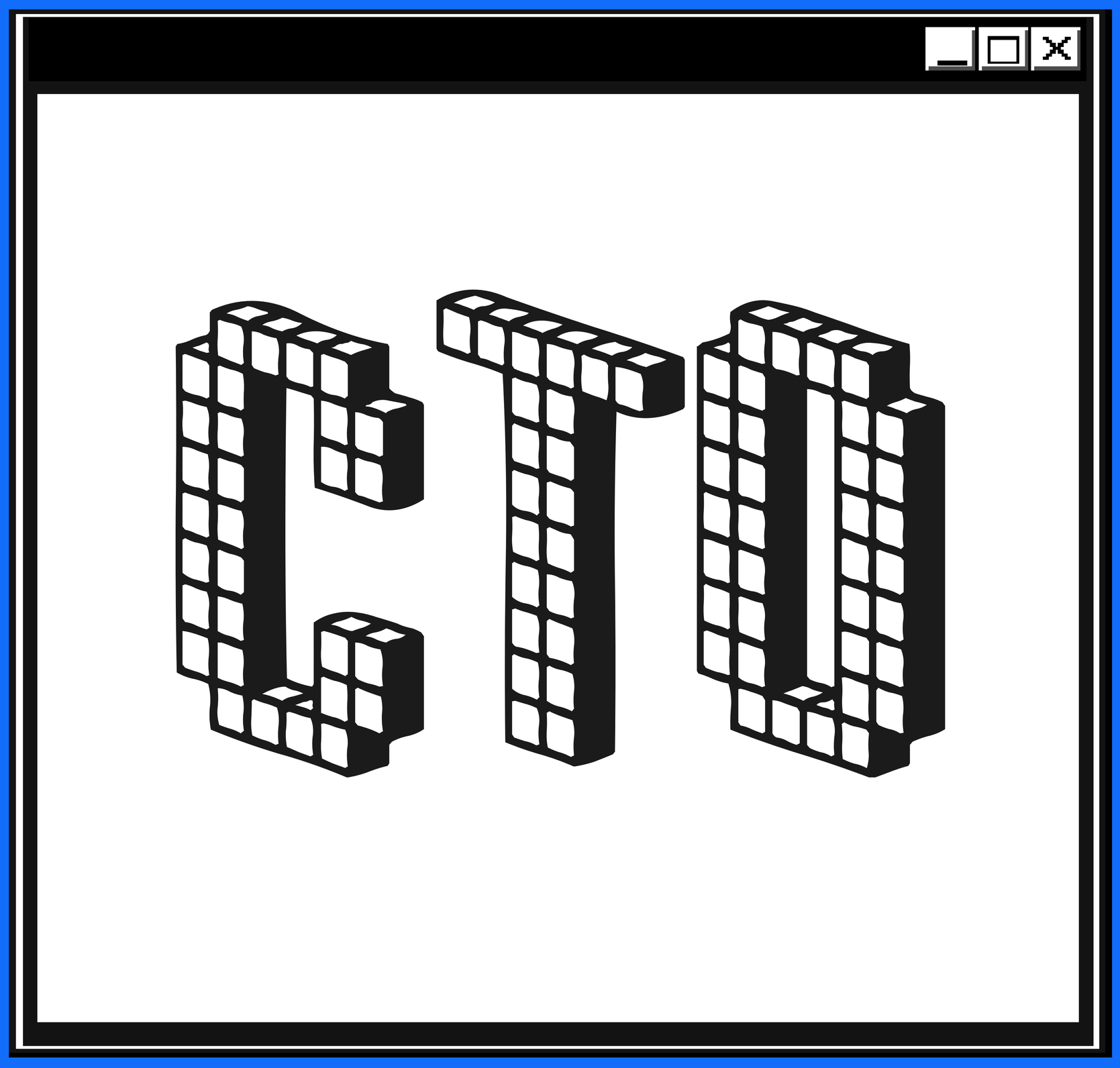

STARTUP CONVERSATIONS

Ft.

Pitch FLIXBUS Personio userlane

Seedtable.

In partnership with circleci

Table of Contents

Startup Conversations — View from the CTO is a collaboration between Seedtable and CircleCI. In this podcast series, Gonz from Seedtable sits down with eleven of the greatest CTOs in Europe to deconstruct how to build and operate high-performing technology teams.

FROM CIRCLE CI

Foreword

A foreword by Nick Mills, EMEA, General Manager for CircleCI.

CircleCI is in a highly privileged position within the European technology ecosystem. We are fortunate to help many of the leading European technology companies, such as Deliveroo, Monzo, Wise and Babylon Health, to achieve scale and compete more effectively and efficiently on a global scale.

We are also proud to support individual developers, early stage start-ups, not-for-profits and open source projects across Europe to formulate their ideas, achieve product-market fit and build a strong foundation for success.

As we look at the impact that this diverse range of European customers have achieved, both within the region and beyond, there are a number of key factors that drive this success. Great ideas, world class talent, availability of financing and favourable government policy and support initiatives have all helped to contribute to making the region one of the greatest places to build and grow a company.

To make the most of these ingredients, strong leadership is required to set the right strategy, establish meaningful metrics and inspire and guide the wider business to ensure everyone is pulling in the same direction and to maximise the chances of achieving success.

It is with this leadership in mind, particularly on the technical side, that we are thrilled to be working with Seedtable to celebrate the role and impact that CTO's have on their organisations, building great products that are commercially successful and creating cultures where their software engineers thrive.

In this era of turbo-charged software-enabled innovation, the way in which companies build software can significantly influence business results. Studies show that the performance level of a software engineering function is now a key indicator of a company's success prospects. This is particularly the case at a later stage, where operating a world class engineering team is a high-leverage differentiator and can generate greater returns on invested capital.

The CTO plays a key role in driving this innovation and this guide provides unique insight from some of the leading CTOs at Europe's most prominent tech companies. In addition to discussing strategy and how to build great software, in this guide and the corresponding podcast series will cover a wide range of additional important topics, such as recruiting top talent, developing high-performance teams and the evolving role of the CTO in high-growth tech companies. Some essential book recommendations are also shared. My reading wish list has never looked so strong!

I have personally learned a great deal from producing this guide. I take a sense of renewed inspiration and optimism about the role European tech leaders will play in pushing the frontiers of science and innovation. I hope that you, too, take away valuable insights from this project that help you solve your big challenges and achieve your personal and business goals.

Foreword –

EXECUTIVE SUMMARY

Learnings

We distill some of the key insights all our guests share when it comes to operating high-performing technical teams.

Each of these guests, we're not surprised to say, delivered. Our conversations are diverse, educational, and entertaining, and we cover a whole range of topics, including organizational principles that bridge the gap between business and tech; how to unleash your team's productivity and creativity; common mistakes CTOs should avoid when onboarding new people onto their teams; and why software design looks a lot more like art than craft.
As a starter, we've taken the time to distill some of the key insights all our guests share when it comes to the life and work of a CTO. These insights fall under five themes:

- **Recruitment**
- **Team Development**
- **How To Build Great Software**
- **The Role of The CTO**
- **Absorbing Information**

Next, we'll detail each one by one.

THEME #1

Recruitment

Key ideas: hire remotely, pay competitive salaries, hire people passionate about your mission, hire hard-working people, leverage your network.

In early 2020, the world changed, and with it the way companies of all types and sizes compete for and recruit talent. When asked about the **impact of Covid-19 on talent markets and recruiting**, all eleven guests agree that the pandemic has made talent markets global. In essence, the world of geographically-dependent hiring is over, and that means both good and bad news for European companies.

The good news is that there's now access to the world's top talent without the added nuisance of relocation. The bad news is that everyone is competing for the same top-level talent. To both reduce the challenges of worldwide competition and capitalize on this newly-enabled talent, our guests suggest CTOs double down on seeking employees who believe in their company's mission, for starters.

Here’s Håkan Holmberg:

> *“When the job market went global, our company went global. All our meetings are online with clients and with our colleagues. Today, almost everyone in the team works remotely (...) And when we're hiring, we're also hiring all around the world. So one of the main things that I'm looking at now is to try to really get a good digital environment working for the team so we can come as close to a physical experience as possible. I believe that the physical experience has advantages. And so the important part for me now is to try to create a digital environment that has almost those features of natural interaction between people.”*

In terms of **what our guests look for in new hires**, and **how they work to maintain team culture as companies scale**, all our guests put special emphasis on the importance of hiring engineers that resonate with the company's operating principles and mission, and who are curious by nature.

Here’s Håkan again:

> *It’s very important that when you have an interview with someone that you think whether you’ll be able to work a lot with this person. Startup employees need to be truly dedicated—It’s not a 9:00 to 5:00 job. That's just not the case. So you need to find people who will be motivated by both your mission, what you're trying to offer to the world, but also by simply learning the technologies and becoming really good at what they do—to see it also as an education. You need someone that sees the job as more than a job, as a part of growing and developing.*

Most of our guests also recommend that CTOs work tirelessly to make sure those operating principles and mission-first culture are maintained at scale; to do so, they recommend CTOs lead by example and work constantly on emulating said principles themselves. Other than that, they should focus on identifying and clarifying the principles and culture from day one—if the foundations are set properly, the rest should take care of themselves.

In a similar vein, a few of our guests had advice on what *kind* of people CTOs should hire; the traits to look for. Georg Schroth, for example, was adamant about the importance of hiring engineers that were, by nature, extremely hard-working people.

> *I'm definitely looking for energy drive and passion because this is something that you cannot teach. You can teach almost anything, but not energy or drive. And so if that is paired with the willingness to take ownership, like full responsibility and the willingness to make this ownership successful, then I’ve found an excellent new team member.*

And there’s more to his advice. Georg believes that a CTO should surround him/herself with people who will outwork him/her this will make sure “the responsibility of being motivated, working a lot, always ends up with you.”

But, as in all things, balance is a must. While Georg’s advice about finding engineers that will keep you on your toes, another of our guests emphasizes the importance of setting up a team culture where hard work is not the be-all-end-all. Johann Romefort, instead, advocates for a smart-work first culture; good engineering work is not as much about working long hours, he says, but doing good work, and that requires leisure and rest. He illustrates this point with a poignant personal anecdote:

> *When I arrived in San Francisco, I was looking for an apartment and from this super nice loft I met the landlord and we started to talk about technology, right away. We didn't talk about the apartment or whatever. She happened to be the former marketing director at Intel. And so she coached me a little bit, though I mostly brushed it off because I was arrogant. But one thing that stuck with me that I revisited, years afterward in one of my notebooks, she wrote, 'Manage your energy, not your time.'*
>
> *I wish I would have followed this advice because it's so true. You can find a lot of time in a day. Right? You can work for 16 hours a day, but do work with a high level of energy? Probably not. So it's much better, in my opinion, to work, less time but with much higher energy. That sounds the opposite.*

Lastly, A few of our guests offered advice on how to go about hiring great engineers practically. Johann, for example, suggests that CTO should first start recruiting talent by tapping into their personal networks, and that of their employees.

Not only will this make recruiting easier, but it will also allow the team culture to grow organically. Daniel Krauss had an entirely different piece of advice: to succeed in hiring great talent, he believes that companies should work to have dedicated recruiters as soon as possible, so as to cast a wide net and find the best talent out there, no matter where they are.

> *I always recommend, whenever young startups ask me how to [recruit great talent], that they find dedicated recruiters. You have to have dedicated recruiters. That's the only way. There is no other way around to find good talent.*

THEME #2

Team Development

Key ideas: set up an IC development path and a Manager path, lead by example, give your employees freedom to make mistakes, treat engineers like humans and not cogs in a machine.

When it comes to **team development**, most of our guests opine that there should be two set paths for all company engineers: an IC (individual contributor) path and a Manager path. Both paths should be taken equally seriously, and should offer engineers the same amount of growth-room; just because, traditionally speaking, all engineers usually end up becoming managers as they move up the corporate, that doesn't mean they should. Only some people are well suited, or even want to become managers.

Our guests' advice is for CTOs to pay close attention and listen to their employees to find out what they want out of their careers, and then set things up in such a way that those goals become attainable. Another thing our guests emphasized is the need for CTOs to continuously support engineers in their development. Arseniy Vershinin and the Personio team, for example, offer their employees a €1500 development stipend, for them to spend as they see fit.

Another key aspect of team development many of our guests highlighted was *leadership*. In their minds, a great leader can help engineers grow into the best version of themselves, and CTOs should work tirelessly to make that happen. How tirelessly? Johann Romefort's advice is a bit counterintuitive but pressing:

> *You are [your engineers'] leader and they use you as a sort of role model. And so if you are the last staying in the office, then they will also stay longer in the office. They will emulate all your behaviors. So make sure that you have a good routine. Make sure you're getting enough sleep and taking breaks from time to time (...) Make sure that you get a bit of exercise. I'm not huge on exercising, but at least get up on your feet and work out a little bit. Make time for friends and family.*

In a similar vein, both Daniel Krauss and Benjamin Lan Sun Luk hint at the fact that many CTOs (and managers) often confuse leadership with micro-management, and that this is a mistake. Great leaders, in fact, and according to them, not only lead by example but also make it a point to give their employees the freedom to experiment and make mistakes. Here's Benjamin:

> *My best advice for the people who go from the IC path to the more manager path is really to trust people. To give them time to make their mistakes, grow, and at some point, to own the platform as well as you did in the past.*

Felix Eichler expands on this advice and relates it to your company's culture. His word of wisdom? He equates company culture with product management and advises leaders not to treat their employees like cogs in a machine:

> *As long as you treat people as humans and not as like cogs in a system, I think your culture will be set up for success. It starts with yourself as the leader, and then the people you hire and fire.*
>
> *That is how you manage your culture — like you manage a product. You listen to your user feedback. And your users, in this case, are your employees and your coworkers. And then you iterate on that. And you open feedback sessions with them. If you do this, you'll be approaching culture iteratively.*

Under the umbrella of team development, there's also team productivity and performance to consider. More specifically, many of our guests shared their thoughts around how to measure and improve your engineers' performance and productivity. The answer to that question, as Georg Schroth would tell you, is not easy, mainly because there isn't *one way* in which to measure performance that will work for everyone, and no set of standard metrics all CTOs should measure.

> *Generally, you can't do metrics right, first of all. You can build your best dashboard, and you will know nothing about your company. Metrics can't tell you a story, only the team can. First of all, you want to understand what you want to improve and why. And then use metrics use them as an answer to very specific questions.*
>
> *Testing hypothesis — that's what I think you can do with metrics. And usually, those hypotheses should support a specific business goal. Say you want to go in a certain business direction — you should only measure specific things as long as the questions and answers inherent in those measurements help drive positive change.*

Following that guideline, Georg wraps up by mentioning a few important metrics he's measured in the past, to great success. Things like lead time, cycle time, team deployment velocity, open/close rate from issue to resolution. But, he says, "these are all just *things*. You should only implement them if you actually have hypotheses that you would like to falsify or verify."

Lastly, also on the topic of measuring your team's success, Arseniy advocates for a people-first approach. There are, he says, a variety of indicators and metrics you can measure, but for him, it all starts with people, which is why the first thing he pays attention to is how happy people are within the different teams that form his organization. There's a practical reason for this: employee satisfaction, in his experience, is one of the highest correlating factors to how both the software team, and more generally, the organization performs. He measures this through regular employee pulse checks and by looking at the engagement score of his organization, breaking it all down by team. The other metrics he follows are divided into three clusters:

Firstly, since we use this concept of mission-based teams, that means that a team is owning a piece of the product end-to-end, and is able to both define its vision and develop it end-to-end — we're looking at the customer happiness associated with this particular piece of product that a team is managing.

Another metric is UX satisfaction. So you can ask a customer, "Hey, how do you find the experience with this particular feature part of the product and also reflected back to the team?" Those are set of metrics on the team level that more correspond to the business-as-usual metrics such as performance quality, reflected in how many buckets have been reported with that particular and other more engineering metrics. But also another aspect of it which we started to get more and more into is this actual engineering performance. And this is all the time a very hard, over-debated question. But we believe that so far what starts to work fairly well for us is to track metrics such as deployment frequency, change fail percentage or mean time to recover.

So that's on the team side. And it all, obviously, wraps together around the whole mission of the product and engineering department, which is to build the best-in-class product with more aggregated, more lagging, but still very true metrics such as Net Promoter Score. And the Net Promoter Score then reflects on how the whole department performing against the goal of making our customers happy and productive.

THEME #3

How to Build a Great Software

Key ideas: build software on a need-only basis; software development is a team sport; focus on your customer's pains first; plan around 6-week cycles; use the 'Now, Next Later Roadmap.'

Though building great software is an extremely complex matter, and not one we could possibly explore in its entirety during these conversations, our guest did have multiple pieces of advice to offer to current and prospective CTOs. Some of this advice was of a general nature (i. E. how to think about building great software. Benjamin Lan Sun Luk, for example, advises CTOs not to rush to technologies that can scale when at an early stage, urging them to remember that seed-stage startups need to focus entirely on solving the customer's pain and finding product-market fit, which requires a lot of flexibility and iteration on the part of the engineering team.

