Interview 5

Speaker1: [00:00:03] Thank you again. Is it okay for you if I record this discussion?

Speaker2: [00:00:15] Of course. You're welcome.I'll be happy to help.

Speaker1: [00:00:18] All right. Thank you very much. So the topic of my master's thesis is to get a better understanding of possible challenges in the low code development through the end user. And I try to see if there are any aspects in the Scrum methodology that needs to be adapted or if Scrum is a possible methodology that can be used.

Speaker2: [00:00:48] Stands for this topic, combining the two. Yeah, that's. Yeah. I'll be also keen on reading the literature after you finish it. Hopefully. And successful.

Speaker1: [00:01:03] Yeah. At least there's a second one who is reading my master thesis. Can you describe your actual position and your daily work?

Speaker2: [00:01:17] Of course. I work in the DWP, which is the digital workplace of company x. My actual title is Officer 365 Architects, or which is now called Microsoft 365 And it's a suite of services that includes SharePoint, OneDrive, the Exchange, the teams, but also a part of it, the bundle of power platform, which is the low code platform that Microsoft is offering, and it includes power apps, power automate virtual agents or power virtual agents and the power BI. So my responsibility is actually across all these services. So I have something to do always within the designed architecture, but as well as lots of hands on engineering and operational work related to the configuration and how these services are running. I mean, ultimately the goal is to offer company x users as an enterprise the ability to work there for better and safer into our organization.

Speaker1: [00:02:33] Thank you.

Speaker2: [00:02:34] And yeah, like the power platform is something quite new in company x. And there were some interest initiations and it was very intriguing. We looked at it and it was like offering the possibility to build apps in a much in substantially shorter time. So we started with it. Like as a platform, even though it was not officially

positioned within the organization. But we start with it. Transforming some legacy applications with it. With it featured like, like, like shift and lift almost as its features. But then we also transformed it with better functionality. And I have helped also in the support as well as in some of the development of these applications.

Speaker1: [00:03:33] Yeah. So when was the first time you got in touch with low code software development?

Speaker2: [00:03:39] Well, I was in touch actually with the very early beginning from I have been working with Microsoft Technologies like my entire career revolves around it even before the cloud when it was server based, but with the cloud also started with the initial beta timeframe. But as for the low code, I started as early as 2015 since Power Automate were called Flow and it was just as easy as like the way of replacement of SharePoint design at something in the past and the way to have a workflow engine. Yeah. So as early as possible and yeah. But we have to remember that back then not many enterprises or organizations fully trusted the platform because it was just not mature. It was just very early on. So there were also challenges even to sell this to the board or the architecture board or. Yeah, because of the preview and the stability and the maturity.

Speaker1: [00:04:40] Yeah. Have you developed low code software by yourself as well or.

Speaker2: [00:04:45] Yes, quite many actually, which combining both power apps as well as power to make together. Yeah.

Speaker1: [00:04:56] And where do you see the biggest difference between low code software development and classical programming?

Speaker2: [00:05:06] Right. It's an interesting way to look at it from outside because the because I started my work as a standard software developer working with C-sharp dot net, ASP.NET Web Web pages and also Windows Forms and then transition to focus more into office risks for financial services. For services. I would say like when you work with it, of course there are differences in the simplicity and the way how you work with it. But like if we look from outside or personal experience that most of or many of the Low-

