# **Interview 3**

**Interviewer:** Since we start recording now, I would like to start the discussion with you about with this first background statement about this research. And then we can go through the questions and opinions you might want to give. And this is not going to be like a classical. I ask a question, you give me an answer. So it's going to be a bit interactive. I would move the notes here from the quadrants that I'll explain in a moment. So it is going to be a bit more interactive discussion, I guess. So what I have tried to propose within the course setting taken that I am trying to intermix experiential learning with growth phase start up practices and I will explain a bit more why this growth phase practices, because there is a bit more stable software engineering practices in this phase as compared to the previous phases and external activities such as Innovation Bootcamp.

And what I want to come up with or what I've tried to come up with was the start up driven software engineering course. And this is like the intersection of all these three practices here, these three domains, sorry. And at some point in time I iterated twice with the experts in teamwork course where we had students that had some technical previous background. We had them develop a project together with external stakeholders. But the difference here with compared to a customer driven course that you might know of is that the customer or the stakeholders, they don't really come to students with precise projects they want to develop, but they come to the students with the challenges they have and they face.

In a certain domain. And then the student team has to come up with an innovative idea how to develop this and they have to actually develop a minimum viable product out of that in the course setting. So this is a bit different with other project courses, project based learning, where innovation is also important as it is the software side of it, the software practice. So hopefully they do a prototype and then they start to think in a startup mindset by applying also to fund whatever fund it is, innovation Norway, student entrepreneurship, bodies or whatever, research council if you want, but not so much. So whatever funding opportunities there are, they should fill in an application to that funding opportunity so that they can hopefully get this project done in a real context, like find the means also to support it further.

That's why this is a bit startup driven in the sense that they have to think from the startup mindset. So find this innovative idea, develop a minimum viable product, test it out a little bit and then apply for some funding and present it also to the pitch it or whatever you want to call it to these entities which provide funding and so on. And hopefully they can come back from one year to the next, which might happen but it's not always the case. Then I came up with a perspective framework, how this could work? The theoretical side of it is that apart from teaching and design, its a bit long. But the theory is based on group dynamics, software startup formation theory, or sort of software startup theory in general and innovation bootcamps. Why Innovation Bootcamp? Because that's what I chose. That is program in context.

When it comes to practical side, there are some interactions here that one might think of. I don't want to go in detail with this because this is the education side and some tools that one can think of. So if this was an interview about the educational side of this, I would have gone through like what kind of tools you'd like to see from the educational side, etc., interactions between the learner-learner, learner-teacher or whatever, the learner stakeholder and so on. But I'm not really interested into that now. What I've come up here was first to observe from a practitioner side and from a person who works in industry, how he sees this approach in the sense that there are four quadrants here. One states, ok, we have startups at early phase, we have startups at growth phase. And if you choose, for instance, we have the epsilon axis, we have multi and interdisciplinary cross-functional teams and in the bottom we have software engineering only teams.

[Why software engineering only teams? Because I make a parallelism here. I've made some comparison. In software engineering courses, it's pretty common that you find software engineering students only. So you don't find any other participant within the team itself. So then if you merge early phase in one, if you match early phase with multidisciplinary teams into your batch growth phase with multi interdisciplinary teams and in three, you understand the idea. So software engineering only and in four growth phase with software engineering only teams. So up to here, is it clear? I know I went a bit fast on this, but is it the overview of where we are right now with the background of the discussion?

**Interviewee:** Yeah. I think, this sounds Ok.

**Interviewer:** Yeah. That's good. So when it comes to these practices, I don't want to stop for a lot, but some are taken from XP for instance, or from Scrum or whatever. But these were practices that were notably present in growth phase start ups when usually they were at growth phase, not at very early phase, because I've asked both cases. So when it comes pair programming, let's take this the first one. When it comes to pair programming, where would you see it fit in one of these four quadrants? I said a bit answer myself.

**Interviewee:** Yeah. Of course it could apply to any quadrant. But I would say that in the beginning, you would focus on features primarily and not so much on software quality. So in the growth phase. And of course it's a practice between developers, but they don't have to have all the developers on the team to use pair programming. But yes, maybe it's okay to put it in somewhere in the middle there, but more towards the bottom.

