Beyond the Commit, Episode 1: Michal Janoušek - Transcript

In this premiere episode, we explore the software engineering career journey, from individual contributor to team lead and ultimately to engineering manager. Along the way, we unpack the essential skills, mindsets, and expectations that define each stage of professional growth.

Our host, **Paweł Dolega**, dives into these topics in a conversation with **Michal Janoušek**, a seasoned technology consultant and founder of PureBrew Tech.

With nearly 20 years of experience in the tech industry, Michal has progressed from individual contributor roles to consulting and leadership positions across various levels. His perspective on career development, team building, and technical leadership brings valuable, real-world insight to the discussion.

Beyond the Commit is brought to you by VirtusLab & SoftwareMill. Our podcast spotlights CTOs and senior engineers, sharing candid stories that resonate with technology and business leaders alike. Expanding on our popular technology blog (45 k monthly readers), the series adopts a more personal, conversation-driven format.

Want to listen to the podcast on Spotify, Apple, or Google Podcasts? Check out the Beyond the Commit website for details. Below you'll find the transcript of the conversation.

Paweł: How are you today?

Michal: I'm very well, thanks. Like, it's a little bit early for me, but that's the part of it because I have just a freshly born baby girl, so I have to find the time to do this together with you.

Paweł: Congratulations, and, uh, and thank you that, you know, even with those obstacles, fortunate obstacles, but you were able to find some time to be with us.

So, um, today, uh, we have Michal Janoušek, who is a consultant and the owner and the founder of the company PureBrew. Spent nearly 20 years in the tech industry, in the software industry, and worked in a variety of different businesses. And I think you entered, you know, Scala, or been in the world of Scala while it was in its bigger growth in 2010, 2012. You worked with a bunch of different companies. Now you own your own company.

And I think what is interesting for our listeners or viewers is your path from the, you know, from this individual contributor to team leader's role, to engineering manager, and finally to manager of managers or managers of various teams, several different teams.

And also, this angle of, you know, founding your own consulting agencies. Because, uh, if someone asked me about, "Look, Pavel, what do I need to do if I want to be an entrepreneur in, uh, in software, if I want to find my own company?" I would say that, you know, you

probably got... it's not a... like, for some people it works that way, but I think that it's good to work somewhere. And even it's, it's better to work at least two, three different places.

And one of them would be... it doesn't have to be long, but one of them has to be, like you mentioned at the very beginning, the small company where you have to do everything yourself. Because it teaches you, you know, all those things that need to be done in a company.

Then, uh, there is this thing, your episode in a, in a big corporate world is also interesting because you, you know, you won't kind of learn a lot about, you know, how to run the company because it's a completely different setting, right? You, you probably do a small piece of the entire, you know, process or something like that. But it can teach you something, and you can learn about, you know, how big companies, you know, work, what are the incentives, you know? How decisions are being made, who is making them. So, that's an interesting thing.

And I also liked this idea, you know, this idea, your history, part of your history when you worked with Cake Solutions. Because Cake Solution was, uh, you know, a consulting agency, and they work with companies big and small. And they were... like, I worked with Cake Solutions, we had the episode together. I didn't work directly for them, but we were doing the project together.

And, um, I always admired Cake Solutions and Guy Remond as the CEO. Even read a book he wrote sometime, you know, back in the day. And I like their approach to consulting and the software engineering work. So, it was like... I'm not sure if that was, you know, something that is groundbreaking in terms of the, you know, entire world. But it, for me, was definitely a good learning point in terms of, you know, how you do consultancy.

Michal: Like, definitely, I'm so grateful to Guy and for this opportunity which I have had. Like, definitely at the time when I was at Cake, I, like, really didn't have that many experiences and I was questioning some of the things being done in, in a certain way. But, like, now, where I am right now, after some more years of experience running my own business, like, I, I respect that even more than when I was in it.

And I, and I liked it very much, and generally everything worked so well. Like, the quality of the people and stuff which we were doing, like, it was a magic formula. I'm still very close with the CTO, Jan Machacek.

Paweł: Right.

Michal: He's, yeah, he is now making... He has just recently released a game. It's called **Kopi**.

Paweł: Kopi?

Michal: One other, like-

Paweł: I have to check it out.

Michal: It's, it's like this coordination... It's like a coffee shop, and you have to serve people. And I think they have a port for Xbox, for, for PC. And they have some, some agency. It's Dream on a Stake. So, they have been working on this for the last two or three years with another ex-Cake alumni.

Paweł: Wonderful. Great story. I mean, like, you know, what a transition, you know, from consulting in, in an enterprise world doing streaming, you know, even base system, moving to the game industry. The game industry is difficult. I know, I have some friends that work there, and it's brutal there. It's like a competition, it's a different industry.

Michal: Yeah. It's, it's, it's really funny. It's not like talking about it at some point in time. I think they started more than two years ago. And I have heard these ideas and been shown these different kinds of games. But then, when it got closer to release, the type of the game changed over the course from 3D to 2D and cooperation, like, without a shooter.

They had those games in pretty good shape, so they have spent loads of time with them. They'd say, "Oh, no, we don't like it anymore," and they have done something completely different.

Paweł: Yeah, it's... I mean, it's very interesting. It's a story for another time. But I do feel this way, when you do business software, you work with the customers in the business setting, there are certain requirements. It's a little bit different when you do a product, I do understand it. But when you do software services consultancy, you think about the requirements, what needs to be fulfilled.

And in games, I think it's totally different. You can have some sort of, you know, requirements, things like that. But at the end, what counts is, will people play it? It's not like you fulfilling some requirements or something like that. It has to have something in it, something. It's difficult to define what it is, this something. So, yeah.

Michal: Yeah, yeah. So, like, that's what I was saying, I'm super grateful for being able to work for Cake and that level of experience which built on my previous experience just enabled me to set up my consultancy and be successful in what I'm doing, and hopefully what I will be doing in the future. So, I'm, I'm grateful for that.

Paweł: Right. So, uh, yeah, I would like, uh... we definitely wanted to focus on those, you know, different expectations, requirements, in the different steps on your career because we look at it through the lens of, you know, various, you know, stages in the, say, in the management development path from, you know, individual contributor or team leader to engineering manager.

But before we do that, maybe you could give us a few words about how you define the role of a team lead? And I'm asking this question specifically because I think that people intuitively understand individual contributors, you do your work with your own hands. Your output is what counts there.

Then there is this engineering manager that people intuitively understand is more into the people management coordination. But I think the team leader's role is kind of difficult to define what is needed there, what are the characteristics of the role, what is the ratio of the... there's obviously some management, there is obviously some individual contributor roles.

And there are all those names like, I think, having a name of the role team lead. In various companies we have team leads, technical leads, or tech leads, then you have engineering managers. And depending on the company, this might be one of the same roles. So, could you give us, you know, a little bit of your insight about the specificity of the role of team leader?

Michal: Yeah, definitely. So I think it's important to say that it really depends on the company. Like you have mentioned, the tech lead and and team lead, in smaller companies and even for a team lead, I think it's not necessary, but usually it's, it's good that the team lead is a tech lead for that given area and what is being built. It doesn't necessarily have to be the case. There can be a senior software engineer who is very strong but doesn't really want to do the other part of the job, which I can go into.

Like, in the team lead, there is still, depending on the size of the company and what are the other supportive roles, it's like the planning and requirements kind of talking to the stakeholders, which depends who that is, if it's some sort of product owner which is one person, or a product team, or the people in the HR, like providing the work intake, depending on what the process is.

So, like, just how the work comes into the team. The responsibility is kind of making sure that the requirements are as clear as possible for the team. And he has to work with both sides. Like, he has with his engineering team to make sure he understands it, he is able to explain it to the team.

Then like in these different ceremonies, however methodology is being used, kind of make sure that the team understands this as well. So, uh, they can pick it up and work it through. And then like planning in the sense of, again, if there is some project manager working with him or technical project manager or whatever, you still have to look into organizing the work into the sprints, let's say, if Scrum is being used or in Kanban. Like, what's the capacity of the team? These kinds of things.

And maybe even looking... being able to smash or some of the velocity. But, like, not as much. Like, off the bat, I would say that the split in between it's like 80% engineering work, because it doesn't necessarily mean that all this 80% are spent like writing the code. It's, like, much more code reviews of, of the team, setting up and designing the architecture for what is being built by the given team.

And so, this, like, designing, architecturing is a big thing. And it's like what I was saying, the tech lead, team lead in a small team, it's kind of similar, similar thing. I think the definition of a tech lead kind of differs and gets blurred in the smaller companies where there aren't that many people, you don't have that many specialized roles. Because a tech lead can stand on its own, next to a team lead. But then, it becomes more like a principal engineer.

So, like, for me, the tech lead and principal engineer is something similar. So, like, you have someone who goes around and helps different teams, while the team lead is solely focused on his small area of expertise and team who he helps to succeed. So I've talked about the design and architecture and then obviously, like, the implementation, execution of the code and everything. So, like, that's, that's the IC part.

But I think, additionally to that, like, when I mentioned the reviews and stuff, it's kind of working with the, with the team members as well. And it's even like coaching and mentoring of the junior, junior members of, of the team, which in the best setup, you can have, it's all the team members who are helping the newcomers and trying to make them better.

And it should be a collaborative environment where the goal of everyone is to get as much as possible from everyone. But, like, the individual competitors probably don't expect and wouldn't be that keen on doing these things, but the team lead should. Like, he should kind of help his team and all the members to get as much... be better developers. That's probably the best thing.

Paweł: Yeah, so it's sort of a mixture, right, of these, you know, people management and the individual contribution role.

Michal: Yeah, and the last thing, I think, it's, it, it depends what sort of other supporting roles you have, but it's the quality of the software. So, like, some sort of testing. Does what the team is building, does it work? Is it reliable? Doing some deployments and these things. I definitely, I'm up for, you build it, you run it. And having a strong ownership, so that has to be pushed from the team lead onto the team members as well.

So, yeah, like, maybe we'll get into it a little bit later in terms of responsibilities and these things. But, like, in some way, the team lead is responsible for the team and quality. He should be able to explain to his team and his members that the whole team is responsible for these things.

So, like, I said testing, deployment, release management and these things. It's usually on the team lead. That doesn't mean that someone, the other team members can't do those things, but he's the one who kind of keeps an eye on that and manages the life cycle of all those things. Because it's, it's usually best if you have someone who can make the decisions and have the final say, let's say.

Paweł: Right, right. Yeah, I mean, I, I specifically like the thing that you said about the, uh, you know, that you should be somehow involved and responsible for, uh, you know... You said this, "You build it or you run it," and, uh, I think that it's a, it's a generally a good idea in terms of being responsible for the products of your work. And many problems, I think, arise from this, you know, from the fact that there is a separation. Like, you have, I don't know, DevOps team or production team or however you call it. Operations, yeah, operations, who's responsible for the running software. And a different team responsible for building it. I think that it sets incentives wrong.

I mean, I do appreciate the fact that, like, in your case, when you have a young child at home, it's difficult to, for instance, be on support and do 24-hour support on certain days. But, uh, so I do appreciate it. But at a certain level, it's worse to have this feedback loop of, you know, people building software, being responsible to have, in a way, so not entirely, as killing the game, right? That you, that you build software and you are sort of responsible.

And because of that, you know, you're building it more responsibly because at the end of the day, it's gonna hurt you, not the other team. But you'll be, you know, you, you would pay the price because you would be, you know, waking up at night or something like that to, to fix some issues. I think that helps, you know, a lot of engineers, I think, that started with this

idea that let's build something clever. And over time, they understand, they understand like, "Well, sometimes simple is better than clever." Okay.

Michal: Exactly. And from experience even, when we... when engineers are introduced to support, sometimes they are completely fine with it. Sometimes they are trying to fight it so hard. But, in the end, when we have been rolling it out, in the end, everyone was kind of okay with it. Because it was lucky enough that the teams worked with me. It was always like building good quality software, so there weren't that many call outs. And, yeah.

And, like, all those people were pretty, pretty good engineers. But it's still asking for something more. And it's, it's not typically something what you would expect in this kind of a job. If you think about it, they are fairly... many other jobs who have to be on standby if they are doing some things and running important things. And, yeah, so we can't have all the benefits all the time as software engineers.

Paweł: Yeah. So, um, you know, we mentioned this, uh, this mixture of these, uh, management and the individual contributor role for the team lead. Now, I wonder, you know, there are some companies that have these diverging progression paths, and progression paths, or engineering progression, and they usually have something like the individual contributor progression path or engineering progression or engineering career framework, where they do have those roles of, you know, typically junior, mid-level engineer, senior, staff, principal.

