it?

[05:10] PARTICIPANT 10:

And that I think that is the most important question here, in my opinion because different people have very different ideas about what Agile is. What it is, and how to do it. And the way I see it, Agile was a response, a very valid response to a problem that is also perfectly clear. We have these principles of doing engineering work that date back centuries that simply don't work in I.T. So, when you're building a bridge, you can spend a lot of time devising the plans, measuring everything, figuring out what kind of concrete you need. And that approach for that type of work, that makes total sense. With software, it doesn't really work. I mean unless you adjust it somehow. So, you do get a lot of bureaucracy, you do get a lot of the same as you going to get in software. Where you have these huge releases and you have a lot of things planned. This

way of working, it just doesn't accommodate for change. You cannot in the middle of construction, say well, actually, we need a different kind of bridge.

[06:45] PARTICIPANT 10:

You cannot do that.

[06:46] RESEARCHER:

No. It's too late.

[06:48] PARTICIPANT 10:

In I.T., that's absolutely normal. And sometimes you truly don't understand the full extent. And in that sense, the Agile principles make total sense to me. Because the idea of valuing, you know, individuals, interactions, over processes and tools, makes perfect sense to me. And working software over comprehensive documentation. Again, total sense to me. It all makes sense to me. I think where it all went wrong is that people tend to look for easy solutions and they somehow missed the idea that people, again as far as I'm concerned, people who came up with the manifesto, they came up with a new response to this overwhelming popularity of the Waterfall approach. The traditional “let's scope everything out, let's spend six months writing five-hundred page specifications and then we're going to build it”. And that is not an efficient way of doing things. That's absolutely obvious to anyone who actually did those projects. So, they did want to change things. Since it was sort of a response to this traditional idea, they explained, this isn't working if you're only putting value in, for example, tools and processes. You also - and maybe more so - consider interactions. And then you have this pendulum swing all the way to the other side. And it became sort of the norm that, you know, we don't need processes, we don't need tools. And now you don't have to deal with all this bureaucracy. And it's all very free-spirited and nice and you can do basically whatever you want, but which is also not the case. Because then you get to anarchy where everybody does whatever they want. This kind of interpretation of Agile as a free-for-all and that the teams know better all the time…

[09:09] PARTICIPANT 10:

[audio breakup]

[09:29] RESEARCHER:

Hi PARTICIPANT 10, I can't hear you.

[09:58] PARTICIPANT 10:

…I get that things will change. And I whole-heartedly agree that it makes total sense to try things out, see what works and what doesn't work. That's perfectly fine. Well, that is no excuse to just skip over the thinking through part altogether, because that's exactly how it happens. And it leads to really annoying results where you spend several months doing something and then you get very predictable results and you say to the powers that be: “well, you didn't have to spend all this money, you could have just thought this through”. But no, that's not how we’re

doing this, we're doing "Agile". So, we have these very different interpretations of Agile. And as long as we take these Agile principles and apply it using common sense, we can do a whole lot of good. I worked with very different teams on very different projects, but in most cases, it was, some kind of hybrid approach, where of course you use Scrum, Kanban boards, you use all those things, you use Continuous Integration, a whole bunch of Agile methods, but you do not replace the idea that you still need tools, you still need processes, you still need documentation. It doesn't have to be five hundred pages long.

[11:37] PARTICIPANT 10:

And of course, to me, there is this thing called Disciplined Agile Delivery. I don't know if you heard about it. So, to me, that's one of the better symbiosis for complex projects. If you're making something simple, anything can work, but if you're doing anything complex, you need to think things through, you need to have some idea of what you're doing, you need to understand what problem you're solving. Again, one of the issues with the Agile manifesto, at least with the interpretation of it, as far as I'm concerned, is that is does overemphasize software. So as long as you deliver a piece of software that works, and again that's...it might be the case sometimes, but in most cases, you're solving a business problem or you're covering a business need, there is a higher level, there's a strategic view of things. You don't just do it in a vacuum. You don't add a new feature because somebody decided it would be a nice thing to try out, something to do. And that's fine, but as a general rule, you have this idea of what you want to do.