**Interviewer:** Ok. It leans more towards . That sounds good. When it comes to MVP with low code minimum viable product with drag and drop things and not a lot of code development.

**Interviewee:** Yeah. So also, I think it could apply to any quadrant here, but if I have to choose, I would choose early phase startup and then multidisciplinary team.

**Interviewer:** Okay. So it's more like. Ok. I'll just put it here. Ok. Before I go in this, limited documentation, where do you see it more?

**Interviewee:** Yeah. So what is meant by limited documentation?

**Interviewer:** Not a lot of documentation written about what you are developing, but focusing more on the actual thing that you are developing.

**Interviewee:** Yeah. I don't think it makes sense to spend time on documentation in the early phase. [cross-talk] or at least if you think of it as getting some documentation that would, I think apply more to the growth phase.

**Interviewer:** So it would be more which quadrant, two or four?

**Interviewee:** Yeah. I'm not sure, I mean I think it would apply equally if you have business people or other disciplines in the team or not. I also may be in the middle.

**Interviewer:** And readable code, where do you see it fit?

**Interviewee:** Yeah. Also I would put everything that has to do with software quality towards the growth phase.

**Interviewer:** That's good. When it comes to testing, these are two scenarios. Automated testing. You can enter DevOps if you want, but I didn't want to enter that. It doesn't have to be DevOps, but automated testing and integrated testing and customer validation acceptance testing. So where do you see this testing?

**Interviewee:** I don't think it makes sense to spend a lot of effort on automated testing early. So I would put that towards the growth phase. But things that are oriented towards the , yeah, so maybe in the middle there as well. Software oriented towards the end user. So like acceptance testing or customer relation, I would put that in the early phase. And of course, it would have to put it a bit up that involves customer.

**Interviewer:**  Ok. Ad hoc versus agile practices like ad hoc project management. Where do you see it more happen in your opinion and agile practices done properly, I would say? Here it says increased productivity.

**Interviewee:** Yeah. Well, I would think that in the early phase you need to demonstrate that you have something which is of value to someone. So I would not put much emphasis on project management practices or Ad hoc practices in the beginning. So I would put both of them towards the growth phase.

**Interviewer:** Okay. So in Ad hoc, it's also here, growth phase. And agile practices.

**Interviewee:** Yeah. I'd also put that towards the growth phase. That depends on what kind of agile practices you're thinking of, but because I would reason that things to do with software quality can wait. The most important thing is to get features up and check that the features are something that provides value, and it can refactor later. So of course some working iteratively. For example, I would do from the very start, but not include practices that focus on software quality necessarily. Ok.

**Interviewer:** When it comes to protecting the code by storing it in secure repositories, when do you think this happens?

**Interviewee:** Yeah. That's a good question. In my head of course it depends on the product that you're going to develop. But I would think that it's very often so much that exists in the heads of a team so that you can't just copy the code base and then compete out a startup which has gotten a head start by thinking about a problem for a very long time. So I would think that it's not so important or less. I would also put this on the growth phase to start thinking about.

**Interviewer:** And patenting the code. What do you think about the patenting?

**Interviewee:** Yeah. So I would wait for that as well. So I would kind of early phase check out if the product is viable. And then in the growth phase, start thinking about formalities such as patenting.

**Interviewer:** So this should be towards the growth phase or early phase?

**Interviewee:** Yeah. The growth phase.

**Interviewer:** Ok. I put them in the axis for the moment. Well, the start up pressure might be there but these others that you see here mainly and these others here. But team development, we can start with team development and the rest of them it's about little bit, these are together and the rest of those here are about technical debt a little bit. So team development, where would you see it more probably happening?

**Interviewee:** Yeah. So that would be more important earlier. So I would put that on the early phase.

**Interviewer:** And do you lean towards one or three with software engineering or multi disciplinary cross-functional teams?

**Interviewee:** Yeah. But I think it's important and they worry about maybe more important if you have different types of roles. So say formal to the disciplinary team, but would be even more important.

**Interviewer:** And when it comes to startup pressure.

**Interviewee:** So how do you define startup pressure?

**Interviewer:** Yeah. It's like the external pressure to get things done with limited resources. In the more lean context, I would say eliminate all the different waste things that you don't need.