And then they have those people management branches, where they have, you know, engineering manager and senior engineering manager, VP of something. Yeah. And, uh, yeah, and, and other roles in that area. And I wonder, you know, in terms of, like, the, the team lead seems like a stepping stone towards the management path, but the team lead role itself, would you put it rather on the management path in, in, in companies where you have the progression, or would you rather put it on the individual contributor path?

Michal: Yeah, I think I, I, I would put it on the management path, really.

Paweł: Like the first step?

Michal: The first step and, like, kind of getting a taste of what the management is and what the different roles are. Because you do much more communication with different people, different stakeholders. You have to mediate anything like what happens in, in the team if there is some disagreement around which library is going to be used or what sort of style and any conflicts, even on a personal level, let's say. You should, you should try to mitigate unless it's like something too tough that you have to escalate some of those things.

But I think it's definitely a good opportunity for the people who would be interested in that to try it out. And I have quite a lot of cases where people just tried it and find out that it's not something for them and they just want to focus more on the technical stuff, which is logical. Because I think it's essential for the people to understand the work which is being done on, uh, the software development life cycle and kind of go through it.

And even like building software, how, how it is. And sometimes people say, "Yeah, like, 'Why does something takes so long?'" or whatever. It's like building a house, right? It's, it's easy. Some of the things over the course of the 20 years, like, which we are doing, are getting

closer to that because everything gets more, more standardized. And you have experience of doing those, building those things. But like, many things are changing all the time.

So, like, understanding this nature of the job is, for me, super important. So, like, recruiting the managers from the actual individual contributors, for me it's, it's super key. On the other hand, it's very complicated because typically, the people who are doing these things, like, they are doing them because they like to deal with the technical problems, solve the problems, where there are some rules that apply.

So, they are not as keen on dealing with interpersonal relationships. And it's just like I have this piece of work which I can go at. Like, it's nicely defined. I have the requirements. I will do it. It does what you have told me to do. And I don't have... like, in an ideal world, I don't have to talk to anyone or to my team lead, and that's it. Just leave me alone to do my thing.

And then this is the pool of people which we have to just kind of upskill or find the ones who are interested in doing more in terms of management, which I think it's challenging. Because, like, I'm not saying that the engineers or software engineers are bad at management, but just by definition, it's the people who wanted to do engineering, not management. If they would have, they would go, like, and study and do some sort of management from, from the start. So, like, this whole thing, it's fairly challenging, I would say.

Paweł: Right. And, by the way, I got to tell you that I have, myself, I have doubts about going into management. Like you said, you know, some people going into management school, you know, going straight into management. You said about, you know, recruiting managers from the individual contributors, from the engineers.

I have some doubts about someone who enters the role as a manager without having this experience hands-on being, you know, of, of knowing what it is to be an engineer. I think that's an important thing.

And it's like... and it's also an interesting thing because I guess there is an angle which we may talk about later on about, you know, um, angle of how do you maintain this understanding of the engineering work? Because I can imagine that if you work in the role for, like, 10 years, some things, you know, some things you forget. And some things, you know, this is a fast-changing world. And some things, by the time you spend 10 years in management, they have already changed in the engineering world. There are different tools, different approaches. So, it's probably another angle that we could discuss.

But before we do that, you know, I wanted to focus on this. You mentioned this, uh, described this team leader's role. And yeah, I definitely, I think I agree with, uh, with, you know, your definition, you know, how you frame this role.

And now going into these transitions, focusing on your transition from individual contributor to team lead, and from team lead to engineering manager. Maybe you could tell us a little bit what were the important things that you did that led you to these transitions, like, what do you have to do? What was appreciated? What people saw in you, you know, your, uh, your managers, people that you were reporting to, what they saw in you that they, you know, promoted you, uh, to these roles?

How can you, you know, yourself, if you wanna... if someone wants to, you know, move, you know, from individual contributor to team leader or from team leader to manager, what they need to focus on? How do you think they could find help? Are there any resources that can, you know, they could look after? So, I'm interested in this area. Could you give, you know, a little bit of detail about this?

Michal: Yeah, so I, I think, like, from the individual contributor to team lead, I think, the first step is coming from the person themselves. Like, we can get into the resources a little bit later. There is always some stuff about what, what you can, what you can do and, and then what, what you should do. But like, for me, it really has to come from your behavior which gets noticed by your team members and the current team lead and, and people, like, within, within the company.

So, like, you have to, like... I wouldn't say fight, but like, you really have to show that you care about what you are doing, like, from technical, technical quality and excellence in the code, let's say, to make sure that you always want to have the best possible solutions. Obviously, in some timely manner or whatever. So just kind of be a little bit of a high performer.

But, like, not necessarily. That's not always the case. But like, I'm saying, you don't have to be the best programmer on the team, but it's always, like, in terms of being noticed, like, you have to do probably a little more than anyone else.

And then outside of the technical stuff, like, you have to express the interest in some of these things, which are the key areas of the team lead, which I have mentioned, like, the ownership of the service. And kind of care about why you are doing things and the features being developed. Trying to get involved and understand and ask questions around, like, why we are doing this, this way.

Coming up with ideas how things could be done in a better way. And trying to engage with the stakeholders. Getting interested in, in, like, being kind of... get yourself more somewhere closer to the team lead and trying to get a say or, like, give a say to the team lead at least of what... how you think it should be done, which is super valuable for the team lead as well to get some feedback.

And then just work together to make the solution better. And just, like, do this in an organized, organized manner. Because, like, obviously everything starts with a decision that you would want to be a team lead. So, like, this is kind of what I was talking about. It's like setting you... setting the individual contributor on their path to becoming team lead, like, once they have decided. It's... I think it's important to try to find out what that is.

But working closely with the team leads and different teams, like, you can see and observe what they are doing. And, like, there are a couple more meetings and these things. So, like, that seems like, from the individual contributor point of view, it's like your team lead is kind of giving you the work and has, like, more meetings than you.

But, like, perhaps you don't see as much other things, like, he is doing because he's not trying to bother you with. Because that's, like, one of the other roles which I haven't mentioned. It's like the protection layer to create the space for the engineers to do their work uninterrupted.

So, like, that's- that's on purpose that it doesn't really overlap. And then the individual contributor doesn't really have the insight of someone else harassing the team lead. And he only likes telling the team like, "And now we will be doing some changes a week later," because it's not really... it makes no difference at that point in time for the IC.

So they don't need to be disrupted that some feature will be done differently because they are doing something right now and there is no impact. So, like, it's kind of this thing, kind of team keeping the team chugging along nicely without disruptions, which I didn't mention previously.

And yeah, like, there are definitely some materials to look at, like, how to become team lead, but I think it's much more relevant in general... like, choosing the management, management kind of path within the software engineering organization. Because it always then starts to focus more on the engineering management roles and soft skills and these things, which I think the team lead needs, but in much less manner.

Because it's... the team lead's role is still, again, dependent on the company. In larger corporations, it's not really as much of people management. You should be working together with the ICs, but it's more about, like, making them better ICs. Kind of helping them to transition from the junior or junior software developers in, like, higher buckets or senior engineers.

So, like me personally, what I have done... Yeah, like, this is exactly what, what I was talking about right now, is kind of super assertive and proactive, interested in, in the project, interested even, like, what the company is doing, just beyond what the team is doing and the project is working. Just kind of general interest.

And that will get noticed by your team lead, by your peers, by the engineering managers. And that's like... it really depends on what the process of selecting the new team leads is within the company. I generally haven't seen anything which would be very well-established.

I think partially that may be because of the background which I have. I haven't really focused on the career path developments as you would have in bigger corporations because you always had fairly senior people with a clear, um, task which we need to do. And nobody was really looking as much into just making some progression.

Even though I, as a small company owner, I need teammates myself as well, because it's better where you have multiple teams or whatever. I can't run everything. But, like, back to the transition, yeah, I think, like, for the team lead itself, I don't think that there would be as much literature.

Paweł: You mentioned that one of the key things is just an attitude, right? Attitude that you need to have, you know, in order to be seen that, you know, people see that you care about a bigger picture, about the entire team, not only about the ethics of your work, which is... I think it's a key point.

The transition is smoother. I think that the more difficult one might be, or the more challenging one might be, to the engineering manager from the team lead. And I can tell you that, you know, I have the experience in our company that a lot of people were struggling in this role. And I partly, I think partially it was because they didn't know what to expect.

So, one, I think that more than one person told me once, that was several years ago, that transition to the engineering manager role, you know, after a half a year of being in this role, what they said is like, "That is not what we expected it's gonna be." But at the same time they also said, "But we would still decide in the same way, we would still decide to go into this direction knowing now what we know."

Which showed a little bit of our failure as an organization that we didn't, you know, paint a very, you know, good and reliable picture of how this role would look like in terms of what would be the activities, and what would be the responsibilities. But most of the people, even though they have sort of different expectations, stayed in this role.

So, I think it's... given that, I think that is a little bit more involved in terms of the changes than the team leader role. So maybe we could expand on that, right? You know, what are the important things, what are the important qualities? What are the attitudes? What are the things that you should focus on both before the promotion and after the promotion?

Michal: I think, like, another thing is the willingness to accept the responsibilities. Again, it just seems very corporation-like company-wise, it's saying like, "Ooh, it's the team. It's the people who always want to willingly do more." But it's... like, yeah, but, like, that kind of comes from some motivation and drive of the person because they really want to do quality work, understand what they are doing.

So obviously, then they are better leaders for that because they simply care about what they are doing. And they want themselves to succeed, and they want the whole business to succeed, and they want the whole team to succeed. And they are willing to help their team with doing so.

Like, with the engineering manager, I agree, I think there are much more resources which I would recommend. And it's mainly about the soft skills. And I think the role depends on much more coordination. So, I was talking to the team lead, you have some kind of interactions with your team, some kind of interactions with the product owner or whatever there is to discuss the requirements or whatever.

There is a little bit of a cross-collaboration because at some time, there are some dependencies between the different teams and the services. So, they probably talk to different team leads. But that's kind of the circle of responsibility. Now for the engineering manager, there is a certain level of coordination.

But then, for an engineering manager, depending on the corporation, we were saying, like, 80% engineering work, and this 80% goes down rapidly. Some of these things which we have mentioned for the team lead, go up. There's planning and requirements, building roadmaps, coordinating the different teams, having the insight and visibility of those teams' capacity. And what they are building, so you can coordinate and build a roadmap for the whole of your organization.

So, that's kind of... so it's much more management and you have to be much more structured and organized and be able to communicate very well with people. So, soft skills. The engineering part, maybe 20%. Typically the engineering manager doesn't code as much, or at all.

It really depends on the size of the organization. It may do some sort of architectural things and designs and define the APIs. But it really depends on what other supportive roles are there. So, then transitioning them from a team lead, I think I agree with you that it's much more ambiguous what the job is actually going to be.

And you are steering much further away from pure engineering work. And that's probably what surprises everyone. Because of this planning, requirements and all these things, I think, go naturally. And maybe even in the general contributor role, you still have to understand what we are supposed to build. And think about it.

And this planning, requirements, building roadmaps, different kinds of skillset, but it's still doable. It just broadens the scale, and you have to lump in different smaller units, fit the Gantt charts or whatever, so you can then come up with when something will be delivered or not. So, there are plenty of materials on the internet on just how to run projects, how to look at the velocity of the team.

So, I probably wouldn't mention anything, but I think the software engineering industry itself is fairly mature in using those agile methodologies and whatever. And from experience, all the... even individual contributors are fairly, fairly up to speed and understand what it is about. It's then more about getting out of the metrics and the reporting and these things.

It's, in some way, a funny thing that the individual contributors are upset when some estimations and these things are being done. They don't want to give them up because they feel like someone tries to control them too closely.

But all this information is like... and that's another management kind of a job of the team to explain to them why this is being done and that's, like, that it's not completely useless. And then as we go higher in the ladder, I think there's a lot of these, you know... a lot of these people are worried about this. And they don't like it because it often seems like micromanagement, like you said.

And I think what's, like, one of the things that's, um... I think that's... you know, when we talk about stakeholders, people for, you know, not necessarily pay money. But, you know, as engineering teams, we report to the stakeholders of various kinds. Like, in my experience, they mostly... they care more about, you know, transparency and knowing where they are, compared to, you know... if they wanted to... compared to how fast it is.

So if they wanted to trade, you know, being extremely fast but having no transparency at all, they wouldn't say it. But typically they prefer in the real world transparency, even if it means that there are some delays that normally happen, but they are, you know, notified upfront. And, you know, they, they, they sort of feel that they have all the knowledge that they need to, to precisely know where they are.

