Interview transcription - 2021-04-22- 13:00 over Zoom (remote)
Speaker 0 = Interviewee
Speaker 1 = interviewer

- Speaker 0 00:00:01 So now we're we are recording. Um, first of all, thank you for participating of course, and spending your Wednesday afternoon or at least up to an hour tops, but usually takes, uh, 35 40 minutes. Um, so basically what we are focusing on on our thesis is to investigate, uh, how the concept or the practice of, uh, splitting bigger things into smaller pieces and kind of working from delivering small pieces incrementally and, uh, the whole, uh, divide and conquer aspects. So, uh, the goal of this interview is for us to get an insight into how you and your team does this, or how you treat this practice, um, and why you do the way that you do reasons behind your choices. Uh, but first off, would you mind, uh, spending some minutes describing your experience and what you do at the moment at your company or where you work?
- Speaker 1 00:01:17 Yes, I, I am working at [NAME OF COMPANY] and I am the manager of two teams. Uh, the first team is, uh, contains the software project managers and a software solution architects, and a software solution architect is pretty much like a business analyst. They collect the requirements and then they write the specifications together with the customer. And then the second team is the engineering team. And in that team, we have two roles is it's technical leaders, which, uh, is responsible for the design and the development work during the implementation phase. And then we also have software engineers that is, um, yeah, doing the, working with the task, which is implementation and design and writing test cases, et cetera. Uh, our, um, context. Uh, so let's say we have a standard product [NAME OF PRODUCT] and that management system, which is, uh, taking care of what we have for customers having in, in those stock, in the warehouse.
- Speaker 1 00:02:24 And we also have processes for handling the products within the warehouse, and then we have, uh, our own developed interface towards the machines. So we can from, from our system, we, which is called [NAME OF PRODUCT], we can quit, we can stop and start the machines, the robots, et cetera, and we can start transports. And then we send telegrams to those sub systems and then they collect the response back to us. And we, so, so we, we have one system that thinks care of the whole warehouse, including automation and also manual processes. If you have like a hand scanner or a PDA or, and do some manual picking. So what we do in our project is that we, we have a scope and then we have our software and then we have to do the customizations, the gaps we call them, uh, that is sold in the project.
- Speaker 1 00:03:13 Uh, and then we also do configuration of our standard ports to get it, to work, uh, with the customers and in a typical software project at [NAME OF COMPANY], we have the software project manager, the solution architect, technically the engineer, and then we also have a tester and a trait in it. So that's the rules we ha we have, uh, in our project, since we are working in a scrum inspired, uh, method and process. So we, we are working in iterations. We call our sprints for iterations and they are usually two to three weeks. A development team is depending on the size of the

project, I would say that they are from three to 15, 20, depending on the size of the project. And the typical software project is around, uh, I would say 4,000 hours and the small one is 400 hours and the big one is 30,000 hours. So we, we have a big variety of the sizes of our projects.

- Speaker 0 00:04:17 Okay. So in terms of, I mentioned, uh, that we split things from big to small generally. Uh, so that's something that, uh, when I mentioned user stories, that could be anything that you think of, uh, which are either say a feature or an Epic, or use the story itself, but it doesn't necessarily have to be written in the, uh, classic format of as a user I want to in order to get some benefits, right. So, uh, we're not constrained by that definition or by that something that looks exactly like the Academy or the literature describes it. And also the concept of user story splitting, which is generally used when you actually divide a user story, as I mentioned into smaller tasks, but we want to look at that concept from a bigger picture as well. So it could be dividing an Epic into smaller pieces, and then those pieces into yet, again, smaller pieces. So when I mentioned user stories, uh, don't be constrained by the, uh, by the format. And when I mentioned user stories thing, that's the practice of making big things smaller.
- Speaker 1 00:05:52 Yeah. And I would think of it like, uh, we call it requirement's more or less than we can. Sometimes we describe them as fuel stories as well. And we sometimes the test cases, they use the story then.
- Speaker 0 00:06:05 Yeah, exactly. Not any kind of items such as those, that's all you're looking at. The, you mentioned that you were working in iterations. So how does the, uh, how is the hierarchy within those iterations? How does that look and also, how does it look in the bigger picture? So you have one integration for two to three weeks and what kind of items do you have in that iteration and how do they look and how do the bigger ones look from the bigger iterations given that you have those?
- Speaker 1 00:06:42 Yeah. And, um, I mean, the, the, the length of the duration is, is, like I said, two to three weeks, but the capacity of that iteration can differ depending on the size of the project. I mean, if we have 10 developers, then we have more capacity compared to if we have a tool. So the scope of that iteration can be very different. But what we usually do is that we, in the beginning of the project, uh, we try to understand what we shall deliver. And one thing that is perhaps important to tell here is that the scope is pretty much fixed for us when we start a project, because we have sold our project on fixed price. So the scope is, is set already when we start the project. And then of course we work with change orders, uh, during the project, but that they are pretty small in the total amount of orders.
- Speaker 1 00:07:35 So we can think that the scope is pretty much fixed. So then in the beginning of the project, we know pretty well, what we want to do, and we know how many resources we have available in the project. So we do usually do a pretty rough split of the different areas that we should deliver in this project. And perhaps we say that the first iteration contains of a startup where we download the source code and we set up the environments, we might map the warehouse, we put in everything that we know. So we are not working that much with any functionality in the first iteration. And then when we start in the second iteration, we usually, we, we are, we are trying to follow a

