**Interview 5**

**Interviewer:**

I'll just start recording to cloud and since we are having this interview together, I'll jump into immediately to the first part of it. I have tried to discover some software practices, startup software practices and I would like you to at least try to map some of these practices with the different quadrants here for instance, let's take the first one and make an example. If you say limited documentation, where do you see it more? Don't write a lot of documents about the source code or anything else. But just focus on the coding in general, in principle. So, do you see this limited documentation in any of these quadrants, for instance, the one on all if a startup or from growth startup with multidisciplinary or software engineering team.

**Interviewee:**

I think that documentation is something that you should limit always. But if you force me to choose.

**Interviewer:**

I'm not forcing you.

**Interviewee:**

So, it will smack in the middle of all of them because the whole idea with multi-disciplinary teams is that within the team, you maintain the knowledge, without documentation rather than with documentation. So, I think that that's why I think it's so important to do what you can to decrease the amount of documentation that you generate within the company. Now, having said that, I typically see that when a company starts to really scale, not a little but really becomes into the, the dozens or hundreds of people, then often there is some need to capture some things in documentation. But in my experience, even then, there are a number of walking libraries in the organization where, which are people that basically a quote, unquote, educate others on the rationale behind design decisions, what they know about the customer and those kinds of things. So, I think in general, we should just be very careful with document spending time and documentation because that time, you cannot spend on other things and if there's one thing that startups run into, it is lack of resources.

**Interviewer:**

This is what I discovered. From the readable code, where do you see it?

**Interviewee:**

There, it becomes more important on the scaling side, rather than on the early stage startups. So, the way that I often talk about it is that you have a nail in stage and then you have a scale it stage. During the nail it stage, you're basically experimenting with whatever it is that what works for customers and at that point, readable code is not important. The only thing is that you can actually learn as quickly as possible from what works for the customers and once you find that out and you start to scale it then architecture code quality, and those kinds of things become an issue. Up to that point, just functional, approximate functional correctness is more than sufficient.

**Interviewer:**

But just to put this into perspective, with not only the startup also with the team, type, should it be a multidisciplinary, or software engineering team.

**Interviewee:**

I think you had it on the line, because, it's your research. So, it's up to you. But I have never seen any software engineering only team be successful, ever.

**Interviewer:**

Okay. But to put some context to this, thank you for saying that. We do software engineering courses, and we only have software engineering guys in the team.

**Interviewee:**

The point what will happen is that some of the students will take more of the sales and business side perspective, some will take more of the user perspective, and some will take more of the technology perspective. So, what you'll see is that even though you have software engineering students, they will still gravitate towards business, customer and technology. That's at least my prediction. The fact that I currently do not teach doesn't mean that I haven't been teaching and I do not have opinions about teaching, I definitely do. But I just haven't done any teaching the last 10 years.

**Interviewer:**

Modular programming for instance, or before I jump to that, agile practices, do you perceive those imply increased productivity in general?

**Interviewee:**

Yeah, they are necessary but not sufficient. So, do you want to say in general, or you are looking for the startup space?

**Interviewer:**

Startup space.

**Interviewee:**

The biggest challenge you have, and every company that I know falls in that trap is building something for which there is no customer. So, I often, quote, Peter Drucker, the big management consultants, the goal of a company is to create a customer and therefore, I have never run into a company that failed because of technical reasons. But I've had several cases that failed because they were unable to create a customer and therefore, an agile is just the normal practice. This morning, I was in a meeting with someone from Ericsson, what do they do? They do DevOps now, which is agile plus continuous deployment. I just came out of another meeting with Volvo, what do they do? They work with agile practices and continuous integration tests moving towards DevOps. So, in large and small companies, agile is just the current best practice.

**Interviewer:**

Should I keep it towards one and two, or in early phase, growth phase or with multidisciplinary team because you somehow excluded this pure software engineering?

**Interviewee:**

Yeah. I think it should be on top of limited documentation. Sorry for being so annoying. This is a generic practice, everyone needs to do this and you do realize that we only booked half an hour, right?

**Interviewer:**

Yeah. So, I'll go fast. Hope project management practices, where do you see them fit?

**Interviewee:**

Early stage on the line.

**Interviewer:**

Protecting code with private repositories? I know it's general. But this matches up with no patenting or IP rights.

**Interviewee:**

I think, for a startup patents are completely useless in my experience. The reason for this is that any large company that wants to kill you will basically have so many patents in their portfolio that they can pull out 100 patents that they claim you are violating for every one patent that you can bring out to the table.

**Interviewer:**

Yeah. Good and protecting code, like private repositories? Is that something like a good solution? In your opinion?

**Interviewee:**

Yeah, I developed something called the three-layer product model. I'm not sure if you heard of that at some point. But basically, they talk about innovative functionality, differentiating functionality and commodity functionality. I think that as long as something is innovative and differentiating, and you should try to protect it in certain ways and therefore, yes, you should have it in a private repository. But I don't care if it's in your private GitHub account, or if it's somewhere else, it just should not be put in the public domain with a license, or whatever associated with it.