Because they, they also report to someone else. They need to have this visibility to tell, you know, how the larger portfolio is going on and things like that. So in a way, I think that if it's done right, you are right, I think, because, you know, teaching about how important these, you know, this being transparent and having this reporting in line, it doesn't have to take, you know, a significant amount of time.

But if it's done well, it could be done in a way that it doesn't, you know, cost you that much in terms of time spent. I think it gives you even the freedom in a way, because no one is bothering you because they already know what's happening. And you can have freedom to, you know, to pursue the things that you want to do with, with your team in a way.

Michal: I think that's part of the job. And, and, like, for the engineering... for the team lead it's more, like, explaining this and running it on the low level with the ICs to make them happy and collaborative about it. And then for the engineering manager, like, that's one of the main, main sources of, like, information. Which is not necessarily saying, like, there aren't many tools, like, which just crunch those numbers.

But, like, as I was saying, we don't want to evaluate the performance of the individual contributors. But, like, we need to see the velocity of the teams or performance on the team, so we can see, there are differences between teams. It's a bit different between people, it's normal. It's only about knowing how quickly things can be done so we can, like, accordingly plan for that.

So we can get accurate information... we can report, like, upstream as accurate information as we can. And it's, like... it's a difficult thing, like, to manage and, like, make all the teams, like, understand because they all... they'll always be pushing against that, and they, they are worried about it.

But, yeah, I, I think, like, everyone accepts up, up, upstream and any sort of changes. Yeah, like, we can, we can then... I think we are going, like, beyond this question specifically, like, what, what we had or other things from the team lead to engineering manager.

So it's definitely more, more of this. Like, definitely, like, a much broader circle of, let's say, a little bit more managerial roles to talk to. So, like, how to prepare yourself for, for that. I think, like, in engineering management there is, like, a fair amount of literature which I would recommend. And, like, especially from the soft skills because it's, it's much more about the pure, pure, um, people management.

And there is, like, a fair amount of coaching and mentoring for the engineering manager to help the team lead. And, like, you could, you, you could say that the engineering manager is kind of training the team leads in, in a way in, in their job at the time, when they got promoted.

So obviously, the team lead... we should pick a team lead with some experience or whatever. But then the engineering manager is there for the team to kind of help them, help them out, help them, help them grow, help them to deal with the problems, like, they don't know how to deal with, like, within, within the team, unblock them.

And, and, and... so that means, like, the engineering manager has to have, like, a different set of skills and have much more experience. And, ideally, some experience in the team, team leader role so he can understand those problems which the team leader has, so he can provide, like, proper advice.

And, yeah, it's, it's about, uh, these, these books I keep on talking about. It's, like, about motivation, like, the book Drive. And, uh... then the, um... um, from Harvard Business

Reviews. So that was, like, a pretty, pretty good book which I have read on emotional intelligence. So, like, that's, that's another one.

So, when we are talking about engineers, like, they don't really like to deal with people. So, like, I think, like, this is, like, a very important book to, to, to understand, like, both the Drive and the emotional intelligence. And it's, it's more about, like, understanding and making the relationship between the people better. And, and then the, like, Radical Candor from Kim Scott.

Paweł: It's a great book. We also have it in our list of books recommended for new engineering managers and team leads.

Michal: Yeah. So, like, I think, like, it's, it's, like, a very effective book, even like how to use your communication skills and how to manage in an effective way, which will allow you to grow and, like, your people to grow.

Because sometimes, like, you can... we are going, like, to some slightly different problems. But, like, I think, like, it's super important to read this kind of book if you want to be a good manager and, like, deal with the people in a nice way. Like, there is this other, uh, blog with George Lee Orosz. So, like, I, I think, like, he has-

Paweł: Oh, Pragmatic Programmer, right? Or programming, I think.

Michal: Yeah, yeah. Yeah, yeah. So, yeah, yeah. So, I, I think, like, he has quite a lot of articles about the topic of, like, engineering managers and, and, and, and, like, different kinds of roles. So there are even some, like, specific blog posts with his own view of definitions of these different roles, like, as individual contributor and, and team lead and tech lead and principal engineer and engineering manager.

So, like, that's another kind of way of looking into, uh, how- how, how it works in the software engineering world. Like, what are the possible transitions in between the roles? Like I was saying, you can go from the team lead manager back to individual contributor if you find out it's not something for you.

But you can find out that it's actually very interesting to do some of these things. I mean, managing people. And another thing, it's kind of... it's definitely about having more influence within the company. And, like, on one side, let's say more responsibility, but, like, we'll probably talk about it a little bit more.

But, like, more influence to affect things. So, like, the higher you go in the ladder. But, like, it's, it's very different way of affecting things. And, like, it's the mindset change from the IC to team lead to engineering manager. Because you have more influence, you have more control over things, but, like, indirect control.

Because, like, you are not writing the code anymore. The team lead has quite a lot of control in the sense of something going wrong, he can just kind of go, jump in and say, "Guys, like, this takes too... I will... I will fix it, and I will help them. We will push and, and do this."

But then, like, if you go two steps higher, and you are coming from this background and you keep on going through the transitions, and it happened to me as well. Like, if you are in an

engineering manager role, you somehow, like... if you don't, if you haven't trained yourself and are familiar with that role, you can get to the stage where you feel like, "Oh, like, this is, like, wrong. I need to now jump in and help the team with the stuff." But, like, yeah, that's- that's not what you should do.

Paweł: Right.

Michal: Like, you need to make... give the space to the teams and, and, and, and people to solve their problems. Because that's the only... because you don't have the time to help them with that as much, like, that's one thing.

And the second thing, if you would do that, you wouldn't give them the space to grow and solve the things, like, themselves, so they would feel micromanaged and everything. You have chosen these people to be capable. You have the team lead and everyone. So, like, you should let them deal with the things themselves, unless they come to you and, like, ask you for help.

So it's kind... so outside of more coordination and planning and requirements, which I have mentioned, like, that, which is part of the role of the engineering manager, which is kind of difficult to acquire, so a lot of literature. But then, like, you still have to keep on doing it to get the experience.

And it's, it's, it's mainly about understanding the different people and what drives them or motivates them. That's why I was mentioning all those books, because all the people are, like, slightly different, so you have to treat them differently. It's... in my opinion, if you treat everyone the same way, then you are not doing your job completely right.

It depends on the leadership style, for sure. But, like, there are different things, like, which works well for one team lead, and there are different things which work well for the other team lead. And, like, if you know them well, based on how, just how they behave. So if you build relationships and trust in each other, it's, it's... if it's with the Radical Candor, then then you can, then you can actually, like, understand.

You would figure out what's going on, even if the team lead doesn't tell you everything. And, like, you would know what you need to do to, to, to help them out. So that's- that's another thing, which is very different in the transition in terms of the responsibilities. And it's really difficult to... I think I acquired the experience just, like, quickly, just, just to be promoted.

You have to be sometime in the, in the team lead role. And then, like, the jump in between the roles, or to the, to the engineering manager who, like, manages multiple teams, it takes a lot of time. It happened. It takes a lot of time to get... become an engineering manager.

And, like, I think what I was trying to say, it's like a big jump. You really do a role, like, changes so, so, so, so massively. But then, like what we were saying, the engineers, like, who are doing the change from the team lead, like, they don't even realize how, how different that is.

Paweł: Yeah.

Michal: And they still keep on trying to apply the same things as and, like, same behaviors as they have been doing as the, as the team leads. And, yeah, and it's not, it's not something which should be... you should be doing.

Paweł: Right. And there is, there is often this, I think, in many organizations, there is this trap of, you know, you, you typically promote or give more responsibility to people who perform well. So we have this thing that if someone performs well and, uh, cares about the team, they become a team lead.

And then when they perform well, you know, you... sometimes they become engineering managers. And it doesn't always seem to be right. Because, like, I think, you know, I sort of... you hinted at, it's like when you move from, from senior, you know, to team lead, it's, it's not that... it is a change, but it's not as significant. And you still rely on a lot, on a lot of skills that you, that you were doing and you were practicing in, in your individual contributor role.

And it's even more true, obviously, when you go from senior to staff engineer, or from mid- to senior, from senior to staff or principal. You know, you base on the skills that you have. Whereas, when you go to the management path, suddenly you are in the situation that in a way you became, you know, you become a junior again because now you have different skills that are the most important ones, the skills that you didn't practice that much in the past.

And the funny thing is that, you know, when the pressure arrives, what I see, what I saw a lot of time, a good portion of engineering managers start to fall back on the skills that they used in the past and they, you know, they practiced. So that's engineering, which means they start going there like, "There is a pressure, you need to deliver something on time."

And then those engineering managers, instead of making sure that the team is working well, they fall back on like, "I gotta be hands-on because we are not gonna make it," right?

Michal: Exactly. And I think, like, that's, at least for me, it's even applicable generally, like, I, I, I can see that it's almost like this thing, which you are used to doing something and you are so familiar with it. So even with work, like sometimes if you have maybe even something difficult to deal with in your life, you're gonna say, "Oh, hey, let, let, let me work."

And like you would rather stay, like, overtime and doing something because it's like nicely defined work where everything makes sense, you are in control. And it's just an escape hatch in, in some way.

Paweł: It's known territory.

Michal: You've, you... Yeah, exactly, exactly. So I, I, I definitely, I definitely agree, agree with that, that I have done that myself, like, a couple of times. But then, like, that what comes from the journey, like, if you have to go through it and like make the realization that in, in, in that sense, like that's... if you, if, if you think about it, so maybe, like, having, like, engineering managers themselves, like, having the less exposure to coding, like, may make this, like, issue a little bit easier for them to deal with because they simply, like-

Paweł: Right.

Michal: ... wouldn't have this. But, like, that would be, that would be like some, some things which wouldn't be as beneficial like, like that.

Paweł: Right.

Michal: If you don't, don't have the experience, then you, you probably may understand a little bit less of what's going on. But, like, definitely at this, at this level, it's pretty, it's pretty detached from the coding. And, like it, it really shouldn't be done. Like, I think, like, there are definitely cases where the roles, like, overlap.

And then the engineering manager kind of can have the role of a tech lead or or a principal software engineer as well. And, like, he has quite a lot of say on, on the system design and architecture. But it's still, like, if there is a fair amount of people underneath you, it's, it's probably not, like, coding what you should be doing.

Paweł: Right. But, but then at the same time, yeah, as you said, it's like it's probably having some skills in that or experience is essential because, in a way, you know, at least I think you've got to be able to detect bullshit. Like, sometimes it happens that someone comes to you and says that can't be done or it can't be done better.

And you need to understand where it's okay, you know, to accept that. And you need to sort of feel when you need to push it a little bit more because, you know, there is something, you know, more that you could get. And it is, it is in a way important.

So you need to be able to, I think, at least have sufficient knowledge to know where to push. Like, I think that the motivation, like, it's a different, like, different people work differently. And some people need to be pushed to, you know, a little bit to, you know, to, to deliver the, you know, the, the great work in certain times.

So, and you need to know where it's possible, right? It's not, not, not even from the people's perspective, that's one thing, but also the, from the perspective on the engineering, you got to understand what are these, these areas that could be done better. At the same time with a, you know, with a budget that is reasonable, right?

I think that in the business line where we work, it's like people want to have, you know, perfect software. But in reality they don't. Like, if they knew how, you know, expensive it is to deliver perfect software... and I'm talking about perfect like, uh, you know, rocket, uh, rocket science perfect, when you set rockets to the, you know, to the orbit, and there can't be any mistakes because otherwise, you know, hundreds of millions of dollars will be lost or people may die.

You know, that level of perfectionism is not desired because it's extremely expensive, like... so, you got to understand that, right? Which- which one is which, right?

Michal: Yeah, yeah. And, like, definitely, I, I think, like, that's kind of important in this, in this role. I still think, like, the individual contributors, like, strive much more over perfection. And then, like, when you go, like, towards these more managerial-oriented roles, like the team lead or engineering manager, like, that has to come a little bit of realization of pragmatism.

And, like, this kind of value the time, or, like, quality kind of ratio, and like to tune these knobs around. Yeah, like, we have to deliver something. And at the end of the day, if it's not like some of these, like, mission critical software, as you put, and depends really on the given software.

But at the end of the day, like, the business really wants to... wants some functioning software which does well what they have asked for. Like, how it is implemented. And, uh, if it's like super performance, and it's great.

But, like, if it's less performant and still works okay for the use cases they wanted, it's okay for them. And, like, we may have spent, like, much less money and time to deliver it for them, and they would be actually, actually much happier.

