00:32 RESEARCHER

Hi PARTICIPANT 35. How are you?

00:32 PARTICIPANT 35:

Hello.

00:34 RESEARCHER:

How are you this morning?

00:36 PARTICIPANT 35:

Okay. It was a very hot night. Yeah.

00:44 RESEARCHER:

Okay. We having nice weather these days. It doesn't get much hot in Denmark. So, I think last night was around twenty-two, which is really good.

01:02 PARTICIPANT 35:

Nice.

01:03 RESEARCHER:

Okay, I'd like to start by thanking you for the opportunity to talk to you and accepting to do the interview.

01:14 PARTICIPANT 35:

Thank you too.

01:15 RESEARCHER:

Okay, I'll start with introducing myself telling you basically what I do and why I'm doing these interviews and explain to you the structure of the interview. And we can start after that. Okay, so my name is [redacted] I'm a researcher at the I.T. University of Copenhagen. One of the universities here in Denmark, based in Copenhagen. I do my research in software quality. Basically, I try to understand how software team manage to achieve quality, what do they do, do the tools, do the methods help them. And currently I'm running this project, try to understand whether Scrum make a difference in a software team to help them to achieve better software quality. We do interviews, because we want to understand people experience and people perspective, we capture the experience as a sort of a source of knowledge. And we analyze it, and we make conclusions. And we try to publish those conclusions and disseminate those conclusion in a research report. That's the outcome of this research. are we hoping for. Do you have any questions for me before we start?

02:54 PARTICIPANT 35:

Um, if I could I get the notification when that paper comes up? I'm really curious.

03:02 RESEARCHER:

Sorry, say this again?

03:04 PARTICIPANT 35:

Can I get the notification or where could I watch for that paper to come out?

03:08 RESEARCHER:

Yes, yes, I can send it to you. You don't need a notification; all you need is to let me now and I'm happy to send you a copy.

03:22 PARTICIPANT 35:

So, thank you.

03:22 RESEARCHER:

Okay, no problem. Also, if you are interested, obviously, I'm also running a workshop. At the end, once I have the findings, and the conclusions, I'm running a workshop with a few developers to get the feedback. And to validate our understanding from what we collected so far. If you like to participate, you're more than welcome, just let me know. And I'll send you the invite.

04:02 PARTICIPANT 35:

Sure, sure.

04:04 RESEARCHER:

Okay, then I will get in touch by email once I am in a position to organize the workshop. You won't have to do much talking, but just give us feedback in what we propose as conclusions with other people. So, there will be discussion with other people. So, for

the purpose of the interview is I have questions of obviously I prepared questions. But it is a flexible and fluid way of discussing. If you want to add anything, feel free, it is not a rigid way of conducting the interview. So perhaps we can start with an introduction of yourself. who you are, what do you do? And what's your experience like?

04:56 PARTICIPANT 35:

Super. So, I'm [redacted] I'm a software developer. Now, I've also taken responsibility as a Scrum master besides it.

05:10 RESEARCHER:

In addition to software development?

05:11 PARTICIPANT 35:

In addition, yes. And so, I'm developing each day. But next to it, I reserve something like an hour or an hour and a half a day, depending on the need to also take care of this responsibility. I've been working professionally for two years in the first company that I held as an example, then another half a year or the half a year in the second one where I switch teams. As a new opportunity came, so I joined [redacted] as the second company, sorry. And then switch in the middle because there was an acquisition and there was a chance to try something new. I don't know where to go right now. I could go in a million ways.

06:13 RESEARCHER:

That's sufficient. I had a look at your LinkedIn and I'm happy with your experience. And yeah, thanks for that. Let's start with some questions. What do you think of Agile? You told me in our correspondence, you use Scrum? You could also tell me what do you think of Scrum or Agile in general? And what was your experience like with these methods?

06:48 PARTICIPANT 35:

So Agile. I really like it. It helps us a lot in organizing, not only software quality, but it’s also a lot of, let's say, a lot less headaches during the development cycle. I did prepare a few examples from each team I had. I have notes written here. I can go slowly over the pages, tell me to skip, if it's not too important for you or not. Would you like me to go?

07:18 RESEARCHER:

Yeah, yeah.