Code platform projects are either individual or very, very, very small teams to run it and created from start to finish, one or two persons would be sufficient to build the whole thing. While traditional development could be a team of ten,fifteen,, seven, five, it varies, of course based on the scale, but usually the power platform solutions or the low code platform solutions are more individual and they can build really large enterprise solution, you know. So that is the team size the difference and that also has advantages and disadvantages. You know, always a team could be maybe delivered better, sometimes better quality due to the like the variety of skill set of each members included. While that one is more, more of individual contribution, that is one of the biggest differences. But while you are the person who is developing it, you don't feel like the difference because you are delivering it, you are working on it. The cycle as well of the lifecycle. You know that it used to be called software development lifecycle.You know, in the basis is a bit different where I feel. In the traditional software development and regardless the methodology is being applied to scrum and you ship features as fast as possible. I feel the delivery and the presentation and the ability to demo with the local platform is much faster. You have screens that are ready in seconds, in minutes, in days, you know, and like you are waiting to deploy, you are waiting for continuous integration and development. You have no it's like it cannot be compared. You can demo much faster. Users can have the ability to access easily, especially with the, if we are talking about within Microsoft 365 within their normal account, they just log in which they use to log in actually to access their email, their intranet. So they use that they don't need additional switch to another environment and they can see the outcome. So also the testing involvement of the business is slightly different where it favours the low code. When it comes to complexity, there is limitations on both sides. But on the traditional development, you have more advanced way, let's say, or more freedom to build something. But that then again touches on the complexity. We can build many advanced scenarios with the local platform. You might end up using certain additional premium functionalities that requires additional licensing in order to achieve that, which could be a limitation for some enterprises. But then also there are some limitations that low code platform cannot do as strong as the traditional software development.

Speaker1: [00:09:00] Where do you see the biggest challenges in Low-Code development?

Speaker2: [00:09:08] I see one of the biggest challenges that we face as not for the company itself is the simplicity of the low code platform. It sounds counter intuitive. Simplicity should be a great thing and it is a great thing. But unfortunately, it drags a lot of normal people, a lot of I mean, it intrigues a lot of users who without technical background, they see themselves, they are able to create a workflow. They are created, able to drag and drop and create business workflows. So they start building things and a lot of things in ungoverned way and uncontrolled way. And because it's simple, things do work and but they might work for a short term or they might work and not a compliant way, you know. So that is a real challenge because you can end up with a lot of things that are not standardized, a lot of things that are not set properly and secure, which could cause damages. And it sounds initially in the beginning, especially for the users, because they feel they did good and they did well, actually, you know, they managed to build something and they could automate a boring tasks or they could. Better their way and their productivity by making small apps for certain calculations or something, which is great. But the other aspect, and because they got into it really quickly and they managed to build it really fast. And intentionally they skipped or they forgot and they overlooked other important aspect for the organizations related to the governance security compliance. And yeah, this is something has to be like thought of in a way that it has to be balanced not to discourage those users who are active enough and safely enough to take the initiatives themselves. No, actually, we should nurture them. We should support them, and we should not put rules to kind of restrict or discourage them. Not at all. But just finding a balance between staying compliant and governed as well as encourage those who are willing to. Yeah, but leaving it as it is, it will be kind of more a trap, you know. Yeah. To get into things really fast. That's one. The other challenge. I would say is. Doing things because it's easy again and fast. But reaching to a certain like implementing advanced or sophisticated features sometimes requires you need to hook it up with, let's say, as your functions, which is a completely code, you know. So you feel like some users might get disappointed, like, okay, I managed to do some screens to create some buttons to run some workflows, but now I need to do some processing and let's say my flow is timed out or it's failing or no, I need certain operations that that needs to be done, but I cannot do it. I don't have the skills. This is understandable, but when you reach a certain sophistication in your application, then you need as a business developer or citizen developer, you would need sometimes help from the technical team or a developer in order to implement those advanced scenarios.

So that's might somebody sometimes be kind of discouraging, like when they realize that we cannot build it all completely from A to Z on our platform.

Speaker1: [00:12:58] what would be possible measurements that you would apply or introduce to tackle these challenges.

Speaker2: [00:13:08] Yeah. The early planning of these services within the organisation and number one acknowledging that the services are there and not blocking it because they are visible across all, let's say Microsoft 365 they are visible in SharePoint in OneDrive. So acknowledging that these services are there and preparing early on that we will have users who will be curious to start those. So considering these audiences and making it easier for them to follow our standards and how to build applications or keeping the channels open, you know, not just it's there, we are not doing anything about it. Now, prepare some governance, prepare some certain framework on how things should run. Responsibilities. If you build something, then you should be at least documenting it. So others would be aware about it. So have certain, at least a minimal process around how to use Low-Code platform. That's like I feel it's essential for all organizations. The second thing is, it's always good to have even though it's a self-learning. In order to get into that mostly self-learning you know people trial and error just read a couple of see a couple of YouTube or some presentations and they get into it. But it's always good to have not a short course, let's say a crash course. Where would they get a small proper intro in order to set expectations? Right. Because like, not everything is a bed of roses. They can build it all. And it's just that is like set expectations, right? So you will have certain limits within this.