process in the warehouse. We might start with receiving because is what we start first. And then we see if we are getting receiving to work, uh, and receiving that big, uh, in this case, we believe that we will be able to penalize preserving, and then perhaps the next step in that process, which might be occasions, election, where we decide where to put this pallet that we are reserving.

- Speaker 1 00:08:47 So we might say that, okay, let's include that in that first iteration. And then in the next, we might talk about allocation and order handling, et cetera, and that, so we make some kind of rough plan or what the content of each iteration should be. And then we say, okay, we have four months, and then we will have, uh, 10 iterations, for example. And then we know that the scope on rough level, on each iteration, what to do, uh, and then we try to, to, uh, to plan, uh, so that then we have this like as a rough estimation, and that might be done by the project manager and the technical leader in the beginning. And then we have this iteration planning meeting where we define in detail, what shall be delivered in this situation. Then we create tasks for that. And then we have a role that the task should not, uh, is not allowed to be bigger than 40 hours.
- Speaker 1 00:09:43 And we try to keep them around eight to 12 hours because, uh, um, I, I, I might get back to that later while we want to have them as a at a decent size. Um, but when we, when, when we, when we'd done it planned that iteration in detail, we, if we, for example, see that we have the need to, uh, see a host message and message from another system, to be able to know what to do, then we take care of that directly. So as soon as we see that we have to fix something, we start in that iteration. We don't think that, okay, let's wait with all whose message to the end. So we want everything to work, uh, as much as possible, but when we have, have done it, we don't want to have whole functionality. We try to fix everything we find during that live within that process, that we are trying to plan.
- Speaker 1 00:10:38 So that's pretty much how, how we, how we do it. And then during that iteration planning all developers, uh, in the team or, or in that meeting, and we try to set the goal, a target for this iteration or release as we might see it. And then we, we, we might have to have a walk through of it from a functional perspective. So everyone understands how this will connect to other functionality in the system. So if you are doing one thing here in this iteration, be aware that that might be used later on as well. So perhaps we shall prepare that to be as generic as possible, or as specific as possible to be able to support the next process that we will work with the next iteration. So everyone understands how, how, um, what the tasks, and now we create the tasks, uh, plan them and we'll put our estimates on them. And then, uh, we'll put them in the backlog for that iteration. Then, uh, we're more or less start that iteration
- Speaker 0 00:11:44 A lot of detail. Thank you. That's good. Um, so my next, my next question would be, uh, if you could, how the splitting is actually done, but I felt that you actually talked about that, uh, in quite some detail.
- Speaker 1 00:12:02 Yes. But they feel, for example, see that, okay, now we have this process. We find that, then we see that that estimate is, is, is bigger than 40 hours. Usually the developers, they might like to have big tasks because they don't want to

divide it. They, they might think that it's more efficient to that. One person will be one thing for a longer time than dividing it. But me as a manager, don't like that, because it's very hard to, to understand how, how, um, the process or the progress of that task. If someone is on a daily standup, say that I work with this task and they have nothing is not being me, et cetera. And then we don't w w we are not able to give that person support if he needs help. Also, when it's hard to see if we are meeting the deadlines in that iteration, as well as we, we, we try to, to have that limit of 40 hours.