**Interviewer:**

Between code and pair programming, because these two go together, code smells, where do you see them most often?

**Interviewee:**

At the very end of the skills. Where you see them, or where they become a concern?

**Interviewer:**

Yeah. Well, where do you see them the most? Where do they appear most?

**Interviewee:**

They are introduced in the early stage startup, because then you're just trying to hack something together without worrying about where they come from. But they become a concern when you start to scale.

**Interviewer:**

How about pair programming? I know it's coming from an end for practice.

**Interviewee:**

Yeah. I don't have any opinion there. I don't have any experience. Well, I do have some experience, but I don't have strong opinions one way or the other concerning that.

**Interviewer:**

Yeah, that's fine. Then when it comes to testing, automated unit testing and integrated testing, when do you see it more?

**Interviewee:**

At the very end of the scaling stage, that becomes important.

**Interviewer:**

Then the acceptance testing or customer validation.

**Interviewee:**

That is valid everywhere. Again, back to my earlier point, my biggest concern is startups. not creating a customer, they implement some nice technical assets and this I think will be the biggest challenge in your course. Because where are your customers going to come from and how can these people behave as if they are real startups?

**Interviewer:**

Yes, those are good points. But I have a plan. I know it's not yours. So, I don't want to go one by one. But these are coming mainly from technical depth concerns, such as modular programming, flexible architecture, code reviews, low coupling, technology and tools and flexibility. So, where do you see them fit?

**Interviewee:**

Beyond the scaling stage.

**Interviewer:**

Beyond the scaling stage. So, these are all related to technical depth in general.

**Interviewee:**

So, here's my thing. You first find something that customers actually wants, then you scale your markets, then you start to mature your markets. No, no, no, I want to hold books that you have now selected beyond the arrow. Because it is only once you start, your growth is starting to slow down, then these things become important, as long and this is something that I learned when I worked with a Dutch startup. They were growing so fast, that any hour not invested in scaling was costing the money and it was much better, to suffer through low code quality, high degrees of coupling and all kinds of other hassles, just to drive the scaling, because scaling was the only thing that mattered and only when growth started to flatten, they started to look into cleaning up the codebase and decreasing the inefficiencies and all this other good stuff.

**Interviewer:**

Interesting point, I didn't expect that but obviously with you, it should be some surprises. Now we have some tools and frameworks and approaches whatever you want to call them that we have been using and this has been introduced through an innovation bootcamp, just to be honest. So, do you see this with cross functional teams, this is the most important or with software engineering teams.

**Interviewee:**

I don't believe in pure software engineering teams, because I have never seen the startups that can win justice on technology. So, lean business Canvas model goes into the beginning of the early stages and multidisciplinary. Kanban is a generic technology that can be used anywhere. Growth, hacking goes into the growth, Diego goes into number two. Design thinking goes at the beginning of the early stage in number I think is box one. So, now I have to start looking MVP with low MVP with low code. Early stage of cross functional, startup pressure is everywhere, sustainable growth, I don't know what you mean by that.

**Interviewer:**

Yeah, it's the growth phase with some degree of sustainability, even if you want to or whatever.

**Interviewee:**

Customer journeys is in the scaling stage. Because then part of scaling is to try to go through the end to end scaling problem, try to figure out, where do I lose my customers pitching those early stages of early of the earliest of the early stages and multidisciplinary. Retrospective meetings go everywhere. Team development is important after the growth phase.

**Interviewer:**

Okay. In the early phase?

**Interviewee:**

No retrospectives are everywhere. I mean, you need to constantly reflect.

**Interviewer:**

Persona, is that important?

**Interviewee:**

I don't know what the Lean personas,

**Interviewer:**

That is something taken from the Lean approaches, but it's fine. We don't have to move everything here. Cynefin model, if you have encountered that.

**Interviewee:**

Yeah, I've heard the name, but I actually haven't looked at it in any detail. So, I don't want to comment on this.

**Interviewer:**

Then the filo code, for instance.

**Interviewee:**

With low code, that goes to the very beginning of the early stage stuff and box one.

**Interviewer:**

Scrum XP, I guess they want to do a certain way.

**Interviewee:**

That's everywhere.

**Interviewer:**

So, this is towards the end now doing it quite fast and productive. So, apart from innovation boot camp innovation workshop, hackathon or something else, what do you feel is best and where did you see them? In this case, they were in an industry context.

**Interviewee:**

Hackathon is in the beginning, before the early stage before we even start, and then they become relevant again, after the scaling stage. Because then what you're trying to do is find the next innovations and build on your successful startup exists. Innovation, bootcamp, is the same for me in the same place as the hackathons. Innovation workshop is typically in option number two, because then you're trying to innovate in order to accelerate your growth and yeah, the biggest one that I'm missing in, what you're actually presenting so far, is where is the customer and how do I work with the customer? How do I get continuous customer feedback? How do I work data driven practices?