Speaker2: [00:15:09] You have to be aware of of that. So when you design the application as a user or as a citizen developer, you would be aware that what you can or what you can't do, as simple as, let's say, external sharing. I would like to build a small applications, let's say, within powers, and I would like to share it with the external world that they can easily take a picture of the images and send it to us. As simple as it sounds, there are certain limitations where you will run into, such as licensing or extra costs, where then traditional development would be a better solution actually of that, even though you can build it completely internally, you have the camera component, you have the screens, you have the storage, you can do it from A to Z. But then a small aspect, which it's not actually unique to develop it, it's just external sharing, you know, a

log in that will sort of change the whole design of the application. So the awareness is really important and it's not enough to just that we rely on the users. Knowledge from the beginning based on whatever resources they had. So also spread awareness is really important, you know, and then again, this doesn't have to be one week course or very intensive. No, just basic intro about the whole platform. That would be good.

Speaker1: [00:16:38] And. If you think about the way of software development, you mentioned that it might be a difference that the team is much smaller in the development process. How do you think this might fit into a scrum process like this? Classical tools and platforms and events?

Speaker2: [00:17:03] Yeah. Well, scrum and save. We did the training in company x, but also I did the training even before I did the certifications of Scrum and Safe, even before joining was. It's a lean and clean way of doing things. I feel it instills and enforces the concept of responsibility. You as a member of the team, you are responsible, you have duties, you know and you have to have contributions within the team. When it comes to local platform, as I said, it's a small team. So you you have, I would say, greater responsibilities. You don't have just like traditional software development. You create a couple of or you have this small subsystem or you have you you need to just develop some features. No, you are building actually sometimes full screens within the application. So the amount of responsibility is, I would say is greater on the part on the local platform because you are working within a smaller team. So each individual contribution matters and it could be up to 30% of the whole project or 40%. So that somewhat brings more. A word more. Yeah. More. Not obligations yet obligations by the developer because they know that if they didn't deliver or if their part is not being developed properly, then it could affect up to a 30-50% of the whole project. And like the traditional. Okay, I'm just a small feature. I can comment the code. We haven't delivered this additional optimization, you know, we haven't delivered this like button, you know, for instance or this is small feedback for no, unlike that, it's like your individual contributions matters much bigger and a bigger scale. It has the flexibility aspect because again, even in Scrum teams, which are supposed to be small, it's say 5 to 7 to eight people. If everyone and everyone is entitled to their opinion, if they take two or 3 minutes to discuss their point of view, which they have every right to do, so, you know, it could drag longer and longer and it's fine. And sometimes these discussions are fruitful and they could spark some ideas. But smaller team, I feel much more focused. Laser

focus on the topic on the target. Things are much clearer, easier to communicate. It's easier, always easier to communicate with one or two persons than larger group. Regardless the personal relationship between that team, it's faster to work with one or two. Now the quality will differ, but I feel it's not a fair comparison because there you have different technologies that you are working with. While in the Low-Code platform, you also have different layouts, different user interfaces, different capabilities to work with. But if we say about. Quality. Because it's more smaller team and more focused. You might have some. Like not challenges, but you need to be. Like wearing multiple hats when it comes to testing, when it comes to the development itself, as well as the administration aspects. Because as a developer, let's say in the if you are a C sharp or expedited code developer, you will just build your code and that's it. While if you are a power platform or low code platform and you want to use a component, external component, you have to have a knowledge at least about the licensing in order to tell the people, okay, I use these additional components, they are a separate license, they are not within your package of, you know, so you have to know more a bit and you have to have more responsibilities and you have to be well rounded, you know, more than the the standard or traditional development.