> *The first thing I advise to early-stage founders is to really not rush to the "hypest" technologies that can scale, that can host millions of customers. Because you don't have customers right now. So make sure that your solution can really help your customers and, that you can iterate easily. It can be quick, and dirty in the beginning. That's totally fine.*

Regarding how to approach software creation, Adam Renklint tells CTOs to remember that great software is not the product of individual efforts, but rather resembles more a team sport, where different people contribute in different ways, all working in tandem to accomplish something greater than the whole. He says:

A few of our guests offered advice on how to build great software that was of a more practical nature. Marijn Van Aerle, for example, believes that CTOs should not try to achieve goals on a yearly basis, but instead plan their work around 6-week cycles. He says:

> *In terms of our roadmap process, we try to keep the time frame short. Yes, we have a year plan — but it's an idea. Every six weeks we have a cycle that we work on in the project, and then we have two weeks to pick up all the stuff that came up in between. And then we again have six weeks to work focused on a project. And that cycle goes for the entire team, and actually for the entire company because there's also other departments involved.*
>
> *Six weeks cycles are quite nice because you can really do something there. It's not like a two-week sprint. It gives also time for planning. Usually, something big you can see as a chain of six weeks projects. So like two or three, six weeks projects. But eventually, every project has to ship something. So it's not like you're doing six weeks on some technical stuff that doesn't*

> *do anything. It should do something after six weeks.*
>
> *You always know that after six weeks, there is no guarantee that your next project will go through. Because that's the flexibility that we need. So you have a project for six weeks, then we decide with the management team and all the stakeholders what's the next cycle going to look like, and that's it then. And, yes, usually there's some continuity. But I think flexibility is far more important than this long-term vision thing that might happen.*

Felix Eichler, on the other hand, advocates for the “Now, Next, Later Roadmap” when it comes to software development and goal-setting, which goes as follows:

> *The Now, Next, Later roadmap communicates really well what stakeholders want to know. And it is also very readable from a very high altitude, especially if you link it with OKRs. So that you're not listing features, but you're listing objectives or maybe key results on that altitude. And then these become business objectives. So anyone will understand what that means.*
>
> *For example, if you want to win 100 new customers in North America, then that means that you're going to build maybe a US data center, or that you build some features that are more relevant for some houses in that market. So basically, the features become a side note or become the description text of your key results or objectives. And then you put them in Now, Next, Later. And people know, okay, we are busy with something right now. There is always something in the pipeline.*

THEME #4

The Role of the CTO

Key ideas: CTOs need to stop thinking of themselves as ‘coders’ as the company scales; CTOs act as go-betweens between different teams; CTOs should educate the rest of the company about what they’re building, CEOs included; CTOs are in charge of making the organization feel the customers pain.

How the CTO role evolves over time and as a company scales was also a topic many of our guests had experience on. Most of them had a similar trajectory; when their company was initially founded, their day-to-day job revolved almost entirely around creating and maintaining the initial version of their product. But as the company scaled and their engineering teams grew, our guests saw their role as coders take more and more of a back seat; slowly but surely, all of them became managers and communicators. Most made a point to highlight how this transition is unavoidable but sometimes painful.

Arseniy Vershinin summarized this transition eloquently:

> *The common denominator to this role is that a CTO should really focus on enabling the organization—whatever this organization is and in whichever face of existence it is—to resolve the most important and strategic business problems with the help of technology, either directly or indirectly. If we talk about the initial phases of the company founding, this happens directly. So the most important task that a CTO together with a founding team obviously has is to create MVP.*
>
> *But then the role shifts into making sure that you have a well-running team; making sure that this product that hopefully found initial traction on the market actually is able to scale, reliably, securely, performantly, while addressing the most important customer needs. And then, at later stages of company development, it depends on what a particular CTO wants to focus on and what particular sets of business challenges there are to be tackled.*

That said, our guests offered advice to make to ease the transition and set other CTOs up for success when going through scaling: CTOs, no matter the scale of their business, should still take the time to code from time to time, both for the sake of their product's health, and their own. They should also put their egos aside and focus almost entirely on sharpening their communications skills; the role of the CTO at a large company is not that of the genius lone coder working until the late hours of the night to code some sort of miraculous product, but one of leader, communicator and perpetual collaborator.

When asked about **the relationship between software teams and the rest of the company**, most of our guests agree that one of the key responsibilities of the CTO is to act both as a go-between between the different teams, and as an educator. Our first guest, Johann Romefort, had great advice on the matter; he suggested CTOs should educate not only the business teams but the CEO on what the engineering team is doing, which of those tasks matter most, and why.

He also suggested that CTOs should, from time to time, embed their engineers into other, non-tech teams (everything from product to customer support) so they can get a good sense of what the rest of the company is about, what the customers want, and how to best serve both.

> *I'm a big proponent of taking your engineers and embedding them into product teams. Making them participate in these discussions. Having your engineers doing customer support as well, because then they're going to feel the pain of the customer. Maybe they will see they hadn't thought of before. And marketing, too. There is a technical aspect to marketing. And sometimes you see the marketing team struggling. There are few things that the tech team actually couldn't solve pretty easily. I think the more connections you create like that are inside the company, the more open mind your engineering team will be and the more innovation will come out of that.*

Similarly, Benjamin Lan Sun Luk explores why the role of the CTO is not only to orchestrate the creation of a company's technological infrastructure and product but to do so while being teaching their engineers to be mindful of customer's pain, especially when in charge of a late-stage company. He says:

> *One of the main missions of the CTO is really to make the organization, your organization, feel the customer's pain. Building a very high-performing team is, in the beginning, about making sure that your guys really understand deeply what they are trying to solve—to make sure that they really understand the customer.*

THEME #5

Absorbing Information

Key ideas: CTOs should be life-long learners.

All our guests agree that to be successful at their jobs, CTOs need to be continually absorbing new information; i. e. they need to be life-long learners. That is why, at the end of each episode, we asked them to list which books/papers/articles have most influenced their thinking around startups, engineers, and life, and which they'd recommend other CTOs read. Below, you'll find said list, broken down on a per-guest basis:

Johann Romefort

- Mirror Worlds: or the Day Software Puts the Universe in a Shoebox; *David Gelernter*

Marijn Van Aerle

- Getting Real: The Smarter, Faster, Easier Way to Build a Successful Web Application; *Jason Fried, David Heinemeier Hansson, Mathew Linderman*

Felix Eichler

- The Meaning Revolution; *Fred Kofman*

Sylvain Wallez

- Getting Things Done; *David Allen*
- The Manager's Path; *Camille Fournier*

Benjamin Lan Sun Luk

- Radical Candor; *Kim Scott*

Håkan Holmberg

- Bitcoin Whitepaper; *Satoshi Nakamoto*
- Philosophy of Logics; *Susan Haack*

Arseniy Vershinin

- Radical Candor; *Kim Scott*
- Crucial Conversations; *Kerry Patterson*

Adam Renklint

- Getting Real: The Smarter, Faster, Easier Way to Build a Successful Web Application; *Jason Fried, David Heinemeier Hansson, Mathew Linderman*
- Adam prefers to consume information through short-form content, namely through platforms such as Hackernews.

Daniel Krauss

- The Hard Thing About Hard things; *Ben Horowitz*
- Any Salman Rushdie Book

And that's just a start. For even more insights, please go ahead and give Startup Conversations — View from the CTO a listen, starting with **Episode #1 featuring Johann Romefort**. We hope you enjoy these conversations as much as we did.

EPISODE ONE

Johann Romefort

About: Johann Romefort is an investor, consultant and former CTO who specializes in Technical due diligence & corporate innovation. Before becoming a freelance consultant, Johann worked as the Managing Director at Techstars for the BSH Future Home Accelerator and, before that — he co-founded Seesmic, a social media account management company, where he acted as CTO building a distributed team of 30+ developers. Seesmic was acquired by Hootsuite in August 2012.

techstars_

To listen to the episode, **click here**.

Gonz: Johann, welcome to the CircleCI and SeedTable podcast.

Johann: Thank you Gonz.

So, let's start with a question. What's a two minute version of Johann? Give us an intro for the audience.

Okay, it's all in two minutes. But my background is in software engineering, working in different industries, gaming, copyright, then created a company in 2007 with another French Cofounder, and we moved to San Francisco to do that company called Seesmic. A very unusual trajectory because we raised 6,000,000 in the first week we started and subsequently like another 10,000,000. Got acquired in 2012 by Hootsuite, then took some time off, started to help with entrepreneurs. Moved to Munich about seven years ago and started to work as a tech evangelist . So like very different work than what I used to do, which was like CTO.

I also get to know the kind of different world that I did for almost two years. Where I run basically like there's a BSH Home Accelerator, focusing on the future of home living. And then it's been about a year that I'm doing mostly tech due diligence for private equity and business. And also training CTOs on what it takes to actually succeed on a technical due diligence. So still a huge part of the Techstars ecosystem as well. I'm also running a CTOs like... Yeah. Still evolving in the scene, basically.

So as the former CTO, mentor, advisor, former MD Techstars, current sort of community builder or know to a big network of CTOs, you have a very wide view as to what being a CTO is. So what do you think is the role of a CTO? I mean, how does this evolve as a company grows?

Yeah. It evolves quite a lot, right? Throughout the life of a company. And when I first started as a CTO, I didn't really know what to expect. It was my first time, and it was pretty brutal, I have to say. Right? Especially because we raised a lot of money early on. It was not nice and gentle. It was really a pretty steep learning curve for me. So I would say, if I look at my own experience and the ward of quite a few CTOs, you're the first engineer. Right?

So you're super hands on, building the first [crosstalk 00:04:40] the door and then starting to build a team behind that. And the more you build this team, the more you figure out that, oh, damn. I don't really have time to cut anymore. Right?

So that's the first frustration, I would say. And this one is, it's a huge one. Like, I see it with a lot of CTOs that I'm coaching. It's really hard to let go of the code and because it's your baby as well. Right? And just give it to the team and offset everything will be fine, essentially. But we need to do it at some point and kind of become a coach for your team. So that's how I saw myself. I had this realization at some point. I was like, what is the best thing I can do for the company. Right? Instead of what it is that I really love to do. Which was coding, obviously.

And then you realize that the most exponential thing you can do is just helping your team members and your employees to do their best. So that's what I focused on. And I think this transition was a bit hectic, but eventually led me to a very different sort of role. And Additionally to that, I would say that because our company was always in the news. Like one of our investors was Michael Arrington from TechCrunch back then. So every two weeks we were on the news. We got a lot of attention, which for a CTO was just not good. Right? It's just way too much.

And I had to transition from being like this, this introvert, which many CTOs are, but it's okay. Right? Yeah. Very, like much more introverted than counterpart CEOs to become more like I would not say extroverted, because I'm still introverted, to be honest, but more social, I guess. Right? And more able to speak in public. So I started to do that as well. I started because I got invited to conferences and getting interviews. I realized that there might be an angle there also to find talents for my team. So that's something that I caution you to do later on when I was a tech evangelist as well.

That is actually a perfect segue to my next question about sort of being public and recruiting talent. So I'm not going to stay to the obvious, but the world changed with COVID. What was once a local hiring market? You compete, like let's say if you were in Munich, like you are right now, you usually compare it with companies in Munich. For talent, of course. But now it's global. So the global hiring market and you got to compete with companies in London, in San Francisco, in Singapore, in India, and so on. So what does it take to recruit and retain talent in a world where COVID made talent markets global?

Yeah. I guess, you know, the are parties that really like finding people. Right? If I look back at my role as a CTO, like doing that, we were, I think, pretty early in building a remote culture. Engineering culture. Only our engineers actually work where we want. But I built a pretty large office in Bucharest, Romania, another one in Singapore. And I also like [inaudible 00:08:18] for the developers. Kind of spread it all over the world. And I would say that one of the most effective way for us to recruit was always through the network. Right?

So we would tap into our employee network to find other people. And that way we actually grew very organically and had really little terms because those guys, they love to work together, we're friends. And it was kind of a family. And funny enough, this company actually still exists after the acquisition. They're still working together. They were not part of the acquisition, and it's been 10 years. Right? But to stay together and are taking project because say... That's something I have quite a pride about that, because I build all of this team and also later find out that they work with another Techstars, this company.

I would say that finding those people... First, you have your employee network. So going, be more public. Right? If nobody knows that you exist, it's pretty hard to find talents. And you need to... I think Engineers general, they're not looking at job offers pretty much. Right? It's always going through the network. So you really need to use that and to build a lot of awareness around what you're doing. So having an engineering blog, speaking at conferences, organizing meetups, and being part of the community really, I think can help. Because it has this inspiring element to it, where if other engineers are aware of the problems that you're solving and kind of get in touch or so with like maybe some of your engineers.

And I have an idea of your engineering culture that all of those things are really good elements to find new talents. Now, when it comes to keeping people, that's also a knot in itself. Right? Because you're competing with a lot of other companies, which can really offer better salaries, better perks, like it's a huge war. I have a very simple approach to that, to be honest. Which is to take care of your people. Right? You see, sometimes companies spend a lot of money hiring talent and then sort of just letting them be, without really taking care of their wellbeing inside the company.

And I think it's a mistake. Right? To me, being a leader in my company means that I do care about my employees. I care not only about the work, the produce, obviously, but who they are as human beings. Even though I had my team in Bucharest, I was flying there like every two months, literally. And spending two weeks with those guys. And we would be drinking together. And I think we will have dinner together. That's how you build trust. Right? And trust is really, like the fundamental element in keeping people. right?

So you need the trust. You need to... You know, I'm French so when it comes to conflict management, I'm pretty direct. But again, I'm very caring. Right? I would not say something, just throw something at someone without saying, *"Hey, I can help you in solving that"*. So I think this communication aspect is very important when it comes to keeping people, and sometimes the simple things are the best.

Yeah, absolutely. Simplicity is usually the best. Beers can't hurt.

Yes.

Absolutely. Absolutely. Other than technical skills, what are some of the core personality traits you look for in new hires to join the team? And I'm asking, because what you built, which is a high performing team that sticks together after a decade, is quite impressive.

I think good feeling is quite important there. And that's also something I learned at Techstars. Right? Because we're interviewing a lot of companies and we're putting money in them. So we need to get good at selecting good people. And somehow the criteria between selecting funders and employees for my company before like roughly is the same. So you're looking for people who are likable first. That's kind of like the good feeling aspect. Right? If I'm sitting in front of someone and I'm like, I don't really want to get a beer with this person in the bar tonight.

Then probably it's not going to be a good fit. Right?

And that's very intangible, but that's just the way it is. So likeability, coachability. Right? Like, is it someone that you can coach and help to get better at what they're doing? Some people are just like, they have quite a fixed mindset and it's really hard to make them move away from that. Right? And in a high performance team where things are changing really fast and you need to catch up to a new constraint all the time. You need people who are coachable, flexible and will follow the motions, basically.

So there's that in terms of likability, coachability, being a good team player. Right? Some people just work for themselves. Let's be honest. And sometimes I would say that it might be the one thing that is the most difficult, maybe, to find out in an interview. Like what their mindset is when it comes to working with others. And then, I generally like to work with people with a high degree of empathy because I think it makes a team much better. And I would say that I really developed my empathy along the years.

And I think it's really essential to create a good work environment when there's no blaming, no gossiping and no things like that. Right? Again, it's all about being pretty direct, but at the same time, being in touch with your emotions. Right? Which sometimes sounds corny, I guess. But I think it's very important.

Absolutely. Absolutely. And one thing you said really caught my attention, which is that some people work for themselves. And as I've been scaling my own team, I've been thinking about this quite a bit. One of the realizations that I came to is that you often want to hire people who think of their careers as a multiplayer game versus a single player game, which is what most people think. And what I mean is they usually need to have this positive sum mindset. Right? Collaboration, being able to give away your Legos with the focus of moving the team forward. Right?

It's not about politics, it's not about sort of territory. So if you think about your career in a multiplayer game, like a multiplayer game, I find that that's one of those other core traits that I look for. So any practices or fun rituals or cultural principles that you've used in your organizations or with the CTOs that you coach that help create a high performing environment.

I look at different things, I would say. First, I think it's important to make sure that the CTO is well in his head. Right? And when I say that it's my experience as a CTO, I say it was brutal. I didn't sleep enough, for example. Right? So I was sleeping, like four hours a night. Like my CEO, even sort of like branded me the CTO who never sleeps. Which sounds really bad, actually. Now that I look at it. It was fun back then. But I would say start with yourself and make sure that you have good habits because your team will copy those habits. Right?

You are their leader and they use you as the sort of role model. And so if you are the last staying in the office, then we can also stay longer in the office. Right? So they will emulate all your behaviors, pretty much. So make sure that you have a good routine, such as sleeping enough or getting enough sleep. Taking breaks from time to time. I think it's also very important. I didn't really take any vacation in four years when I was doing a startup.

And it's a huge mistake. Right? Because I was really obsessed with time. I was always stressed that something would go down if I go on vacation. A lot of irrational fear, to be honest. The reality is I could have done it. I just didn't do it. And the reason it's important is that your brain needs a break, otherwise it's burning out at some point. And if I look at myself during this period, maybe the first year I was okay because I was so excited and so on.

But as a year passed, my productivity level also decreased because I was too tired. I also made some wrong decisions, to be honest. Right? Because I was too exhausted to make the right ones. So I think that that's a good first one. Make sure that you get a bit of exercise. I'm not huge on exercising, but at least get out your feet and work a little bit. Make time for friends and family, which I also totally forgot. And a lot of CIOs, they forget about that. Right? You do have a support network like your friends and family.

They are supporting you in your endeavors. You need to be somehow grateful for that and spend some time with them. It also regenerates a bit of your brain power.

Totally. Thanks for that. You said something or something that I found is that the counterintuitive thing about sleep is the more you sleep, the more productive you are and better decisions you make, not the other way around.

Yeah, absolutely. When I arrived in San Francisco, I was looking for an apartment and from this super nice loft and met the landlord. And we started to talk about technology, like right away. We didn't talk about the apartment or whatever. She happened to be the former wall to wall marketing director at Intel. And so she coached me a little bit back then. I mostly brushed it off because I was arrogant. But one thing that stuck with me that I revisited, like years afterwards in one of my notebooks, she wrote something like, 'Manage your energy, not your time'; Right?

And I wish I would have followed this advice because it's so true. Right? Like you can find a lot of time in a day. Right? You can work for 16 hours a day, but do you work with a high level of energy? Probably not. So it's much better, in my opinion, to work for less time but with a much higher energy. That sounds the opposite.

What are the key things of productivity or high performance teams is tooling. So this might be a bit tactical, but do you let developers pick their own tools or aim to standardize tooling as much as possible?

I think it depends on where this tooling sits. Right? Obviously you have some tooling that we see in the middle of the chain, like your CICD, for example. Right? So it goes to the CICD pipeline. Obviously, you need to make a choice and just impose it on everyone. Right? But other than that, I've always been super open about if it was either using Windows DS code, MG, or whatever kind of IDs affect productivity. Because all of them have been working for quite a long time in these environments, and they are productive for environments.

So it doesn’t make sense to serve like that at all.