07:20 PARTICIPANT 35:

So, in the first company, as some background info, it was a young medical company. It was a new team, based on an acquired firm, we were nine developers in the team. So, it also firmware. But we are full stack. Also, they had a little bit different practices, but they also were in our Scrum team, it was a bit over mix there. But most of the team was composed over fresh interests. Nobody knew Scrum, including me. So that would prove a bit of a headache in the long run. But it was sold, there was only one mid-level developer and one architect. So that was all the seniorship in the team. The PO was external. And we didn't know him until later, which also caused some issues in the start. And we will continuously on board one person every two or three months as if the team was expanding. So, it started off without Scrum. So, this is how we it was bad before it will come out how it was good after. So, we started implementing Scrum slowly step by step because they didn't want us to get overwhelmed with it. We had no project owner. And we had to build a new product without this knowledge, without his knowledge. We had no documentation, no instruments to base our decision on, we had nothing mostly just some heresy and some emails, which was hell. It was a very bad start for a project. I still to this day, don't know how it came to be. So bad management, I will say. So, the first few month’s iterations, we had four-week iterations, sprints. We did not fully implement Scrum.

09:31 PARTICIPANT 35:

We had missed commitments. No estimates yet. So, it was kind of hard to estimate how much we could complete and have a good picture of what the team speed would be from sprint to sprint. We had some missing tests or feature that will also have implemented, it was pretty chaotic. Because of the missing info, as I said, we have no idea of the requirements, we had the old program for the medical instruments, so we could just analyze it, look over it, understand some behavior, but we didn't know what we need to change in the new version. So that was a problem. Also, I say no documentation, no source code was in Spanish and written in Delphi from fifteen years ago. And all previous developers left. Just imagine our surprise as a junior team. So now how did Scrum help us. Well, first of all it helped us take care of external factors. From standing to facilitate meetings searching for people that knew the knowledge, so our Scrum Master relieved us from this problem so we could focus on our work. So instead of spending a few hours looking for people who would know about this information, sending emails all the time, or calling people, we could just let him do that. Then we could focus on testing at the end of the sprint instead of spending all that time. Another thing was we managed to have a project owner as the Scrum sense, we would have weekly meetings with him. So, we could discuss every story or requirements we had and get the confirmation that's what they wanted. This is also where some problems came up with what we already implemented because it had been a different vision of the project. So, what we did until that point, so let's say until the third space of three months after we got the review that okay, this, it was a wheel where you could see each biochemical component in the machine, and also each blood sample or urine sample, whatever it was.

12:22 PARTICIPANT 35:

And the way we implemented it; he didn't like it. He said, well, yeah, it's a bit complex, the user might need some extra info there, but some place is over there. And it, it kind of messed us up, because it also meant we had to change a bit something in the infrastructure to manage it. Also, when we started having estimations, we no longer broke commitments. At the first estimations were a hard for the bigger user stories, because we had no experience, we couldn't really estimate the big ones. But it was much better anyway. So, we could say, okay, this sprint, we have, let's say, twenty points as a team, we can use as we knew how much we could commit. So, we have no longer broken stories at the end of a sprint. Less bugs. Yeah, not as many problems. Regarding quality, as today to the estimate, day to day basis, we had during our attendee's screen in the meeting room will also always have statistics on broken builds. And bugs, bug tracking. So, each day, we'll look on those and see for the role that are lowered. And based on that we could also like take quick action during the sprint to see if we need to change anything. Or create a special task to solve them. Because one second, I want to see where I wanted to go with this. Okay, let's get this, I already talked about this. Oh, retrospectives. That was a big help. As we didn't do them initially. We would always complain between us every day about an issue. But nobody took any action because it was nobody's responsibility. It was just between developers and also something as big to take to the management or anything. So, it's okay. That's how it works. Well, it doesn't. So, once we started to have these retrospective, sorry, yeah?

14:58 RESEARCHER:

I'll stop you there. This is not a perfect implementation of Scrum, obviously. It has a lot of problems.

15:06 PARTICIPANT 35:

Yes, it was not.

15:07 RESEARCHER:

Do you think it still helped you to deliver software quality?

15:12 PARTICIPANT 35:

Yes, yes.

15:14 RESEARCHER:

And how?