Speaker2: [00:21:35] And this is good because also it brings more expertise for the persons who work in the Low-Code platform. Yeah. As for amount of efforts, there is some challenges for the resources. When we talk traditional software development, you have hundreds and hundreds of libraries of frameworks, of open sources, of 30, 40, 50 years of accumulative knowledge that are out there, you know, which you can use copy frameworks entirely, copy classes entirely and you have a lot of while in power platform are talking about a couple of a few years just of accumulative knowledge and most of that are let's say not fully open source. I mean, there is a good community, but it's not incompatible, you know, so you need to do more research. You need to do more work by yourself in order to develop advanced functionalities. But this is balanced out with the simplicity of building thing. When you create a menu, you just drag and drop an item. When you create a button, you don't need to write the description of that button. You just drag and drop, for example. So it's a matter of a balance. You don't have that community to it's it's growing and growing very well, very fast, steadily. But the history is just very large, you know, like centuries actually, or three decades compared with a few years, handful of years now.

Speaker1: [00:23:13] And if you think about that the local developer might be a business related person who is just developing part time. What do you think? How could a scrum approach be suitable like with a daily with maybe they do not even have the same product goal because each of them is developing their own application.

Speaker2: [00:23:45] Right. I see this point for sure, like because the low code platform will be I mean, the audience of that local platform will be. Right in. Between it as well as the business users. They do it part of their job and sometimes for their own individual agenda, just for their own department or their own team they work with and they may not share. Of course, the like the bigger vision of the whole corporate, I see it in two ways. Number one, I see the favor or the advantage. Let's say this application is needed for the corporate who knows the business best. It's not actually the technical developer nor the I.T. it's that person who knows actually what is really needed. And they are they might even not really write down the requirements, negotiate requirements, because they are the requirements themselves. They reproduce that. So they are the best people actually when it comes to what the business needs and what the business want. And this is a great advantage. I mean, we have built tons of applications, not empowers like generally as like it built tons of applications that never met the requirements or just forced the the business to change their way of working just to adapt with how Salesforce work or how Oracle or its solutions work, actually. So but in this side, the business actually built what exactly they want and need, and this is a great advantage. Now, this also brings the challenge that within Scrum or Safe. Let's say frameworks get agile or scrum. Sometimes you need to do the daily, let's say, and you are in business. Let's say you are in marketing. You are a procurement. You might not have the time, you might not have really the interest within as like how things are running when it comes to a system like they don't affect you in in any way or shape or form. You don't have the interest to listen or. Yeah, that is completely separate of what you have built or what you are doing. So you didn't see a value of sharing or communication or hearing that side. It might be beneficial in the long run or at some point of integration, but it's not easy to realise that early on you will feel that you are fine on your own. You do want to be participating with other things or other events, let's say, within Safe or Scrum, because you feel like you are good on your own. And this kind of maybe encourages the Silos or the isolation of things, you know, where you would have or other colleagues could might have already built exactly the same applications with the same amount of time while it's already being built by you. So that is less sharing, maybe. I mean, of

course, such things can be overcome by having a proper community within the enterprise. Some, let's say, champions, if we call them, or power users who would run, for example, monthly or every quarter about showcasing what we have built in the local platforms, what our actually business users have contributed. So share awareness. Yeah. But involving those business users into the scrum also has the advantage of stronger connection with the IT. Because now they speak almost the same language. When you work with Low-Code platform and you have built an application, you really experience what the developer has been building, what the administrator has been configuring. So now you understand their point of view. When you talk with them, you don't say, I want an application that runs today and tomorrow, or I want an application that is being published on my web on my mobile phone. No, you would at least understand that. Okay. My application is not really scalable. You know, I used when I built my local platform, I use this certain resolution now. It's not fitting. I need the help so it won't run. Even if the managed device colleague is able to install it, it won't run properly because I understand how I built my app. So there is there will be a common understanding with the IT and much stronger like it's not layman's terms, it's not just normal business talk, but you already experienced first hand how applications are being developed. So that brings you closer. It brings actually you as a business developer, it brings more credibility for you to the IT because you are no longer sitting out there in the business just giving requirements. No, you are actually hands on and you built something, you know. So you talk from experience, you talk from real challenges. Not at all expected from you to solve all these technical challenges because that's not really your job. We are happy and grateful that you are participating, but that brings more credibility and more voice to you to take everything into account and support that. So it has like, you know, multiple aspects to look at it. Not really positive or negative, but like each situation will be different. But when it comes to scrum and safe, I see that it will be leaner and faster within those teams. But within those who works with two or three small group, they can implement between themselves, especially if two or three colleagues participated and collaborated together. But involving them on a bigger IT might not be easy to have them involved in every single event or every participation of daily they could come, for example, in the PI planning, that's for sure. It's the forum. It's the place where there is a good collaboration for the overall, but not maybe every single step, not let's say the retro, you know, because of the relevancy, it might not be completely relevant at all for them and not beneficial.