So, like, it's another, like, skill which comes, I think, from the experience of doing, like, multiple different projects and delivering them for clients to realize this. That it's at times much more important to, like, deliver something which is good enough, rather than something like what's perfect.

Paweł: Right. In- in a way, as you said, that's- that's sort of funny, because I- I've said that sometimes you need to push some people to, you know, to, um, to get, you know, better performance or, you know, better quality at some point. But the opposite is also quite true. You often have to stop some people from being, you know, perfectionists. For, you know, delivering anything in a perfect way.

And I think maybe, in a way, what you're saying is even more common. This pragmatism coming from the business reality is often what is required, right? Because it doesn't, you know, make... it doesn't make sense to spend, you know... like, we had examples in the past, not only on our side, but I saw in many companies. Like, you know, there were, um, you know, software being built for startups.

They- they didn't even have, you know, a product-market fit. And maybe they have, like, you know, five customers, a total of 15 users. And yet they build something for the scale of, you know, they were thinking about, "How do we handle million users?" Like, you know, when you gonna go... before you go to a million, you probably gonna have the first thousand, and then maybe a few thousand.

And on the way, it would be a good, you know, problem to have, how do you scale at some point. It's probably... Now, it's better to just focus on the business instead of taking another. Because it- it costs, right? One thing is money, the other thing is time.

You know, building something that handles hundreds millions of concurrent users is much more difficult and time-consuming than building something simple that will just handle, you know... like, these days, you can build anything that will handle a thousand users, right? Most of the time.

Michal: Yeah. And typically, and typically it's like even difficult to manage, right? Like, because you choose some of the technologies which may, which everyone in the engineering world may not be familiar with. Because, you know, they give you the best performance and everything. And like, now everything is better as, as it, as it comes through compute and performance for sure.

But, like, yeah, typically these systems which take longer to build are... even though, like, if you have them scaled down, they are more complicated to manage. So that- that's kind of an additional cost.

Paweł: Precisely.

Michal: So like if- if... So, if- if you don't really have the income and like enough clients to just kind of keep on paying, so it's- it's much more additional cost. Like, you can- you can say that, like, rewriting- rewriting it, then again, if you see, like, everything is working, it's growing, it's- it's- it's another cost.

But, like, you can be clever about some of these things, and even with some basic stuff you can get high quality performance. And then it's like a nice problem to have, to have something, which is small. Now it's growing, making money. So you can invest more to make it more- more- more performant.

But like, but it's usually with the clients, like it's sometimes it's very, very difficult to argue with them. But it's- it's the part, like, of the, like, engineering manager's jobs as well, like, management of the stakeholders and everything. So it's kind of a parallel in between, like, consultancy work and work in corporations and like the roles.

So in a smaller company you have clients, in a corporation you have the stakeholders in the organization. And in some way the smaller organization almost works like some small- smaller companies, like within the org, where they have to manage different stakeholders.

So in- in some way, like doing these things, I think, like they are **fairly transferable** in between the small world and the corporate world. Like that is much more about what we are talking about, like engineering management and soft skills, and like management in general.

I think that's much more specific to do- to do corporations and bigger companies, where you have bigger teams, bigger projects, more people being involved. So, like that's kind of the specificity about the role.

Plus, plus for me, we haven't mentioned what's the difference between an engineering management- the team lead does a little bit of it. But like I was talking about capacity, management and these things, it's even like the engineering manager sees that all. And like it's even the question is, is this team big enough?

Like, are they performing well? Do they actually need some additional members? So like the engineering management's role typically is and like the transition and the change in the mindset is like being involved in the **human resources** kind of, um, part of the job.

Like deciding. We have some open positions. We have some more people. Do we need another team? Like helping hiring that- that kind of team. Like team leads and ICs, like they do that as part of the job. Like they kind of go and interview some candidates, like if they are a good fit into- into this team, right?

But like and they can say like, "We probably need someone- someone else, some other person because we have plenty of work coming our way." And they'll- they say to- to the engineering manager.

But ideally, the good engineering manager should see it, like and and should see on the roadmap that this team would be underutilized so you have to plan like one more person. And already like giving them that one person, like I had this new additional work coming, so because you have always like some lead time until the new hire gets, um, productive.

That doesn't happen immediately. So look, you have to keep on planning ahead. And that's like another skillset and responsibility within the role of an engineering manager in my opinion. Because like he has the best insight on- in- in- into the teams. And that's maybe something that people wouldn't typically expect. It's- it's part of the job.

And so each kind of mindset in my case being like some small business owner and like managing different projects for clients, like having enough- enough people. So like tough ones. But like that's- that's kind of closer. If you are talking management, this one is another bucket of skills, like running a business I would say almost because like you- you need- you need this, and that's another difference between the team lead and engineering manager.

And definitely, it's just specific to corporations to, um, kind of evolve and and support their people on their path like through the company. So like, working for corporations, I think it's interesting for the people mainly because you have some paths of progress and you can get promoted and the company will help you with that.

They'll help you to grow and it's another responsibility... of an engineering manager in some way, like to help grow all the people who are reporting to him. And even like kind of setting up some goals for a given year and then evaluating it with the, with the, with his directs and and these tasks. It's just like again, another skillset.

But everything kind of builds on top of the strong communication skills and like building the relationships and trust with the... with the people who you are work- working with because then everything kind of goes much better and effectively.

Paweł: One thing that is difficult for new engineering managers is this responsibility expansion. And I'm talking specifically about being responsible for the work that you are not doing directly, and more than that, for the work that you might not be able to fully grasp.

Because, uh, what I'm saying here is like when you are individual contributor, and as you said, like in the- for the team lead, team lead is so close to the actual work, the actual hands-on work that even if someone's typically, you know, goes for a vacation or whatever, disappears for any reason, you know, they can pick up, team leads can pick up the work and, you know, fill the gaps.

It's a little bit different for engineering managers, mostly. I mean some do that. But especially if you have a larger team and you have like six, seven, eight, ten people, it's becoming much more difficult. You are not exposed that much to that work.

Sometimes you don't even review all the stuff. You don't even like... some engineering managers don't, you know, only look occasionally into the, you know, how things are changing in the code base, so they wouldn't be able to pick up, you know, hit the ground running if, if, you know, they had to be hands-on from, you know, from the next day.

And yet, they have to be, and they are responsible for the work of the entire team. They cannot say things like, you know, when they talk with the stakeholders and their managers, they cannot say things like, "It's not my fault. It's like someone from my team," right? They are responsible for the work of the entire team. And it's difficult for many people.

Do you have any advice on how to, you know, train this muscle of being, you know, responsible for other people's work, even if you don't entirely grasp what is happening there?

Michal: I think like this responsibility piece, like it's, it's probably like the most challenging part of making the transition from the team lead to engineering manager. Like I have kind of touched upon that, you know, like myself, I had some feelings that I have to jump in.

But I think you have to work with the things you can influence directly, which is telling the team, teams on how, how to manage on like the capacity and everything. That's so, so they can provide you with the good data so you have a good grasp of what's going on and where and like how long it will take them to do things.

And then eventually, like if there is a real crisis, it may be that you may help them a little bit more, in some instance with some specific thing. Like it, it does help, but like it's to you to deal with that stress.

But like, maybe not as much with the overall project and responsibilities as we have mentioned. But like it, it will let you get out some of that stress. Like it will help the team which is under pressure. So it's about also adding those teams and like motivating them if there is some issue to push harder and like doing, doing those things.

Like you as an engineering manager you can rely on influence on like... You can rely on the teams which are underneath you and like what sort of data are providing you. And like you can only influence what is the intake of the work on and how you, how you manage those things, like upstream with the stakeholders.

So for me it's like it's the key to really building strong relationships with all the team leads so you can really trust them and you can rely on the data like which are being provided with you. And like they are not really afraid, so like it's leadership style as well, like, being a super like strict boss and, and just be a little bit hard on people may not be as, as great approach because then, then like they would be afraid to come to you to say like there something is a problem.

And they can be like hiding it for some time and the late- later you know the bigger problem it will become afterwards. So like it's still, it's still the communication, building, building the trust and relationship with people so you have visibility and like transparency across the organization.

And then at any time and point like- any point in time you would know what's happening. Like you are still responsible, it's, it's, it's stressful. But then you can report all these things like upstream and you manage the stakeholder.

So like these relationships and trust like, you have to build it. You have to build it upwards like with the people who are providing you with the work, with the stakeholders, with the product owners. Like ideals... and build those relationships.

Ideally it goes best like when you actually are able to deliver something with those teams so building the trust like kind of coming from delivering some results. So they can understand that like, yeah, if they are saying something, they, they, they, they will do it that way or like within the bounds of some, some, some error. That's fine.

But like it's those things like which was the mantra at, at one of the cakes like which we had like under-promise and over-deliver. Like these kinds of things. Like you can build out all these safeties in your managerial process so you feel comfortable.

Which doesn't necessarily mean it's like on purpose saying like we will do less and like you will overestimate everything that it will take too long. But it is kind of being close to what's happening, like and trusting your people that they will tell you if, if, if, if something is wrong.

But then obviously the trust works in a way as well, like if there is really a lot of pressure, like and something needs to be done. And it's really important for the business with some deadline, like you can then come to the teams and like explaining that. And like that's this thing. It's really, really crucial.

And from experience the teams are like and people are willing to work harder for a certain period of time. But like you have to, you have to use that track really carefully because like long term or obviously you will burn down the team. You will burn yourself. And then it will have negative outcomes over the long term.

But like sometimes if there is a push, so like that's, that's kind of one way. And the responsibility kind of comes out of like how you organize your work over the relationships you have with the people. And if you have visibility and everything and trust then there is a pressure but like you can then report up like this is like how we are looking.

And like from my experience as well if you go to the stakeholders and there are some dates or whatever, with exceptions like the dates can move and, and like everyone understands that. True.

So it... yeah, and and and at times it's even like the conversation goes, goes towards a way, um, instead of like the stakeholders like coming and saying like, "When it's going to be done, like give me a date."

So it's, it's, it's a date of your choosing. They may give you some indication like this is the deadline we need it by. But if you come back and say like, "Yeah, we, we, we have done like this analysis or whatever and based on work which we already have planned and like then doing this we think it's going to be like this."

And then like you, you negotiate with them and they, they may understand, they may not understand and they would tell them like then we have to like discuss the scope or whatever. And kind of these are the skills which I think help relieve the stress and like of the responsibility.

Like you are, you are responsible for those things for sure, but like you are only responsible for what you have negotiated for yourself. And somewhat in my opinion, everything is negotiable, and you are in... like you can influence up a little bit. And, and, and then anything under your teams.

Then what helped me as well is the peer groups of other engineering managers or, or people who are in the same position as, as, as you are. And just like discussing how they are trying to deal with some of the problems and just generally venting about some frustration and things because that's super, super important.

Because everyone kind of gets stressed and needs to vent out, like directly, and complain about things. But you shouldn't really be doing that. Like, with, like kind of use that to, to channel some anger on, on, on your reports or, or like be... really complain like in a, un- unconstructive way to, to your managers.

And like it's, it's useful to have a mentor, like within the company as a single person or a group of people where you can, you can talk through the things in, in, in some sort of unf- not as formal way. So that thoroughly leaves quite a lot. And then you will realize that-

Paweł: That's a- that's an interesting thing also from that perspective that, um, that's one of the things that companies could do. Like I was wondering about the, sort of the follow-up questions, what companies, organizations could do to help, you know, people with the transition to team lead or to engineering managers, and to help them with the... after the transition in this very, you know, fragile period of the, you know, first few months.

And I think there are at least a couple of things that we have from this conversation. One of them was, you know, this providing this network of peers that you have in the companies. It might be peers, but it also might be, you know, mentors, like a specific mentor, probably one or two, and a larger group of peers.

What I found out, I don't know if you agree with that, but what I found out, we were running a few circles of that, of, of peers. And typically it takes some time, as with any team, for those, you know, for those teams to gel, to, you know, somehow connect together. And what typically happened is on the first meeting people were a little bit shy about talking with... about their problems.

So someone told me even that, like that... the groups of peers of managers, when they didn't build trust yet, someone told me this, "In those groups that barely started, everyone... someone asks us questions and everyone has solutions." In those groups that were working together for some longer period of time and there was some intimacy built already and there was some, you know... that they... the, the team gelled already together, in those groups, you know, a lot of people came up with doubts.

And there were always more questions than answers. And those teams that were not intimate enough and more, more answers to the questions, like almost everyone felt vulnerable to, to say that they also have the same problems that they were, you know, giving, um, some answers. So that was an interesting observation.