15:18 PARTICIPANT 35:

So, because we always keep a watch on the bugs. So, and discuss them also, because we were discussing between us, but it wouldn't reach the manager or the PO because the PO also had a say on how we took priority in our tasks. And because our Scrum Master slash manager, it was the same person, he couldn't also prioritize our work. So, because of the daily meetings where we talked about the growing or lowering of the number of bugs, and the retrospectives where we would look at all the, all the sprints in length, and see or observe the issues that came up between us and talk with him also. Because, okay, let's say I had a bug, I saw it, I had an issue. So it was, but I forgot about it, but then another person or two, saw the same thing, but didn't talk about because they saw it only once. We will sit at the retrospective discuss every little issue we had during the sprint levels of observe patterns, we take note of this pattern, feedback the action items, and we'll go to solve them. So, an issue was that we from time to time, we need to for example, we need to instruments, medical instruments, and we had no instruments. We had intermittent bugs, so we just wouldn't observe them until the end. So, when we get that we say okay, so that bug was also in there. So, see saw him so we that's the problem, because it always fails. And we will lose time on it. And it also might hide some other serious issues behind it.

17:07 RESEARCHER:

So let me recap and get more details from you. Despite it's not a perfect implementation of Scrum, what I got from your, from your description of this particular experience is it created a highly collaborative environments. And this collaborative environment allowed you to share knowledge and to share bugs amongst the team. And that's facilitated resolving these bugs, right?

17:48 PARTICIPANT 35:

Yes.

17:49 RESEARCHER:

Okay. So, now you're working in a different Scrum team, right?

17:54 PARTICIPANT 35:

Yes.

17:55 RESEARCHER:

Are things better now?

17:58 PARTICIPANT 35:

So, in the first team, I joined, so I tried, I went into more after. So, the first one I joined this company, it was, it was going very smoothly. So, they are doing Scrum for a few years. I joined later. So, for me to integrate into the team, it was very easy. So, for the integration purposes, it was easy. Also, the estimation for it, for me to estimate they had good examples, and baseline set. So, I could so easily estimate my work and say, okay, I need to take this much time for me to do it. And I will not have to sacrifice documentation and test at the end to meet the deadline. So that was extremely helpful. Daily meetings were very short, because when they were planning these stories, they would, at the planning session, I mean, they think about it, how they could rewrite them in such way they had minimal interaction, for a few reasons, once that they wouldn't lose time, with bureaucracy to see who's doing what, who's also doing that, then also not have more issues in code. And also, let's say alleviate issues that would come from this impossible path. So, if two people would work on the same code path, they wanted to not have the issue where they would come with two different philosophies on it. And once they merged it, it will cause bugs and other problems. The management of the PO was composed of also ex developers now. Also, they had combined experience, maybe ten, fifteen years ago. That helped the team a lot to prioritize issues. So, when the team had an issue, they knew that had to be fixed. It was like okay, it's a bug. Continue to develop, deliver the product. No. The team had an issue that needed to be solved.

20:22 RESEARCHER:

Can I follow up in a few things?

20:24 PARTICIPANT 35:

Yes, sure.

20:25 RESEARCHER:

So, the first thing you mentioned that you did the estimation yourself for the user stories, and you said it was very helpful. Can you elaborate a little bit how it was helpful and how it did help you achieve quality?

20:46 PARTICIPANT 35:

Um, so by doing the estimation, for my work, right, that's what you want to ask. It helped me. Okay. Let's see, if somebody else did it for me. He would have to guess how hard that user story would look for me. And, and also guess, how much time I needed to also work the issue completed documentation, write the test. So that's what is needed to complete the work. And that could be a mess. He could overestimate, which would end up in needing too

much time and I could do more work. But I wouldn’t take it like, because I could add, I would have to take another user story, but I might not have time to finish it. If we he was underestimated. I have to, I could sacrifice some of the quality of my code to meet the deadline if I couldn't extend it. When I estimate my own work, I always make sure I have time for quality.

21:53 RESEARCHER:

That's a perfect example. Yeah, thank you. Yeah, continue, sorry. I interrupted you. Yeah.

22:01 PARTICIPANT 35:

No problem. Yeah, by allowing me to make the estimation myself I could say, okay, for you, it might be five points. But if I am to develop this, like, for me, it's let's say seven. Because I know how fast I can work on I know, my experience level. And also, the metrics that are used to develop the numbers in the Scrum estimation are pretty also important. So sometimes, it is useful to take also the experience of the team in account, not only the complexity of the code, because if the team is more junior, a big chunk of the time spent on the issue will be also on documentation and learning. And if you estimate only for complexity, you will end up with an issue where the points would be less. The estimation, you'd say, could do a lot more than just paint your over assign the user stories or requirements. And you'll end up in a place where they would rush every piece of code, sacrifice quality and other parts this what happened to the previous company, because they also had to study and learn. So that was, that could be a problem.