And sometimes that’s even true with the languages. Right? Particularly if you are a micro services architect. Why impose it like one single language? I mean, obviously, you don’t want, like, 10’s of them. Right. But you can have a bit of flexibility as well, which really is very appreciated. So I would say, yeah. If it’s not too hard for the company, having this flexibility of choosing your tooling, it’s quite important.

How would you quantify the impact of software engineering teams on company level metrics? As a CTO, of course. How do you bridge that gap between what my team is doing and these are final revenue growth rates, ROIs, market share? Is that something that you’ve got to usually think about?

Yeah. It’s just difficult to really sort of identify what beats the other. Right? Obviously, if you produce bad quality code, that will have an impact for sure. On the other hand, if I again look at my own experience when I was doing Seesmic, we were kind of the King of partnerships. So we would have partnerships inside all the big ones like Twitter, Facebook, Salesforce, Microsoft, and so on. Way too many partnerships. And what that created was, first, a lot of distraction. A lot of pressure to deliver on their really tight deadlines.

And essentially, that creates technical debt. Right? Because you need to go fast and so on. And it’s that you need to repay at some point. And if you don't repay it, that has an impact on the business as well. Very clearly. And so that’s something I... When I talk to CEOs and also to CEOs, I’m trying to make them aware of all these things that are not, not features. Right? Like the features of a product. Everyone understands that. Right? But really, few CEOs actually understand what it takes to build good software.

And it’s not just about features. It’s not about taking and you add this new thing on the front end. It’s about repaying the tech debt. It’s about having a great infrastructure. It’s about taking care of the scaling and kind of pulling the architecture. Right? Like the securities and monitoring, like all the things that you don’t really see. And if you don’t have them at some point, it will just break or put you in a pretty bad situation. And that will have a real impact on the business. Right?

Yeah. And this a great sort of follow up to my next question, which is how do you actually balance the short term requests, revenue, growth rate, team stuff? Again, it’s like long term technology development, having a great infrastructure, all the things that are behind the scenes and that people who are maybe not technical don’t understand that need to happen. And that might act as a roadmap.

One of my mistakes back then when I was CTO was to not properly educate my CEO. Right? About again, what else am I doing besides building the product? So I think that you need to have a realistic development roadmap to take into our conferences. When I talk to CTOs, one of the first questions they will ask is, how do you manage your tech debt? Because that has a tendency to kill companies. Right? Or to make companies miss investments. My company needs acquisitions because of too much tech debt.

Which means that the money that you're supposed to inject for growth actually is going somewhere else. Right?

And so that’s the tricky part. So it’s like everything. Right? There’s like, short terms, wings that you can do sometimes. But you still need to keep in mind the longer term and make sure that you’re taking care of that as well. And again, educating I think the whole company about what it takes to build great software is important. And not only our CEO. Right? Because I ask, for example, the Director of Partnership really early on. We had too much money. Right? So we prematurely scanned everything we could.

And the Director of Partnership, what does he do? Well, he's trying to find new partnerships. Right? And then, he's like, "This supposed tech is not executing fast enough." So, again, a lot of indications and transparency to be made there.

Absolutely. And speaking of the long term, how frequently do you discuss innovation, new technologies with your own teams or is that usually a top down thing? How do you think about that? How do you see the CTOs that you coach go about that?

I mean, it varies, I would say. But I generally think that the CTO would be the one at some point, sort of like taking care of the technical vision. Right? And of the RND topics. So one topic that I took care of at some point for exactly that. Just building prototypes. And honestly, I'm much better at building prototypes than production code. Right? So building prototypes to try to showcase an idea that I have, something that could have an impact on the business, and see if you could fly or not.

And then, discuss that with your entire team and see if there's potential for it. And then, handing that over to the engineers who are like, "Oh, my God. What is this code? Okay, Let's build that property now." Right? But another thing that I did with my team as well is many actions? Right? And that has this tendency also to bring a bit more motivation in the team when everyone knows that they can actually contribute to the innovation. Speaking of which, I think one mistake that is being done in many companies is like siloing the different departments.

And so I see many many startups who have this tendency to see the engineering team as almost like their monkey. Right? Yeah. Let's give them the buildings. Right? Instead of taking advantage of the fact these people offer these models, where they can have really smart comments about products. So I'm a big proponent of taking your engineers and embedding them into product teams as well. Making them participate in these discussions. Having your engineers doing customer support as well, because then they're going to feel the pain of the customer. Maybe they will fix these bugs faster than they used to.

Something maybe with the marketing. Right? There is a technical aspect to marketing. And sometimes you see the marketing team struggling. There's a few things that the tech team actually couldn't solve pretty easily. Right? So I think the more connections you create like that are inside the company, the more open minded your engineering team will be. And also the more innovation will come out of that, because they will be in touch with these ideas, and we bring their own as well.

Yeah. A lot of food for thought, particularly the piece about embedding, detecting within other functions. Marketing, product, even support. How do you A) convince engineers to support and B) convince product and marketing or whatever that might be to let intruders get into their day today?

That's a good question. I think you need to sort of bring the positive. Right? First and foremost. And try to convince them that it's important to... Yeah. I guess in the case of the customer support, which might be the one thing that sort of like eh, I don't really know if I want to do that. I did a lot of customer support myself, and I think you got to show the example, actually, as a CTO. Right? If you're not talking to product, if you're not talking to marketing, you're doing it wrong. Right?

And if you're not doing support, you're also doing it wrong.

So again, when your engineers see you doing that, it's up for them to say, "No, I don't want to do it." Right? And I might have a few of them being on that side, but I think for the most part, again, they will just emulate your behaviors. It's just going to be a problem if you say, "Do something." When you're certainly not doing it.

Lead by example.

Absolutely.

I think that's a way to wrap it up. What do you think is keeping most CTOs up at night over the next 12 months? I asked, because as I mentioned in the beginning, I think you have a pretty wide sort of view into what many CTOs are thinking right now.

I guess it depends on the stage. Right? That will be different problems. I would say that early on what kept me awake at night was just something that is still more unknowns. It just doesn't go down too often. So just taking care of everything. Right? That was what was keeping me up at night. And then, it shifted more towards team issues and things like that. I think generally speaking, the real struggle is time management.

On one end, being at home can help. Depending on your home, I would say. If you have kids, it might also be much more difficult. But you might be able to find better cans than if you're in an open office sort of context. But really make sure that you group meetings. Right? That you are being cautious about time management. So grouping meetings, avoiding going to meetings where you don't need to be. There are plenty of them. Probably. Making sure that you get this long stretch of time either to think or to build something.

And then, if you go through a round of funding, obviously there's going to be consideration like, "Okay, how am I doing with the TechDB stuff?" Right? But that also can keep you up at night, at least it did for me. For sure. Because I didn't know what to expect. Yeah. I would say good time management. And that's already something to improve quite a lot to your condition as a CTO.

Let's switch lanes for a bit. What writer or book has got the greatest influence on your professional career and why?

I would say that it's a book called, Mirror Worlds from the Developer. So the full name is, A Mirror World of the Day Software, [inaudible 00:35:30] in a Shoebox, something like that. It's a very interesting book because it's from a computer scientist from Yale, who invented [inaudible 00:35:43] systems back then. Created a language called Linda, which is extremely aesthetic. Right? And that has a lot of influence on my way of thinking about software and issues. Not only as a way to do things, but almost as an art form, one could say. Right?

Right? Taking into it. It's like when you look at an object. There's a function, but there's also the aesthetics that goes with it that might serve as a function. And I think it's pretty much the same with software. And so this book really had a huge impact on how I see architecture and how designing things with aesthetic in mind is actually quite important.

I think that is a perfect note to end on. Thank you so much, Johann.

Thank you, Gonz.

It was a pleasure talking.

Likewise. Thank you, Gonz.

EPISODE TWO

Marijn Van Aerle

About: Marijn is a co-founder and CTO of Floryn, a company providing SMEs with smart, short-term working capital solutions. Marijn has an impressive track record in the world of technology: he was the director of Engineering at Trimble and Gehry Technologies, and co-founded several other startups, including Mishare and Perfect Blue.

floryn

To listen to the episode, **click here**.

Gonz: Hey, Marijn, welcome to the podcast. It's great to have you here. Super excited.

Marijn: Thanks. Great to be here.

Let's start with a quick one, with an easy one. What's a two-minute version of you?

I'm the CTO of Floryn. I've been an entrepreneur since I was 18. So I've been working on software, basically websites at first, and then that's escalated into applications and companies, and that escalated into startups.

So I've always been working with my CEO, Sam, who is the commercial guy on our end. And we've been developing products. Our company before Floryn was a 3D viewer for building models, which is something entirely different. And we sold it to a US-based company. And before that, we had a couple of smaller initiatives. But right now, a CTO of Floryn.

Cool. So as CTO and founder, of course, of a fintech company, what is the role of the CTO now? And how did that evolve over time as the company grew?

When we started out, we were with just the founders, of course, and one software engineer. So there was a lot of coding back then, obviously. So I worked on all the technology myself at the start, and that's just for getting the proof concept and the prototype in. And once you have some validation and once we got funding, the role started shifting towards hiring, especially. So building out your team, and explaining to everybody what we're trying to do. And now that we've grown even further, we are with about 50 people now, of which I think 16, 17 in the Development Department.

So now it's more about what kind of company you want to be, and is the team all right, coaching people in management roles, leadership roles, trying to really build out a good structure.

Absolutely. And we'll dive into recruiting, we'll dive into team culture on how to build a high-performance team in a bit. But any recommendations for first time CTOs? Imagine someone who came in at 20, 30 something person company. There's a bit of a team, they are asked to accelerate things. Any recommendations for them?

That's hard. I've never done that because I've always been a founder. Let's think. So once you join a company with an existing thing, then the most important thing is to become knowledgeable enough and to become like an authority. Because I think that's quite important if you're a CTO, that you have this sense of control over what you're doing and an idea of what's happening everywhere. As a founder, I have this naturally because I was there when everything was set up.

But if you come in, I think that's something... And if you don't do that, then the other departments or your engineers or people, they're going to run circles around you. So getting the knowledge of how everything really fits together and where the gaps are, I think that's something that you really need to do. And that goes further than what platform are we using or what technology is it? I really mean like, how does the CRM and analytics and what are we really doing here?

How all the pieces fit together, and where we're going, and how all those pieces help us get there. Absolutely. And at that stage, at least from what you said, that's where you need to start recruiting, building a team, and start thinking about what that looks like. The question is, the world changed quite a bit since COVID. Before that, most companies were local, the talent market was local. You were competing for engineers in a, let's say, 30-kilometre radius of your office. Of course, you could relocate people, but roughly, you were competing with the companies next door.

Now we are competing with companies everywhere. Startups in the US, big tech companies in Singapore, whatever that might be. So how are you thinking about recruitment right now? What does it take to recruit and retain talent in this new post COVID world, when the talent market is global?

That's a good one. So I think it's always worked to our advantage, this location thing, because we are a bit in the South of the Netherlands, which means that everybody who doesn't really want to go to Amsterdam every day but does want an interesting startup to work for, can come to us. So the location, I think, worked a bit in our favour. And right now we see that we are competing more with the Bookings and the Ubers and the big Amsterdam tech companies as well. So it's become harder in a way for us.

So what do you do? What I think and what I've always thought is that the challenge in itself, that's what entice people. So you should have nice events and days off and all the checkboxes you need. But the most important thing is the actual challenge. People want something really interesting and cool to do, where they can make a difference and where... Especially for engineers, the technology needs to persuade them, basically.

Yes, everything needs to be good, but everybody expects that. Everybody knows if you're working for a startup with money, that you get all these nice perks. Well, it's very rare that we can entice people with that. It's about the actual challenge. So we are wiring a billion Euros every few months. We are doing machine learning to improve... So those things, that's what brings people in.

So it's a complexity of a technical challenge.

Yeah. And for most of the people where it's been a real success, that was the differentiator.

Absolutely. And probably you could make the case that if someone is there for the foosball table, they're not there for the right reasons.

Definitely. Especially at a startup, if you don't really want to dive into the work and want to have the work eat you alive, basically, then you should go somewhere else.

Yes. I can definitely relate to that. What are some of the core personality trait you look for in new hires outside of their technical skills? Those are the foundation. But when you think about core traits, culture, what do you look for?

I think most of it is the standard stuff. You need to be enthusiastic and motivated. And you want somebody who really goes for a challenge and who basically wants to change something drastically. You want to see that somebody has strong opinions on certain things, and is really motivated to do something about it. So you should not be completely neutral towards all of the things, because then you probably don't care. So that's one thing.

And the other thing is, maybe the opposite of that, but we have quite a stable team, I would say. I also like people who are, how do you say, professional. So you don't come here for the foosball table. And also, we're not going to let you work 80 hours a week, we know this. You're supposed to have a good work-life balance. And if you can do that and still do great work, then you're in it for the long game, basically. So if you have kids and if you have other things to do and hobbies and if you can be professional about that, I think that's really important.

We have that in our team. We have people that they can separate their private stuff and their work stuff, and they know how to keep the balance basically.

Is that how you ensure that you have a stable team? That you retain that talent and they stay there for years and years? That's the goal?

Yeah. It's always the job. If you have this methodology with sprints, but this is not about sprints. This is the marathon. This will take a while.

Absolutely. What sort of career tracks or career progressions have you developed for your tech teams? You're thinking of people who want to stay there for a decade-plus.

So it's a bit natural. We took a good look at other companies that have open-sourced all this stuff, basically. And I would encourage everybody to do that. Base camp is not the best example right now, probably, but their [crosstalk 00:10:10]. And it really hits a lot of interesting topics and value. And GitLab has a really nice one that I would encourage everybody to take a look at if you start structuring this, because it's very good. You can really learn from what they've done.

So what we have is two tracks, basically. A management track and an individual contributor track. And if you're an individual contributor, you're about the technology. We have people who are really good at technology and not that good at leading people, but also, if they're professional they understand that about themselves. So they know that they're not natural leaders, but they are so good and they are becoming slowly an authority within the team. So people are listening to them, and they are being an example for people. That's the individual contributor for us.

And you can grow from junior to staff engineer or something like that. So the main point is it goes higher in senior and also in salary. You can keep developing yourself, even if you're about the work. You don't have to be a manager.

That is very, very interesting because most companies and most people don't recognize this. It's like the only obvious career path is to become a manager when some people are great ICs. Why do you come to this conclusion? How do you recognize this? Because it's something that I try to do in my own work, but it's something that most people don't.

We had a big overhaul of this whole system. Like how are we going to do all our job profiles and everything. So we took a good look and we got inspiration from other companies doing this. And we saw this, I think, in GitLab, and then we saw the track of... And we immediately could think of two people in our team, like, "Yes, this is them. Exactly."

We basically bounced off the job profiles to those people and said, "Hey, this is some role we have in mind for further growth." And they said, "Yeah, this is me." So [crosstalk 00:12:10] right? It's really hard to come up with a role if you don't have as a person there. You can do this when you are three people, but it will never click, because once you are bigger then different people will emerge.

So my advice would be to really look... Every job profile has this list of, these three guys are in here. These two girls are in there. So that's what makes it work.

So what practices, culture, principles or just routines or rituals, maybe, do you use in your org to create a high-performing team, high-performing environment? And that could be anything. It could be from stand-ups to culture stuff. But I'm curious, so keep the question open.

So I'll not go into the stand-up things and stuff because that's fairly standard. I think two things we maybe do differently. One thing is we don't have QA and we don't have a release team or something like that. So we take ownership of monitoring, doing releases, and doing technical support for the other departments. We take ownership of that within the development team, and we have this rotating responsibility.

So there are sub-teams of two or three people who are working on something. And every sub-team has the responsibility, for one day a week, to take care of these tasks. This, I think, is really important, because if you have a bug or if you have a support issue or stuff, it comes back to you. So that gives a huge sense of ownership on what we're actually doing. You can't have everybody looking at everything. So that's why there's this scheme in place so that you also have other days of the week where you can work completely uninterrupted, and you still are with two or three people, so you can divide up the task. So it's not like you have to be alert the whole day, every moment.

But I think that's something that we're doing differently. And that means that the people who have that responsibility, they have to check all the errors and exceptions coming in. They have to check all the messages coming in from users or our internal department coming in, like, "Hey, this button doesn't work. Why is that?" And stuff like that? And they have to manage the release.

So we do releases every day and sometimes multiple times per day. And it's a very automated process, but you do have to take care if stuff goes wrong, or you do have to make some release notes, or sometimes you have to communicate. So there's this list of stuff going live, and you have to communicate to people like, "Hey, your feature went live."

And this is the responsibility of this group. It's especially the responsibility part that's important. So it's not that much work manually. But if something goes wrong, then it's always clear like, "Okay, we own this problem ourselves. It's not some other department that has to fix this."

It's the ownership of the thing that's key.

And not saying they have ownership, but actually pushing the ownership by letting people handle the issues.

What indicators, if any, of course, do you use to measure your team's performance?

If you look at indicators or measurements, I think we're not that good. We take the human approach. So what you do is you talk. You talk with each other after every project. So we take care to do a good retrospective at the end of every project and share that with the rest of the team. It's really important to us that we do that every time, and that we do it well.

So then issues come up. You have a lot of one-on-ones with all your people and we structure them. So it's not like we're talking "Hey, how are you?" It's about the actual performance. We have these things, like the quality of your work and the performance and the culture, the communication part. You have a checklist of things that you want to discuss every time.

And I think that's how we manage performance and how we bring up issues. We are not that good in the metrics part. Maybe we want to become better at that in the future, but I think we are very structured about the human parts.

I'm curious. Can you dive deeper into your one-on-one structures and how you approach them? How often do you do them? What's the agenda like? I'm a big fan of structured one-on-ones, so I'm curious.

So right now, I don't really do all of them anymore because that wasn't working out that well, because I have too many other things. So now we have a layer below, and they have like five or six people in their team that they do the one-on-ones with. And that's far more manageable, I think.

So the one-on-one structure, I think it has the human part, but we try to drill down from the actual performance review that you're going to do. So we have a quarterly performance review and a yearly performance review, and the points that we look at in this performance review should be addressed in one-on-one so that is never a surprise. [inaudible 00:17:32]. You really want to talk about these things all the time, because you're going to grade. It affects your salary in the end. So you really want to talk about those topics, and you might talk about other stuff, too.