Michal: Yeah, I think like it's... like with every new team and like when there are no strong relationships, like people are, like much more competitive and like want to prove their worth as well. So like it's exactly... it's maybe a sign of vulnerability to say that like, yeah, like I wouldn't say like it's really a problem, like let's, let's, let's come up with a solution on this part, so it, it... I look a little bit stronger. Yeah, yeah. Yeah.

Like, with support, like from my experience on the path, like I have been on, like in... for those transitions, like which I have been provided, like it, it was not great. Like at The Cake, like we, we had like this peer group for, um, improving the soft skills, like for the, for the team leads. But like that it was more training-wise organized.

It was partially a reading club. So I, I can... so I would encourage everyone to, to, to do that at, at their company. So we had like bi-weekly meetings where we were discussing some topics. Then at the end, like we, we got another book, which the goal was to kind of read it or not read it, up to you, up to... before the next session.

And then like it was the next session, like we discussed it. And it was mainly focused on the book, like which we have read, which was mostly about motivation of people, the psychology, the management and like different kinds of things. And like discussing the problems, like which we are having, like kind of role playing and like understanding that the role is really wearing those different hats and like always like focusing on one or, or the other thing.

So like that was, that was, that was really helpful. What another thing like which we were doing, it's outside of the peer groups, it's kind of shadowing in, in between the, in between the company, like in between the team lead, like, uh, it was the same, same thing which we have done. Like everyone kind of had, let's say, like some sort of a wingman, so which was the designated person like who can, who can come up and, and, and do things when the team lead is away.

Because everyone likes taking holidays and, and everything. And it was already like some sort of strong person which was on the path already, eventually like becoming, uh, becoming the team lead. Even though they haven't decided or, or, or whatever. But like if they were in that role and like doing those things and like enjoying that, that may have set them on the path of becoming a team lead.

And it's like the team lead is kind of training up, and like mentoring the person who he works closely day to day to day. And, like, I think, like, that's probably the best way of doing these things. And, like, we're talking, like, mentoring and coaching. But, like, leading the team, like, day to day, I think, like, that's probably the best way to do it.

And then- then the guy who was working within the team under someone, like observing what they are doing. Then maybe over time we are figuring out that he would really be keen doing that. Even start, like, doing some things of purposely taking that guy on, on... that person, like, on the meetings with you and, like, give him more and more responsibility. And then eventually taking him out and, like, giving his own team to run. So I, I, I still think, like, that's- that's the- that's the most valuable thing.

Obviously, this is, like, super difficult to do in terms of the other transition from the team lead to engineering manager. Because then the relationship between the two, it's usually not as strong as, like, within the unit of the one team because the daily day-to-day work, it's- it's smaller.

The engineering manager kind of has more responsibilities. But, like, it really depends. Like, I, I have had, like, very, very strong relationships with my engineering managers as well. But then, then, then it comes again. It's another part of the engineering role, it's developing his reports.

And, like, discussing and, like, helping deal with the problems which are more of an engineering, like, manager's nature. So that's... in some way, like, it's part of that relationship. It's already in progress, like, some sort of preparation for eventually going towards the engineering management role.

Because, like, you see, like, how the conversations are going with your engineering manager, you have much more insight, like, what's happening. You are providing the input for the things, like, he wants. You maybe don't have, like, the full visibility, like, upwards into the stakeholders as much. But then that's... that's- that's- that's something, like, you- you can be trained on, like, a little bit later on as well.

So, like, that's... it's kind of what I think works. Obviously, for soft skills or whatever, there is, like, a plethora of different courses. I think for coaching, mentoring and these things, I- I was looking into this and unfortunately, like, they were fairly, like, expensive.

And even though they probably are, like, working and very useful. But, like, at this level, like, we have plenty of software engineers. Like, it was something, like, which was fairly difficult for the companies, like, to afford for their staff.

But, like, lately, with the emergence of AI and, like, it seems there are quite a lot of courses and things. Almost for free or very, very cheap. Uh, and, like, quite a lot of new materials to go through. Like, and to go via, like, interactive manner.

So that's probably something, like, to, to look into. I, I, I think, like, generally, like, teaching software engineering or whatever, like, these days you can find quite a lot of material online. Like, now, it's maybe, like, even let... you know, a little bit bad in the sense, like, there are so many so it's difficult to pick one. Pick- pick the quality ones. So, like, it's either, like, someone recommended something, so it's probably, like, getting better.

Paweł: Sure.

Michal: Yeah. And that's- that's probably, like, what I have in terms of, like, training up people on... like, getting them prepared for the roles.

Paweł: Now you mentioned a couple of, a couple of positions already, you know, throughout the conversation. I think you mentioned these, uh... just to sum up, you mentioned these, uh, Drive by Daniel Pink, then you mentioned, uh, Kim Scott's Radical Candor, then HBR's Emotional Intelligence, I think.

Michal: Yes. Yeah, yeah.

Paweł: I think, like, what, what we are, uh, you know, suggesting people, uh, at our company is also this, uh, Camille Fournier's book, Manager's Path, which describes the, you know, all those roles team leads to individual contributors, team leads, the engineering manager, manager of managers, VP up to a CTO. It is an interesting book because it describes the entire path.

Michal: Yeah, I have it on my reading list.

Paweł: Sorry. You al- I also- we also have this other book, which is High Output Management, it's a pretty old position, by, um, by Andy Grove, the, uh, you know, the, uh, I think original CEO of the, uh, of Intel. It's really interesting. It's a bit of a classic.

And yeah, like, what is... what you also mentioned from the coaching, that's an interesting thing. And the conversational soft skills, because I think it's also important that... you know, you cannot just put, you know, too many books or reading materials because the- this is the- the soft skill area.

It's one of those areas where it's very difficult to learn effectively theoretically. Because you- you read about something. And, you know, they- they say that if you don't practice something, it doesn't really, you know, it's- it's not really ingrained in your brain.

And so I think it's especially true in soft skills. There's so many different books. And most of them and many of them have some, you know, interesting insights. Those insights are often repeated. But it's one thing to read about something and the other thing to be able to practice it in the heat of a daily work, you know, many different priorities, you're often tired or, you know... you know, uh, you know, you've- you- you think about something else.

You are distracted. And it has to have some things ingrained, especially when you talk about, you know, coaching, one-on-one and things like that. It has to be practiced in order to, you know, for you to be able to, you know, use it effectively. And I think that's the difficult part.

Michal: I, I, I think, like, you have to practice it by doing, really. And, like, like mostly getting the experience from- from being at the receiving end, let's say.

Paweł: Right. Yeah.

Michal: If you are, like, climbing the- climbing the ladder. But like, then you are thinking about, um, stepping up eventually. Like, you have to try to visualize or try to- try to, like, think about how it would look if you would be on the- on the other end.

Paweł: Mm-hmm.

Michal: And like that- that's I think like one of the other important things. Like, to just use those- those meetings and everything. If- if you have these aspirations, the thing about it is like this way, if- if you would be doing it the same way as your manager is doing it.

'Cause at times, like I can see, like in the engineers, they don't really enjoy some of these meetings and like some- some of these, like, growth kind of plans and setting up the goals. And they seem... like, not- not- not everyone, but like they may- it may seem a bother to them.

But like these things are in place like for exactly this, to be able to give you the opportunity to- to learn and like develop yourself. So, like that- that's I think like it's- it's- it's super important. And like, try to get as much time as and like value out of these meetings as- as- as possible.

Which I may not have done like myself in the past as well. But over the time, like you kind of come up through this realization that like there is some value in it.

Paweł: Yeah. And I'm like... and also I loved what you said about, you know, allowing people to make mistakes. You mentioned this, because, uh, you know, you don't always have time to prevent all the mistakes in work or something, because as a manager you want to be doing something else.

But I also think that the same is for the managers themselves. You learn a lot from your mistakes, from those, you know, about soft skills, about things like that. And I think it's, you know, mistakes that you have are very valuable.

Assuming, of course, that they are not terminal, I mean, not terminal, you know, directly as a... but a terminal for a company, or at least very costly. And I think that like w- I loved what you said, because, you know, when someone... You know, you see sometimes, especially if you are a new engineering manager and you typically have some experience, longer experience than some of the junior engineers, even in the software engineering, uh, directly.

And you see that some people, uh, yes, they are going to make a mistake. Like you see that as a bad approach. And you got to restrain, in many cases, yourself from, you know, trying to rectify it too early.

Because I think that's like... I think that what you hinted at, is like you have to let people make those mistakes, again, as long as they are not extremely costly or terminal for the company or the, you know, or the team. Because, you know, you really learn well on things that you- that you made... You know, assuming that you have a feedback loop, if you did something yourself and the outcome was, you know, was at least somewhat bad, not necessarily very bad.

And you were able to see that, it's very good, you know, it's very good teaching for you. And I even heard someone saying... I think I read it somewhere, like this story about someone saying that- that when I see a junior manager, junior engineer doing something that might be a mistake, and this mistake is not costly, I usually let them, you know, do it that way.

Because in both cases... There are two cases. Either they were right or they were wrong. In both cases, a mana- as a manager, I win. If they were wrong, they'll- they'll learn something. If they were right, I would learn something. So if bo- in both cases, you know, I won.

Michal: Yeah, yeah. Yeah. I, I, I agree. I agree. And like this part is definitely excruciating, like, or at least like have been for me to change the- to change the mindset. And especially like being used to do different things to kind of help the team and really make sure that it doesn't come down to any too costly things.

But like I, I, I think like if, if you look at it in one way, like that's like when you go to engineering manager, maybe you shouldn't even have that much of a visibility, like to that low level. Like it's probably more applicable for the- for the tech leads. Like internal, like you shouldn't have like that- that kind of visibility.

Like you have some architecture or design, but like if you have more teams, like you typically wouldn't have that much of insight into these things like going wrong. But like I have definitely have had, but like that for me, that's almost like an indicator. You are looking too closely and like what's [laughs] what's happening.

Like doing it for good reasons, but like there should be other people or the team leads and like how you have set up and built the teams, like they should see it themselves, or or learn about it. Like you would assume that like the- the team leads on the projects like which can have like this massive impact on the whole company or whatever, like that there are some seasoned people who- who wouldn't really do an- any- any of those- any of those mistakes.

But like they're learning. It's super, it's super important. Like that's- that's definitely like something which I'm struggling with. Like just give the space, give the space to the- to the people. You have mentioned as well, like the feedback loop. Okay, it's like that's- that's- that's another skill.

Uh, to kind of listen to the people, like what they are saying. And encourage them to give feedback so you can adjust your behavior and implement some changes which would be, like, helpful for, for everyone. It's like the managers and, and can kind of be empowered to do the decisions and like to drive the teams and everything.

But like, they need to make sure that there are very good listeners as well, so they can actually pick up the- the different flavors of things which are happening, to receive feedback on their management style so they can adjust things. They can adjust their behavior slightly when they are dealing with different personalities of a certain team lead so everything gets a little bit more effective.

So that's, that's probably like another specific soft skill for the managers. Like, it's not all the managers are doing this, right? Like, but what I've mentioned before, it's, it's, it's about the style, like, which you want to have. And like, some things are even natural.

So, like, I have some things which I'm, like, struggling with myself especially. And, and this kind of delegation of things and, like, leaving the people to do their thing. And, like, letting them make their mistakes, you know, it's like something which came to me, like, over years and years and years and experience.

And it's, it's still at this point like I know, like, what I should do, but it's always, like, a struggle to, to let it go. So, like, it's not all the things, like, you will be able to change and control yourself. But as long as, as, you know, like, you can adjust some other things to compensate for, for, for your behavior.

Paweł: Yeah. I'm smiling because it's almost like those, you know, sort of the coaching sessions that I'm, you know, that I'm hearing about. It's like, you know, what the coaches say, like, most of the time people already have the answer. It's like, it's like in your case you said, "You know what you should do."

It doesn't mean that you always do that. But if you really think introspectively about things, you know what you should do, right? It's like, it's almost- like the coaching in terms of this classic coaching is like go to someone, tell about your problem. And they ask you, "What do you think you should do about it?"

And you say, "Well, I should probably do this and this." "Well, just do it." "Okay, yeah." "Here is 100 bucks." "Thank you." Yeah, so- that, that is funny, but, um, the- the other interesting thing in that is like, uh, you mentioned this, um, you know, these hands-on work that is still happening on the team leader side.

And then you move to the engineering manager, you become less hands-on. And the question is like, I think it's, like it has its own challenges when people become engineering managers because they already have this working knowledge of the software engineering and the hands-on work. So it works a little bit for better or worse because we mentioned also that as the pressure piles up sometimes people, you know, fall back onto the hands-on work, engineering managers fall back onto the hands-on work.