[12:59] PARTICIPANT 10:

You might not know how you're going to do it, you might not know all the details, but it's important to understand what's necessary right now. And that's this concept of Just-In-Time delivery. You are preparing things as they need it. You don't prepare absolutely everything six months in advance, but you also don't go to the other extreme where you prepare absolutely nothing and then just wing it and say, you know, oh let's just see how it goes. So, this was a sort of long answer to your question. But Agile is immensely useful. And doing complex I.T. projects in that sort of old way, the traditional way with the Waterfall approach, is utterly wasteful and sometimes outright impossible. And again, it might vary from setup to setup, because I guess in some projects it would make sense. I haven't worked with anything like that, but I assume that you're designing software for, for an airplane, you don't want to do a lot of experiments. It's probably there is a lot of vigorous testing and product calls and documentation. And that's perfectly fine but if you're working, for example, on something like healthcare (and this something if I do have experience with), you don't need 500-page specifications, but you do need to think things through.

[14:37] PARTICIPANT 10:

You cannot push something out and see how it goes because this is the patient's livelihood. It's not some guy or gal for whom a game crashes. Who cares, ultimately speaking. But if you're in a healthcare setup and you're a system, and you have a lot of people and their health and well-being at stake, you cannot afford to wing it and not know what the end result is going to be. You do have to think things through. But the Agile practices themselves, they're insanely, incredibly useful, and important. And in all of my almost sixteen years of experience, I haven't worked on a single true Waterfall project. It's just, it almost never works. At least in sort of normal setup outside of highly specialized projects, like, software for planes or something. So, in my

experience, there's always some kind of hybrid with some thought and processes and Waterfall elements within it, but the core is always Agile one way or another.

[16:02] PARTICIPANT 10:

You can use the Kanban board, you can skip the Kanban board, you can use Scrum or you can call it something else, like a coordination meeting maybe, every day or every two days, but whatever the implementation is, it's always Agile. In all my experience, that's been the case. It's just in some cases would go way, way too far and I might be straying into the next question here, if there are further questions, but for example, when I started working for a startup that I'm working with right now, their understanding of Agile was the Kanban board that was updated literally every night. So, the technical team would sit in Russia, the founders and marketing, all those guys sitting in Mexico, so there's like eight or something hour difference. So, during the Russian night, the Mexican guys would go to the board, add some tickets, move things around. Then the morning, the Russian team wakes up and looks at the board and sees what they're supposed to do today. There is no planning. There is no idea why we're doing this. It's just this endless flow of stuff appearing on the board. And it's led to absolutely catastrophic results. But I guess we can go into that detail later.

[17:32] RESEARCHER:

Yeah, I'd like to follow up on two things you mentioned. You mentioned...before I do that, do you mind if we drop the video? Hopefully, we can improve the quality of the audio. Thank you.

[17:47] PARTICIPANT 10:

Does it help?

[17:48] RESEARCHER:

Yes, much better. Your voice is much better. You mentioned something about anarchy. Does Agile create anarchy or is a poor implementation of Agile create anarchy?

[18:10] PARTICIPANT 10:

[audio distorted]

[18:24] RESEARCHER:

I'm sorry. I'm sorry. Your voice is really terrible. I don't know why that is. It's getting better.

Can you try now?

[18:36] RESEARCHER:

No, it's still very far away. It sounds like a very far away.

[18:55] RESEARCHER:

I still can't hear you.

[audio distorted]

[19:48] RESEARCHER:

Hello?

[19:54] PARTICIPANT 10:

And now?

[19:54] RESEARCHER:

Oh, fantastic. Perfect. Thank you.

[19:59] PARTICIPANT 10:

It's the one case where Bluetooth works better than the cord. Anyway, so, yes, as far as I'm concerned, it's always the implementation. Because Agile, again, we're talking about Agile manifesto, it's very generic. It's just 4 of these big headlines with twelve supporting whatever they're called. So, it's not some comprehensive… I mean, if you, for example, take my favorite Disciplined Agile Delivery, there is a book, which is like several hundred pages long. So, you can read it through. And it's a comprehensive way of doing things. You can agree, disagree, use something, discard something, that's fine. Same thing with RUP, which is essentially a predecessor of disciplined Agile, which was more sort of Waterfall-oriented, but still with a lot of very good ideas. So, you can agree or disagree with the approach. But that's, again, well defined. Whether you agree with it or not, whether it makes sense to you or not.