So performance is one, which is basically what can you get done? Is it above or below expectations? How many other tasks do you perform? How many issues did you fix? How many new things did you create? Are you happy about it? Are we happy about it? So that's the easy one.

Then there's quality. So quality is you're doing a lot of stuff, but it's breaking, or it's not well documented, or your planning is always off. So that's also quality. For developers, planning is part of quality. Because giving clear expectations on things. If you're blocked, are you gone for a week? Or are you communicating about, "Hey, I'm locked with this, and are you managing this process?" So all those things are quality for us? How is your technical knowledge? Did you come up with a good solution, or does it simply work?

Development is a clear one. Did you learn anything new? Are you spending enough time to develop yourself? If not, can we help you with that? And then the last one is engagement. Which is more about, it's not about working 80 hours a week, it's more about communication. Are you transparent on your progress?

Usually, if things aren't going well, then people tend to hold their cards close to their chest and not really talk about it because it's not working. If you're a good communicator, then we want to see those. That's what we mean with engagement, like communication. Do you take responsibility and ownership if something goes wrong, are you pitching in during meetings, during retrospectives? So those are the four things that we want to look at, at basically every one-on-one, and every quarter, and every year.

That's great. You said you were bad at the performance metric stuff, but I could argue that this is better because it's a holistic, all-around approach. It's not just hard numbers.

It works less well. That's what I'm still struggling with for the whole team. So if you want to see how the whole department is functioning, that's a bit harder. So all I can say is with this structure, with this coaching structure, we're surely trying to get the maximum result out of the people. I'm confident about that. But on the Department level, it's always really hard, I think with engineering.

That makes perfect sense. Switch lanes a bit. Going back to ownership, do you let engineers pick their own tools? Do you aim to standardise tooling as much as possible? What do you think about that? Because that comes back to ownership, but also having a team-wide processes.

There is freedom. I think in the end, the engineers pick the tools. However, it's not like an individual picks the tools, that doesn't work. So you have a certain amount of, some people say, credits or tokens for doing fancy stuff, and you really want to use the tools that you have and not rebuild stuff, because then you're wasting a lot of time in connecting things to each other.

There's a good argument for standardizing. So Yes, we do standardize. It's not like you can't do anything else, but it has to make sense within the tools that we already use. So, yes, there is freedom, but then you also have to own the trade-off.

So we're going to switch all our react infrastructure back to generating stuff on the server. It's pretty cool that you're enthusiastic about this, but what's the consequence of that? Are we going to have all the application in there, and what then? So I don't think a lot of freedom... I'm not against anything. I'm not really opinionated.

But if you give it a good thought process, then usually you're going to standardize.

I don't want to jump in and say a thing, but at the end of the day, it comes down to... The theme of the conversation is ownership. Not the tools, but rather just it's ownership of the entire thing. As a CTO, and a co-founder of the company, how do you translate your ideas and your vision and the mission of the company into terms or plans that the technical team can actually get behind?

That's hard, always. It takes time. What we do is we have a demo. So that means that we demonstrate everything that we've done to the company every month. But we also reversed it so that... The sales and marketing and finance department here, they also have a demo where they demo what they've done. So that means that everybody's directions, basically, that everybody's going into are always communicated. That was not a direct answer to your question.

Don' worry, that helps.

So what we do in terms of the vision, we have a plan for the year, of course, that we communicate very clearly at the start of the year. And during the demo meetings, I usually give an update. "Okay. So we had this plan for a year. It's been three weeks into the new year, and we are already doing different things." So that's fine. Let's address that. And let's say, "Okay, this was the plan. This is where we deviated, and this is why."

So I think regularly repeating this stuff... I'm always the one who pulls out the year plan again, only to say how we didn't do it. But that does give a real sense of that we are thinking about these things. It's not just an accident that we didn't do it. No, we considered it and there were alternatives. And then we made this decision. So I think that's the way that you do it.

And on Monday, we always have a check-in with the whole development team where we basically address the major projects and how they're progressing. And one of the things is that if there is anything in general to update, so any direction changes, I address them. And I give a short presentation or I let everybody know, "Hey, we're thinking about this." So communicate a lot and repeat it. That's what I would say.

You mentioned Monday meetings and then one-on-ones and then team meetings with demos, but you also mentioned uninterrupted work. How do you think about meetings? How do you make sure that engineers have enough deep work time to actually get things done?

If you have these meetings and then you have the stand-ups, and then maybe a project check-in, but that's it. So that leaves a lot of uninterrupted time. The only problem is for the manager people, because they are in, like, five different projects. So then you have five check-ins, and then you have the one-on-ones, and then you also have all the other demos that people have. Then it becomes a problem. But I think for developers, it's... You should ask them, but I think it's quite okay.

I think it's maybe two meetings a week and your stand-ups. Two or three a week or something. I think that's the goal. Unless, of course, you're actively preparing a new project or those kinds of things. But then I don't think that's meetings, that's more working together.

Absolutely. But this goes back to the IC versus manager thing. ICs need that block of time, but managers don't. Their actual job is to be in those meetings. It's not a waste of time. It's what they need to do to keep a pulse that, to keep the ball rolling, to make sure that everyone in the team is okay.

And another thing you could do is, we try to have regular time slots for these things. So if you have a project check-in meeting at the start of the project, we lock in that slot for six weeks. So you can plan around it. You try to not do it at 10:00 in the morning or at 2:00 or 3:00 in the afternoon, but you try to keep it close to lunch or close to the morning stand-up, those small things. And by keeping it at the same time always, it's much less of a distraction because you can plan around it. You can plan your day around it.

You mentioned the reading of the conversation. One of the things that used to attract and retain talent is the complexity of the technical challenges. How do you balance the short term revenue requirements, like any company has, any startup has, against long term tech development? You're not running like an obvious... You have a lot of complex technical challenges, that's what I'm trying to say. You're not running like a very standard thing. It's not an e-commerce company. You're dealing with money and lending.

So how do you balance the long term things versus the short term task, right?

Right, short term revenue requirements with the long term tech development. How do you make sure that you still have the North Star over there?

They should be connected. If you have to do something in the short term, it's probably because there's a good reason for doing it, and then we should do it. And then it should also be part of the long term vision. So I don't think the two have to necessarily block each other. Unless you're talking about things like improving and maintaining technical depth, for example, that's something that eventually you have to push that through once in a while to make sure that you're doing great, and that's long term.

What we do in terms of our roadmap process, is we try to keep the time frame that we think of as short. So, yes, we have a year plan, but it's an idea. So every six weeks, we have a cycle that we work on in the project, and then we have two weeks to pick up all the stuff that came up in between. And then we again have six weeks to work focused on a project. And that cycle goes for the entire team, and actually for the company, because there's also other departments involved in these projects.

So everybody knows this cycle also, because there's a deadline when you can get a new project in.

And six weeks I think it's quite nice, because you can really do something there. It's not like a two-week sprint, and it gives also time for planning and stuff. So usually something big, you can see as a chain of six weeks projects. So like two or three, six weeks projects. But eventually every project has to ship something. So it's not like you're doing six weeks on some technical stuff that doesn't do anything. It should do something after six weeks.

But you always know that after six weeks, there is no guarantee that your next project will go through. Because that's the flexibility that we need. So you have a project for six weeks, then we decide with the management team and all the stakeholders what's the next cycle going to look like, and that's it then. And, yes, usually there's some continuity. But I think the flexibility is far more important than this long term vision thing that might happen.

It's like the six weeks is the forcing function to A, to ship something that's usable instead of just having, like, a six-month project, and then to put out a hard stop so you can actually rethink priorities.

So you cannot change the time. It's six weeks. That's the only thing that's not flexible, the six weeks. So you have to scope it, basically. You have to think, okay, I'm three weeks in, and we have this huge list over here, this won't work. So then you can't change the six weeks, so you have to change the scope. You can't say, "Okay. Yeah, we'll take two more weeks and then everything will be done."; That never happens. You have to cut the scope, basically, if you can't change the time.

Interesting. It's all trade-offs.

We stole this from Basecamp, by the way. Or stole this roughly from Basecamp, adapted this. They're doing it differently, I'm sure. But we did really like this six week, and we tried it out a couple of times. And first we had them staggered. So everybody had their own six weeks project, and then this year we synchronized everything across the whole company, and that really helped.

I'm curious, why?

Because it makes it easier to switch people around, because we have specialists, and most people are really full-stack with some things they like to do, so you can switch them around a lot easier if the projects end at the same time. And also for the manager people and the planning people, it's a lot easier to centralize your planning moments instead of... If you have staggered projects, you're basically planning all the time. And it got really annoying, also with the rest of the business. It's also, "Yeah, we're going to do this project now, and in two weeks one more." And people are constantly thinking, "Okay, is there a capacity to do this?" Because they're never sure when people are free again. So this is far more efficient and nicer for everyone.

Absolutely. I can see that. What's keeping you up at night over the next 12 months or so.

I'm a founder, so the technical things. We have a few fancy technical things that we want to do and not so fancy. So we have a lot of security and compliance projects coming up where we have to do a huge audit and get a certification. I'm sure we'll manage that because we are already supervised by the Dutch Central Bank and we know how to do this stuff. But the thing that's keeping me up is if we can remain as flexible as we are right now while doing all that. So that's something that I'm thinking about a lot.

And the other thing is COVID and the recovery from the economy, but that's not really technical. But for us as a business, this is a really interesting time for SMEs and for all businesses, and for everybody. But from a financial standpoint, it's really interesting how our market is going to react and interest rates and lending and government support programs that are slowly being phased out. Tax relief is going to be phased out. I think we're very, very well prepared for this, but it's new territory. So that's...

It's new territory for everyone.

Yeah.

Cool. Let's wrap up with a fun one. What writer or book has got the greatest influence on your career, and why?

Book, I need to think hard about that. So I'm reading Extreme Ownership right now.

Interesting.

I'm already talking a lot about ownership, so it really triggers me. And I think I love it so far. So that's one thing that I really like. It was actually recommended by somebody from the sales department with us, but it really works well, for me as well. And another book, I think the first one from Basecamp or from 37 signals, it was called back then. I have to come up with the name, but they had Rework, and they had the first one they had that was really a big one for me.

The one on developing software?

Yeah. I should have a list somewhere with all their books. It was a long, long time that I read it, but... Getting Real, that one. Getting Real, the first one from them. That was really important one for me.

Why? You reference Basecamp quite a bit, what attracts you so much about how they work? I know what's going on with Basecamp, feel free to skip that one.

I have no opinion on that as well, because we're... I don't know. I don't know what's going on there. I don't know anyone there.

No, the question wasn't on the politics or the work culture, but about the sprints and the book, and you think about working similar things. But I'm not going to dive into politics at Basecamp.

So what I like about their approach to this is the common sense approach. Maybe it's because we're Dutch or something, but we're simple people. We like to keep things simple and think about stuff and not use the buzzwords and come down to getting real, basically, about the real topic that you're addressing. If you make a job profile, you think about, do the people understand it? Does it hit it? Not, does hit all the buzzwords? And is the work culture in there?

And it's about the real stuff, basically, it's about... And not performing emotions like with scrum. I love the methodology itself, nothing wrong with it. But a lot of people are just going to the motions, and Basecamp is really, and 37 signal, and with this book, they are really against all that stuff, and I love that.

Absolutely. Getting real, having clear ownership, being clear. That's a perfect place to end on. Thank you so much for your time and I really appreciate it. It's been great.

Cool. Thanks for having me.

EPISODE THREE

Felix Eichler

About: Felix Eichler is a co-founder and CTO of Userlane, a digital adoption platform, that helps people master software instantly. Before founding Userlane, Felix worked as a software engineer at Parkpocket, the app for parking space management. Felix is also a graduate of the Technical University of Munich, and was named one of Forbes 30 under 30.

userlane

To listen to the episode, **click here**.

Gonz: Felix, welcome to the podcast. I'm looking forward to this. This should be fun. How you doing?

Felix: I'm doing great. Thank you, Gonzalez, for the very warm welcome.

Perfect. So Let's start with some formalities. Let's get going. So what's like the two-minute version of Felix? Let's start there and then I will start branching out.

I love this starting with formalities. And so the two-minute version of me as a small kid, I somehow drank too much of the magic potion. I started getting into programming really early on. I was like 10 or something when I wrote my first pieces of code. And then it was made possible for me to attend the Computer Science University, Technical University of Munich, a really good one here in Europe, at the age of 15, even before I finished my high school diploma.

And so I was really nerdy as a kid, but then life took a different turn. I got interested in entrepreneurship and really having an impact on society. So at the age of 20, I started Userlane, a software as a service company now.

And we are now five years in. We have collected around 15 million in funding so far. And have just, last year, grown from 50 to around 100 people in the company, now located on all over Europe. And also, I'm very humbled to be featured on the Forbes 30 Under 30 list this year.

That is quite an introduction, and you did in less than two minutes.

Yeah, the formality part right? Somehow I feel [inaudible 00:02:48].

Cool. So let's dive into the fun stuff, really. So co-founder, CTO of Userlane. I'm more in to break it down in a bunch of different ways. But I would like to start with the tension between being a co-founder and the CTO. How do you perceive that?

Yeah. The role of the CTO is really an interesting one. You're in the beginning, there are probably 10 times more business co-founders out there than technical co-founders, so it feels like almost you're receiving pitches and offers and, yeah.

So I met my co-founders just at the pizza booth, really, on [inaudible 00:03:32] in Munich, and we were really hungry and then got talking. And the dynamic was really interesting. So my business co-founder, Hartmut, he's 15 years older, has a background in consulting. So very different also for me.

But this is really... The secret ingredient and the magic sauce of many, well, working relationships in business or in private, is that you share the values, but you have different skills and preferences, and that has really taken us all the way.

So I can look at things from the technical perspective. I can really imagine how to build them. And Hartmut is really good at envisioning how this will work for the customer and what they will anticipating their demands.

So figuring out the chemistry is like the first job. As a CTO, you're basically a business guy who also knows tech. And I think that there is a choice you have to make whether this is what you want.

It's like the business will play a more important part in technology in many aspects, because at the end of the day, you can also die of technological fanciness. It doesn't really help your customers and users to benefit from that. And it's just really beautiful architecture and something like this.

But that is not likely not the maximum, greatest company you can build, because then you will lack growth and that will lead to other issues.

And on the other hand, of course, you also don't want to end up facing insurmountable tech steps. And this is really the job of the CTO, of balancing these two extremes and finding just the right balance for any given company. And any given company has a different taste for their balance, and we'll have to find a different sweet spot.

Yeah. What I'm curious about this role of the CTO is how it evolves over time. So you started the company, you're, well, over a hundred. You raised 15 million. So how does that role evolve over time? And what else do you think are some of the most interesting inflection points?

Yeah, sure. So I've been doing this for five years, and basically every year I had to reinvent myself.

So in the first year, it's basically building the prototype. You're basically coding day and night, building the first version of the product, whereas your business part is building the business, and find the customers and users for that.

Then you basically get into the second stage. You hire your first between one and 10 engineers. And then you're basically acting like a tech lead and team lead. You're enabling other engineers to work. You onboard them. You basically set a lot of principles, guidelines, like you envision the whole thing and plan out the whole thing. But there's also some other coders next to you.

And probably here. This is one of the most interesting parts, really, because there you have to split your attention between... You still write code, probably. But also you have to do investor reports, PowerPoint slides, a bit of company communication and everything.

And in my experience, this is a tricky stage not just because you have to context switch a lot, but actually because you have to change your attitude of the perspective a lot.

So from writing the next 10 lines of code to then, in the next hour, in the next meeting, then pitching the vision, or maybe convincing another investor or employee to come aboard, that is really stressful on the mind much more than actually switching from product strategy to sales strategy to marketing strategy, that is all. If you discuss it at the same altitude, that's pretty normal.

So then if you keep growing, you're lucky enough, right? You get another funding round or have good revenue. You hire more and more engineers. You'll have your first managers in the team.

So you have now indirect reports. And for me, it was also the time to look for a VP of Engineering. And since then, pretty much the role of the CTO really, really changes. And in my situation, I was also responsible for product.

So basically, it's the same story for any product and design team. And then you are tasked with less... there's more and more tasks that are abstract and that don't have immediate deadlines.

So when you still are involved in the feature teams as a tech leader, as a coder, you pretty much know that next week is the deadline and the week after the customer gets it. And you basically see what you do now. You have the effects of your work really, really soon when you sit with the customer and see their excitement.

But now when you're the CTO, and you're responsible for strategy that takes longer to develop, to build up. You need to talk to dozens of people. You need to make time also for yourself to think about it and to find inspiration. Then you express it. And then it takes multiple development cycles for the product teams and the engineering teams to build that and to, then, release that.

And then eventually you'll earn the fruits of that by having many, many customers or a whole market segment, your niche that you try to find. You'll figure that out by getting the feedback from them in numbers.

And it's all much, much more abstract, and so you need to... one learning I really had is you need to focus on spending your time proactively on what you should be doing, because, of course, you have a lot of context in the organization. You know probably large parts of the code base and the whole product. You probably just know a lot because you've been in the company since day one.

And people will pull you into things which is good. But you need to really... The doses makes a difference between poison and medicine here. So you really want to make time for proactivity and working on strategy. And meeting with customers is really important, especially for anyone, also in the engineering and product side of the business. This is not just for sales and marketing people because you got to understand what they really want and what really are their jobs to be done.

There's so much I want to double click on, and I'm not trying to go in chronological order. But the second stage, sort of that inflection point that you mentioned is tricky from the moment you are the problem solver, the hacky coder who builds the first product side, to raising your first round, hiring your first team member, and just having to do this context switching between managing, recruiting, coding, strategizing. What do you think some of the common mistakes that people in CTO or technically at this stage should be mindful of?

There are many, but I'll just say them in the order that they come to my mind now. But one of the biggest things, the biggest risk that can happen to yourself is burnout. You're just between everything you think you need to work more and work as an entrepreneur or also for many folks that work in startups work is a bottomless pit so it will never end. You'll never finish work. It's not that you have shifts for one day, and after a certain amount of time, you go again and was great.