23:27 RESEARCHER:

Yeah, that's, that's a very good example. I'm going to ask another follow up questions. I've noticed when they ask you to estimate your own work, they give you a little bit of control and a little bit of empowerment, how it makes you feel?

23:49 PARTICIPANT 35:

Good, of course. Well, it gave me a sense of control over my work.

23:59 RESEARCHER:

When you have a sense of control over your work, do you invest better in writing good code?

24:06 PARTICIPANT 35:

Oh, well, yes, I see it as my responsibility better. So, when I know that I own something, it's my name on it. So, if something goes wrong, I will be the one which will have to face the consequences. If somebody says like, okay, it's my work, but you can help me with this little part, okay, let's help you with that. But it's still your responsibility. It's not my mine. You said you need help with that. So, it technically is the same thing. But mentally it is different.

24:38 RESEARCHER:

So, this accountability, you're talked about, does it motivate you to write better code?

24:45 PARTICIPANT 35:

Oh, yes, it does.

24:47 RESEARCHER:

Yeah. Okay, I like to follow up in another thing, you said that the meeting is short, and there is less bureaucracy in the process. Can you describe to me or give me an example of how less bureaucracy has helped the teams to achieve better quality?

25:12 PARTICIPANT 35:

Going back to the older company. Where the Scrum was mixed. So again, at the start, we had a lot of, wasted a lot of time, each one of us sometimes on the same thing. by searching for people to help us understand the feature we have to implement. And it was very hard to find someone. All of us had some shared knowledge about what person would know which area of the code but this wasn't always discussed. So, we will some time send similar emails to similar people everywhere. And it caused us to lose a lot of time. And that was time was accounted in, again, missing documentation, missing test cases. It had to come out, that was the point. After we delegate this responsibility to the Scrum Master, so he would stop all the external interference from the team that helped us relieve us of that issue. So, no more wasted time on that. To take care of it. He would get the information faster, because he already knew that all the persons. It was no longer shared knowledge; it was concentrated in one place. So, he knew that people would have to contact directly. So not only it, it went faster, the quality improved a bit because now we had prompt responses and wouldn't have to wait sometimes even after we finish the user story to rewrite it again, if there were misunderstandings there. So, we could adjust our tasks during the sprint to cover the story as it was meant to be written. Does that answer your question?

27:23 RESEARCHER:

Yeah, it does. Thank you. I need to go back to my questions. And we've been talking about quality and how do you define quality or software quality in the context of Agile or Scrum software development?

27:45 PARTICIPANT 35:

Oh, this one is a bit of a doozy. It's going to have to wrap my head around. I don't know how I could ask for an example without getting biased.

28:01 RESEARCHER:

No, I mean, it's your perspective. I mean, how do you define it?

28:09 PARTICIPANT 35:

Well, general quality, we define it by our guidelines, or public methods must be commented. So that's the public API of any class package. So, those need to be commented, we need a software design documents for any bigger feature we implement. And we also, what else, we have a zero-bug policy preferably we try to stick too much as long as much as possible at the end of the sprint. So, it will be these three things. Also, unit test and end to end if possible. We account for all of these using the estimations. So, let's say it encompasses them.

29:17 RESEARCHER:

Okay. Thank you and you mentioned free of bugs, but how about the design and the code itself? Is it part of quality?

29:29 PARTICIPANT 35:

Yes, it is. So, during the STD, we have an estimation session. I see sorry, for the STDs we have an estimation session. So once the sprint at the beginning of it, it's next to the planning usually, we have another meeting where we gather all the team and discuss what solution was proposed. What architecture was proposed, what patterns were used and so on, so forth. So, we discussed the design in depth before going further. Also, before it goes into implementation and after our team looked over it is also reviewed by more senior staff in all teams around us, because in the first team I was it was part of a bigger software, each change we had to do would also impact other things. So, it was it was pretty, a group choice.

30:45 RESEARCHER:

Hello. Can you hear me?