- Speaker 1 00:12:53 And that's also one thing to be for being, being able to have some progress in that iteration. So, um, we're trying to make them as to can have this overview of that task. And it, you know, be that, uh, to complicate that, and when a task gets too big, then it, it might also be pretty much a how to say, uh, that it's one person that knows how it works. Then we might have a risk with that. And we won't have the code that everyone is comfortable in changing orders, codes, et cetera. And if that person gets sick later on, then it's better to have that task divided onto more people. So more people are aware of the functionality and how it works and so on. So that's, that's another advantage of splitting them.
- Speaker 0 00:13:42 Hmm. Are you, are there any type of tasks which generally become big as you say, or is it just some things are just bigger in their nature, so to speak?
- Speaker 1 00:14:02 I mean, yes, definitely. Some things are bigger by nature. We will see that this is a very complicated thing to fix in the beginning when we planned the iteration. So then we try to split it up, but usually a tasks tasks might get bigger during the implementation because we see, we understand the requirements during the work with that task. So then it's easy to just continue to work with it and fix what you find. But then we are trying to say that now you should not do that. You should create a new task. If you see that this task is growing too much, we get some more control and we can distribute it on other people as well. But, um, to what we know, then we can split it. But usually there is that the task is growing during development or implementation.
- Speaker 0 00:14:49 So in terms of the time of executing this practice, you've mentioned that both during, when you notice that something becomes big during development and also before. So in the beginning of a iteration, I guess, are there any other times when you know that, or when you actually do the splitting?
- Speaker 1 00:15:18 No. I mean, no, it's, I, I will say that it is during those two times when it, first one, we plan it and then during the work, and we see that this, this task is just getting bigger and bigger, and then we have to split it up. Okay.
- Speaker 0 00:15:33 Who is, uh, who is typically involved in this process?
- Speaker 1 00:15:41 Uh, in the beginning, I think it's the technical leader. That is the responsible person for that. And then of course it is the actual developer that finds out that this one is getting bigger and bigger. And I didn't foresee that we have to do this as well. So then it's up to the individual engineer, but of course, to, to discuss it with a technical leader. Okay.
- Speaker 0 00:16:01 Okay. And, uh, are there a reason why the, Let's say more roles are not involved or is this something that you have found out works good for you?

- Speaker 1 00:16:19 Why, why not other roles are involved in those discussions? Is that your question?
- Speaker 0 00:16:25 For example? So why the technical lead and not say some other role, for example, or why, or?
- Speaker 1 00:16:33 Yeah, but that the technical leader is the guy that's responsible for the development process and that they might sit in another country or somewhere else. So the product managers should not be in there and touch that the responsibility of the iterations or sprints is on, is on the engineers and the software engineer and a technical leader. That's the two roles we have that is actually working in the iteration. So it's pretty natural. And then the technical leader is the one responsible for the delivery of this iteration. So, yeah.
- Speaker 0 00:17:10 So looking at the, looking at the developer's perspective, which deal with these tasks, what do you think that they would say, uh, regarding the tasks that they are delivered, so to speak? What do you think that their perspective is, or their feelings on how this process works?
- Speaker 1 00:17:34 Eh, I mean, the process is there for, for, for them. So I would say that they, they think that it works pretty well, but I mean, sometimes it's, it's, um, can be stressful with estimations and that you feel that you overran the estimation. Uh, then I think it's, it can be stressful for them to do that, but that's, that's one thing that we try to mitigate by adjusting the estimates for the next iteration. If we see that, no, I mean, we're too low in our estimates, then we have to hire them because it's very much based on the ones that is doing the estimations and, uh, what we know when we estimate them, sometimes they grow. So it's not because the lack of skills of the developer because of that, we did not understand the requirement or the proposed solution. We found out that that is not working. We have to refine that.
- Speaker 0 00:18:28 And they, you mentioned that you, uh, take into account certain things when you split, uh, when you split a story or a big item into smaller pieces, for example, are there any guidelines or techniques that you can kind of rely on to guide you on how to make those judgments or choices?
- Speaker 1 00:18:54 Now, I would say that the, uh, more or less, only based on experience from the technical leader, because he, he knows what part that is included in this big task. And he knows where to put the line, where to split the tasks to say, you can start, but you can end off this telegram, or you can end up with yeah. That position or whatever, because then he knows that then that scope is pretty much boxed in a good way. So it's connected and it's easy for someone else to take over or perhaps start the next door before that one it started. So, so it's based on experience. We don't have any methods or, or best practice or so it's, so it's experience based.
- Speaker 0 00:19:38 And what do you say, do you have any insight into, um, what aspects he takes into account when he makes those choices or what his reasoning is behind splitting in this way? Instead of some other way, for example,
- Speaker 1 00:19:58 I would say that it, uh, it should be, uh, the split, the tasks should be big or small enough to have some kind of, um, meaning and so on. And also perhaps it takes the test perspective into account. If it's possible to write the system test for this

part of the scope, that's feels pretty natural too. We have a, perhaps a natural clear cut. So to say that this ends here, and then we start a new process. So we have to create a new method, or we actually use something else we're waiting for a, for a decision from someone else here. So, so, so, um, I, I will perhaps, I don't know this, but I would guess that he has the test perspective. Also they start and it starts here, it stops here. And then during this, those two were important. It's pretty easy to test that it does what the we should want as well.