**Interviewer:**

That's a good point. Actually, these activities, bootcamp, is the one that is bred stakeholders, external stakeholders that present their challenges and their potential customers and the students. The students are supposed to find solutions for those challenges.

**Interviewee:**

But now we're mixing two things. One is, how should you set up your course and the second one is, how do startups work in practice and in startups and practice, it is finding those first early customers that are willing to work with you, making sure that they don't hijack you, transitioning from working with the customer and their qualitative feedback to working with the customer, and the quantification of their feedback. So basically, the database and if you want to have a distinguishing factor here in the early stage, I typically work with qualitative data and in the scaling stages, I often work much more with quantitative data and that bits in startups, very important. But to be very honest, if you're doing a course and your customer, your students are starting from scratch, they will never reach the scaling stage. So, you will should only focus. In practice, they will only get to work, experience the early stage stuff and not the late stage. That's at least my prediction.

**Interviewer:**

Yeah. The idea here is that, of course, within one court setting, it's that difficult. But when it comes to some of the practices that we see in software engineering, in general, you put all these bunch of things here, you put it here and decide it is beyond the growth phase, but we want to be aware of this thing.

**Interviewee:**

That's one of the reasons I don't teach because I prefer to teach when the need is there and not years before they use that knowledge, which I'm sure that you realize that they may run into the need for that knowledge. But currently, they are so far from it that often it is hard to motivate students to spend time.

**Interviewer:**

It's an opinion, so, it's important to listen.

**Interviewee:**

I have lots of opinions. I guess if you want more, I can share more opinions.

**Interviewer:**

Of course, but you have little time. This goes to my first question here that is part of the ending phase of the discussion. On which of these quadrants 1,2,3,4, should education focus more? I think we were entering this discussion.

**Interviewee:**

One and the reason is that in my experience, the hardest part that most people, students and professionals struggle with this to get going with something. Once you go in, you can often figure it out and getting going is by far the hardest part and having tools to go from an absolute gold star to having concrete actions and ways to take something forward is for me the most important.

**Interviewer:**

Since you mentioned tools and other things and methods and so on, when should we adopt this tool? Because, of course, we can focus on one, but what kinds of tools should we use? In the early phase or something else?

**Interviewee:**

I think it's early stage. But the biggest problem is often not the tools. It is the in ability to get off yourself, and to go and sit down and talk to customers. That's where I see everyone sits in a little Ivory Tower, having all opinions about the world, and about what customers should want and what the customer do want but it is the actually going out to customers and deeply understanding their reality. In my experience, the biggest challenge, the most successful startups that I'm involved in is people that have deep domain knowledge in a particular field, and then decide to go do something in that field, to start a company in that field. That is what works in practice.

**Interviewer:**

Okay. Again, sorry if I insist on this, the point is that when I read all these researches about early phase, they mentioned a lot of things that don't happen. But we move a bit more forward and you see the growth phase and beyond, then you see some things appear like, I see this startup pressure, combat boards, IP protection, agile practices, and so on. These are practices that mature more in the world when they grow, rather than when they are in very early phase. So, you never see a die, or almost never see agile in early phase. I'm not saying never. But in most cases, you'll not see. Limited implementations, they might not understand, retrospective meetings, and so on. So, it is a bunch of things here that you have listed in the middle and you say they go everywhere. But then I guess we should teach those things. But they appear mainly towards growth phase.

**Interviewee:**

Your experience may vary. But basically, every startup I work with early stage or later uses agile, because it is a heartbeat method. Even if you three people just aligned, they may even have much faster sprint, sometimes they even go into one-week things just to keep a very fast heartbeat. Because what is the one of the biggest problems in the early stage startup, you need to learn what works in practice and what doesn't work in practice and you have a limited amount of time. So, you want to maximize the rate of learning that you have as a company, which means that you need to go through very fast cycles. So, that's why whether you call it agile, or whatever you want to call it, it's about very short.

**Interviewer:**

Frequent feedback, I guess.

**Interviewee:**

Yes. A frequent is literally weekly. Is there anything else I need to do to improve the quality of your research or of your life?

**Interviewer:**

Thank you for the question. I think you've given me important insights. I believe I will send you some information, more insights from this. Some things have been published, and you might not have the time to go through them. But I can just recapitulate what we have discovered. But, yeah, thank you so much for the discussion and I really appreciate. To be honest, it's one of the best discussions I would get.

**Interviewer:**

Yes. One more thing in case we get to publish this and the aspect, would you mind if I mentioned that you have given this feedback to me and be part of the practitioners or the people that have contributed to the paper without hiding?

**Interviewee:**

I'm fine with that but I want to see it first. I want to see how you want to attribute things. Otherwise, I'm fine.

**Interviewer:**

This is like you sat here, that's why it's recorded. Yeah, sure, that's obvious. But thank you so much.

**Interviewee:**

Excellent! Take care. Bye.