[21:09] PARTICIPANT 10:

But still, it's a well-defined approach. Agile is four main sentences and then twelve supporting ones. Well, it's always about interpretation. I cannot imagine how Agile itself could be good or bad, because when the level of those generic statements makes total sense, how you interpret, how you apply it, that's a different story. So, to me, the only thing can lead to chaos is the poor implementation and absence of competence and discarding the idea that developing software solutions is still a very complex process, and it requires a lot of, again, competence and attention to detail. And just general ability to set up a very complex process. And being responsible about it and being realistic about it. Actually willing to do something great instead of selling the approach, getting the money and hitting the road, which is sometimes the case. But to me, it's not about Agile. It's about people, who… some do good things, some do not so good things.

[22:36] RESEARCHER:

You mentioned three interesting concepts, which I would like to follow up on which is discipline, competency, and the people. Do these three elements make Agile work better?

[22:53] PARTICIPANT 10:

Yes, absolutely. I mean, with competence, I mean, everything is better if you have competent people. Discipline to a certain extent. I'm not a disciplinarian. It's more about sort of discipline as an absence of undisciplined behavior and... as long as people are competent and as long as they care about the work they're doing, I think you're good. Competencies, they could grow. I'm talking about sort of the internal sense of competence and the value of it. So, you might not know things. That's perfectly fine. But if you understand that, okay, I don't know this thing, but I'm willing to at least listen to somebody who is competent in that area and make a decision based on that, rather than, I'm going to use my incompetence to make whatever decision I want. And that's exactly what I'm seeing quite often where you have people who are inexperienced and they - that's weirdest thing - they read the books and they get the soundbites and they try to use the tools without understanding the problem.

[24:30] PARTICIPANT 10:

I don't know if you understand me.

[24:33] RESEARCHER:

Yes.

[24:34] PARTICIPANT 10:

So, for example, if you take very, very simple example, like, the retrospective meetings, which is sort of the norm for virtually any team where you go through a sprint, or through the whole project or whatever it is, and in the end, you hold the retro where you discuss how things went, what wasn't working, what was working, how we can improve. So, the whole team gathers and discusses and that's a wonderful tool. It's very helpful. And it does lead to great results because it just makes sense. But it also depends on how you hold that session. If you just read “you need to hold the retro” and then you download a template or something and just go through these questions and just try to replicate a process you don't understand. You don't know what the need for it is, you just know that successful people do it, successful companies do it, so I'm going to do that as well, it's going to work for me, and it won't. It just doesn't work because there is a reason why this happens.

[25:52] PARTICIPANT 10:

And you need to know how to hold these meetings, you need to know how to talk to people. It's not enough to just gather everyone in the same meeting room, or on the same call and have people say things and write it down. It is what that meeting is, of course. But if you don't understand why you're doing this, if it's not something that rises out of your personal, or personal in this generic sense, if it doesn't rise from the need of your project, then it just becomes a formality. And there is a lot of that where people just take these tools and processes and solutions, and they apply them indiscriminately. It's akin to going into a pharmacy and saying, Okay, I heard that somebody got better by taking this drug, so I'm going to take that as well. Well, you need to understand how that decision came about that that person took that drug. So, I'm sorry, I hope I answered your question.

[26:59] RESEARCHER:

Yes, you have. You have in a great level of depth and detail. Thank you.

[27:07] PARTICIPANT 10:

I understand that we have an hour but if we need to go over that, that's perfectly fine.

I know I can be wordy.

[27:16] RESEARCHER:

It's OK. It's a good thing for this type of work. The next question, can you describe your Scrum environment to us? I know you have been mainly working on Scrum. You could choose any one of your experience.

[27:31] PARTICIPANT 10:

I think I can describe a couple. One that I love and one that I hate that's currently what I'm working on. So, I'm working with a startup with the founders and marketing and everyone else in Mexico, although they are French. The one is Polish, they both live in America, so it's all very international. But the technical team, the engineering team is in St. Petersburg, Russia, for the most part. There's one guy in Azerbaijan , there's one in Ukraine. So, it's a little bit distributed. But the core team is in Russia. And the other team is in Mexico. And when I arrived there, I already described the setup to you, that there is a guy who is one of the founders and he just spews out requirements onto the Kanban board without any format, just however he feels. He just does it and throws it on the board and it's up to developers to figure out what he actually wants, and how it's supposed to work with no regard that there is a process, that it's complex, that you need to figure out the requirements, you need to discuss it with the team, you need to understand the architecture, how it fits into the architecture. He didn't care about any of that.