- Speaker 0 00:20:56 Okay. And in terms of having those perspectives, is there any, uh, or do you think that there is a, uh, some kind of a definition of when, uh, this process has been successful? Or do you have an, do you have an idea when you think that it's been a successful process in terms of the outcome?
- Speaker 1 00:21:25 I mean, when we have been good at dividing the tasks into two to a good size. Yeah, no, I don't have any good example of that. I mean, usually that the, uh, the works in projects and I cannot think of a good example, but I have examples of bad examples where the tasks has been as big as 200 hours or something, because it does. And that's usually when we try to do too much within the same task that we have, perhaps we have this, uh, uh, yeah, you have to, we call our user interface screens to work with a screen, and that can be pretty much time consuming. And then also sometimes we, we have to put in some logic into that screen, and then you have to take care of the, or the backbone parts as well. And then that does get very, very big.
- Speaker 1 00:22:13 And then perhaps we have, uh, a demo for a customer and we get new requirements. So we get updates to the requirements and we see that, and then one person is the one that knows that task. And then it's not good because, uh, um, yeah, it's, it's got still dependent on that person. And, and also for planning, uh, issues. It's also hard because then we will not be able to finalize this because these tasks got too big into this iteration, and perhaps we should have created a new tasks and then take this prioritization. And shall we take those out of this situation and planning the next one, or shall we try to finalize them, but then we have to move something else out or a situation. And then we can take that decision together with the software project manager, because, uh, usually, uh, we tried to do this iteration demo for the customer after each iteration, so we can get this, um, yeah, we can show what we have done and we can get the proved that we have understood, uh, what they need, uh, in an, in an, at an early stage, instead of finding that later on in the, in the process when it's much more expensive to fix.
- Speaker 0 00:23:26 Hmm. So in terms of, uh, um, usually in, when you work in a scrum way or inspired by scrum, or, uh, you usually have a retrospective ceremony or yeah. Now, but perhaps?
- Speaker 1 00:23:46 Not after each iteration, but, uh, I, I usually recommend that we should have that after the first one, so we can have a chance to adjust. What's not working and so on. And then, uh, uh, we can have that, uh, depending on the status of the project, if we see that this is working and we are meeting the target so that we don't have to have a retrospective after each iteration, but, uh, after the first one I recommend, and then two to have it, when we see the need and then perhaps in the end as well, to

try to learn something for the next project, what would include and what we, what we can improve and so on.

- Speaker 0 00:24:22 And do you do during these retrospectives, do you, do you discuss the practice of how the split thing has been taken care of or if it's been successful or not, and why, et cetera?
- Speaker 1 00:24:39 Yes. That's is one topic that, uh, most likely will pop up, but it is. And another one might be the estimate and another one might be the division, a leader, the definition of done, because we went, when you have finalized the task, you set it in to review, and then someone else should review that. And then if that one that the review where that's not accepted, then it has to send it back to the first developer to review again and there, we can see that. That's one thing that we usually talk about during those retrospectives, that how can we align on what we mean? We, the dump. So everyone has the same ID and that, uh, I mean, you should not be commit to anything that has not passed the system test, too. I mean, if you commit anything into the trunk, you should have done what we have really done before you do that. So you don't break any other thing by your commit feeds. I mean, we try to set up those rooms in the beginning of the project and, uh, if needed, we discuss them again at that retrospective to make sure that everyone has understood them has the same that we have an alignment on what we mean by done.
- Speaker 0 00:25:55 Hmm. So if you look at a, if you take the bigger picture and look at the more general, uh, benefits of dividing or splitting tasks, what do you think that the main, main thing or main benefits of that practice is for you or your team?
- Speaker 1 00:26:22 For me, as a manager and as a project manager, my benefit is that I have better control on the progress in the iteration. I know, and I can see if someone is sliding away to meet the target. So I think that's good and, uh, fall from a perspective, I think it's, uh, it's easier for them to get the understanding of what I shall do if the task is smaller. It's um, and also I think it's important or the energy level, or, well, I don't know what to call it, but that you, you feel that you tick off things and you don't work for one thing for two weeks. And so if you divide that into five tasks, then you know that now I've done three out of five. So I have some progress here. And I think that brings energy into the group as well, to see that we have some progress. Um, yep.
- Speaker 0 00:27:22 Do you see that there are any, or can you think of any drawbacks or negative aspects of the practice?
- Speaker 1 00:27:30 Yeah, it might be like what I mentioned earlier that if one developer thinks that it's, it's better and more economic, if I continue with this, because then I don't have to understand the previous part of the task, et cetera. So, because I'm already into this, uh, this, um, problem or whatever. So it's, it's a, if I fix it, it takes five hours, but if someone else should understand the problem, it might take 12 hours, 20 hours for them to be able to start where I am right now. So that's the drawback, I would say,
- Speaker 0 00:28:02 Uh,, have I missed anything? Have you thought about anything? So in terms of the conversations or the conversation that we've had, do you feel like there is anything, or can you think of anything that you would like to add? Anything that