But it also helps because you have this understanding of the field. But then when you work, you know, a little longer, it might be, you know, a few years or maybe 10 years or so, I think that the knowledge might become a bit rusty. Things change. Like one thing is that you forget.

The other thing is that, you know, the environment is changing, especially these days, you know, introduction of AI tools, things like that. The development today is not how it was five years ago, and it's not probably not going to be five years from now how it is now. And who knows? Maybe the difference would be even more significant than, you know, comparing today to five years ago.

So how do you... Do you have any advice on how the engineering managers or manager of managers in the technology field, how do they keep their working knowledge up to date? And even if they should keep their, you know, this, this working knowledge up to date.

Michal: Yeah, so to start with the last question, I, I like, like let no, like let me start like with, with, with the story and like what we have been over, over already. So I may repeat myself a little bit. But personally, I just like when I am a vendor to this like team lead roles, like that's fine, but like in management roles, managing multiple, multiple teams and things, like I, I always like to start, like try to contribute to the code.

But like we have said, that's something which you shouldn't, which you shouldn't do. And like I paid for that, and I paid for that by working quite a lot of time. And like it's kind of having a second shift after I have done my day-to-day work with the teams and everything.

And like just let's look at something which I can do myself, some- something and a task and to keep like... it was still at the time when I, when I was programming like day-to-day myself. So it was really easy. There- there wasn't... there was just like keeping, keeping the muscle memory going. And and and all, all, all those things.

So that kind of worked well, but like obviously you can't keep on doing this. You know, family comes like some different things. And you can't keep on working. Cause like you burn yourself out. And so I don't recommend it. And you say, "What I've tried," and like, "How I have set up." Like, so I, I, I had to. I had to stop it.

So I, I, I don't recommend doing exactly this to everyone.

Paweł: Cause it, it becomes a night shift, right? When you do.

Michal: Yeah, yeah, yeah. It, it, it, it, it definitely becomes, becomes a night shift. And, it distracts you as well. Like what's... In some ways like it at times like what we have been over, it just gives you too much of an insight into the service, which is more focused on a certain team.

So like you feel empowered to kind of give them some more unsolicited feedback on how things are being done. And that ends up being like a little bit more of micromanagement which then breaks not, not entirely, but like a little bit the dynamics between the whole team and you because they would feel, oh, like, you are just, like, going and, like, looking in their code.

So maybe you don't trust them, like, what, what they are building. And you want to micromanage. And, and, like, they are, like, engineers who know, like, what they are doing. Like, nobody... no one, like, likes this, that if they are tasked with something. And, like, then they have someone, like, just to come in and review whatever they have done, then, like, nitpicking.

And as we have discussed previously, there is some solution, like, you have some opinions on how to do it. It doesn't necessarily mean it's always the good, like, the, the only, the only solution and the best solution regardless. So, like, that's, that's something, like, to be very, very careful with.

But from the background of being IC and, like, technologist and liking software programming, I, I still kind of, like, I can't let go. I can't let go of, like, doing, doing any, any of these things. But, like, you have to be careful.

Like, that, that, that brings me to the last question. Like, you have had like, does the engineering manager really have to, uh, still be current in terms of doing the actual code or whatever. Like, I think like a couple of years back I would say, like, yes.

But, like, now I'm, I'm, I'm not so sure because I understand the role, like, much, much better. It's more about, like, having the experience of doing that work and in generally being into computer science.

And over a certain amount of time, I, I, I think, like... well, it's quite funny, is, is well the... like, there is a big shift with the AI and everything. But in terms of, like, computer science, there are patterns, like, which are repeating over the time.

Like, we kind of keep on going, like, having more powerful machines. Like, different frameworks. Then, like, every couple of years there is a new framework so... which does things better, but it's kind of like a spiral where we are improving and, like, going up. But, like, the same things are going back and forth.

Paweł: Servers are rendering, yeah. Front-end side rendering, back-end side rendering and so on, right?

Michal: Yeah, like, service-oriented architecture and soap and, like, monoliths and, like, then soap and XML and, like, base dollars and, schemas and, like, definition. So it is great, just give it to that. It just generates the clients and servers and everything is fine. We are just working on, on the definitions. That's all. It's bad, like, the authentication, everything's taken ages. We need something, like, which is quicker, faster.

So here you go, like, JSON, schemaless or whatever NoSQL databases, like, that's, like, great, it scales so much. And now, like, oh, maybe, like, the relation databases are, are, are not that bad. And some schema is, like, pretty good.

And, like, do we have some schemas for these JSON objects? It's, like, it can be, like, whatever garbage it is. And, like, it's difficult to, to... Swagger comes and all this. And, like, Smithing and IDEals. And just, like, it's already a repeat. It's, like, definitely better technology. But the principles and everything, it's very, very similar.

Paweł: Yeah.

Michal: Like, it, it has... it has some differences for sure.

Paweł: Sure, sure.

Michal: But, like, if you are-

Paweł: It's, it's a little bit different, like, that's what they say is, like, history doesn't repeat itself, it rhymes. So that's like, it's different, but it's sort of in the same, you know, direction.

Michal: And, and then, like, when you have done those things, and, like, you have some sort of analogy which is not completely, like, exact, there are differences. But, like, you can understand, like, very well of what's going on. Like, then, like, synchronous and, like, asynchronous way of doing things. Like, messages and, like, and, and queues.

Like, they were, like, they were message brokers like, before, before, like, Kinesis and, and Kafka and, and, and all, all these, like, message- message logs. Like, Oracle have been using them, like, in between their databases and, and, like- so it, it's kind of reusing the same technologies, like, making them slightly better, coming up with different approaches and different patterns.

But you can always then build up for the fundamentals which you have acquired over, over, over the years. So I think there is some specific amount of you keeping yourself current of how the technology develops and if there are, like, these big shifts as the AI.

And, like, how that impacts... how that would impact actually the work and the developers' experience, how you can utilize it to make it better. Like, what are, what are the feedback on different languages, on developing languages?

Because, uh, like, being current on the technology stack, I think it's, like, super important from the responsibilities which we have been talking about, what the engineering manager does. So, like, if you need to plan the resourcing, like, if you need to pick up, uh, the people and, and the development team in, like, different roles, it may be your responsibility depending on the size of the organization.

It's, like, even what languages are being used. Even though it matters less and less. But then it impacts the pool of the developers you can hire from, how it... how expensive they would be, what's their availability, what's even the status of the language or whatever it... Is there, is there enough, like, people, like, supporting it, writing it, maintaining it?

So, like, the technology which you have chosen will not die. And then, like, you would have to replace it. All, all those things. So a certain level of keeping yourself up to speed is, is, is definitely important. But I don't think, like, it necessarily means that it has to be... that it has to be, like, on the code level, like, writing code and being able to deliver code.

Like, definitely that means... like, it's a, it's a big choice in, in, in a career. And, like, it may take some time to... change that career. Like, the more time you spend in an engineering management role without, like, touching any coding, it would make, like, much more complicated to come back to some role which is more closer to the individual contributor side.

But I, I, I think, like, it's entirely possible, and it really depends on the engineering manager, like flavor. So like you, you, like, you're, like, spending some time in that role. If you want to go back or whatever, you would have some time to, to boot up something.

But, like, it really depends how- strong software engineer and developer, like, you have been before. But then, it's entirely possible just to come back completely.

So I still like, uh, I keep on saying, like, it's individual. Like, I, I haven't stopped myself, like, doing something. And, but, like, I wouldn't, like, use that as, as, as a recommendation. Like, in, in, in general, it's, it's more about, like, being so passionate about what I'm doing.

So, I still, like, kind of use some small projects or hobby projects or something, like, for friends to do, to find some time as a hobby, like, throughout the course of the week so I can do something. And I can still program because it's something which brings me immense joy.

So, just steering towards a different question again, a little bit. So, like, that's different... like, it's a big mind-shift for the engineering manager of being notable. Like, how to measure success. And how to understand, like, it being successful.

Because when you are an individual contributor, you do some code, it just works. And you get, like, an immediate feedback and, like, some sort of satisfaction that you have done something. And it's, it's, it's fairly fulfilling.

So, for the engineering manager, it's kind of setting up the mindset that, like, generally in the middle... for the middle managers. Like, if you do the actual work, like, which you can see in front of you, it, it gets, it gets, like... in a way, if you were used to doing that, it gets less, less rewarding in some way, like, being the manager.

So, you have to set your- set yourself, like, these goals as well about, like, "Did we deliver at all on time? Did someone get promoted? Did, like, the project were successful and, like, brought as much revenue to the company?"

And kind of celebrate those things for, for the teams, motivate it, but, like, for yourself as well. Because, like, that's, that's the main value, like, which, which you are bringing to the company. And, like, and it's... like, the biggest reward is someone... like, something, like, coming to you and, like, really be grateful about all the leadership and mentorship, like, you have provided him as, as an individual.

Like, that's, that's what I see. Like, it's an important thing to, to, to think about. So, going, going back to the original question, which was like how you keep yourself current. So, I'll tell you about this, like, already nipping in some programming, but it's a little bit extreme. It really depends on the individual.

And I don't think it's, it's required. So, like, that's kind of personal choice how much time you have left. But then, then definitely, like, newsletters and, like, consuming, consuming all the stuff, like, which is currently on the internet, right? Like podcasts. Yeah, I have mentioned, like, the Pragmatic Engineer, TL;DR, the newsletter.

So I do that. Like, there is quite a lot of content. It's probably... it's probably impossible to consume all of that. Uh...

Paweł: Okay.

Michal: Just by, by...

Paweł: I have, like, four hundred, you know, newsletters each waiting in my inbox.

Michal: You have some rating systems, right? So, like, the interesting stuff or, like, people... what things which are being discussed or, or talked about, they come, come up. So, like, if you have a limited time, you just go for the ones, like, which come, which come up.

And with Reddit and, like, quite a lot of the communities are, like, somewhere and you find the right space if you want to follow up, like, with scholars or things like that. So, that's- that's super useful.

And I think, like, then going on conferences, like, development-focused, like, kind of hardcore developer-focused. So that... like, that, that depends. But I don't think, like, it's only an opportunity for, like, business development, because, like, for people in the- those roles, like, it, it could be.

Like, the conferences are quite often, like, business development opportunities. But, like, listening to some of those talks, which are in there to see, like, where the technology is moving for the plants in some specific language, what the people are doing, like, how that works, I think, like, it's pretty good source if you can find the time and, like, really savor the conference and, like, consume the content which is there at, at the time you are there.

So, I was failing usually with that as well, like, going on a conference, like, help working on that, it, like, doesn't really work that well. If you really find time to go to a conference, try to use it on investing into yourself and, like, the technology and, like, updating your mental model of the technology and to keep yourself more, more current.

And then, like, on the... on the coding and programming, into that, I still think, like, there are so, so great courses for different languages or, or whatever. So, like, from time to time, new languages come, come around... come around or it becomes a little bit more popular.

It's, like, super easy, like, throughout the week or couple of weeks, just do a course to refresh yourself. And, like, that, that will, like... make you, make you really current and like-

Paweł: Hm

Michal: ... it comes back very, very quickly. And like if you do that, like frequently and if you plan some, some sort of schedule of things to do, then I, I, I don't think like you may not get

as rusty as you would think as, as an engineering manager. But it's, but it's still something to be done more on, on a personal, personal time.

Paweł: Mm-hmm. Sure. I mean, like there's, uh, I think a lot of, you know, CTOs or engineering managers are doing this. Like, even if there is not a, you know... they cannot, uh, participate or contribute to the core projects that are core products they are building, there are often plenty of, you know, opportunities, as you said, for some smaller apps.

Like for management, maybe you wanna have a one-on-one app which, you know, where you can store notes with your reportee. And then you can, you know, have the next one-on-one and you can look at it. So, many different things, many different, you know, small tools that you can make. And even with the introduction of AI tooling, I think it's simpler than it was in the past.

Michal: Yeah. Yeah, yeah. Like, you can as well, like, go into some, like, code reviews and like reading the code and like helping debugging. I think this is kind of what keeps me interested, like building some dashboards and on, over, over the services like, which my teams are managing. So I have a higher level view if everything is healthy and, and, and I can look and like quickly find out if something is wrong.

Which kind of helps with the support, like in identifying the errors. So... and like, understanding the, if the service is really bad forms. And, uh, and like, I don't see it as a control function, but like, it's always better to have some sort of visibility or like what's happening where.

Paweł: Sure.

Michal: Mm-hmm.