[28:57] PARTICIPANT 10:

And he would come and say: “Well, it's a simple thing. I mean, why does it take so long. So just do it, just do it quicker”. And, you know, all the attempts to tell him, okay, it might look like a simple thing, but we need to do a lot of refactoring and we need to adjust architecture for that, just to accommodate this little thing. These would often fall on deaf ears. He'd say, “no, I just need it done, it's important, the clients want it. So just go ahead and do it. I don't want to hear anything”. And that's the setup where I came in as a product person, project person. Basically everyone, because we had a bunch of engineering people, no managerial staff whatsoever. [inaudible] So I came in to be the middleman and I started with explaining the value of the process, that it might seem that things will go slower, but that is just how it works. You can only go fast for a while, but when you're going fast and not thinking about the consequences, you are increasing your risks. And you are basically digging a hole, which you not going to notice for a while, but at a certain point, things will start breaking where you need to change really a very small thing,

[30:34] PARTICIPANT 10:

…but due to all there is, I don’t know if there’s such an idiom in English, there's a Russian word “kostyli”, which means crutches. When people break a leg, they use crutches. So, when people write the code, they basically use crutches. So, it kind of works, it's not the best solution, it's not the elegant solution, but we need to do it fast, so whatever. It's a makeshift solution. That'll do for now. And when the amount of those “that’ll do for now” decisions reach a certain point, you just have this weird code where everything is based on those little makeshift solutions and crutches, and the duct tape and prayers, and a little bit of cement. And that is a very unstable structure where even a small change, which is honestly a small change, it's not like you need to change the architecture. You need to adjust a little thing. But when you adjust that little thing, you touch upon something else, and then it breaks because it wasn't a good solution to begin with. And then it just starts the chain reaction and you end up rewriting and refactoring the whole thing and spending weeks on that.

[31:56] PARTICIPANT 10:

And that's just the reality. You can go fast at the expense of a good, well thought through architecture solution, but you're going to end up with something very, very unstable, or you can slow down a little bit and allow the team to accommodate those changes and think things through and think through the possibilities. When a requirement comes in, it's not just one little piece of, “we have this process, it has a number of steps, we just need to add another step, so we’ll just add the step and that's it”. Just so you understand what I'm talking about, the startup is about online verification. When you encounter something like that you provide a selfie or a video of yourself and you provide images of your documents and then some magical process says “yes that's actually you on the document” and your identity is verified.

[33:05] PARTICIPANT 10:

So, for example, we need to add a step for rotating your selfie or whatever, then nobody on the technical side has the time or capacity or the skill to ask why. Why are they doing this, what is the bigger context, what are we trying to achieve by going with this particular decision. What else can we do, how else can address the bigger need. None of those questions are asked. And just people keep doing things, doing things, doing things. And it becomes a mess, understandably. So, I try to instill a sense into all of that and bring a little bit of order and process into this. Again, I'm no disciplinarian, I really don't like processes for the sake of processes. So, for me, they have to arise from the very real and understandable need or a problem. I identified the needs, I presented a certain process. The requirement needs to go through a certain funnel before it reaches development. Let them do their thing. First, we talk it over. We figure out how it's going to work, what it's going to be, what are the alternatives, what else is it going to affect?

[34:35] PARTICIPANT 10:

Then we'll come up with some design ideas so that, you know, all this time, this is happening separately from the development team, it's just not their business yet. So, once we have something more specific and feasible, then we would come to the development team and then we'll discuss it with them. They would throw out ideas. Maybe they'll suggest something slightly different that would save us a lot of time in development or that would allow us to do things differently, or would give us more possibilities later on instead of getting locked in on a specific solution. Sometimes you can spend a little bit more time right

now on this little thing, but then later on, you have more options and you don't have to redo the whole thing. And that's really helpful. So that's the kind of process I try to build and also try to get away from this insane Kanban board that would just change daily and move on to sprints because you cannot really estimate anything if it's just this endless board where things appear and disappear without any structure. And thankfully, we also hired a head engineering and that person helped us move to sprints, because I just didn't have the time to do that as well, because educating those guys was a full-time job, in addition to the actual work on requirements. But I'm still seeing the same approaches, they just come back all the time.