**Interviewee:** Yeah, I would. Well, I think that depends very much on the journey of the startup. But you get probably more pressure when you involve more people. So say that if you secure external funding, for example, that would be something that would provide more pressure and probably that would become more of in a growth phase. So I think that's a nice placement.

**Interviewer:** Yeah. So this was here, as I mentioned, like modular programming. We don't have to go with each of them one by one. But let's say modular programming, flexible software architecture, code reviews, low coupling, technology and tools flexibility, early on architecture decisions, etc., market validation. So all this enter the technical debt sort of to say scenarios.

**Interviewee:** But why does market validation come in there?

**Interviewer:** This is market validation. I mean, this would be like include or accept technical debt for market validation. In that way that you have to make a decision quicker. You don't apply the best software practice just to get it done, get the product out.

**Interviewee:** Ok. So you increase the technical depth.

**Interviewer:** You accept it in a way so that you can fix it, manage it later, or something like that.

**Interviewee:** Well, I think that would be natural to think of in the early phase.

**Interviewer:** Ok. This one's?

**Interviewee:** Also except technical debt.

**Interviewer:** Yeah. I'll put it here. Then the other ones, like modular programming, flexible software architecture, code reviews.

**Interviewee:** Yeah. I think that can be later but requirement validation, a little bit early. Get feedback on the product and developing dummy MVP's very early.

**Interviewer:** I'm trying to zoom in. I don't know if you are going to read these things.

**Interviewee:** Yes. I think the things we have on top now, I'll put to the growth phase.

**Interviewer:** Does it matter the team type or is it like two or four?

**Interviewee:** Well, I think as long as you have, I'll say develop a technical product, you would need it also with that.

**Interviewer:** I'll just put it like this for now. I can rearrange later. So you'd feel like any of those goals in two or four or all of them can be here like you say, it's okay.

**Interviewee:** Yeah. I think you would need them also in the team. We're not only software developers. Let's see, lack of developer competence. I don't think it's so important in the beginning maybe to develop a competence. But of course, the focus should be on providing functionality very fast in the beginning. And if you don't have people who are skilled in the right frameworks, then you might have a challenge to produce things quickly. So if it isn't like , it might make sense to put it early.

**Interviewer:** And the lack of best software practices.

**Interviewee:** Yeah. So I would say that supplies for the growth. Ok. So that gets a little bit early.

**Interviewer:** Obsolete MVP development to clarify this a bit. This also goes maybe with the market thing or to test requirements or stuff like that. Like you make some MVP's and you put it out there just to see, try out prototypes, to try out what the people think and the customer think and their feedback to listen to their feedback.

**Interviewee:** Yeah. So like experimenting with the product. So that would be early phase. At least a lot in the early phase. It might apply to all phases. A cold smell, so I don't think. One needs to care about until the growth phase.

**Interviewer:** Ok, great. Now to go a little bit towards the tools, frameworks and different approaches, like there are a bunch here. I'm not saying that these are the most exhaustive list because this is a list of things we have used in a way. Actually not all of them, because some of them were proposed from the educators such as this growth hacking or design thinking. But some of these things, theories and the tools or frameworks have been already used or models or whatever. So like sustainable growth, lean persona, cynefin model, which is you might have heard of it in the agile like categorize in the four out of five domains where you can enter a problem. Like simple, complicated, complex and chaotic. And customer journey is lean business . Actually this is the first one I would like to ask. Where do you see this, for instance, lean business model.

**Interviewee:** I would think about this would apply very early.

**Interviewer:** Ok. And which kind of team would you choose between the two?

**Interviewee:** Well, you would need someone with insight into the domain, not only the picked up in the first quadrant.

**Interviewer:** Sustainable growth.

**Interviewee:** I think that's something that should always be thought of.

**Interviewer:** Ok. I put it here in the middle.

**Interviewee:** And also that model. I guess to me, it's that you categorize the product. I'm not sure if you really use it a lot during work. I will think of it as it would be probably in the complex domain. If you are working on developing a new product. I'm thinking about innovation.