Paweł: Yeah, and after all, you, you mentioned that the control function, and you mentioned these people don't like, you know, when you assess performance. And there are two levels of performance, the team level and the individual one.

And I think, like, people sort of understand this, you know, team level performance, but also, you know, individual performers in a way. I mean, after all, you know, some people report to you. In a way, in a classical structure, you are, you know, um, you are, you know, they are your subordinates in a way.

So, you sort of like... if we imagine the worst situation when the shit hits the fan and as there is a, for any reason, layoffs in the organization, someone will come to you as an engineering manager and ask you, you know, "We need to shed some people." And you point me in your team who that person should be.

And you better have in your mind, I'm, I mean, not instantly, but you better have in your mind some sort of the performance comparison, and you better have this performance comparison as much... uh, you, you don't want to have it to be biased because you like something. You would really like to have a clear picture of the performance even on the individual level.

After all, it's not a, you know, it's not a fun part of the work, but it is part of the work.

Michal: Yeah. Yeah, yeah. Yeah. I agree. I agree. Like that's what it's like. It's another of the HR responsibilities. And yeah, like it may happen, as, as you said, like it happened in, in the past. And like these tough choices have to, have to be made.

Like even firing someone eventually, that's kind of an interesting experience to go through and like you have to prepare and train yourself for that. So, yeah, it's- it's part of the job. And like has to, has to, has to be dealt with. And that, that's definitely something which doesn't come naturally, like for software engineers or, or whatever. Like that's kind of pure management.

Paweł: And often there are some legal consequences. So you got to be trained on that because, you know, sometimes it's the word that you say that you shouldn't have said and things like that, depending on the contracts in the country.

But, uh, yeah, as you said, it's a serious thing. Like no one likes to do it, and hopefully it won't be... you know, we won't need to do it anytime in the future. But, uh, realistically speaking, you know, if we have another, you know, 20 years or something, you know, or 30 years of our careers, it's probably going to happen at some point.

So it's, you got to be prepared, especially if you... like, you got to be aware when you are starting as the engineering manager that it might be the case. And it'll likely be the case, you know, if you spend enough time in this, in this position.

So yeah, you need to think about it. So, uh, but that's also an interesting topic about, um, uh, you know, about the size of the team. And I think what, what I'm trying to say is that I think that's, you know, in pandemic or pre-pandemic, so that was the years when, you know, they basically, the tech industry was on the rise.

And then we had 2022 when it, you know, the layoffs started and things like that. And companies started to optimize in various ways. But before that, I think it was the case that, you know, some people said that the ideal ratio is to have one manager, engineering manager for, say, five, six people.

And then, you know, obviously the big companies, uh, Big Tech, you know, started doing huge rounds of layoffs. Like it might have been, you know, five or, you know, few percent or even two percent. But you know, for those companies having hundreds of thousands of employees, there were thousands of people.

And it... you know, one of the reasons, one of the areas where it hit for certain companies, I think was Meta, but also other companies too, that they specifically targeted, you know, management.

And I see sort of it like including engineering managers, I sort of see some rationale into it. Like if you have six people, you know, in your team, probably you can still make it. If you have, you know, seven or eight people, maybe, maybe even nine, if you, you know. For a... you know, it might be suboptimal, it might be unsustainable for a long period of time.

But for some period of time, it's doable. So I'm wondering, you know, if it that, if you think that's a trend that is going to, that is going to continue or do you think it's, it's gonna be, you know, revert back to these five or six people depending on the organization.

That also includes the changes, I think. Apart from the economy, the changes in the tooling that we have, things like AI. So my question is, what is, in your case, uh, what is your opinion on the ideal ratio of, you know, individual contributors to engineering managers? And the other thing is you think that it's gonna change for various reasons, but for instance because of the introduction of various AI tools.

Michal: The team size, generally, I think, like, for the, for the engineering managers, and, like, I think, for me, the good size, it's always, like, five to six people for a team. We can discuss and argue about the difference between the team lead and engineering manager and what, what we have been talking about.

But, like, for me, it's still, like, the best, like, kind of size of, of a unit to operate with. And, like, that's, that, that's how I would like to do, to have with my teams.

But it's similar, I think, it's a good number of people which a person can manage and have strong relationships with. So, like, that kind of comes if you step up. And it's a little bit harder if, if you have this more hierarchical structure, that you have an engineering manager with up to six teams.

Each team has a team lead. So you have around, like, five to six, like, direct reports for you. But, like, then indirect, it can go higher. So, like, I, I, I still think, like, that, that, that, that works, that works very well.

Historically, I have operated within those teams over, over the course of time. Then I moved on, like, the one layer up, and it was, like, two, three teams. So that kind of worked well.

And, like, for me, it's, it's really like the skillsets on, like, what we have discussed already, in, like, building the relationship and, and trust and, like, the control. And it's, like, it's a really, very different, like, mode of operation, I, I, I think, in.

And yeah, like, why all these things, like, came about because there was, like, this over-hiring period. And then, like, if you have new hires, they take some time to kind of be embedded into the teams, like, kind of realize what the culture of the business, how, how, how they are operating, how the team works. So they need, like, quite a lot of support.

So it did make sense that this shift has, has happened, like, in those recent years. And now, like, we are coming slowly back into, into, into different numbers. For some of these, like, big tech companies and corporations where there are, like, like 20 or 40 directors, I think, like, that may not work as well in terms of retention of the people.

So it probably... It's, like, very high profile, like big tech companies, which probably, like, hire the best talent, like highly motivated and ambitious people who are self-driven. So, like, that can work in a certain way.

But I think and, based on the data which I have seen, like, the retention, it's not very low. And, like, these people kind of jump between these bigger companies. So I think, like, it's a viable way of doing business and, like, make that work.

And it, it literally depends on what the companies, like, would be optimizing for. So, like, if still they have this much of pool over talent, in some way, it feels to me, like, that it's okay for

them to manage them this way and, like, burn them a little bit and, and, and, like, replace them quickly or, like, take them from, from some other company.

So it may be more cost-effective. It makes the business, like, go faster. But it's probably not sustainable. And, like, the people don't like to work like that over a long course of time.

So it makes sense for me, like, but I- I don't think, like, not every company can afford, like, doing this. And because they just... it just wouldn't be attractive as much. It's probably, like, related to the compensation and the purpose.

Paweł: Not everyone can afford to pay big tech monies, right? So, uh, they are...

Michal: Exactly. Exactly. So, like, you have to... part of the deal has to be, like, different, different sort of compensation, right? In terms of, like, efforts. And, like, how, how, how they treat you and some, like, other benefits, the, the progress. And, like, kind of climbing the ladders as, as, as, as we have been talking about.

Like, I can, I can imagine that outside of the things which I have said, like if you have those bigger teams and, like, even, like, 40 teams. Like, when we have talked about the work of the engineering manager, like, not as much about the tech lead, but, like, this coordination, facilitation. Like, helping the teams to come up with a consensus because it's not always like this.

But in those kinds of setups, I prefer to have, like, self-contained teams who kind of work independently and are enabled by this, this independence. But they are usually, like, very opinionated about how to do certain things.

And, like, what their service and, like, the services, which a manager shouldn't do. And then there are some, like, negotiations and mediations and quite a lot of work, like, on these big projects. And big corporations had spent in those discussions on... or, like, finding what the consensus and how the APIs, like, should look like. Rather than, like, the actual implementation, it's, like, kind of straightforward. And, like, the engineers can do it.

Paweł: Mm-hmm.

Michal: So, like, that's where I see the benefit of this, like, higher team. Of like, having, like, 40 people and like, one, one engineering manager who is much more technical. And then, then, like, having the... like, it's kind of, in some ways, some sort of, like, dictatorship.

So you, you, you kind of are much freer in terms of, like, reassigning people working on, on different things. And, like, moving them around, giving much more direction. And I think it's possible to be effective in, in, in, in this setup. But it's super demanding for the engineering manager as well.

I think, like, it has to be, like, super strong person. Definitely needs to be current. Probably part... like, not necessarily code. But, like, has to have, like, even more understanding. And, like, it's, it's, it's kind of more of a technical lead with, like, a little bit of management.

So it, it, it just kind of the team lead role almost. Like, not necessarily with the, with the coding.

Paweł: By the way, that's... I've been told that's, uh, how NVIDIA is being run. I think Jensen Huang has 40 direct, uh, reportees. Uh, but you are, you are very right. I think that what every person, you know, what people say basically about him, is that he's... even though he's CEO of a huge company, he's very, very much into details. And he works like a hundred hours a week or so. It's like he works all the time. All the time.

Michal: Yeah, so, like... that, that, that's the trade-off. So, like, that's like going to... on your other question, like, what's the future? Like, I, I think, like, that's too much. I, I, I, I think, like, that's a certain way of doing things for certain companies who can afford it.

But like the middle companies... like, I, I, I'm not so sure that, like, people in general, like, would be, would be willing in, in, in that setup. Because I still think, like, even if you will apply this, and, and, like, there are no official team leads or no, no... like, not the middle layer between the 40 people.

I still think, like, people as we are, like, they would form some sort of, like, smaller teams and groups of people who either like to work together, work on the same things. Because it's not like everyone is interchangeable. Like, and if within that group there would be a senior engineer or whatever, which would be like the unofficial team lead or something like that.

So it's in some way... like, it may be like even, even, even cheeky. So, like, the hierarchy- the given hierarchy is not there. There is not a role or anything. But I still think, like, that the people will form the hierarchy.

And, like, that's generally how it is. Like, but I have no, no data to back this up or whatever. But, like, that's what I think, like, would, would happen. That there would be someone who will be, like, more vocal and would like to assume, assume the reins.

Obviously, in a super competitive environment, they can keep on, like, constantly clashing. And it would take them... so, like, you need a specific type of person for, for this with, with, like, very, very specific mindset so they are able to do things independently.

But on the other hand, like, they are able to come up with some consensus between, between the, the other people. And they wouldn't have the team lead or five people to kind of babysit them and mediate them as, as, as much if you have like one person for, for, for 40 people.

Paweł: Mm-hmm. Yeah, it's like... maybe what you're saying, it sounds like it's almost a humane thing, not a technical thing. Like, it's, it comes from human nature, right? That, there is a limited group of... limited number of people that you can really form a coherent group with.

It's like this Dunbar number, if I recall it correctly. It's like, I think we have those, you know, Facebook accounts or LinkedIn, God forbid. LinkedIn where you have like 5000 people who are, you know, who you know. But realistically, you know, you don't know 90% of them.

You, you probably... some people didn't even see that, right? Those, those 90% of people. Like, I know some people who have this rule that I can only add you to LinkedIn if we shake hands together. Which I, which I see it's... yeah, it's, it's... it makes it a little bit more real.

But it's still, like, it's not people who you really work with, who you really know. It still is the same as it was 20 years ago. You have a group of maybe five people that you really, really trust on a personal level, then maybe you have 20 people or something in the organization that you really work with, right?

Michal: Yeah, exactly, exactly. So, like, that's, that's kind of tricky. And like I was talking about attention. So it's even like hiring some sort of junior developers or like little bit more junior people. Like, does it, does it really work? Or, like, are the... this kind of juniors, like, who are so excellent that they are not really junior people.

That, like, they are a little, like, lacking the skills of, like, working within, within the teams. And maybe like a little bit of experience. Just like some sort of level of juniority. But what I mean is, like, they don't necessarily need as much support because there wouldn't be that much support provided.

Or, like, who is, like, providing the support in this setup? Like, the engineering manager. Fair enough, but, like- how many, like, junior people you can then onboard, like, within the team? Because they would need, like, some substantial support.

Or they would have some, some mentors selected from, from, from a given team. And they will be working with it. And maybe with two other people. So they will form, like, this unofficial team as well. Like, that's my view on it, that it, it, it's kind of an interesting way of doing things.

Um, but saying, like, it's much more empowering for the, for the engineering manager. And sometimes, like, I can, I can see, like, the benefits of it. Like, sometime, like, the, not necessarily evil, but like, the mastermind, like, pulling the strings. And, like, can have the last say.

Paweł: Mm-hmm.

Michal: Mm-hmm. It's a little bit more effective. Like if you have, like let's say, a little bit more of, of a dictator who would say, like things then will get done in, in a certain way, so you don't spend too much time deciding things.

Paweł: Yeah, yeah, sure.

Michal: Um...

Paweł: And in IT, you have smart people. They can argue. And they are smart, they are eloquent. And, you know, as IT people, we can discuss things, you know, with no end because everyone is so eloquent. And, you know, for everything there are always some perspectives.