30:53 PARTICIPANT 35:

Yes.

30:53 RESEARCHER:

Okay, fantastic. You froze for a second or something. That brings me to the next questions. You mentioned code review, and which is a quality assurance practice. What else do you do to assure quality in a Scrum environment?

31:15 PARTICIPANT 35:

I lost you for a second. I mentioned code reviews.

31:19 RESEARCHER:

And yeah, code review is a quality assurance practice. What else do you do to assure quality in a Scrum team?

31:34 PARTICIPANT 35:

Reviews mostly.

31:37 RESEARCHER:

Do you have automation testing?

31:41 PARTICIPANT 35:

Oh, yes, of course. We have gated builds. We have automation, we have for each pull request is built automatically on Azure. And we run integration tests before it's merged into the master branch.

32:01 RESEARCHER:

And do you have testers in the team, do you have QAs?

32:04 PARTICIPANT 35:

No. We have we are trying to not have specialized testers. Each developer is responsible for his code and the test that they run on it.

32:16 RESEARCHER:

Who does the end user testing? The developers as well?

32:24 PARTICIPANT 35:

I lost you then.

32:24 RESEARCHER:

Who does the end user testing?

32:29 PARTICIPANT 35:

We do.

32:29 RESEARCHER:

You do? Okay. Okay. Yeah, that's, that's quite interesting.

32:35 PARTICIPANT 35:

So yeah, we trying not to hire QA, as a company. The reason is that, first, for unit testing and integration tests, they have no, no role there. And for the user testing, we could test those features much faster, because we know what was added. And we know which bad interaction we could have with other solutions, with other pieces of code, or other reviews, or what not. And each let's say we had three sprints with four releases a year, each third sprint, we would have an extra week, where all the teams or development teams would not implement, but spend time using the application. So, we will use the application we will try will play with it, we will check if the installers are written correctly, we will install deploy by hand as a client, we will just, we would in the client's position, we will try to look through their eyes. And we did play with it, and we will discover a lot of small bugs that might not be seen

easily. Because we also knew what should be there and what should not. Let's say if there was a value of two instead of a three, the client might not see it, but we knew it shouldn't be like that, because we implemented it. So, it did increase the quality. I'm not sure if how much compared to QA because we had no QA. We were the QA. It did increase it because we work closer.

34:50 RESEARCHER:

Scrum or Agile doesn't define a QA within the team. It has this generic roles which are developers. Yeah, that some companies consider QA as part of the developer's team. That's not a concern. My next question is you talked about testing, and you talked about code review. And those are practices in a software development team that existed before Scrum or Agile. So, what do you think this Scrum setup that you've been describing in this company has helped to produce quality? And how did it make a difference?

35:37 PARTICIPANT 35:

It's hard to compare not knowing the thing before Scrum. That would be an issue. It's how would you compare it? If you are new, how could I compare it to something before?

35:56 RESEARCHER:

So, let me rephrase the question. What's good about Scrum that helps producing quality?

36:11 PARTICIPANT 35:

Let's say, the set meetings that the Scrum proposes, so the ceremonies helped a lot because they would be repeated. People would know what to expect from them. It was this pattern, this I don't know how to call it, the routine, because the routine, people get used to routine. When you do not have this set meetings, people like don't focus on one thing, they discuss whatever comes to mind, but when you have a routine, say, okay, the retrospective, we discuss what issues we had. People will think of the issues all the time, they will stay and think of it, they will reserve time, if you have normal meetings, let's say that happen randomly. They'll just say whatever comes to mind, they will not say anything, they will not analyze stuff. It's not reserved time, as part of their time. You know, the random meetings are not a team event. It's an individual event that takes their time out for from implementing the features. Their test, their documentation, when you have these meeting patterns, they know that they have to take time out anyway. So, they concentrate better on it. They, they start thinking, okay, what issue we had? Why did it happen? How what can we do to solve them? Because they're not rushed anymore. They do not rush to finish it. They know they have to finish anyway. So, they just stay and enjoy it a bit more.

38:04 RESEARCHER:

So, I'm just going to recap what you said. So, these Scrum ceremonies facilitate a better communication to share errors, right?

38:15 PARTICIPANT 35:

Yes, to share problems that appear, any kind of problems. Be it code or processes.