**Interviewer:** I put it just in the middle or do you want to leave it out? It's fine if you want to leave it out.

**Interviewee:** Yeah. Maybe we could put it in the middle.

**Interviewer:** Lean persona again from lean principles.

**Interviewee:** Yeah. So how do you define, what you mean by lean persona?

**Interviewer:** It's like you define this person that is your typical customer and you define it with the way he is intended to use your system and to benefit from your system. So how he gets value out of it and how he's going to use it. Like for instance, let's make a supermarket and you have a common user. This user is called John for instance, and John is a typical user that is in his twenties. He buys this kind of stuff, etc. He uses an app and stuff like that. So it describes in a certain way the person so that you can make sense how the system is going to be used in the requirement standpoint and so on.

**Interviewee:** Yeah. So I'll put that early and in Quadrant one. It's also insight on design thinking for me would go up there. And customer journey. So things like , I think one could start with very early. And maybe in the middle there.

**Interviewer:** From XP. I know you're going to have a as well like this framework. But from XP, where do you see them?

**Interviewee:** Yeah. So I think it's impossible to split them because I would start with scrum and then introduce XP practices later because I think that's not so important in the beginning. It's more about developing features and then they could start considering software quality.

**Interviewer:** Ok. So XP should happen later, you say.

**Interviewee:** Yeah.

**Interviewer:** But scrum can also happen later or just in the early phase.

**Interviewee:** Well, I think it might have things that you start with in their life as you would also continue it. So like team development is not something you stop.

**Interviewer:** I put it in those both places, I think. Ok. So we have a bunch more here. I mean, retrospective meetings, this can be removed because anyways, it's part of the scrum framework. Would it separate because it is a bit about reflecting how the team works, the process and so on. That's why it is there separate. But if you feel it's not critical, it is covered by Scrum, it's fine to remove it.

**Interviewee:** But in my head it's the most important practice and developers, so we can include it in the early phase. Just to emphasize that point, even though it's covered in scrum.

**Interviewer:** Pitching. Pitching is, you know, to pitch the ideas.

**Interviewee:** So that will be earlier. And also maybe more towards the first quadrant.

**Interviewer:** Loss to growth hacking. Actually I heard this from Becca, they use this growth hacking practice.

**Interviewee:** What is meant by growth hacking?

**Interviewer:** It's like to do some rapid growth of the customers for a product, like to do some boosting in that direction, like acquire a lot of customers very fast. There are some techniques for that, I guess. I am not a specialist in this. I just heard it from Becca. Maybe this is something that in the industry they don't use it much. I don't know if it comes from academia or something like that.

**Interviewee:** But getting a larger user base, I guess it would depend very much on the products that you're making. If it makes sense to if you want to test it, if it's critical to test with a larger customer base or I think you will learn. I think it would make sense editorially. But I think in most cases I would think that one would focus on the features first and test it on the few users and then expand the user base later when things are more stable and you have something more to show.

**Interviewer:** So you lean towards growth here or towards the early.

**Interviewee:** I would put it maybe on the second quadrant.

**Interviewer:** Ok. Customer journey and last one here.

**Interviewee:** Yeah. So that I would start early as well, in the first quadrant.

**Interviewer:** Ok. So external activities, like I have picked four here. I have used innovation bootcamp. And maybe there are others. But what do you think? Which of these actually three of them are the most critical one. So Innovation Bootcamp, where do you see it happen?

**Interviewee:** What's the difference between the Innovation Bootcamp and the Innovation Workshop?

**Interviewer:** The boot camp is more about brainstorming ideas, finding solutions, building some MVP. The workshop might be a little bit more theoretical. I guess not theoretical, but with some hands on examples during the workshop. But it doesn't mean that you sort of develop some rapid MVP or something like that. You might go through different exercises, how the innovation could be part of the product and so on. But yeah, it doesn't have to end up with concrete value. It's more about hands on examples. While Innovation Bootcamp its about exploring a certain idea, coming up rapidly with some MVP and then pitching it at the end, hopefully afterwards.

**Interviewee:** A couple more of the innovation process that would come.

**Interviewer:** And it doesn't have to be technological immediate MVP. It can be some sort of technology there, like low coding, etc. While the hackathon is a bit more tech oriented and the reason I want this kind of activities external to the course is that because they provide some good communication with the external stakeholders, finding customers, whatever. But yeah, this can be whatever thing you might think of. But the reason is to have collaboration with external stakeholders in a way so that you provide the solution to them.