And then you really have a dedicated time for friends, family, social, exercising, doing sports, just doing nothing also. And you need to make that. You need to really build that into your schedule. And you need to prevent yourself from burning out. And you only feel that after two or three years.

And that has been, for me, one of the biggest learnings and emphasis now. And I now want to build an environment that is not just for me as the individual person, but even for, also, other team members who also feel that, to build a healthy work life, balance of work life integration model, where it's really about the impact and about the results. And this is very good if you have OKRs and stuff like this. But then how much time you put into the work does not matter that much anymore because it's about impact.

And also you should prevent yourself from feeling guilty when you're not spending anymore the 70 hours that you probably did in the first phase. And then one of the other mistakes is just not thinking about these four topics that you mentioned like strategizing and managing. Sometimes people just don't prioritize them. And then you build also... You have to strategy depth and management depth, same thing as tech depth if you're negligent about these things, and that you need to clean up at some point.

And so one of the common mistakes that I think is very likely that you might fall into it first is not even thinking about them. And so you need to actively work on yourself. And work on what you should be doing and exchange with peers and exchange also with founders and other CTOs who have been there or done that. And this is where I learned the most.

So really building up this mentor network. It doesn't have to be a dedicated mentoring. You can just probably start with reading a few blogs, listening to podcasts like this, and probably also join a mentorship network. And there's many free ones out there.

Yeah. Absolutely. I think that's going to be tremendously helpful as people go through those stages. The other inflection point, and this comes from my own experience as well as this shift that happens when you start... Or you go from managing ICs, from managing individual contributors to managing managers. You mentioned this idea of recruiting your VP engineering. So how do you think about the difference between managing ICs and managing managers?

The short answer is it becomes easier because when you're dealing with managers, they basically also sit on the other side of the table with their direct reports. So they likely are more aware of what's going on and what the expectations are. And they are more of the whole meter level of the process.

So I generally find it's much easier to manage senior people. One of the biggest differences is that you have to really trust them, and that you have to... likely they're going to know more than you in something. And you need to enable that, and you need to allow them to build out also their vision. And so you got to get really in sync with them and think about what type of company you want to build. And then essentially, you'll pretty much work with them, like you would as a partner.

Absolutely. I think this idea of trust is key when you think of managing managers?

Definitely. It's basically like a chain of trust and as a chain of, also, enablement. I mean, the same framework still apply. It's not a different except for it gets easier.

So it's always about providing a purpose, providing them with autonomy, and providing them with opportunities for personal growth. And if you think about these three things, you probably cover 80 percent of what you have to do as a manager or as a leader that you said, actually.

Speaking of being a leader, you said that... you chopped it up into five stages. In later stages, one of the hardest things is to block time to think and strategize. Why do you think that is important? And what's your daily routine right now to make sure that you do that stuff and everything else, of course?

Perfect. Yeah. This is a really important point. And there has been a moment where I got so frustrated with my schedule. And I really felt like so dizzy and confused every evening. And I really felt like I'm missing out on my responsibilities so I started to play Calendar Tetris. So looking at the calendar from a game point of view, just how much free space can you get, or how can you optimize it so that it suits you, basically.

So what I've done is I have that 1:01 PM rule, and this means I only do meetings after 1:00 PM. This works not every day. There's townhouse in the morning, for example, in our case. But basically, if you do it 80/20, that's fine.

And then you have all mornings to yourself. And the trick is, in the morning, don't start with reading anything. I don't even look at social media in the morning. I also don't look at my email or Slack in the morning.

I try to have a cold morning, eat breakfast, maybe go for a brief walk, and then start with what is really the priority one in the mid and long term so that I work on that for like 2-3 hours. And then it's basically already lunch.

And then in the afternoon, I get to manage and communicate. And it will also be easier to communicate because you will have basically done your homework in the morning and many things will become easier. And then, of course, it's really about, yeah, do you need that meeting?

There was one moment I was saying to our entire engineering and product teams, if anyone feels useless in a meeting, don't join it. That week was a bit chaotic, I must say. But it was good because actually, it resulted in five meetings being turned into two meetings because people started questioning a lot. And it's really good.

So basically, I motivated the team, again, to self-regulate, which meetings they feel impactful? Which meetings do they look forward to? Which meetings do they say, Oh, no, not again?

And then this is feedback. This is a signal that either the meeting is pointless or the point was not understood, or the purpose of the meeting was not understood. And then it's the manager's or the leader's responsibility to reinforce that point.

And now also, I try to have no meetings at all on Tuesday. And that also, of course, does not work every day, every Tuesday. But that is at least a good ambition. And then on Tuesday, that's either also often becomes a workshop day then, and you get to really spend it on the mall in the long term.

And I think also, it's completely normal if you're in an early stage startup and this schedule totally wouldn't work for you. I've also only implemented this after I had already my VP engineering in place.

Yeah. Absolutely. I want to go back to the point of meetings. And you said something that's very, very important, which is empower your team to make these decisions. And I think that's something you should do consciously, because meetings are very, especially recurrent meetings are very easy to set, but they are very, very hard to remove because you end up hurting people's feelings. Sounds like why change something that we've been doing for months?

How do you communicate that to your team to make sure, like, hey, it's okay if you don't come to this meeting. It's okay if we just kill this meeting?

I mean, that's not what I said. It's not okay to just randomly kill meetings or something. Meetings have a purpose. And you need to ask yourself, what is that purpose, and can we also do that without meeting synchronously at the same time, where connections will be bad, people will be late, people will be not there that day.

So prefer asynchronous communication written or recorded, maybe in a Loom or in a brief video of yourself, rather than just calling everyone and demanding their attention.

So this meeting could have been an email. It's a write-sentence. And every meeting maybe should be an email. And then only if you don't get a response after following up, then you set a meeting to really grab someone's attention. It's also like you don't just phone call people before texting them or something. And the same thing is with meeting. Maybe don't set a meeting, rather just first send a message. And then only if there is clarification needed or written communication doesn't work, then maybe meet.

Look at all the other options that you have to convey information. What if these things can be automated? What is really the meaning of the meeting?

And then, of course, if you always keep talking about the high-level company goals, the happy faces of the users and customers, then every meetings will also feel more meaningful. And it's a beautiful way to connect everything together, also with the use of the OKR framework, for example.

Yeah. Absolutely. Of course, I'm not implying that we should kill meetings for the sake of killing meetings, but the mentality of reassessing if the way we are communicating is the right way.

I can tell you're very intentional about the communication structures in your team. So tell me a bit more about that. What structures you have in place? What tools do you use? How do you think about sync versus async?

Yeah. These are good questions. And when you say that I'm intentional about the communication and my teams, well, we from a technical background, we are also very intentional about why our microservices and APIs communicate.

And so when you look at a company or a team, it's actually very similar to microservices. When you scale up, microservices is usually a good approach because you don't need to be in the full context of the domain in order to do a change. And same thing with teams. At one point, after more than 10 engineers, it's just not ever everyone will know everything that's going on.

And you build communication structures that allow individuals to not know the big picture and the full picture. And therefore, you need to basically start different squads, as we call them, so different sub teams. They will post their updates. You can read through them like a newsletter, but you don't have to be in their sprint planning or something.

This also means to include, for example, the sales marketing teams and customer success teams. This is a really good practice we found over the last year at Userlane, that this creates also direct feedback channels where you want them and actually cuts down on some of all this repeating the same topics to the same people again and again.

One more point I want to add to this that this has very much changed with the age of remote work. So then very much, it should be asynchronous first for many reasons, because also, if you're scattered around different time zones, if you are getting people together, it's also always should be the last resort. There is this order. First, write something down, then maybe call a meeting, and then only as a very, very last resort, you get people together in a group.

So you also need to just map it out. So for example, we have a small little spreadsheet where we say, okay, these are the topics we want to talk about on a weekly level. This is what we can save us for the monthly level. This is for what we do on the quarterly offsite. And that really helps us to also prioritize and to remain focused.

Totally. You said the importance of async communication as the world changes and then goes remote. What I'm very curious about is how do you think about hiring and recruiting talent in a world… Essentially, COVID made talent markets global. You guys are based in Munich, right?

Yes. I said we are originally based in Munich, but basically, we are now remote first. Many of our people are in Europe, some are outside of Europe. And as we grow, we will definitely become more global.

And well, the talent markets are also global. And going remote is also not about saving costs. It's really about access to talent. But also now that everyone's going remote, the competition for remote workforces has also increased. So we also see salary is really increasing in many, many regions and markets.

And one thing we try to do in hiring is we basically... I approach hiring conversations, just like I approach also one of one conversation. So I talk about autonomy, mastery, and purpose.

And basically I look for this win-win. So for any given person, any given candidate. What do they really want? I'm going to really try to see through that.

And when they tell me, yeah, I've studied computer science, and then blah, blah, blah. And then I asked, no, but why?

Why did you study computer science rather than listen to their story? And often they need to think a little bit, because apparently in many interviews, these questions don’t get asked.

So I tried to do that. And I try to really understand if I would be just a friend of this person, would I even recommend applying this person to... That this person applies to Userlane and it works here. Is it really a fit? And I tried to really elaborate on this win-win. And of course then, sometimes position in Userlane, sometimes rather not. It feels very organic and natural. And this also works remotely.

And at the end of the day, my site mission at Userlane is also to build a workplace that people really love working for. One of my inspirations of why I became an entrepreneur was that, for me, it was not taken for granted that work is fun.

And actually, many of my early childhood friends or also family members, they did not enjoy their work so much. So I specifically was like, no, this cannot be truth, that you just go to work. You look at the days to end and the years to end until you retire. That’s so sad. I don’t want to do that.

So I met entrepreneurs. And they struck me with some of their enthusiasm that some of them had. And so this is how I got into entrepreneurship and starting to enter the, hey, companies don’t have to be bad. There can also be good companies. Work can actually be fun. Work is actually, for me, now and for, hopefully, many at Userlane, is self-realization and self-fulfillment.

And this is what I try to manifest into the culture. And this, basically, starts with hiring. And you basically ask what people find fulfilling. And then you see, does that fit what skills and types of people you need in the company?

And if it’s a fit, then you basically make them a good offer. And your culture is part of that offer. And basically, I sometimes say that if a meeting is not fun, then it’s not a Userlane meeting.

And so people really enjoy collaborating with one another. And I’m especially proud of this. Since in the last 12 months, we grew from 50 to 100 people. And it’s actually feels like we’re even having more fun than before, and I’m really proud of that dynamic. And I hope that I can keep it going until we are much larger and become a good enterprise, not a bad enterprise.

But one thing is for sure, you’re going to become an enterprise. You’re not going to just slack your edge offers anymore to get some paperwork done. But you’ve got to create a ticket in some system, and they’ll work over it in the next few days and stuff like this.

But this is not bad. This is actually just also to help them focus. This is also just to... The form can ask you questions that you wouldn’t have thought about otherwise to include information that you would have omitted. And it makes just things good.

And as long as you treat people as humans and not as like cogs in a system, I think your culture will be set up for success. It starts with yourself as the leaders. And then the people you hire and fire, that is how you basically manage your culture, you manage a product.

You listen to your user feedback. And your users, in this case, are your employees and your coworkers. And then you iterate on that. And you open feedback sessions with them. And you’re really approaching culture iteratively.

And we listen to our employees regularly, and we know that it’s not culture. It’s not something we define top down, but it’s really a collaboration of everyone. And some people are more interested, and they can really shape it. And some people just enjoy and come along.

I’m very curious. What are some of the core traits you look for in new hires? I’m not talking to hard skills or soft skills. Essentially, what are some of the personality traits you look for in people who you think would fit the culture?

The people who are also, first of all, able to enjoy what they’re doing, because this means that they’re going to be in a good mood when they work. And they’re going to be inspiring also to other colleagues that we bring on board.

So many people decide for a company based on the people they meet in the recruiting process. So you start with yourself. If you’re having good days because all morning you didn’t have meetings, and then in the afternoon, you only have the interviews, it’s much more likely that you also are not stressed out in the interview, that you can actually take the time, take a breath, and really try to look this person in the eye and understand their context and their story.

And that starts with you. Then let’s say this is an amazing person. You bring them on. And then, of course, on day 1, day 2 interviews, they bring on along the same vibe and feeling and in the same culture to anyone they interview.

And then I think at Userlane, this is what kept happening over the last five years. And now we are in... I’m incredibly thankful for the situation that we are in that regularly, I hear the feedback from candidates when I interview them at the very end of the process, that they love everyone they saw, and they find this is one of the most vibrant cultures that they’ve experienced in a hiring process. It makes me really glad.

And what we are looking for... Well, as I said when you’re having fun at doing your work, you probably also spend a significant amount of time in it. You probably got good over it over all these time that you spent with it, the 10,000-hour rule. And yeah, then it’s usually that’s just a fit, yeah.

Absolutely. One thing that you said is you grew from 50-100 in the past year or so. And there’s a saying that every time a company doubles or triples in size, everything breaks. What are some of things that you advise other CTOs to look for as they rapidly scale their organizations?

I mean, as any good software architect would say, it depends, I think, on what you... under unique set of challenges in your company. I think generally, if you already the awareness that things will break is a good point to start with. So just think about what you do more then a handful of times, you got to automate. This applies not just to engineering tasks, but literally all around the company.

And you probably want to bring on recruiters that will work for you when you want to hire that many people. And then they define a scalable process for onboarding people for interviewing people. And that makes many things more scalable.

And generally, if you look at a company and any kind of business process with an engineering mindset where it’s about automation, then you know where is the biggest pain. And you know that probably you just need to structure it or automate it or provide some guidance assistance to your users and then any process will fly.

Absolutely. We spent a bunch of time on recruiting things. As I said, we don’t want to make this a two-hour thing. So my question, we’ll edit this out, is essentially, there are a bunch of directions we could go. What are you most interested in talking about in the last 15-20 minutes? Do you want to keep on on this operating tech team? You want to talk about impact, or what energizes you right now? What are you?

That’s a good question. Usually, my answer is culture or something like this. I can really talk long about this, and what does really set a good company from a bad company apart. But actually, if you would ask me that question, I don’t know how I would answer right now.

Culture, generally, maybe we should go... Should we go technical? I'm not sure. You know better than me if that would be helpful for the audience or anything.

I think, maybe, how do you communicate roadmaps, a bit of this product management side of stuff, that would actually be interesting me. And one of the things was also like, how do you report to your CEO board or something? That actually is, maybe, yeah, I think that I'll come up with something on the go.

Let's start with at the top, which is demonstrated impact to the CEO and the rest of the company, which I think is an interesting subtle dynamic. And then we can dive into communicating the vision of the product and everything to your team.

Good. Let's try. Cool.

Perfect. So as a CTO but also a co-founder, how do you demonstrate impact to your team at different altitudes? Both the CEO, potentially, the board or the execs/management team and then the rest of the company. How do you think about that dichotomy?

Yeah. That is a really good question. The short answer is do good things and then talk about them. And the long answer is way more complex: you starts with also, how do you think about your roadmaps. So we adopted Now, Next, Later roadmaps.

So we don't really structure them in quarters or months anymore. And we don't try to predict something that we can't predict. And actually, building a piece of software is more art than it is like a manufacturing work. So you cannot really... there's more pressure exerted on timelines or something.

You're not always going to get better results. And the Now, Next, Later roadmap communicates really well about what stakeholders also want to know. And they are really readable also from a very high altitude, especially if you link them with OKR, so that you're not going to list features, but you're going to list objectives or maybe key results on that altitude. And then these are business objectives. So anyone will understand what that means.

For example, you want to win like 100 new customers in North America. And then that means that you're going to build, maybe a US data center, or this means that you build some features that are more relevant for some houses in that market.

That's basically the features become a side note or become the description text of your key results or objectives. And then you put them in Now, Next, Later. And people know, okay, we are busy with something right now. There is always something in the pipeline. Then we run something like a monthly release radar, if we call it.

So where we only go over the topics that are very close to the finishing line or have been released in the recent past and explain them and explain the impact on the business outcome and on the vision. And then in Next and Later, we keep all the follow up items.

And in Next, it's the topics that we are currently planning or already researching or already have a pretty good idea about what they're going to be, what the scope is going to be.

And then at Later, it really connects to your vision and to your long term goals like, why do you exist? How does your company or your product make the world a better place? Because ultimately, that's what any product should do. And you keep these things there. And you often break them down into smaller bits and pieces that then fall through the Next and Now column so that you continuously communicate on what you're working on. And then you show progress this way. That it has been working for us pretty solidly over the last year.

I’m curious. We’re going back to operations and culture. But why OKRs? Why do you think OKRs make Userlane a better place to work? And I’m also curious, what would you advise a first time technical leader who’s trying to implement methodology like this one?

Yeah. Everyone is talking about OKR. So this word, the meaning of it was dissolved. Now it’s basically just another word for goals.

And the point is that it should be served in a fractal manner or in a repeating manner. So you have a higher order goal, you break that down. And it’s just about keeping, like, hold on to that chain about whatan individual or a team, a small team, will do with their team goal or OKR that links on, maybe, a department goal, and that maybe to the organization goal. And that links to the vision.

And then you know what you’re doing now in the next few weeks. How will that impact the long term vision? And that’s the magic of OKR.

And then one more thing that is often misunderstood is it shouldn’t be things like ship feature X as your objective. That works. It’s better than nothing. But what’s even better is if you look at some product metrics, like active users, like time spent per session or how many sessions there are, how sticky your users are, these should be, probably, your objectives. You can find your own metric. But pretty much, if you’re building SAS product, start with number of users and stickiness. That’s a good starting point.

And then iterate from there. And you can have hypothesis of which metrics really make a difference between valuable customers with a high value to you and not so much. And then you can try to optimize for them and to basically level up your users and bring them to level 1, level 2, level 3, so that they’ll be more and more engaged and more and more deep into your product. And that’s basically what you’re trying to build.

This methodology really connects well with the Now, Next, Later roadmaps that I’ve just mentioned. And together this makes them also really simple to show your impact and what you, as an engineering or software team or product team, are doing for the SAS business.