38:21 RESEARCHER:

Yeah. So how does this quality of Scrum helps improving quality?

38:32 PARTICIPANT 35:

I don't know how to answer that.

38:40 RESEARCHER:

Do you have an example? You could share with me an example?

38:46 PARTICIPANT 35:

Simply that... it's easier to, so during the sprint okay...

39:11 RESEARCHER:

It's okay. It's okay.

39:12 PARTICIPANT 35:

No, I didn't know what's happened. I didn't know she was behind me. So, for this discussion, the Scrum ceremonies, they do not help directly, it's indirect help. So, because we discussed so often in the ceremonies, so they mostly base and we have the one set that is retrospective. So, if we work on our own. Let me think of an example. We had some very obsolete code. We had an obsolete code for reading it was called the rules. These rules would watch for data coming In. And if something happened that match the rule, it would execute some other code. The logic was very complex, extremely complex and it was slow. Our story was to improve the performance, which was a headache. And in this, we had a huge problem, because it was always expanding the scope. Always. Luckily, because of the daily meetings, we could stay a bit easier on top of it. We were multiple people working on it. And each of us had always new tasks, added new problems that arise. And we had the usually fifteen minutes, morning meetings during the day but it could expand into a full hour. And this helped us a lot because we had the set time where you could discuss all these problems and what the entire one day entailed. Why did this help us? Because we could talk as a team to see how we want to handle it. And we would know what future problems it might pose for us. So, we could account for it in time, if we didn't have these meetings, then we will wait for it to become an issue, it would have a much bigger impact on our code as a whole. So even if daily's do not have a direct impact on the quality, they help a lot in again, seeing repeat offenders and discussing things. You discuss things as they come as they go, because you the daily meetings happen at a very short interval of time, let's say it's eight hours. So, there are a lot of issues that can come together in that amount of time. So, it's a lot easier to see problems that appear. It if anything comes up, you will easily see in the time of the meeting. So, from a quarter of an hour, which everybody said what they did and what they will do, it will extend, I had this issue, he had that issue, we had that issue. And it's easy to identify and resolve. Very easy. And again, all the issues at the end of the run. It's in their account in time that is used for bug fixing and documentation. That's what usually sacrificed, that's the quality I'm referring to, always. And as an extension to testing, quality also, the code might be a little bit worse, because it was not tested properly, and might have bugs that escape. So, it helps a lot with time management and an issue management how we prioritize it, which extends in everything I thought before.

43:43 RESEARCHER:

Okay, that was a good example. Thank you. I will get into more specific questions. How does for example Scrum helps finding bugs?

43:54 PARTICIPANT 35:

That's a bit hard. Okay. I know a good example. We had our meeting with a PO. It was, not the retrospective. It was a demo. At the end of the sprint. We implemented the feature, how we saw it in the user story and the requirement. We had all the tests we wanted to have, so no bugs there. We had all the documentation. No issue there. We had plenty of time to do it. So, we could say the feature was perfect from our perspective. We had no idea. We had the demo meeting with a PO. We showed her what we did. And she was a bit confused. To say the least. Why does those rules come in that order? Well because that's what we understood from the previous code and from what was written in the story. Well, yes, but there was a BUT. As it turns out, there was a small requirement that you could summarize it in a line of text that was not written there. Which if we didn't have the demo meeting would escape in production. So that's an example of how these demos help and it had a certain naming ceremony.

45:53 RESEARCHER:

It's called the client demo or the demo. That's the right word. I have seen it in Scrum teams. They do the client demo or to the product owner demo. It calls the demo. Yeah. Most, most Scrum team call it demo.

46:12 PARTICIPANT 35:

Yeah. So yeah, that's how it helped us.

46:17 RESEARCHER:

That's a good example. Thanks for sharing that. I need another examples. How does for example, Scrum, from your previous experience or the current experience, helps producing high quality code.

46:35 PARTICIPANT 35:

Sorry, helps producing.

46:40 RESEARCHER:

Helps producing high quality code. Should I repeat the question? No?

46:46 PARTICIPANT 35:

How does Scrum help produce high quality code? The answer would be the same.

47:08 RESEARCHER:

Okay.

47:18 PARTICIPANT 35:

Again, depends on what would be your definition for high quality code?