**Interviewee:** So the question is where to put this in quadrants? I think I would think that it's a good idea to do a lot of creative work in the beginning. So I think all would fit. Lock it down. Of course you would need developers. So that's a little bit more towards three, one. Would also be one. Yeah.

**Interviewer:** Okay. Anything you might think of? Anything else?

**Interviewee:** Other activities? [Yeah]. Well.

**Interviewer:** Like this. Second question here what kind of external activity would you suggest for industry and education settings alike?

**Interviewee:** And so in order to develop an innovative product. Well, it might have. That's most important with processes that enable feedback. But I think you covered that quite well here already with having MVP's. And I would think it would be natural to try things out in larger and larger scale. So that you would after a while have a larger product and have a larger user base or a set of stakeholders.

**Interviewer:** Yeah.

**Interviewee:** So that would be an okay approach to risk and that you also would have a small team in the beginning, and then you would expand on the team as you understand that you need more technical competence or you might need more other types of competence. For example, if you want to file for patents, you would need to invest in that.

**Interviewer:** Yeah. But we don't have to find out what other things can be out there. It's not a problem because anyways, I've chosen the innovation bootcamp and one might say it's good, might say it's not so good, but I saw it happened a lot in industry in a way for internal entrepreneurship as well, like internal innovation and so on. So I found it would be a bit more cost effective. And some research papers also had the similar approach from before, so it seemed reasonable. But on each of these one, two, three, four quadrants do you think education should focus more and provide maybe some insights and reasons that should education focus on one, for instance, or two or three or four in terms of practices and in terms of how they would?

**Interviewee:** Well, I think for us to do software engineering education, we should be more towards the right so that people that are in business schools, focus on things more towards left and people are into product design and other disciplines. So I would say that when ideas have matured a bit, then you would be more engineering practices. If you are a very small team, you could do without much engineering practices. But as soon as you get all the team, you would need to coordinate better within the team. And they would need to also focus on the quality of the product or the code base on the products increases. If it's very early, you could just rewrite everything, but that gets more expensive as you mature. So it would make sense to kind of refactor things that you've developed early and then establish a sense of quality in a good time before you have a product that will face real users.

**Interviewer:** This question to clarify it a bit briefly, this question is because what I observed were poor practices in the early phase. Of course, a lot of testing from business side, but in growth phase, the practices in software engineering were a bit more stable and one could say that, okay, they are like you have clear code reviews and stuff like that. Technology tools, they try to do a bit more better XP and Scrum together and so on. So all these practices are a bit more stable in growth phase. And when we teach software engineering, from my perspective, my opinion is just an opinion. We teach best practices to students, not bad practices or things that are done poorly. So we try to teach good practices from industry to the education setting. So I understand when you say you lean towards the right a bit more because it's more like better practices with two and four better practices in that way. And yeah, I don't have to express an opinion, but I see your point here. When you say two and four. And between two and four, which would you prefer in the next experiential learning context? And this is sort of the last question.

**Interviewee:** Sorry, which I would prefer?

**Interviewer:** Yeah in two and four would you prefer multidisciplinary cross-functional teams to work together or like this experts in teamwork?

**Interviewee:** I think you learn a lot more if you have multidisciplinary teams. So if you have a software team . Yeah. So it's multidisciplinary.

**Interviewer:** Yes. Okay. I think it's 1:00. And if you have any question about this for me or anything you might think of and I hope it wasn't very stressful discussion. We can put it in a different manner. I didn't ask you questions and have you talk all the time. It was more like a bit interactive. When you have time, we can maybe go through this other one, which is totally education side like this. Here the questions are a bit more focused and and then of course we move these things around. But yeah, and this is the framework repeated in a certain way for me to have an initiative, the things I've done are correct or what are the differences I could have done better, etc. So yeah, we can maybe schedule some other time for that. But in this one, do you have any further?

**Interviewee:** No, I think just to jump over to this other meeting, it was interesting exercise.

**Interviewer:** Yeah. Thanks.

**Interviewee:** I hope it was useful for you.

**Interviewer:** Yes, it is. It is. And I will write to you when you have time for another discussion, hopefully in a month or before a month. Because I need to wrap up.

**Interviewee:** Yeah, sounds good. Pretty good to talk to you.

**Interviewer:** Later then. Have a nice day and nice week.