[36:19] PARTICIPANT 10:

You think you got through to them. It's just the same thing all over again. It's great that we have a sprint, and that sprint has a scope, and we agree on that scope. And unless something really, a real emergency happens or something really bad happens, we stick to that sprint, to that sprint scope, because we need to understand how much work we can actually do, to plan better so that we can provide better deadlines. Otherwise, we're just, well, we're going to release it sometime next week maybe, or the week after that. We'll see. You cannot have realistic deadlines magically appear out of nowhere. It's a process and it's an iterative process. You do things and you learn what was wrong and what was not optimal. And then you change them and then you see what's the effects, and then you do it again and again and again. And eventually you get to the point where everything is great. But these guys are very impatient and it's a nightmare. Because whatever process you have set out for them, one day somebody could jump in and change the whole thing and “oh, we need this right now, right away. Well, yeah, I understand that there's great scope, but we really need it so please go ahead. Just do it. No questions asked.” And so that's kind of the setup, right now.

[37:58] PARTICIPANT 10:

It's chaotic. There are a lot of very professional, competent people on the team, but it has a devastating effect on the team morale because people are bickering. They are you know, there's endless conflict and as much as you try to bring everybody together, it's also part of my job to be sort of a psychologist for the whole thing because all the reconciliation happens through me eventually, because I'm the source of requirements for the technical teams, so when there is a conflict, I also try to help people let go of their own perspectives and just focus on that thing at hand, that we are here to do a job. You know, we're here to make something happen. And it's just a matter of finding what's the best way. It's not about choosing some person's option versus another person's option. It's not about you, guys. But I am also no Buddah, I can get caught up in my emotions as well. So, it can be challenging, especially comparing to some other more established processes. For example, I worked on a healthcare project for a chain of physical therapy clinics in the US and they were doing what's called HER or EMR (electronic health record / electronic medical record). They were doing their own proprietary system for managing a patient's health record throughout the whole lifecycle.

[39:56] PARTICIPANT 10:

So, the patient comes in with a condition and we need to do a Plan of Care, then Daily Notes, so all kinds of medical documentation that comes with it. We were working on that as well as connection to billing because, well it is US health care, it's kind of a Wild West. It's absolute insanity with all those insurance companies. It's really complex, but you got to do it. That's the system. So, all the billing had to be incorporated with medical documentation. So that when a doctor performs a procedure that it's correctly identified and billed correctly to the insurance company under certain rules with certain codes and calculated in certain ways. So, it was an incredibly complex project. And the setup was so there was a string of clinics. There was a guy in the US who had a web development company which consisted of himself effectively.

[41:08] PARTICIPANT 10:

So, he was handling all the sales and all that stuff. I was the product owner, I was responsible for the requirements and we had a partner in Belarus, an I.T. company with their own established development team. So, I would gather requirements from the clients and decipher them and translate into something actionable for the team. And that worked beautifully because we had the chance to actually talk it over before we involved the developers. But then again, it was a sort of self-contained project with very clear goals and objectives, at least on a high level. That's the beautiful thing about healthcare projects, because at the end of the day, they just want to do what's best for the patient. And that's really helpful because that sort of gives you the sort of this lighthouse that helps you focus. So, I would talk to very different people from all the different departments. We would agree on what the process is, what is not working, what is working, what we would like to change, not on the level of we need to put a little button here or a little checkbox here. But just on a higher level, what we're doing here, what are we trying to achieve?

[42:30] PARTICIPANT 10:

So, going from this high level, then we would break it down further and further, and when we've broken it down far enough, then we would involve the team and that would extend to the whole project. So, we would have a first batch of requirements. So, the team is aware of what we're building. I would just talk to them and explain. You're not going into a lot of detail and just explain the concepts to them. What we're doing, why we're doing it. And then what was helpful is that it's an established I.T. company. They have a lot of competence. So, they had absolutely everything to set up this particular project. They knew how to do continuous integration, they had QA and DevOps and all that is necessary.