47:22 RESEARCHER:

You the developer, you should tell me? What do you consider high quality code? I can tell you, my perspective. I can tell you my perspective, a high-quality code is readable. It's clean code. It's scalable. It takes in consideration a future audience of the code. And it's also maintainable. And obviously it's free of bugs, and it's compliant with the team standards. That's what I would consider high quality code.

48:02 PARTICIPANT 35:

Okay, now got it. Yes, we tried to do that thing as well. An example for that I would have would be, again, from retrospective. We started to see to have, be sometimes confused. So, it was where we didn't really implement Scrum properly until a certain point in the future. We would sometimes miss documentation in some places, we would not abstract classes properly, we will not put the interface where we should. And we started to get issues because of this, but they weren't really observable. They were more of a gut feeling. We didn't really get into problems with it. So, during the retrospective, I know that someone said this exact thing. They said like okay, I remember I had the issue. I don't know what some methods will do because they were missing documentation in a few places. It just skipped past the review. Somebody took some of them. And I had issues with coming up with another component that would be used in a strategy pattern. They want to implement a strategy over a certain component. It wasn't abstract that there was no abstraction there yet. And okay, like we said, like, okay, let's look over the code. So, we took like, half an hour to look over the code. And we saw together as a team, there were a few missing places where there was no documentation, or the abstraction was wrong. Also because of that happening, all of us started thinking more about it in on the moment and said, okay, I also remember I saw this line somewhere in that package. So, we also took that package to look at it, and so forth and in the end, I think we stayed three hours in the retrospective to find all the possible issues that occurred similar to this. So, from the quality, extra quality perspective. That's how it helped.

50:41 RESEARCHER:

That's a very good example. Thank you. I like that example. Yeah, I like it very much. And I liked how you guys collaboratively shared the knowledge and collaboratively managed to break down the complexity of the code and identify the issue. And I think that's one of the qualities of Scrum brings to a team. Yeah, I agree with you. I have a last question. And I also have a follow up questions. My last question, how does a Scrum environment from your previous experience and your current experience motivates you to achieve code quality?

51:26 PARTICIPANT 35:

Okay. Ownership. That's how I would summarize it. Ownership. That's it. That's the motivation. People feel more responsibility if they own it.

51:42 RESEARCHER:

And when they are more responsible, they write better code in your opinion?

51:46 PARTICIPANT 35:

Yes, yes. Because they worry more about it. They say it's my risk. It's my face on it. You know, if something goes wrong, it will be my name on it, it's not like, another issue for some other time. You know, it represents me as a developer. Do I want to look like that? Do I want to work like that? You start thinking about it more. It's not just some other code you just finish and come later and say, okay, I need to finish that again. And it will be solved in the end. No. It says like a robot in the corner. It says it's a lot of issues listing down that you would feel bad about it. It's like you wouldn't make a mess in your home. You wouldn't allow to trash your house. Is the same for the stories, you don't want to make a mess in them, in the code, in the documentation, anything that has your name on it.

52:51 RESEARCHER:

Okay. Next questions. You talked earlier about Agile, or Scrum help you to organize. Can you elaborate a little bit on that? And give me some examples if possible?

53:15 PARTICIPANT 35:

Okay. Ah, well, it's the first, it will like the classical Scrum definition, the planning, it's purely organizational, it helps a lot, just to see what we want to do the sprint. So, I don't think I can add anything that it's like, this is the perfect concept for it. The daily meetings will help us organize because we would always start to see if this interferes between us. So, we had, for example, we were working on a UI piece of code in Angular that depended on the same service. We didn't notice it at the start. Because we didn't know that functionality didn't exist also on the back end. So, the story was only front end, it was estimated for front end. The code that was important, a long time on the back end for that area, and we thought it was already implemented. So, when we started analyzing and working on it, after finishing the front-end part, we notice that during a daily meeting, we said okay, I finish that, somebody says the same, the same thing on the front end, so his side of it. And we said okay, I'm going to do it was a code for finding an instrument that, what was he doing, I think was similar to like, it was busy with something. So, it was logging things, it was busy with something. I want to see what it was busy with and what it was logging. And he said, he will do the backend side, and somebody also said he will do the same thing. So, from an organizing point of view, that was a really, it was a clash of interests, both people want to do the same thing. They had a bit different opinions on how to do it. So, because of the daily meeting, we had, we all took something like half an hour after it to discuss it properly. So, we managed to notice it fast as the moment occurred. And we didn't have to backtrack on it, spend time on rewriting code or merging code or any other issue that would arise from that process. Is this a fine example?