Rarely things are, you know, black and white. There's always some shades of gray. So those discussions, it's... I think it's an important thing. I think that at some companies, like, you know, there's this thing from Amazon, disagree and commit. They, they sort of, you know, embedded this thing into their organization culture that at some point we gotta stop.

And we need to have some ways of, you know, going forward with the decision that not everyone really agrees with. But we still, still need to, you know, still need to go on somehow.

Michal: And, and, and even if there's agreement, it's right. And like that, that's the tough part because the people who have the counterarguments, they are right as well. But like it may easily be like th- there is no better solution.

So, like you have to, you have to weigh the trade-offs. And like pick the one like keep on going. And like to provide value. And then over time even, then even change it. So like that, that's big for me as well, like going from the background of the bank or wha- whatever, like which I have started.

Like there was always a year of planning or writing documents and things and like designing. And then like we started to work. And like there was this technical document which was describing what actually has been done. And it was like 80% different than what has been spent.

Like the requirements were fulfilled, but like the implementation or like a plan for implementation or whatever, like that kind of went through the bin. So to me, it's kind of getting better at execution. And then like you can always change the decisions.

Like there would be some cost to that for, for, for sure. But like what's the cost of waiting too long, like making that decision and doing nothing? Because you can find yourself doing nothing than being outdone by your competition or whatever. And like, and, and, and, and, and it's finished, right?

Paweł: Yeah.

Michal: So...

Paweł: That's- that's super- that's difficult.

Michal: Yeah, super.

Paweł: So- that's difficult because that's tough because even if you... you know, that's the, the... we talk about the making decisions as, as teams, but there's also the other side of the story on the... when you make this decision.

And obviously for any more complicated decisions, there are going to be some people that won't agree with it. And we disagree and commit, like we mentioned this from, from Amazon, for instance, and many companies.

But, uh, I, I, I have this feeling that in many organizations it's almost like, say, this is the 100% of the effort that you can give for a given thing. And sometimes if you don't agree with something, you say, "Well, I don't agree. I'll still do it because they tell me to, to do it. But I'm probably gonna put in a half-baked effort."

And I think what, what is happening in many organizations is like unless the organization has a really solid grasp on how things go, which is like, I think 90% of organizations don't have. And it's like, uh, you know, it's enough to have some important people or key people, and that doesn't have to be necessarily, you know, VPs or, you know, some directors.

It can, it can be some key contributors who don't feel that it's a really good decision. They sort of agree to it, but they give a half-baked effort. And I think what happens is like, when enough people give 60% effort, this thing is not gonna be... ever gonna be successful.

And then they come back and say, "You see? Well, we did it, what you wanted to do, but it doesn't work. So, you know, we told you so, you know, at the very beginning that it's not a good idea. But we sort of committed."

And I think that it all boils down to, you know, how do you motivate people? How do you keep people motivated? And I think that is also an interesting angle in that. Do you have any, you know, recipes or guidelines for engineering managers, you know, how they could keep people motivated?

Michal: I think it comes, like from the methodologies. And we have touched upon some of these things, like frequent, frequent meetings, uh, like one-to-ones and trying to overcommunicate the things like which are happening with the, with the company. And like, discuss things about the roadmaps.

Even like discussing some of the interesting technical developments, like in the AI space and the new technologies which can be utilized within the organization. So, like, keep yourself as a current and encourage people to do some introductions on, on, on things as well.

So kind of keep them interested and share their love for tech and, and, and things. And like give them some space where they can express themselves outside of, uh, just the sole work for the, for the company. Because it brings quite a lot of joy.

Like one of the popular things is like having these company **hackathons** as well. So, if people get fairly excited about it, they can come up with ideas. Then some of those ideas can even be taken on. And some projects can be spun up out of that if it was some feature which the products do.

Kind of it empowers the engineers to become the product, uh, people basically, and come up with some things. And like kind of influence back. So I think that's a great thing.

Then generally, I like some sort of demos, which, uh, depends what the cadence is. So where the different teams like which you manage can show like what they have built to the other teams. And it just depends on how you frame it, really.

So it can be just internal, not as much for outsiders. But like internally sharing, what other teams are doing. Because then obviously that motivates the other people within the company. Like, someone, like, does something great, then they would want to be, like, doing and compete with the others.

And, and show them what they are worth. And, like, obviously, like, these are the, like, things which you can do. But then it's, it's within the company communication as well, like, being super transparent about, like, what is being done.

Kind of give credit to people for what they are doing. And for me, I am really big on distributing the credit to the specific teams or people who are helping me to come up with

some designs and solutions. Kind of always, like, mention the people who have helped with that or the teams, like, specifically. And, give them good feedback.

I think, like, it's super important. Like, basically celebrating any of those wins and making sure these individuals or these teams were essential in, in, in, in doing certain things. So, like, that's, that's super helpful.

Obviously, like, giving some feedback, some honest feedback. We call, like, talk about Radical Candor. About if something is not right, like it's the... like, kind of the worst thing to not provide the feedback that something is not happening, like, exactly how you like it. And not have conversations about that.

And at some point, like, it may end up in, in a disaster or be demotivating for the people, because they are not-

Paweł: So that's where this emotional intelligence comes into play, right?

Michal: Yeah. Yeah, yeah. So, like, you need to, you need to give the feedback so... and then, like, the emotional intelligence like, it depends, right? Like, it depends on the person. Like, a little bit of negative feedback means motivation for someone, it can destroy someone.

So, like, that's kind of understanding and having those, those skills. And, like, how to serve that, that feedback. It's like a motivational tool for the teams. Because, like, some people you tell them, like, they have done something wrong, it will, like, fire them up with, like, with a little bit of spite and anger.

Like, I, I, I know, like, many people who kind of work in spite and, like, "I will, I will show you, like, I, I can, I can do much better." But, like, you have to be careful about, like, utilizing, like, those emotions and and and things. And, like, and different things. Different people work differently.

So, like, that's, that's, like, it's definitely like... I'm n- I'm not saying, like, pitting the teams against each other, because then it comes, like, through discussions which then promotes, like, indecisiveness. And, like, fights over, like, how things should be done.

So, like, you still have to be, like, super careful. But, like, that's, that's another, like, motivational thing which I think should be used quite, quite a lot. Just let me, let me think what else comes to mind.

Paweł: Yeah, I think it's really important that you said about these, you know, you don't want to set things, set teams against themselves. Because, uh, like, it is... it might be a power of- a competition, might be a powerful, uh, motivator. But at the end of the day, we work most of the time at the same organization in terms of our, you know, context of our conversation being an engineering manager.

And even though it sometimes works in a short period of time, I think it's... a long period of time is detrimental. You still need to have... and it's difficult sometimes because it's easy to, in many cases, to set up this, you know, this, this, maybe not incentives, but this culture of that we are against the old things, against the legacy.

We are building, building something new. We are good, they are bad. It's a powerful motivator. But I don't think that it's a good motivator in the long run. At the end of the day- you are still the same company, right? You are, you know, you are, you know, trying to achieve the same goal.

Michal: Yeah, yeah, yeah. Like, yeah, like, it was... like, it's for, for that. What it was I meant, like, with the, with the demos or anything. Like, kind of I didn't mean it in, in, in the way, because typically if you have some legacy teams or, or something, like, there are slightly different reporting structures.

So, it's more, like, within, within the teams that you manage. And you can say, like, "Look, like, this team, like, it's so great. They have done this implementation, like, it's worked so well." And, like, kind of use that. Like, ma- like, so mainly, like, pitting the teams, uh, like, you manage, like, just kind of showing them like, "Look how this one is good."

Like- "Maybe you should do something similar or, or like a little bit better in, in that way." I think, like, what you are saying, like, it's very relevant in terms of big corporations. And, like, going into some sort of mergers of companies or mergers.

Paweł: Oh, super difficult. Yeah. Because these are-

Michal: I guess-

Paweł: ... different cultures in, you know, different organizations. You are now in the same boat. But it's almost like... I think I read somewhere that, you know, not entirely sure if most of the mergers, but a significant chunk or significant part of them are not really successful in the long run.

And I think that it has to do with technology, obviously, and how you merge staff with different companies. But I think that most of it comes from the culture. It's that... it's not like that you can mold existing companies with a history somehow to this new setting. And it's, it's always... It's very difficult.

Michal: Yeah, it's always... It's always like a painful process. I, I, I would say, like, for, for most of the people involved. But, like, it has to, it has to be done. Like, you always have to have... like, that's probably like the biggest challenge which I have faced so far.

And, like, how to, how to navigate it. Like, you have these, like, views where you have to, like, behave like a, like some, some conqueror. And, like- there are some, like, legacy teams. And, like, new things are being built. And you, you have to eliminate, like, eh, everyone, like, from the legacy. And just replace it by, by the new teams.

So, like that's, that's kind of extreme. And like that's something, that's something which I wholeheartedly disagree with. But it, it sometimes, like, it, it makes some logical reason as well. Because then, like, you eliminate all this discussion and, and clashes.

You know, I know, like, it's tough to say that, like, that I have some understanding of, like, why some fusions, like, go, go like this. And, like when there is some sort of replatforming happening, the other way it's just kind of trying to co-exist and merge the teams together.

But like then, then, like that's super, super difficult. So you have different software development life cycles, different types of people. Usually, the culture, it's like very, very different as well.

And just like to tie everything together, I think, like, it's, it's very good, like, if there is a strong leadership giving direction, there is, like, plenty of stuff to build. But if the company, like, struggles itself as well, it's like, what's the next big thing?

And like that they kind of keep on buying these other companies. And trying to include them in, in what they are doing. But, like, they are not building as much new stuff. And everything is just like merging things together, trying to figure out what, what, um, service can be, like, de-commissioned.

And like, only one is utilized, so things are more efficient. Then it's, it's really, it's a really difficult situation to be in for the engineers because it's not really just building new stuff. It's kind of like standing their ground. And saying things like, "Our stuff is... should stay."

And then, then it, like, everything becomes, like, slightly different. And like it takes the joy out of building things. And the truth of engineering it's much more, like, management.

Paweł: Yeah.

Michal: Mm-hmm.

Paweł: So, um, Michal, listen, I, I would love to have... like, we didn't even run through, you know, maybe half of the questions that I had in my mind. So I still have, you know, many more questions. But I also know that you have other responsibilities now with your, you know, recently expanded family. So I don't want to take you, you know, too much of your time.

But, uh, last question. If you were, uh, sort of to sum up these, you know, the conversation that we had, you know, if you were to give, you know, two, three sentences of advice for a new engineering manager or team leader, generally technology, uh, leader, what would it be?

Michal: I think, like, the most important thing from my point of view, it's like working on communication skills, like, and understanding that, like, the main part of the job would be, like, listening, talking, persuading people. Like, that would be, like, the majority of the things.

Like, in some way the part of the job, it's kind of being, like, a psychologist for your people and, like, working with people and talking with them. So, just try to read some literature and prepare yourself mentally for that.

And depends on, on, on your style, but I still think, like, that's the part of the job because, like, you are there for them, you're there to help them out, to deal with any problems they have. And, like, to enable them to do their work correctly.

So, just, like, focus on the delegation and execution being done by the teams and the people you report. And, like, kind of looking from, from the outside and what's, what's happening.

Because then, like, that way you would get the best results. And you will allow them to, to grow.

And then some of these recommendations which I am having, it's like what I have found out to be best from my experience. But it doesn't necessarily mean that everything I have said it's the, uh, universal solution for everything. So everyone is individual.

So from you going into the role of engineering manager, it's, it's like about finding yourself, what leadership style you want to have. What are your, like, what are your strengths? What are your weaknesses? Improving the weaknesses.

And just kind of... it's quite a lot of work to take on yourself to improve yourself and incorporate the feedback from the people to be successful in, in, in that. So that's what I feel is interesting. It's because you are there to enable the growth of other people while you have to be... you have to accept that you are growing as well to be up to the job.

Paweł: Great advice. Um, listen, Michal, it was very, very great to have you here. I enjoyed the conversation immensely. Thank you for, you know, showing up and having this conversation.

Michal: Thank you very much for, like, inviting me. And, like, being part of this new podcast series. Like, it's definitely, like, an additional step in what, what we are doing. And hopefully we will talk about something else in, in the, in the future.

And, like, this series will become immensely successful. So, like, I have enjoyed talking to you. And, yeah, I'm, I'm really grateful for being here. Thank you.

Paweł: It was, it was a pleasure. I am excited too. And, uh, as I said, I mean, we, I think we barely scratched the surface. So I'm, uh, hoping, you know, to have another chance to talk with you. So, thank you again. And, uh, see you next time.

Michal: Okay. Thank you.