Speaker1: [00:30:34] And what do you see if you think about maybe adapting a Scrum model to the needs of potential citizen developers? Do you see any need for adaptations in the roles that are currently used? Like the product owner? The product owner, a Scrum master, maybe a developer? Are there any additional roles needed or?

Speaker2: [00:31:09] I'm very fine and supportive of adaptation and customisation. I mean, let's say the institution of Scrum or skilled Agile, they say this is our methodology, but it's not a single pants that fits all organization. No, we should be able to change certain settings. We should be able to drop things that are not necessary. I don't want to have ceremonies or celebrations of every single milestones. No, that example doesn't comply with my corporate culture, neither with things that I'm developing. So we should be flexible as organizations to change things. And this is important, I think, when it comes to the applying scrum, that is where there is a room for improvement. Where I believe we should not take the business into the IT should be the other way around where Scrum should fit more and get closer to the business. So the Scrum Master should not be a typical I.T. person. It could be actually one of the business owners where he is the one who is like have a bigger picture. You know, not the whole review, but a bigger picture of the necessary solutions are running. They know their team. So that could be from their side as well. And it could run entirely in their organization with a small extension if they need help from the infra from the i.t. Infrastructure or I.T. or advanced development that would require it then that i.t personnel would be attached to the scrum within the business organisation. It's not the other way around. Not IT should be running it and dragging the business and involvement, hearing, making the business hear about servers, about firewalls, about network connectivity, which they and they don't understand nor really matters much to their. Daily Work or yeah, so I see that it has to run more within their organization. As for the roles themselves. Uh. They can be they can be adaptive as part of their work, not full time, you know, as you can be like the scrum master, but you can be also the person who develop. One of the solutions as well. Doesn't have to be a dedicated full time or like the full responsibility because it's in the business. You still have your other work to be working on. And, and maybe this is also important for the, like the corporate as well to give more flexibility to the business teams. So they encourage their like the technology and the development of their users or of their employees who are not in the i.t or engineering aspect. But on the other business side, just to be aware that things are moving, the trends moving a lot towards technology and things have changed. We run and build applications and we do things.

So yeah, technology plays a role in everybody's life and maybe there should be more attention to support these kind of users to help them faster. But have I seen like on the other hand. Have I seen fully running Scrum Team within the business? I can't say I have. You know, it's not. I always seen the other way around. We always drag the business into the IT organization and to the mainly technical meetings. Yeah. Which is good. Which is million times better than the past. Where we are two separate, isolated, we just hear requirements. We deliver them. No, now they are better. They come to us. We have a common platform or a common stage to talk. Yeah. But also maybe getting closer to the other side would be better or worth trying, actually, you know, giving it a chance to, to see the outcome of that. I mean, this one works, but maybe getting closer to them would be also better.

Speaker1: [00:35:41] Where do you see the boundaries of software development through the end user. For example, low code development limitation or maximisation of software development through the end user.

Speaker2: [00:35:59] Yeah. Well, the limitation could vary. Sometimes the organization would limit or block certain capabilities which affect. So they would hear or see or read certain blogs or see tutorial videos that things can be done. Yet the organization are blocking those capabilities. So you are losing potential here. You know, you are losing as well as an organization. You are losing ability to better your business. The second one would be skills. And this is quite understandable. Like because if you are. A marketing person or if you are in the core business. You are. You have so many hours. We have only 24 hours. You have your other. Yeah. So. And. Sometimes learning is not. Sometimes most of the times learning is enjoyable. It's one of the joys of life, you know, growing and experiencing things. But you have your own responsibilities and this is additional load on the person to grow and learn. And when it comes to technology, there is always a new way and better way every single day. You know, there is always a new best practices, there is always a new features. So keeping up with that in addition to your existing, that is exhausting and that is actually very demanding. You know, you could sometimes, unfortunately, you might to sacrifice on this side of the side, you know, in order to there might be some compromises. But then others with passion, dedication, and there are actually quite many. They can do it quite well on both sides and they grow and they develop on that side. The third one is also to the skills where. No. You reach a limit where you would still need a trained developer in order to build