[43:28] PARTICIPANT 10:

People who know their job and as long as they got coherent good requirements that we could talk through and actually plan (not necessarily in great detail at first), they were good. Going in this non-frantic manner and not having to deal with very hectic people who just demand results no matter what, that's another thing that's helpful. These clients knew that some things take time and some things take patience and that you cannot just wish that everything were great. They cannot wish a person's bone to heal immediately. It's just not how it works. They can do a lot to make it heal well, to make it heal fast, to make sure that

the patient gets in amazing shape as soon as possible, but there are no magic solutions here. And this absence of irrational and wishful thinking on the client's part that helped us do a great job ourselves because we would come with the requirements that were well enough thought through. Defined well enough. If something wasn't clear, we would figure it out. Some details would obviously...you cannot accommodate for everything, you miss things.

[45:01] PARTICIPANT 10:

That's the nature of life. You try to miss as little as possible. And when you do figure out, "OK, we missed this", then you just correct it. And if things change, you're just "OK, these things changed". But that's also where this initial planning helps. When you're thinking things through in a very high level, at that point, you see this is an area that is somewhat volatile. We don't really understand. And there is this guy who knows more about it, head of finances, but he's on vacation right now. So, we'll just leave off for now when that guy comes back. And we'll discuss it more. And we know that we shouldn't touch it just yet because there is a lot of uncertainty around this. So, we would start with things that are clear and that's us being agile. That's us not being stuck to some predefined plan. But at the same time, not sacrificing the value of thinking things through before you actually do it. And that was one of the best projects, if not the best project I worked on, because it was huge,

[46:21] PARTICIPANT 10:

…it was immensely complex, and it was very successful in the sense that we delivered everything as we were planning, and mostly on time. There were some complications, of course, but again, nothing too critical, nothing too bad. And the clients were happy with us and we gave them what they needed, not what they thought they wanted. We actually went through their needs and their problems, understood them, and designed a solution for those particular problems. So, we had happy therapists, we had happy nurses, and that was the best thing of it all.

[47:17] RESEARCHER:

Fantastic. PARTICIPANT 10, I have to ask you the final question, because we're running out of time and I have another interview. Yeah, I have another interview.

[47:28] PARTICIPANT 10:

Oh, sorry.

[47:29] RESEARCHER:

No, it's OK. Do you think Scrum produces software quality and how, from what you have experienced?

[47:38] PARTICIPANT 10:

If implemented correctly, yes, sure. Absolutely, because what Scrum did, because in a sense, agile doesn't speak to quality directly. I mean, it's just sort of implied that you should provide things with high quality. You shouldn't put out crappy results, although sometimes you have to compromise. But still, it's the implied sense that you're providing something that is good. And in this sort of hectic environment, you don't always inject the classical QA that takes like two months to test the whole thing. So, you have all these techniques appear with continuous integration, with automated testing, with scripted testing, where you write a script that actually clicks things through. You have the developers testing themselves. You have code reviews where developers review the code of each other and that helps with education and that all produces better quality.

[47:49] RESEARCHER:

What do you mean by education and how does it help software quality?

[47:55] PARTICIPANT 10:

I mean when people collaborate in Scrum, they share their knowledge more efficiently. They review each other’s code and learn from each other how to write better code and what is the best option to code a particular problem. This knowledge exchange help producing the best code the team is capable off.

[48:58] PARTICIPANT 10:

But this does not a replacement for good old QA, especially on complex projects, because you have to do a lot of regression and cover absolutely everything with auto tests. It's an insanely costly endeavor. It just really costs a lot if you want to cover absolutely everything. So, when implemented correctly, agile absolutely helps with producing quality software and not just on the software, but on the solution for the business, because in all my experience, we had exactly one project without QA. It was an internal system for operations of a small airline. We had three developers who were testing themselves. And I was testing and the people who worked there also effectively were testers because it was an internal system. So, when we were releasing something new, we would tell them in advance. And if it's something really sort of dangerous, we would push it to staging first where people could take a look and see if it works.