you think that we have missed asking or would be beneficial to continue discussing or describing or telling us about?

- Speaker 0 00:28:45 Reasons behind your choices or, uh, the outcome or the benefits, et cetera.
- Speaker 1 00:28:34 And then you think, uh, or the perspective of, uh, dividing big things into smaller things?
- Speaker 1 00:28:55 I mean, one thing that I've not struggling with, but both I think is really important is to, to put the target or a goal with every iteration. So everyone in the group in the team is aware of what we are going to deliver here. Um, then I think it's easier for everyone to understand that they, their participation into the project is important and that we get the commitment from everyone. And, uh, also this, that if you work in smaller tasks, I think it's easier to at the end or iteration to, to, to get support from someone that is, um, as his task. And then he can continue with what's remaining of that other task instead of getting into the same task as another one, because that is also one thing that, that iteration isn't done until all tasks are done, and then everyone's work is important.
- Speaker 1 00:29:57 And if someone is struggling, it's up to the team to support that person, instead of taking a new task and continue what's planned for the next iteration. So if the tasks are smaller, then it's easier to distribute them on to more people as well. And then it might be easier too, for the team to be able to succeed in their delivery for that iteration. I think that's a, that's one good thing. And, but to, to, to have this as that, every iteration should have a release. It should have some kind of incremental something to deliver at the end of that, then perhaps we can install it under the customers test environment so they can start playing with it. And then we can get instant. We can get very direct feedback. And like I said, it's very, very good to get that feedback as early as possible. And that's sometimes hard for the developers to understand and to get their acceptance of that, because then they think that, okay, now we have to do this demo and then we will get complaints and then they feel controlled, but it's not about that. I mean, we get that feedback later. So, I mean, then that's why I like to get it at as early as possible.
- Speaker 0 00:31:07 Has this, uh, in your experience, has this changed over time? Have, has your, uh, has your mentality in terms of this subject or topic, has it changed or have you always had this approach or ideas behind it?
- Speaker 1 00:31:25 I think I have always liked it, but I, and I don't think I have reached the target to get full acceptance from everyone, for, for these mindset. Um, but it has improved. And I think that everyone sees the benefit of it, but it's also a lot. I mean, it's, it's very much, depending on the customer. Some customers are very, very demanding. And then, then it's not that nice to do a demo because then we know that we will only get complaints, but if the customer understands and have the right expectations of the demo, then, then it's very, very constructive and good. And that's not up to the developer to, to, to create those expectations that has to be done very early in the project when we describe how we will work. Um, so that's the responsibility of the project manager. Um, and I was thinking about another thing related to, um, yes, and, uh, I

mean, sometimes I can feel that, um, it's hard to get an understanding from the developers that the reporting is important.

- Speaker 1 00:32:35 So, so sometimes the, the languages of the product managers, the developers are very far away from each other. So they, when I tried to motivate them in why this is important, they don't understand why this is important. So I have to adapt my language to set it into a context that they can understand. And at the same time, I think the developers has to understand that I have a yelling customer on the other side that is asking me why this is, this is they have, we have to, we can work with the understanding of the different, the needs for the different departments. The one very important thing,
- Speaker 0 00:33:12 Unless there's anything else I'm going to stop the recording. Uh, thank you. Yep. Thank you.