I mean, you’re at the core of it anyway. So without you, everything would be just a bunch of PowerPoint slides. So it should be pretty straightforward to justify from there.

Yeah. Absolutely. Speaking of KPIs, do you have any internal KPIs that you use to measure your team’s performance?

That is a good question, because it’s like the holy grail of engineering management, like how to tell whether someone’s productive. But the short answer here is you just don’t. You just focus on these OKRs. And if the team works and gets to... If the communication is consistent when it comes to deliverables, if your product is successful on the market, if you’re really listening to your customers. But how exactly engineers spend their days is not too much of your concern. But there is another thing.

What I do is I think about the culture, but also about the developer experience of our code and our platform that we’re building like a product. So I do product management on it.

So this means I interview users, in this case, my own developers and ask them, well, what do you think we should improve? What do you like? What do you hate? And that is really valuable feedback. And you can also survey your engineering team, ask them on a scale of 1-10, how well do you like this microservice or this component?

And then you get basically map out the health of your task stack and what engineers would like to improve. And usually, your engineering team has a good sense for that.

And then you don’t say yes to all these things because, of course, you also want to balance it with delivering a visible outcome to the users, visible changes there. But this is a way to tell how good the developer experience is. Is everyone in pain about your ICT system, or how often do you release?

These are feedback points about your developer experience and engineering culture. And then you basically take it from there and try to incrementally improve it.

I’m going to have a change of lanes. Now, what’s keeping you up at night over the next 12 months?

Yeah. If only I would know that because there is, as I said, probably I need to reinvent myself this year and next year again. So I don’t yet know what exactly it will be.

But generally, over the last years, it has more and more become the long term impact the culture. I’m a bit paranoid. So many entrepreneurs tell me that, yeah, when you are past 100, when you’re past the 50 or the 150, that you got to make really careful that you don’t mess up your culture.

So I’m basically doing more and more work there and trying to listen to people, trying to really understand what is a good enterprise. And really acknowledge that you’ve got to introduce some process and some... It’s going to be more superficial to work with 150 people because you can’t be everyone’s best friend as like you were when you have been the first 20 people in a company, basically spending day and night together and being basically a small revolution. And that is normal. You got to just acknowledge that and be aware of that. And address it proactively.

So every time we go through a funding round or through a scaling phase like this, I basically preach it to the team many, many times. Things are changing. Change is the only constant on earth. It’s not about preserving culture or, if culture was a fire, it’s not about passing on the assets to the next generation. It’s rather about keeping the fire burning.

And of course, as you, let’s say, put different wood pieces in and out of the fire, of course, the color and maybe shape, size, everything will change.

And the question is that, will that be good? And also acknowledge that probably through a funding round, you’ll lose 10 person of your people just because they don’t like the bigger company anymore, or maybe because now they get better opportunities to maybe even start their own business somewhere.

And it’s just completely natural. Whereas if one person leaves the team at 20 people, it’s like a family member nearly dies or something, but just a larger company, it’s okay, that it’s normal because statistically, it will happen.

And just be open about your culture. Do it again, like product management, listen to your customers, your users, ask for feedback, develop your culture roadmap or your vision about what kind of company do you want to be when you are at 200 people? And these are kind of the things that I think keep me up over the next year.

The idea of that the company grows and not everyone is a great fit for different stages of the company is very powerful. So if you sort of set those expectations beforehand, it would make things a lot easier as you grow. I guess, it’s fine. If people just maybe they don’t like 150 people companies.

Yeah, totally. And Yeah, and setting that expectation, communicating that openly, also like treating people with respect if they decide to leave, of course, all of that belongs together. And thanking them for the impact, anyway.

And basically, your culture is also defined and measured by how you treat people that left and that's part of it. So just don't be an ass, just be nice. That's I think the short advice out of it.

I'm curious what writer or book has the greatest influence on... I was going to say, your career, but I'm also curious about life in general.

Yeah, and career and life are basically the same thing anyway. And one of the books that actually expresses this idea also very beautifully, and I think is the one book that inspired me the most is The Meaning Revolution by Fred Kofman.

He's also the author of Conscious Business, which you might have heard of. And he really explains very, very beautiful that work life integration aspect, that work is about offering a purpose, offering a way for them to pursue their own goals in your organization so they don't work for you, they work for themselves in your place.

That is pretty much the mindset and the attitude that kept inspiring me. And that basically was also one of the first books I read about leadership when I originally got into it. And I think for me, this defines the best practice, and I would recommend it to anyone who's going thinking about being a leader and not just the manager.

I think that is a perfect place to end. Felix, it's been a pleasure. Thank you so much for joining me on the podcast. I hope you have a great rest of the day.

Thank you so much and have an awesome weekend.

Ciao Ciao.

EPISODE FOUR

Sylvain Wallez

About: Sylvian Wallez has over 30 years of experience working in the tech industry, and is currently a principal software engineer at Elastic, the public company behind the Elastic Stick: Elasticsearch, Kibana, Beats and Logstash. Before joining Elastic, Sylvain worked as a tech lead and developer at OVH, the hosting provider, and has been a member of the Apache Software Foundation since 2003.

elastic

To listen to the episode, **click here**.

Gonz: Welcome to the podcast. How are you doing? How's everything today?

Sylvain: It's going well, temperatures are going down, so it's a relief. So very good.

That's great to hear. So let's dive right in, but before we do, what's a two-minute version of you. What have you been doing these past few years? Just to give some context to the audience.

So past few years, depending on how far in time you go back. I have 30 years of experience. I've been... And the topic of the podcast, I think co-founder, CTO, tech lead in a number of startups. But for the last five years, almost, I've been working at Elastic, which is well known for products like Elasticsearch or Kibana, acting as a tech lead in the cloud team first. And for a bit less than a year, I moved to a different team where I'm responsible for the SDKs, what's called the client libraries. So it's more like an individual contributor job, as we call it.

Absolutely. So you said you have 30 years of experience and after all this time, and these are your words, you're still hungry and foolish. Why do you think that is important?

Actually, this is not my words, this is Steve Jobs words when he did the Stanford opening keynote, and I love that. The key point that has driven all my career is that, I cannot be bored at work. I'm a passionate engineer, and if I start to be bored at work, you can be pretty sure that in the next six months I will change job or create a new one inside the same company. It happened more than once. So basically driven by passion, willingness to learn all the time and creating things.

Do you have any frameworks for learning? You mentioned that, that's been your thing for the past 30 years. You reinvented yourself, reinvented your roles. I'm curious, do you have any frameworks for that?

Maybe I have some unconscious framework probably. It's always something that comes in my way, it's both interesting technically, and happens to solve a problem at the time where I encounter it. So I would say this is probably the combination of both that drives me into learning something new, or looking at new things. It's not formalized, but it's probably the combination of the two that drives me, basically.

Absolutely. As you said, you're right now at a senior IC role, but you did hold the CTO or tech lead a bunch of times. So I'm very curious, what do you think is, or how do you describe the role of the CTO at an early-stage? And then how has it evolved over time?

So CTO at an early-stage. To provide some context, I was in CTO positions in companies that started from zero, where I was a co-founder basically, up to 20 or 50 people. So still small teams, we can say.

I think the main role of the very early-stage CTO is to be like the Jack of all trades, as we say. But not only in the technical aspects, but also doing some product marketing for management, being very open also. I did a lot of investor [inaudible 00:04:52] when we were looking for funding because you need to have some technical background when investors have some questions. So I was there with the CEO at the investor round table providing all the technical details they wanted to have.

So you really need to have this small key multiple hats, I would say. Of course, my main area is everything tech. But I've always been interested in non-tech things around products that can range from even unit design, for example, to how to approach customers, how to collect feedback and more generally, how to evangelize your product and show your love and your passion for your product. I think that passion is something that shows when you explain it and helps convincing people.

Absolutely. And then that's usually when you're a small company, as you hire more people, you said around, like 50 or so people, how do you see that role evolve when you start to build your first team?

I actually talk more about the external aspect of being a CTO. So a CTO, I think, is the main person within the tech team that is outwards-facing. And then you also have a lot of activities internally in the company. As a CTO, you should be the technical leader. And I say leader on purpose. You're not the boss, you're not the manager. Actually, I hate everything that is related to people management. I'm super bad at that. But being a respected leader, in my opinion, is what drives your entire team.

So you have to act as the technical architect, the technical expert. But also as the leading by natural authority and the technical respect that people have towards you. So you have to inspire them so that they can get involved. And it's not an easy task to work in a very early-stage company. So people have to put a lot of themselves in the company and you're there to inspire them and get involved physically.

I would like to double click on that very important distinction that you made. The difference between leaders and managers or bosses, you said. Can you break that down for me? I think it's a very interesting distinction.

To give an example, something that also has driven my motivation and interest in my different positions is also the respect that I have for the people who are above me in the hierarchy. I have very difficult times working for someone I do not respect. Having some respect doesn't mean there are technically superior, but that means that I trust their decision, I trust their advice, I understand their motivation. So it's probably more about transparency, openness, and, of course, doing a good job in the position that you are in, much, much more than giving orders to people that you are managing.

Absolutely. Like leadership by example, if that makes sense.

Yeah. Well, example, you cannot lead by example when you, let's say you are, managing technical people, but you have to inspire trust and confidence, and that goes through openness and transparency.

I'm wondering, do you find a situation in which openness and transparency backfires? I usually default to that, but I've been thinking about what are the drawbacks of being open and transparent and how have you seen that play out with your teams?

You're right. It can be a double-edged sword, especially when things go wrong. That being transparent doesn't mean that you have to say everything, especially when things go wrong. That is, if the company is not doing well, you should not tell everything. You should say that some things don't go well, but describing the entire full feature as bad as it can be, can have some very negative effects.

So it's also your role as a leader, either people manager or technical leader to protect your team and have them... Not in a bubble because they don't have to be isolated from the reality of the world, but provide them conditions in which they can keep on doing their job well without worrying too much about the future, let's say.

Let's go back a bit. And you said that you developed this very strong interest in a bunch of other areas that are not technical, communication, products, UX, business. Why do you think it's important for technical leaders to have such a breadth of abilities versus just being focused on one thing?

Because ultimately you're building a product so that users use it and customers buy it. So you're not developing for the sole purpose of technical joy, I would say, even if it has to be there, but ultimately you're building a company.

And by the way, this is what I love in startups, product-centric companies compared to, let's say, services company is that the product basically is the company. There is no such thing as different teams competing for resources or having a better customer than the other team or something like that, but it's everybody has one single goal, which is the product and the product is the company.

And especially in a very early-stage, the company... You're working towards basically the survival of the company. So that creates a very strong uniting purpose for the entire team.

Now, how do you install that mentality or transmit those skills to your team, or is that not necessary?

It is necessary. This is something you have to work on all the time. And I would say that for the team, as the technical leader, you have to give people freedom to do their best and be there to help when they need help. But you don't have to constantly look after them, even if maybe sometimes the solution they come up with is not the one you would have favoured. Or maybe it's not the best one, but as long as it works, it's fine. And you're here to just be there, ready to help if they need.

And also your role is also to work on conflicts when there are people who are in disagreement on the solution or the path to take for this other problem. So you're here to jump in and not only impose your own solution, but listen to the different opinions and have the team make a decision. Ultimately, if the team cannot make a decision, it's your role also to take that decision. But that should be, I would say, the last thing you should do. And most often, if your team is in a good shape, they will manage to find a solution and come to...

Going back to that phrase of leadership by example, this could be something like leadership by empowering or by providing the right context, so your team can make the decisions, not a top-down version of that.

Absolutely. I'm a stronger believer in that. And also I do think that a technical leader should keep some hands-on activity on the code base, for example, so that you don't distract with what people are actually... Because you need to understand what they're doing. And you need to be able to understand the pros and cons of whatever suggestion they make when there are some conflicts to solve, for example.

So you should not be this high-level architect that only draws diagrams and hand them over to someone who is supposed to translate them into code. You have to be there, I would even say, inside the same open space if you work in an office. Because then you're really part of the group. So it's important not only technically, but also from a communication perspective because you're not isolated, you're part of the group.

So, you went from CTO/leader to IC, which is usually not that much of a common transition. So any recommendations or mistakes to avoid for others going through that phase?

I know that this is a bit unusual as a career path, and I'm not saying that I will not take leadership or CTO positions in the future. You never know what things will be. But this is also probably related to the fact that I'm really a small team. To give you an example, for example, when I joined Elastic, the cloud team where I was, was 15 people, so pretty small. Which was perfect for me because I was able to provide my best in many different places and again, ranging from working with the product manager in problem definition down to low-level optimization work or solving production issues when things went down in production.

So I was able to do a bit of everything, and this is really what I like. Now Elastic is growing like mad, and over four years that team grew from 15 to 150 people. So basically 10 times bigger, which is when you necessarily have to add structure because 150 people you need to have, directions, management, more fine-grain leadership. And I came into a position where I had to choose between, I would say, more high- level tech leadership or hands-on activity.

So I went on the tech-lead side for a while because I was already a tech-lead, but over time I felt a bit too disconnected from the actual work to my taste. Some people like it. For me, it was not the right fit. So I went back to IC in the cloud team, but then I was frustrated because I was not in a position anymore where I could work on those higher-level things. So there was an imbalance in... I couldn't find the place where I was happy, basically.

So I moved to a different team that was small, we were 10 people. And where again, I can have again, this activity where basically I'm defining the product that I'm implementing. So it's really about me, I think, but probably a number of people are in the same situation.

So for me, I would say it's not like going backwards in your career. It's just like depending on the context and depending on time, you can go from one to the other and back again. And, if your company or the context allows that, then fine, perfect. And you go back into sharpening your technical skill, while you're in an IC position that can help you when you come back into leadership positions. In my opinion, there is no unique path. You can do many different things.

I'm wondering because, it's like the default path to keep growing your career is to become a manager and then manage more people and then manage managers and then manage more people. When there's this alternate path, which is at least equally valid, which is growing as a senior technical IC. Why do you think most companies don't get that dichotomy and only think about being a manager as a career progression?

I'll tell you about my early career. So I started working, like I said, 30 years ago, that was in the space industry. So a large, I would say traditional company working on satellite control systems. And after a few years, when I was 30, my manager at the yearly review told me, "Hey, Sylvain you're doing a very good job. We're super happy about what you're doing." I was already in a technical leadership position in my project. And he told me, "Now you're entering your 30, you should consider becoming a manager."

And I was like, "But no, I don't want to be a manager. I love doing technical stuff. I'm good at that." And it came coming for two and three years. And ultimately I just left the company to be a co-founder of very small startup. So the thing is... I live in France, and in France, I would say in traditional company, this is your career path. That is, you should at some point leave the technical activities to become a people manager.

That was true, I would say a few decades ago. It is not changing, but companies are recognizing that technical expertise has a big value and that there can be a career path that is based on technical expertise. This is something that has been known and recognized for longer, I would say in UK and US companies.

So Elastic typically is a Dutch company, but it's very much US-centric in its culture, I would say. And this career path is clearly identified, and we can see inside Elastic people jumping from IC to people manager and even back, if it doesn't work, the company allows us to experiment. And if it doesn't work, as long as there is an open role to come back into an IC role, that's feasible. So that's a perfect environment, I would say.

Absolutely. The crazy thing is that the other alternative, which is just being a manager, what it does, it takes people from their zone of genius to something that is a completely different skill.

And you know Peter's principle, that is, you go up until you cannot go up anymore, and you become basically bad in your position, and then you're job consisting protecting your position. So that's the worst that can happen in a career path. And for me, becoming a manager will be typically reaching that point.

That is a fascinating mental model, which explains probably quite a bit about growing organizations. You mentioned that as a company grows, you need to add more structure. And you've been in Elastic for the past five or so years in multiple roles. What practices, operating frameworks, cultural principles have you seen being applied to build and operate high-performing teams?

I'll tell you a short story about my previous job. I was working at OVH, which is a well-known European and French cloud provider. That company is also growing very fast, but the management was saying that "Hey, with the growth as ours, no organization can stand the time. And we have to refactor basically the organization every three or 6 months."

And so that meant like giant reorgs all the time, and you never knew what was your current organization or how long it would last. So that was quite disturbing, I would say, for the teams. And that also meant that new organizations were planned ahead of time without, maybe, an actual need for them at the point where they were decided.

Elastic has a completely different approach. That is, we add structure when we feel that the team grows, for example. And at some point we feel pain because the team has grown too large. So this is when we decide to split it into different responsibility areas. That's where we decide to add a people manager for a team so that that person can help the team do good work. This is where we may have dedicated product manager for sub-areas of a launcher product.

So I would say that the very interesting thing at Elastic is that the organization has grown on a is needed basis, but within a framework, that has been defined beforehand. That is more like, when we reach that point, then we should consider that, but do not do it now because we have thought about it now. Do it when we have actually found that there is a need for it. But we have prepared how we want to evolve if the need arises.

Because the problem is that when you set up a structure too early, then you have managers that are not really needed, and then very quickly their job becomes about protecting the wrong role rather than doing a good job helping their teams basically.

Human beings respond to incentives. What you're describing is like a minimum viable process or a minimum viable structure that aligns and structures the entire team, but doesn't put big constraints around it. And what I thought was interesting, it's like you're describing a decision tree. If this happens, then this will happen, and this is how it responds. This is planned beforehand, so everyone knows what to expect, giving them stability. Is that the rationale?

Yes. I would call it evolving stability. That is, do not do big bang reorganizations because those ones are extremely disturbing for the people working in the teams. But more like being ready to... You mentioned MVP. I would say it's an evolving MVP, and not like pivoting every six months because the company has grown. You see what I mean?

That is if you look at Elastic today, the organization is rather different than what it was five or six years ago. But that has been a continuous evolution. And it has been driven by both learning because some teams grew faster than others. So we also experimented some kinds of organizations in early growth teams that have been then transposed into other teams. And then when we see that something doesn't work, it doesn't work. We just change it, and that's it.

You reserve the right to change directions. It's always useful because maintaining a decision just because you made the decision in the past, like it's a sunk cost. So, it's always good to start from first principles.