[50:24] PARTICIPANT 10

But that was literally the only project. Everywhere else we would have QA who would test things and make sure that quality is on the level that is required. But again, in my opinion, Agile is by no means anathema to quality. I think it helps. It helps a lot. As long as it's implemented properly and as long as it is implemented by competent people who do wish to produce quality solutions.

[51:03] RESEARCHER:

You mentioned that in Scrum, you do have QA. So, at what stage do you engage the QA in a Scrum?

[51:15] PARTICIPANT 10

QA are involved in the very beginning. So, when in a normal project, when I have the feature or whatever it is, described to a level of detail that is sufficient for the development team, during those meetings, QA is always present because they need to understand. The developers need to understand what to code and testers need to understand what to test. Some would write detailed test cases, some don't do that. So, it's different from set up to set up. But QA is always involved at the same moment as the developers are because it's important for them to understand what to test. And how that new thing, or that change or new feature, can affect something else. Because of course I can think things through from my end. Developers can think things through from their end. But neither of us have the same perspective as QA. So, they need to be aware and that's why always no exception that QA must get involved at this point.

[52:31] RESEARCHER:

This early engagement of QA and team members, developers, and this early engagement, it empowers the team member. Does it change behavior?

[52:48] PARTICIPANT 10:

Whose behavior, sorry?

[52:50] RESEARCHER:

The team members, the QA, and the developers?

[52:56] PARTICIPANT 10:

I don't know if it changes, but I think it's the right thing to do because it sort of...I don't know if it's an agile thing, but I believe that everybody should be doing their job well and have the opportunity to do their job well and have the chance to be competent and do things they're good at. So, I am good at requirements. This is what I do. And it is my job to prepare the best requirements I can. And these requirements should explain what is necessary, what is the end result that the business is looking for or at least the solution we arrived at. But I am describing the end result and not telling developers how to do it and not telling them to design the database in a certain way or use this library or something else. It's not my business to tell them that. It's my business to do the best I can to give them the best requirements that are unambiguous, that are clear, that cover all the possible scenarios.

[54:05] PARTICIPANT 10:

So that then they could get that information and make a decision based out of their competence and out of their knowledge of programming languages, architecture and all of that. And I find this very empowering when people can actually do their job by not being told how to do their job by other people who are way less competent than them in that particular area. That doesn't mean they have to be absolutely separate, of course. I can voice my concerns, I can voice suggestions. But again, from the standpoint that “guys, you are the experts in this. What do you think about that, or that, or that”. Same thing with QA, “guys, you are the experts in that. So, it's up to you how to make sure that the quality of the product is on the level”. And that, I think helps. At least on the teams I worked with, the projects I worked with. It's been a very satisfying experience when you're doing your job and as long as you're doing your job well, you can then pass off the result of your work to someone else and you know that person is going to do their best to work towards the final goal of actually providing a business solution.

[55:30] PARTICIPANT 10:

And I think that helps a lot with just the general atmosphere in the team, where you don't have which I am seeing the opposite in my current setup with the startup where you have incompetent people telling competent people how to do their job. And that's what creates a lot of tension. That creates a lot of angst and anxiety. And that really doesn't work.

[56:03] RESEARCHER:

Fantastic. Thank you very much, [Deleted to preserve the participant anonymity] Do you have any questions for me?

[56:10] PARTICIPANT 10

No, it's just. If you need anything else. But if you need me, I know you have to run. But if you have any follow up, we can just talk again over the hour, that's perfectly fine. I know I talk a lot.

[56:31] RESEARCHER:

No, it was good. Thank you very much.

[56:37] PARTICIPANT 10:

Thank you. And if you can send me the final work, I would appreciate it.

[56:44] RESEARCHER:

Ok. Yeah. Fantastic. I will do. Just give me a second. I'm going to check if I have your e-mail. So, it's [Deleted to preserve the participant anonymity]

[56:55] PARTICIPANT 10:

Yep.

[56:56] RESEARCHER:

All right. Fantastic. It will take a few months before the paper is ready. But thank you very much. That was a very insightful interview. Thank you.

[57:10] PARTICIPANT 10:

Thank you. And again, if you have any additional questions, please don't hesitate to reach out. I'll be happy to help you.

[57:22] RESEARCHER:

Okay, I will. Thank you very much. Bye.