certain components, let's say, especially with advanced scenarios, you know, very advanced scenarios. You would need the involvement of others. And besides the skills, it's also dependency, as we say, Low-Code platform, it feels like you have a freedom. You have the entire ability to build the things all on your own until you hit a wall. Now you are dependent on identity management and then you will have to wait for their processes, their administrations, their, you know, their way of how they handle your requests or infrastructure or licensing. Procurement. When you feel like you are depending on other things, then you will be waiting. You will not be able to progress. So sometimes you reach a limit where there is. Yeah, a dependency of course, unlike the traditional development where it's all about dependency. Dependence. It's just part of the job. Part of that. Yeah. You can't do anything or it's all that. Dependency is just accepted as a norm while in the low code. It's not the norm to have really dependency.

Speaker1: [00:39:18] And now we have some general questions about Scrum. Where do you see the advantages in the Scrum method in general or also for for software development through an end user?

Speaker2: [00:39:39] What I see come again.

Speaker1: [00:39:41] About the advantages of Scrum mythology. In general or also in terms of local development through the end user.

Speaker2: [00:40:00] Well, scrum and agile and safe. Regardless the naming they all about. Lean shipping fast. Drop the waste and just being focused on delivering value and low-code platform. The main premise of it is just delivering value as soon as possible. As easy as possible. So in the mantra or in the main concept, they are very much alike together. And yeah, they are very close together from that concept because local platform is really fast to deliver, to develop things. And this is exactly what Scrum wants to, to push the value to the organisation, to the business as soon as possible. So there is this strong connection. Quality of work and I mean, the speed should not be also doesn't mean we are compromising quality. No, not at all. Because the platform is simple. With the local platform or relatively simple, you can build really good quality with with reasonable amount of time. So sometimes we hear, okay, I can deliver an app within one two weeks. There is unfortunately older mentalities or, you know, the previous school like, okay, the quality then will not be that good. We haven't used to

that. Or we think the maturity is not that good. No, that doesn't really compromise the quality at all. The focus. You know, the like when you talk about backlog scrum or say backlog is the core backbone of how features are being delivered in a local platform. You would have a shorter focused backlog. Everything is just relevant, not really dependent on your application on other things. So it's just entirely matching your product or whatever you are building because of, I would say, a smaller scope, more focused. While in the other traditional software development, you have a lot of dependencies, that the backlog is quite large, the team is large, you have maybe people from quality assurance testers, you have architects, you have different roles, their own different responsibilities, their own tasks, where in the backlog you are working with two or three colleagues who are just developing as you screens. So it's just smaller, more focused, precise and concise. Just tried to fit your Applications or what you are developing. Then again scrum or say if it's about the business and. Involving the business while in the local platform. You are the business per se. You are the business themselves being building the i.t solutions. So this brings it more to home rather than just getting closer to business. No, the business is actually doing themselves. The business are taking initiative, mitigating risks. Scrum is also and agile about mitigating risk. That's why we have shorter sprints. We build things and ship them in two weeks. Show it to the business. No, this is bad or this is not really meeting our requirement. Okay. We have lost only two weeks in the past. We lost two or three months in the cycle, the old cycle of software development, two weeks. But when it's running on a low-code platform being developed by the business, this risk is slim to none because the business knows what they want. And it's like almost I wouldn't say impossible, but the risk is way smaller to develop something that the business doesn't need. You are the business, so you will definitely build something that is needed for you and this is what you want. While the it still there is a chance, even a small chance of risk running into building. Not really something that is matching or something that's fulfilling. Efficiency. Again when within it's all about efficiency and lean. So again, with the business, if they are, let's say, skilled enough and they are building the things enough, less dependency, it's themselves who do that. But then again, it's not really I'm not really overlooking the role of it because this applications needs to run within tenants, within environments, within tools that are supported by the I.T. So the it has to be involved in a way or another, not as a controller but as a collaboration. And this actually matching on both sides. I mean, this is what Scrum and what Safe is preaching for and this is what encouraging, you know, so like I would say stronger bond and collaboration between