And again, we can go back here to openness and transparency and accepting mistakes, basically or... Well, mistake maybe is a strong word, but at least recognizing that maybe the path that we've chosen is not the best one. And if the company is in a open-minded a state of mind, then that's not a big deal. The management can say, "Hey, okay, that was a wrong decision. So let's consider something else." And you explain why it was wrong and why you got into another direction.

I think it's all about expectation setting.

Yeah.

Speaking of expectations, you have a very unique perspective as both someone with three decades of experience, technical leadership experience, but also IC experience recently, what do you expect of a CTO from your perspective?

A CTO should have, in my opinion, the product vision from a technical perspective. So you have product management that has it from more like a customer-oriented perspective. And I would say the CTO should have it from a technical perspective. What I mean by that, is that the CTO and his team should work on defining this high-level architecture of the product, which becomes more complex as the company grows. And communicate and drive and make sure that it is implemented with the different technical teams.

Because in my opinion, there are two consequences to that. One is technical and one is non-technical. The technical one is that you have a system that works. It's not like a set of disconnected pieces that have nothing to do with each other. It's a consistent thing. And the other thing is that you also improve cohesiveness, I don't know if it's the proper English word, among the team. That is, everybody, again, is working towards the same goal, and they don't feel like they do not understand what the other team is doing, and that what they are doing actually fits. And they see where it fits into the larger picture.

Because something that can be very frustrating as a company grows is that engineers, typically, at some point we may no longer understand what they are doing is meant for and where it fits in the bigger picture. So as the CTO, you also have to have a clear understanding and a clear communication of that big picture so that people understand where what they do fits in the picture.

Absolutely. I think the word you were looking for is alignment, maybe.

Yeah. Probably yeah.

I'm curious, what writer or book has got the greatest influence in your career or maybe your life if you're up for it, and why?

I'm not a big book reader. I read a lot of technical books, however. I'm a big fan of O'Reilly's books, which are awesome. So let me look at my bookshelves here, what do I have? Getting Things Done, because time management is a big issue for me. Things around Kanban versus Scrum. More like methodology books. Although I'm extremely careful with methodology that are written in a book. We say this is the book, because once you start applying methodology by the book, then you lose basically, the real meaning of agile. So you have to pick what makes sense in your context from those methodologies.

And I also read The Manager's Path, I don't remember the author's name. It's a woman who worked at Yahoo back in the days. And she explains extremely well the career path between IC, Senior Principal Engineer, a tech-lead, and then if you want to go into a more managerial career path. So she describes very, very well the different expectations that are associated with each roles and how you should even prepare for it. And this also confirmed that I was not [inaudible 00:27:42] manager guy and I had to stay in the tech career track.

That makes perfect sense. It's funny you ended up reading a management book to decide that you wanted to be an IC.

I would say know your enemy, kind of. You have to understand what it means before you go that route. So that's what I did as well. Other than that, also reading books like Deep Work, because a problem when you're in a leadership position, being a management or tech leadership, is that you have so many things to do that you can feel overwhelmed. So have to find the time to have some deep work sessions. Because if you don't do that, basically you use your ability to think at a higher level.

So it's not easy because as you're leading more people, you have more interactions, I would say, more meetings and so on. But still, if you want to do a good job, you need to have those time blocks moments where you go deep into thinking. And for me, thinking, sometimes people were surprised when I was working in the office. Because thinking means, for me, putting the feet on the desk and looking at the ceiling. So people were like, "Are you sleeping?" No, I'm thinking.

That, and know your enemy are the perfect place to end on. Thank you so much for your time. It was a fascinating conversation.

EPISODE FIVE

Benjamin Lan Sun Luk

About: Benjamin Lan Sun Luk is an Angel investor and CTO of many successful technology companies, ranging from small-sized startups to enterprises and including names such as Launchmetrics and the non-profit Voxe.org. Now, Benjamin has decided to go the founder route and is starting a new technology company. Today, he is the founder of Leto Legal.

To listen to the episode, **click here**.

Gonz: Ben, welcome to the podcast. I've been looking forward to this. How are you doing?

Benjamin: By good. And what about you?

I'm doing great. I'm doing great. It's been a great Friday so far. So let's start writing. What's a two-minute version of Ben? And I would also love to know, what are you building right now?

So in two minutes, it would be very short. But I started like a lot of CTOs as a coder. Then very quickly, I founded the first company, and then I failed like most of the companies. We tried again, and this time, I tried to not redo the mistakes.

So this second company was called Wisematrix. We were doing prescriptive analytics on social media for brands. That I've later sold to a bigger company, where luckily, I mean, I made my way and I became, after that, the CTO of the company who bought me, which is quite ironic.

And finally, I mean, during this pandemic, I thought a lot about at some point I will do a company and I will do it again. And there's no good or bad moment to start a company. So I said, "What about right now?" So I'm now building a new company where I try to accelerate the world's data privacy. So basically, I'm helping companies to be more compliant over the different regulations that protect personal data.

And what's the name of the company? Just for the audience.

Yeah, sure. So the name is Leto. L-E-T-O. Very simple.

Perfect. Perfect. So what's very interesting about you and your career is that you sort of did it...You did everything, right? You're an entrepreneur with a failed company, and exit. You then took on CTO after a couple of years at the company that acquired you. So you essentially went from companies with zero revenue to scale ups with dozens of millions in revenue.

Correct. Yes.

So that breadth of experience is very, very interesting. So I'm curious, what do you think is the role of the CTO as from the early stage, all the way to the late stage scale ups? How does it evolve over time?

So it's a very good question, because it's actually a question that I got very often when I'm mentoring all those startups. And yes, the role of the CTO is definitely completely different from a narrow stage company, compared to a major late stage company.

So I would say that at the very, very beginning, when you are starting a new company, you do it with your friends. You are just like maybe two or three. A CTO at the beginning is really the solution guide. So you and your friends, you got an idea and they want to see it live. You want to try it, you want to experiment it and do some user tests.

So at the very beginning, the CTO is really the guy who brings solution. It might be quick and dirty, but whatever. What matters at the beginning is to really try, test your idea and put it on the market and see how customers will react to your product. So at the beginning, the CTO might really be a kind of lead developer, maybe, if I would call it this way. In a bigger company, they would call this kind of guy lead developer.

But really, really quick, things get serious. So you start maybe to raise some money, and you are getting more and more customers. They're asking you for like and SLA. And at the beginning, you are like, "What is an SLA? Should I be like 99.9, or 90 percent is okay?"

So then since things are really getting very, very serious. So then you need to learn all the aspects of how to provide a very good enterprise software. And in parallel, you need to scale your team. So you basically need to hire the first developers and so on. And then you become like a hiring manager where it's totally different skill set.

You need to have very nice soft skills, you need to be super empathic, and challenge the candidates, and making sure that those guys are like the good fit for the future of the company that you are building at the same time.

And to finish, later on, once you spend a lot of time growing your teams and so on, then you start to spend some time on stuff that you believe that the beginning was not your cup of tea, stuff like finance, budgeting, and pitching to investors as well to say that yes, why my assets are good, how I'm going to scale my team from 50 to 100.

Basically, they are challenging you, what you're going to do with my money. So I think yes, at the very end, I would say on the late stage company, a CTO is not someone who code anymore. I know it's a big debate right now. There's a lot of CTOs who are like, "Yes, I'm going to code even if my team is like 1,000 guys of developers."

But I think at some point, you don't need to code anymore. Of course, it's super, super important to be aware about what is going on into your engineering organization and what kind of potential big decision on the architecture are made of.

But at some point, no, it's all about, yes, building a nice engineering architecture, making sure that the guys are higher then can grow into your organization, making sure that anybody into your organization got anything they need, like tools, budget and on. And basically, yes, I think that's really my goal as a CTO on a late stage company.

That is fascinating. And I want to double click on something which is that inflection point, in which you go from the individual contributor, you optimize for speed, you're the problem solver to the moment that you need to start thinking about building a team, scaling as an organization, building the foundations of the culture. So that inflection point is usually like the hardest one, sort of a shift in mentality from the 30 coder to a leader. Do you have any recommendations for people going through this transition through this inflection point?

Yes, I remember that inflection point. I remember as well that it was really only myCode. And I know that back in the time, it was like super hard for the external people to get into myCode. Really, at the beginning, it was really hard to explain, yes, why I made this kind of choice and so on.

So I think that here, what I did really was to put my ego on the side and really said, "Okay, there's a way to document everything." I mean, 'm just one guy. So it's not super complex to understand what I did. It's just a matter of time.

So really, at the beginning, giving time to onboard new developers and somehow to delegate part of the code is really the first baby step that I did. Then slowly and slowly, I think that trusting people, really, was the difficult aspect to go through because of course, anyone who is onboarding on a new platform, they will do mistakes. And that's part of the learning curve. It's the way to learn.

So I think that my best advice for the people who goes from the IC path to the more manager path is really like to trust people and giving the time to the people to make their mistakes, grow, and at some point, to own the platform as you did in the past.

Yeah. Those are some great points, this idea of letting go, being low ego. I'm curious, you mentioned that as the company grows, what you need to do is hire, build a team, make sure your team is empowered, has sort of proper career development path. So let's spend a few minutes on that. Because hiring and recruiting talents, especially right now where COVID made talent markets global, it's a challenge for every organization.

So do you have any frameworks, strategies, tips, tactics for doing that? Or how do you think about hiring and retaining talent right now?

Yes, it's a very, very hot topic, especially as you said, during COVID, I mean, the market is...the world market now. I mean, any developers who can speak English, and actually they all do, can apply to any of the companies, wherever it's based in Silicon Valley or in Europe, or even Asia or whatever.

So first of all, I think that before hiring, I would say to make sure that your guys stays. I think that's the first challenge that you should not consider it like done. It's definitely a continuous task to make sure that your guys are happy. When I'm saying "happy" is really that they find purpose on the stuff that they do. As well, that the packaging is super satisfying for them.

And as well, make sure that they work with great co-workers, and the ambiance is good. So this is super, super key. And finally, everything that I'm saying here, it's part of making them growing into an organization. Because at some point, like me, I was an IC. And at some point, I'd like to become a manager.

So really giving all the framework, like for example, a job ladder. Even if your company is not so big, because usually job ladder is for super big companies. But actually, giving a job ladder, even if your company is small- or mid-size company, it's super important. Because basically, you draw the path for the future of your different contributors.

So making them grow and retain your talents, I mean, that's the top one strategy to adopt. And then, yes, once you are good at that, then you are considering to, okay, what is now my strategy to hire the best talents?

And actually, thanks to the fact that you kept the best guys, you...Actually, those guys are the best ambassadors. So they are starting to informally discuss with their friends, their former colleagues, this kind of stuff. Or even there are some guys who are so passionate that they go on meetups, or they participate on open source, and they contribute to the open world that those guys are really the best ambassadors to hire new talents.

That's fascinating. Your strategy is backwards, start with retaining the right talents, a very talented team. And that helps the recruitments, not only because of course, people stay and it's not a leaky bucket, but also because that team will become ambassadors. Is that how you think about that?

It's how I think about that. And at some point as well, those guys are doing so much great stuff for you that you are starting to think that, "Whoa, that could be a nice blog article for my engineering blog." Or, "This could be a good stuff that we could share to the next conference or webinar." There's a bunch of organizations that produce content over there. And it's really like bottom up approach. And this is what works best for me, at least.

Absolutely. Going back to career path, something that's rarely discussed is this idea of...or the situation in which ICs want to have a career path, but they want to remain ICs versus becoming a manager. What's your take on that? How have you empowered engineers in your team who want to keep growing but not become people managers?

This is definitely hot topic. So the first thing that I'm saying is that the manager role is not the natural evolution of the IC role. It's not that because you are good, the best individual contributor, that you will become the manager. So for me, it's definitely two different skill set.

So of course, I would say that there is a common trunk, until you reach a role, like a senior engineer, for example. At some point, you start thinking about, "Do I continue to become an IC or do I continue to do...do I will become a manager?"

And what I'm saying definitely here is that, it's just like a Y path. So it's not because you are going to left or to the right, that you are better than the other branch. So for example, let's say that the senior level is, I don't know, like level three. There's a junior manager, which is level four. But there's, as well, potentially like the lead developer who is the level four, and the same level.

And what I'm saying to the guys and that's pretty obvious as well on the job ladder is by saying that, "It's not because you are becoming a manager that you will be better paid." And then the guys are starting to think in a different rate. And actually, what you need into an organization of 10, for example, you don't need nine managers and one individual contributors. It doesn't make any sense.

What you need is like a bunch of very talented and with a lot of skills, individual contributors who can really push your product forward.

That makes perfect sense. What I'm curious about is, you hire, you retain them, and now you're talking about pushing the product forward. So what practices, principles, operating cadences do you use in your organizations to create a high-performing team that pushes the product forward?

We were talking about the CTO role at the beginning. And I would say that on the late stage company, one of the main mission of the CTO is really to make the organization, your organization, to feel the pain of the customer. So really building a very high-performing team is at the beginning, making sure that your guys really understand deeply what they are trying to solve. So to make sure that they really understand the customer pain.

So that's why one of the aspect of making a very high-performing organization is really making sure that your engineers are super close to the customers. So whatever they did, like a shadowing during an interview, whatever, they went on site, or they are, as well, analyzing the data of the behavior of your customer, I think that this is one of the most important point.

And of course, then, you need great managers to make sure that your guys, your individual contributors, get all they need, like tools, like as I said earlier, the budget, and as well the access to the data or any governance issues, it needs to be solved.

That is fascinating. I like to go back...Actually, not go back, move forward in time a bit. Now, CTO, you're the co-founder and CTO of an early stage startup. I'm curious, what are some of the common mistakes that you made before in your transition from founder/IC to all the way to a CTO that you're trying to avoid right now, and that you flag to others who are in a similar stage? "Hey, be careful of this; I've done it before. I messed it up. I'm going to try to do it correctly right now."

So I would say that the first of that I'm advising to really early stage founders and something that I'm trying to do right now is to really not to rush for the "hypest" technologies that can scale, that can host like for millions of customers. Because basically, you don't have customers right now. There's no visitors, nobody.

So actually, my best advice is really make sure that your solution can really help you to challenge the customers and to iterate easily. So it can be quick, and earlier at the beginning, it's totally fine. It's okay.

So now, after spending six years on building enterprise great software, now I'm forcing myself actually, to not do [crosstalk 00:21:26]. I'm forcing myself to be quick and dirty. So it feels super weird because it's been a while. But now, I know that it's best decision because it allows my team right now with my co- founder to really spend some time on what are really the pains and the problems that we are trying to solve.

So it allows ourselves to really spend more time on these instead of spending too much time on the solution.

Absolutely. What is keeping you up at night over this, let's say, the next 12 months?

I think it's financing. Because at some point, I know that I will need to raise some money. And you never know at what stage you should do that. Because if you do it too early, you might sell too much of your cap table. If you're doing too late, it's maybe too late and maybe there will be the competition will over pass you. And so I think that's my main challenge in the future.

What's very interesting about this conversation is that you think of like...And I can tell this because of what's keeping you up at night is the future and health of the company, per se. So it's interesting that you think of technology, a bit of a means to an end, a way to solve a problem. And that's why right now, even over obsessed about new technologies and scaling to [inaudible 00:23:04] futures, you think about solving a very specific problem. Do you think that's the right approach to think of technology as a means to an end, not the end itself?

Well, it's a really interesting question because this is something that I actually asked myself yesterday night. I will -

Great timing.

Great timing, indeed. So I was like, I remember that my first companies, I focused a lot on the technology. And almost forgetting at the beginning, the distribution, the marketings and sales. And definitely I would say, as like more mature entrepreneur now, as a founder, I know that I need to find the perfect balance between delivering the technology and solving my problems and challenges.

And yes, making sure that my technology is well pitched and making sure that it has the right positioning on the market, etcetera. And finding the right balance, and of course, this really depends on your project, actually, and your market, and your technology as well. Because there are some technologies in the world that are more like deep tech, if I would say, that requires a lot of investment in RND before.

But there's also that is like lower. So you know that there is a balance to find between the two, I would say, aspects of an early stage startup.

That is fascinating. I'm going to have to go back to larger scale. I'm sorry for just going back and forth between startups and larger companies.

No worries.

But it's like your background is just fascinating. And I want to double click on a million different topics. But something I'm interested in is software teams performance. As you start, you're a tiny group, so you just get a pulse of how everything is going. But as you grow, how do you think about measuring your software team's performance? Is that something you do?

Yeah. It was a very big question when I was leading larger organization. Because there's a lot of metrics that you can watch. I would stop from the lower level.

There, you got your infrastructure, you need to make sure that the response time is great, you need to make sure that all your load balances are optimized, you've got enough servers and so on, and you anticipate your workload.

Then on the higher level, you need to make sure that your application can be iteratively upgraded with new features. So you need to put...Making sure that your HR and organization is working very well. So from coding, to deploying onto your staging environment, and making sure that your integration from end to end is actually working.

And this is very, very challenging because I got, very often, this story that, "Yes, it works on my computer." At some point, it's not your computer that matters. It needs to work on production at the end of the day. So I know that one of my main challenges back in the time was making sure that the integration works from the end to end.

And once that everything is delivered onto the staging, you need to make sure to your QA team got everything to test everything and push into production. And of course, then, the work stuff comes with the velocity of your teams. So monitoring velocity is really something that is subjective and very hard to monitor. Because in the past I had like eight or nine squads. And so they are all independent teams working on their scope.

And definitely, it was very complex to find a metric that could work for everyone, for each team. Because they are not working on the same scope. It's not the same complexity. Theirs some sort of is more front end or also that is more back end. There are some of the team working on data, or the team working with more stakeholders, with more humans. So velocity was definitely a challenge.

So in the end, the conclusion is that there's something that is really comparable that you can really do some benchmark, is basically measuring your velocity compared to the past performances for one same team.

And I think that this is the best KPI that we end up with my different teams, to really making sure that the team is really delivering what the product team asked for.

Totally. So this idea of being better than you were yesterday.

Right.