55:55 RESEARCHER:

Yeah, that's a good example. So, the teams and the closeness, I'm just recapping. So, the closeness of the teams and highly collaborative environments that Scrum brings in, helps you to, to manage conflicts, helps you to manage the work together, helps you to communicate. Yeah, that's what you meant by organizing better, correct?

56:22 PARTICIPANT 35:

Yes, yes.

56:26 RESEARCHER:

Okay, we come to an end. It's been an hour and I enjoyed the conversation with you very much. I like the examples. They were very good. Do you have any things you want to add to the topic how Scrum helps quality?

56:41 PARTICIPANT 35:

As the top of my head, I don't know. It would be easier for me for I have like specific questions was that, like, if you want, I am not against staying a bit more. I have time. I don’t mind. I'm open to stay a bit more if you want, if you have any curiosities.

57:08 RESEARCHER:

Okay, yeah. Do you have examples or things you want to share with me? I'm happy. Yeah.

57:15 PARTICIPANT 35:

If you ask me for a specific example, or interested in or a specific issue, you want to find the answer to I will try to find example.

57:25 RESEARCHER:

Whatever examples you have, I don't have, whatever examples you want to share with me, that helps the quality, I'm happy to listen.

57:38 PARTICIPANT 35:

Let me look in my notes to see if I have anything else. Okay, so [inaudible]. Another example for the retrospective that was helping us a lot was we were struggling a lot with, so this would be regarding the also the speed of the team, the development speed, or how the first validation of it. We started observing that our pipeline was slow. So, during this, they will just do pull requests to the master or the main branch. And we have, we have multiple pipelines. So, most of them were not observed. So, most of them are okay. But sometimes we had one certain, one on a bigger chunk of code that had to be compiled together. That was slower. We didn't give it much attention, because we didn't push code that often there. But there came a sprint, which had a feature directly linked to that area. What happened there is that it started, let's say slowing us down a lot. We will have to wait something like two to three hours for it to build and test, which for us is unacceptable. We want to wait maximum twenty minutes for the whole feature, maximum that will be the slowest we have. And we are always looking at this, so while developing, we're always checking them because we don't know to leave the apps hanging in the back in there. We want to close them as soon as possible so we can focus on other issues. Because this is another problem that we try to avoid. We do not want to context switch too often. Because it's confusing. It takes time to rebalance yourself. So, we discussed this again at the retrospective and we thought about how we would solve this problem because we couldn't edit the pipeline, it was enough in our responsibility. And it was actively slowing us down. So, what we did, we talk to the management and the DevOps team. As an action item that we created, that we take ownership of that pipeline, and we write it ourselves. And we managed to squeeze it down from two and something hours to twenty minutes. That's the slowest one I was talking about, which was a big improvement. And it helped us a lot for that specific issue.

1:01:38 RESEARCHER:

So, the ownership that management has given you, motivate you to resolve the problem, and subsequently better quality. That's what my understanding, right?

1:01:52 PARTICIPANT 35:

Correct! Not only don't rush it, but the prospect of if you need it, you can take it. Because we want it to have the pipeline, so we can change it and own it to be faster. And they gave it to us. So, owning that. And also observing between time and talking about the retrospective helped us solve it. And in turn, having that run faster, although help us like see if we have bugs or build errors. So, we can solve them faster. This will be especially crucial at the end of the sprint when release would follow. Because we wouldn't have to wait for two hours on each pull request to see if there are other bugs that need to be solved.

1:02:44 RESEARCHER:

That's another good example. Thank you. I need to go, and I don't want to consume more time, you're being generous.

1:02:56 PARTICIPANT 35:

I hope this helps you.

1:02:56 RESEARCHER:

Yeah, it did. It's really very good examples. Thank you very much. I will get in touch by email once I have a date for the workshop, if you still want to participate. So, in the workshop, you won't have to talk much, just give a feedback on our understanding of how things work. Okay. I wish you good day, and I hope today won't be that hard.

1:03:24 PARTICIPANT 35:

Thank you. Thank you. I hope too.

1:03:26 RESEARCHER:

Bye. Thank you.

1:03:28 PARTICIPANT 35:

Have a good day, bye-bye.