the two is, is always good. But I feel like. The value would be more realized in the business when they are. Run it themselves the planning as well of solutions. Let's build these features or let's. Think what problems do we have when we communicate with all that way of doing things with it? Even within Scrum, we have to bring history. We have to explain why we are building this, why we need to bring examples and stories. If we are building this within Scrum team of where its members are part of the business. They all know the story, they all know the pain points they don't need to talk about because they experience that. So they have more focus on the priority and they know what is really needed more than. Like the. Got it. Who knows nothing about that department or what challenges they have had. So, as you can see, it's just a. I feel it's easier on the business side but it has but that also running the risk there is okay we know the business we can now build an app we have we are empowered to do things so they might overlook important aspects related to security, governance, the corporate policies of running things. And they might forget simple things related to basic technology things. Just as I said, external sharing. They might build an app completely internally and at the end we cannot really share it. So. I would say maybe it's all about balance between the two. But give more power, give more strength, give more opportunities for the business to grow on this side.

Speaker1: [00:47:25] One point I'm really curious about is like if I think about a classical scrum team, there are most of the team members are some kind of seniors or do have experience. Do you think like. If the business brings the ideas and the motivation. Does the team need a kind of an enabler who does have the I.T. skills and the knowledge about the integration ideas or testing ideas more? A kind of a coach.

Speaker2: [00:48:08] Yeah. This is really important. Again, it comes back to the decades of accumulated knowledge when it comes to traditional software development. As I said, there is a length and there is already an established best practices, patterns, better ways of doing things. We have learned over tens of years of doing things and in Low-Code platform within the business, it's just a handful of years old. It's still maturing. It's not like not fully mature, but at the same time it's not really a preview or it's not really not unstable. No, it is stable and it is outside mature to a certain extent, but still maturing. So definitely the i.t or the technical members of the traditional i.t. When they join such scrum teams, they can bring a lot of value to the business and to the scrum that is running, let's say, within the business. And that's why we see, for example,

trends of, let's say, Microsoft now in the past and until now they have these frameworks for, let's say, Visual Studio, where you can write your code, you have the unique testing, you have, everything is there. Now they are adopting and they are building extensions for low code platform. So they are also trying to bring what we have as a patterns and best practices from the it make it fit into the scenarios of the low code platform. So bringing those learning experiences. Into testing. I mean, yeah, as I said, like the shipping of those software CDCI, like the continuous integration and continuous development, scaling it down and making it fitting the Low-Code platform that is being done actually by larger providers like Microsoft and others in order to share this knowledge and share these practices with the low code. So it's not just. They can run it on their own 100%. No, there is a lot of they can, but there is a lot to value. That i.t. and senior scrum like senior i.t. Members within the scrum. The traditional way they can bring a lot of value into the low-code platform. But at the same time. It has to be done with, I think. Certain sense of responsibility. And what I mean by that is. I don't think it's right for let's say or let's face it this way, it's all about collaboration. So you take best of both worlds, the low code scrum and the old, not the old. The classical software development. It's all a combination between what is best and delivering a value for the enterprise. I don't think any of these should come with stronger authority. Like, yeah, I know at best we have doing it, but for many years this is how it should run. This is how you should adopt it. No, I think they are equally should be responsible. They should equally collaborate about this. It's not about really a source of authority, only this or because it says this, it has to be this way or because the business says this. Then we neglect other aspects that are very important for the corporate, again, like security, governance and things. So an equal responsibility both ways. Discussions is really important in order to run proper sprints of Scrum. But Scrum is also about master servant, so it's always about being humble, being supportive, collaborative, removing the impediments and the blockers that regardless of where it is, whether from the business or from the I.T, it's all about collaboration and stronger collaboration together.

Speaker1: [00:52:37] I just take a look at the clock and I guess the appointment is planned till ten. Do you need to leave at ten or you have time.

Speaker2: [00:52:50] It's okay. Yeah.

Speaker1: [00:52:53] And like as always, like there are advantages and there are possible disadvantages to choose. Do you see any possible disadvantages if applying scrum?

Speaker2: [00:53:11] Applying Scrum Scrum into a?

Speaker1: [00:53:17] To end user development.

Speaker2: [00:53:23] Yes. I can see some like. And maybe not disadvantages, but let's say like new things or challenges, I would say they are not used to that. Scrum sometimes talks a lot about shipping, shipping, you know, daily meetings. So it could bring a newer behavior, a newer concept, you know, to. To the business team who are not used to that nor even familiar with the concept. I mean, if you talk about Scrum or bring Scrum to the software team, even they haven't used it. It still sounds many of the terms and the concepts are kind of not really vague, not ambiguous while in the business. No, they might not have been used to such behaviors like the stand up in the morning where you see what you have done, what you are going to do next, or what I'm going to do next. I'm going to work on my business project, not even touch my power app or Low-Code platform, for instance. No, I only will give one day a week of my role to the Low-Code platform and my participation. So the daily thing, like, like I wouldn't say Scrum Forces and that's why I said I'm not a fan of whatever the institution says. We have to fit it. No, it's always about customisation really what meets my what is fitting my organization doing it. I have a team in the business who are doing local platform but they can give 10% of their time.

Speaker2: [00:55:03] I would encourage that, not say no, let's do this in the it and you just we deliver a solution for you know I would encourage that and having scrum their weekly meeting instead of a daily it's all about balance between the two. So I don't think we should push every single principle of Agile into Scrum. We should adopt those to what feel to what fits the team locally. Into building things and but brings the. The core of Scrum, which is about delivering a value, being fast, being responsible and being adaptive dynamic. So bring those as, let's say, core essence of scrum into those low code platform, but not making the traditions which is which are beautiful of scrum, which is the daily, you know, being having this strong connection with your colleagues on a daily basis, you know, what is happening and it doesn't have to take 15 minutes, but still

maybe there's 15 minutes are really worth more, I don't know in the business or they don't have that luxury of offering that. So not really focusing on these small things. No, you can run the scrum event without the daily. You can have it a weekly or or say twice a week. Like you can adapt the scrum principles into more fitting and just focus on the core essence values of Scrum.

Speaker1: [00:56:39] So that leads me to my second last question. Like, do you see any other points that might be interesting for my research?

Speaker2: [00:56:53] Uh. Well. Like. I didn't know how. Like that, he says. I mean, are you planning some experimenting? Because, like, it depends on the, of course, the university or another. And also from the topic of the thesis. But also experimenting is always good because it shows you empirical facts in the field. Did it work? Or to what extent it worked or what other challenges actually appeared? We are now sort of speculating based on what we know, what we don't know, what we might run into, you know. But experimenting is something like would be nice and doesn't have to be like, let's run falaise. But actually, I mean, this interview is part of the experimenting because I'm sure you will be also maybe talking with other people from the business or not really from IT. So I mean, that's also part of the experimenting. So combining these and seeing it how like transferred into a real world practices that would be really nice to read not just for the academia but also. I feel like this topic is really important because Low Code platform is just massively being adopted. I mean, Gartner and Foster, which are one of the largest research institutions and there is a lot of statics stats out there, statistics saying that the top 500. Like maybe if I'm quoting right, 90% of those top 500 fortune companies are all are running into those local platforms. By 2030 or 40, there will be X amount, large amount of all organizations around the world adopting Low-Code platform. So Low-Code platform becomes a reality, not just a niche. And then scrum or safe or agile or lean, you know, these concepts are always are growing because they have proven to be a proper way of running projects and running things. So combining those is just I think it's the future because. It combines the both. Yeah. So I'm just really also curious and looking forward to see how this will be translated into reality. You know, with the growth and the, let's say, the skill level being uplifted and extremely jumped from the business side into the I.T. and not in technologies, a lot of the business now know a lot, much more about technologies than before. And yeah, this is, I think, how local platform bought in the first place to target them and to even make

them more into a position where they. Where the values is more realized for the organization. But yeah, definitely. It's just a combination of both. Both directions or both. Things. Which are what? What is the future is about? Yep.