I'm curious, how do you think about quantifying the impact of software engineering teams? But not all between the technical organization, but rather at a company level. Do you think about metrics like a profit margin, ROI, stakeholder returns? Is that something that you think should occupy a CTO's mind?

It's something that I had in mind, but it's very, very complex to challenge yourself around this kind of KPI that are really more, I would say, more finance KPIs. But I know it's definitely super mandatory to go through. So actually, what I was doing, it's very simple, looking around on the web for benchmarking data.

Because it's kind of very common that large tech companies, they are spending around 30 percent of their revenue into RND. So of course, I'm looking at the cost like this to compare it with the revenue. But this is really a macro metric.

But at the end of the day, in terms of micro piloting, I think it's really hard to, when you are working on a project, to really see what impact it will have on the finance. Because sometimes what you are doing got a huge benefit for the customers. But maybe the next time, you won't relieve anything for your customers, but you will improve maybe the internal efficiencies of your sales teams or your customer success. And of course, you know that it would improve the performance of the company.

But it’s really hard to quantify.

That’s why I think that the best way to do it is really to be HR. So by being HR, you know that at every iteration, you are solving something critical for your company. And you know that you are producing value, it’s rough. I know it’s hard to quantify, but you are producing value.

This internal tension of the short-term requirements of a company, of a start up, of revenue, of growing, and then the longer-term things, like making sure that you’re thinking about the right technologies and innovation and managing tech. How do you balance those two things that are usually conflicting?

I think it’s not only a question of tech. In any decision making that you have to do, there as well like pros and cons for the short-term solutions, and pros and cons for the longer-term solution. And one of the biggest and main challenges that any tech organization have is really about the technical debt.

But for me now, I wouldn’ call it “technical debt” because actually, if you go for, let’s say like for the short-term solution on purpose, you know why you do it. For example, the context might be, “Okay, I’m testing something new that we don’t know if there will be any customers. So we need to try it out.”

So it doesn’t make any sense to start building the whole factory and making sure that it will work in 10 years. You’re making the decision to go for lower scale, for example, solution. As you did it on propose, it started, you’re okay. And once you grow, let’s say that there’s detraction over this part of the product, okay, then you reinvest, I don’t know, like one month or two to make sure that your solution will grow and will be suitable for the long term.

And this goes back to this idea of...And maybe you’re responding to your question, which you asked yourself last night of technology as that means to an end.

Yes.

So what writer or book has got the greatest implants on your career, and why?

So I’m reading a lot of books. I don’t know if you expect a tech book, but actually, the book that really did a huge impact on my career was Radical Candor by Kim Scott. It’s really a book about, yes, how you manage ICs, how you coach them to grow, and what is the manager path and so on. So it’s really for me, the best management book that I found recently so far.

Interesting. Interesting. That is a frequent answer when I ask people what are the most impactful books? So i’s absolutely a fascinating read. I think that is the perfect way to end this. We just jammed for the past 40 minutes. I really, really appreciate your time, Ben. It was fascinating to talk about this breadth of experience from IC all the way to CTO of a large company. So I appreciate your time.

Thank you, Gonz.

EPISODE SIX

Chico Charlesworth

About: Chico Charlesworth is a tech entrepreneur and full stack developer based in Dublin, Ireland. He is the founder of Code&Robots, an open source mobile app robotics kit, and the CTO of Bridge, a platform that helps people all around the world own, build and help their personal networks. Before this, Chico was the CTO at Usegate, and a Senior NodeJS developer at nearForm, the global software consultancy firm.

CODE & ROBOTS

To listen to the episode, **click here**.

Gonz: Chico, welcome to the podcast. It's a pleasure to have you here. How are you doing?

Chico: Good. Thank you, Gonz. Really happy to be here and excited.

Absolutely, absolutely. I'm excited as well. So just to get warmed up. Let's start with what's the two minute version of Chico. Let's give the audience some context.

Okay. Well, I'm 42. I live in Dublin, Ireland. I'm English, but I grew up in Portugal. I have two kids, one dog, and since the age eight, I was coding on my Spectrum ZX and where I hacked the game Snake to jump levels and since then that's what I've been doing, playing with computers and trying to now currently work in cool startups.

Walk me through what you've been doing for the past few years.

So from the beginning of my engineering career, I started at big companies, Software AG, IBM, BBC, and since then I felt this need to go further down the path of finding the right group of people to work with. Normally, that's a smaller group, and that's where startups felt more like the right environment for me and working with startups and getting to a place where I can hopefully become part of a successful startup and a few start and stop and eventually getting to working here at Bridge.

So you are currently the CTO at Bridge. So what I'm very curious is, you worked at larger companies then went into startups, now the CTO at Bridge. I'm curious, how do you see the role of the CTO at small startups and then how else do you think it changes over time as a company grows?

Great question. So I don't know if I got the perfect answer, but from my perspective or my experience for the startups I've worked with, I've always taken a lead role in terms of the technical side. But generally, that would be more on the engineering side of things on building the solution and getting the right technology and delivering on that for the end-user. And more and more so I've become interested in going the next step. And for me, for the CTO, there's a balance really, especially for a first-time CTO like myself and where we're at the moment at the startup I'm working with it's very much a balance of leadership, management and still doing engineering when needed.

So it's less about when I'd like to code which is quite often, but unfortunately, I don't always get the opportunity it's more about building the right team and building the right structure from engineering perspective, but also playing nice with other teams like product teams, and QA teams and design teams and making that work. We're a fully remote team so that extra challenge as well of setting up the right environment where everyone is happy, be productive.

So for a CTO like myself, I still got a lot to learn, but from the experience I've had in the last 18 months to two years, it's really been a roller coaster of just trying to build a team and learn how to get the best of everyone, but also create that environment where everyone contributes and helps each other out. Because as a startup, you generally have to wear many hats and that includes the CTO obviously.

I'm not sure if it's a perfect answer, but it's a great answer. So thank you for that. We're going to double click on a bunch of those topics in a second, but I'm going to spend a few minutes on something that's very interesting and very hard to do, which is context switching from coding to building a team to managing the team to talking with different stakeholders. How do you manage that context switching in your day to day life? It's often a challenge with leadership positions at early-stage startups where you still have to do both, right? Be a leader, but also be an IC.

Yeah. I think it really depends what works for you. My experience is in contact switching a lot, to be fair, learning new languages, learning new technologies even when you're coding, that you're bug-fixing, or you're implementing something new, or you're trying to figure out how to increase performance. So even within that context, there's a lot going on. And I generally thrive in that environment actually. It can lead to obviously burnout now and lack of focus and productivity, so you do have to be careful of that. But generally, I quite like it.

It keeps things fresh.

The one thing that you do need to do is try and focus and learn how to execute with the right speed, almost. That's not always easy. For me, what works really well for me is not to have too much in my head any given time, because that way I just lose track. So what I end up doing is putting everything down into lists. Before it could be any tool but at the moment I use Notion and on my personal list, or if it's around engineering, around the team then we have issue-tracking systems and trackers and all that stuff to keep tab of all that stuff and monitoring and whatever else.

So as long as it's outside of my head, I can deal with it. But it does lead to me spending evening sometimes when I need a bit of quiet time to focus on one specific thing, or potentially we can, especially if I got something I want to work on that I just haven't had time to even look at over the week, and then I might spend a bit of time just trying to look into that. Might be my personal interest or something that I want to get into, and the same applies to the rest of the team, I think.

But what's interesting is you do have some people that really struggle with contact switching, and that's absolutely fine. You just have to be aware of that and try and not overwhelm them with too much stuff and just let them do what they're good at as well. So it just really depends on the personality of the person.

Yeah, absolutely, absolutely. It's about identifying how the people in your team thrive and helping create that environment, which are your words. I'm curious, how do you structure your days to manage these multiple responsibilities? What's your routine like?

Yeah. I don't normally have a day-to-day stuff. I normally reserve Fridays to have more if possible a chilled-out day where I can do some engineering. I spread my workload across the week or even sometimes a couple of weeks where certain meetings are alternate between weeks. I just normally find the right pockets of time. So I would have engineering syncs generally during the morning and then, if needed, help with deployments and stuff like that, and code reviews. And once that's done, I would then generally sync with the design team, the product team, and generally any firefighting.

Normally, if there's any of that, then that normally comes first, which was the case today where everything else goes out the window, and you just have to figure that out first. So yeah, being flexible in nature, but trying to really focus on what's in the pipeline to be done and what's going to happen next as well like what's ahead of time, what needs to be done, and trying to fit that in. And again, listening to the team if there's any pains or blockers and stuff like that. I'm obviously very conscious of that stuff and try and have a quick turnaround if possible.

So I don't know if that really answers your question. It sounds probably quite chaotic, and maybe it is, but in my head, it's so much streamlined, it can always be better but...

Yeah, absolutely. You mentioned this word, which is "firefighter." As leaders, we all spend a bunch of time firefighting. So I'm curious, how do you balance that? The immediate problem solving, the short-term emergencies versus the long-term work?

Yeah. So it’s a great question. It depends on severity and prioritising that and planning ahead I suppose if you know you got a deadline you need to address, and then you try and allocate your time and probably the rest of the team, their time to focus on that area. So what we end up doing quite often is to put a goal in front of us, a deadline, even if it’s an artificial pressure. But that will get the best out of us and give us the right focus, which I’m not sure is firefighting but in terms of that.

Now, if there’s a deployment issue or a customer issue or anything like that. Then again, we would prioritise that and get the right person to look at that. That’s not always me. I’d always try and delegate where possible as well but at the end of the day and there’s my responsibility to make sure those issues get resolved, at least from engineering perspective. So luckily we have a great team where that responsibility is shared and especially around the senior engineers, everyone has the freedom to help on those issues.

I love that phrase. The freedom to help. I might steal it. It’s pretty cool. That’s been super insightful. I want to go back to your first response and I want to double click on this idea of building a team, especially as you said you’re a remote company like most companies are right now. So the question is, how do you think about recruiting and retaining the right talents? Essentially Covid made the talent markets remote, right? You’re not competing with people in Dublin. You’re competing with companies like Spotify, who just said they’re going to pay San Francisco salaries to everyone in the world, including people in Madrid. So how do you think about recruiting global talent in a world like that?

Yeah. Recruitment in general and hiring is something actually quite new to me since I’ve taken this role and we’re really proud of the team that we’ve assembled and actually feel really lucky to have an awesome fully remote team, work wonderfully well together. What we try to do is build a remote team that fits our company values and which are around communication, going above and beyond. Attention to detail is a really important one which is actually a hard one to put your finger on but everyone knows what lack of attention to detail means, it’s the inverse, right?

And also, we adopt this philosophy at Tacos, which is an insider term where we’re showing appreciation to the team as often as possible really. And around recruitment, there really is no magic formula. We try and hire internally first, asking the team, hey, let us know if you know someone amazing that’s available, right, that you think would fit well with us and that’s not always the case.

So generally we then source candidates through platforms like Upwork or something like that and then we’re just experimenting on how best to source those candidates, how to hire the right people, making sure that they’re the right fit for us but even more importantly, make sure we’re the right fit for them because we want them to stick around. We’re really keen on building the core team before we expand and that’s really key for us to achieve.

And I think, by and large, we’re doing that, obviously, we can improve and it doesn’t stop there. Once we’ve hired someone, we try and make the onboarding as best as we can and to even improve that process. Every time anyone comes into the team, we ask for their feedback, hey, how can we improve this and so forth? So that’s really what we’re striving to do.

Diversity is another big one which I think everyone’s struggling with and we’re also trying to find ways of achieving those goals of making sure that the team can be as diverse as possible. Right now we’re across three continents so we want to get to five and also have a better gender balance and stuff like that which we generally believe will give us a big advantage in terms of ideas and execution.

Yeah, absolutely. Absolutely. That’s what gets you diversity of thought and problem-solving. So 100 percent. You mentioned onboarding. So any common mistakes you recommend new CTOs to avoid when thinking about onboarding new team members to a team?

Yeah. So I think the most important thing is to be welcoming. So we use Slack and we welcome everyone on board. We give them all the access they need and with instructions. So we have what we call a boot camp doc that everyone that joins, at least through engineering, goes through that document and we try and have all the documentation they need to get started. More importantly, we partner them up with a buddy, and that buddy will make sure that they don't get stuck. If they got any questions we get quick answers to them.

This is particularly important with the different time zones, we're fully remote, so it can feel a bit anxious to join a new team and especially if you get stuck and asking questions. So it's really important to try and make that process as smooth as possible and that goes from senior developers to junior developers, it's all the same. When you join a team, you got these expectations so it's really important to try and meet those expectations and make that as seamless as possible.

I appreciate that as well because I've done that where I joined a company and onboarding hasn't been that optimal and you have to figure out stuff and nobody wants to be unproductive for like a week or longer. So we try and get them in the code, developing, fixing bugs, writing code that gets shipped as early as possible, that stuff.

Absolutely. Absolutely. The reason I ask is because I think you said that you've experienced some non-ideal onboarding experiences and to me, onboarding is probably the most important inflection point you can provide someone in the first few months.

Yes, I agree. I've had friends who joined a team and within a week they've left because there was no effort in bringing them into the team and embedding them in the right culture of that team and I think that's super important. It's not always about the work initially. It's about getting to understand what their needs are, how they expected to communicate with the team is really important as well. And also as they work, the way we do stuff at Bridge won't always tally what they used to. There'll be new tools, all that stuff and that new information.

So it's really important for us to bring them along into the same rhythm that the rest of the team has as quickly as possible.

I wanted to go back to this idea of creating I think you said the ideal environment. So what are some of the culture, principles or operating methodologies that you apply to create that environment?

So this ties down very much to the company values that we have and these company values weren't something that we had from day one. It's something that we've learned or even noticed in the team and said, yes, that's what we're looking for and that's what we want to try and adopt. Like I said, the environment that we want is anyone's comfortable in talking to anyone. Importantly as well to communicate in the right way, ask the right questions, ask the right person and because we're remote, we want to do as much asynchronous as possible, so what tools to use so we use Loom extensively to help us with that.

And in terms of making sure or enabling other teammates to have your back and vice versa and that means that you can build that trust, and that trust really helps them to have everyone on board. There's always going to be conflict but as long as you have that in the team you know you're generally going to arrive at the right decision and you're not going to get unbalanced in terms of what work is getting done and favouritism or anything like that, right? It's a team effort. So that's really important.

And going back to attention to detail, I think it's about really being aware with your work, how that impacts other people. So from engineering perspective, we have to take what products and design have been working on and then implement that and it's really important that what's given to us has got high-quality definition of what&'s required from a visual standpoint and also from a logical standpoint so that we can do our best job.

Then once we've implemented our work, we can then communicate that back, but also be able to even within the engineering team to have code reviews to make sure that we're writing code, but also communicating that.

So we do code walk-throughs via Loom and also do working demos and add documentation because even people that aren't on the team, people are going to join us at some future date. It's really important for them to understand what's going on. So that attention to detail really makes a huge difference and it's probably underrated somewhat.

So the last one really is the appreciation. So we really value when people put in that extra mile, do amazing work, find out why this thing isn't working, have a super-quick turnaround on bug fixing, whatever it might be. The way we do that we use a Slack tool called HeyTaco, which sounds a bit cheesy, but it works actually extremely well just to... It's like giving someone a high five and everyone loves high fives and on Bridge, that's actually Tacos.

That's actually pretty cool. Yes, I absolutely think that attention to detail is underappreciated. I have no idea why, though. I'm curious, how do you think about measuring team performance? Is that something you do at this stage, or is that something you're so close to your team you just feel it? If that makes sense?

No, I don't think you can feel it. I think it's definitely something worth measuring. It is something we're measuring on an ad hoc basis but it's something that we want to improve on in terms of automating it. And it's very much to understand velocity in terms of how well we're doing at the team level, not so much at the individual level. What's really important with that data and this is why we do it on ad hoc basis is to identify bottlenecks.

I recently learned about a technique called value stream mapping, which does exactly that and it doesn't have to be engineering. Any process. It tries to see how long things are taking in between each stage. And if you think of how engineering work, generally, you will get a task to work on. You'll get assigned that task. You will then look at that task. You will estimate that task. You then work on that task and see how long that has spent. Then that doesn't stop there. It then gets moved to being code reviewed, testing, being shipped, and all those.

So it's got like a issue life cycle that it goes through and we have all different types of issues, bugs, tasks, performance issues, DevOps, whatever it might be. So it's really important to understand where are the bottlenecks. So as a given example, what we were doing was once an issue was completed from engineer, that would then go into testing, and once testing was finished, then that would go into code review, very sequential. And initially, that made sense but then once I applied this value stream mapping, I could see that there was a bottleneck there in terms of it was taking up to a day, sometimes two days for that particular issue to get shipped to our staging environment.

And what was happening because we are fully remote and there's time zone differences, that passing that issue to the next team to be tested and then passing it back to engineering team to be reviewed was taking too long. So it wasn't the actual testing or the reviewing, it was actually the wait time. So what we did is we actually combined both of those processes to happen in parallel and that saved us automatically a lot of time and ships were just getting done faster. And you could feel. So this is as well like something that's not necessarily obvious until you look at the bigger picture and try and map things out and that worked extremely well for us.

What's keeping you up at night over the next 12 months or so?

I generally sleep pretty well. The things that normally keep me up at night is if I haven't figured something out, and then my brain is spinning. But generally speaking, I'm not too bad at that. I do have obviously... There's always a mix of emotions of how things are going. Some days you're struggling, some days you're flying.

I'm just generally really excited about where we're going and I want to with the team achieve that. I don't really suffer from insomnia, so so far so good. Maybe that will change.

I'm obviously very, very aware that as we scale I think there'll be definitely different challenges and at that point maybe a few more sleepless nights, but so far so good.

What writer or book has got the greatest influence on your career so far? And why, of course?

That's a great question. I read a lot, but generally, it's around articles and I deep dive into something and then come out of it. So I don't really have a book in mind that has changed much. There's those obvious books around lean startup and all that stuff but I don't generally follow a particular book or even author. I try and get a rounded view, and then make my own opinion of what I think makes sense. So I don't really have anything in particular in mind, and I generally change my mind about how I approach things as well and I'm generally open-minded to changing my mind if that makes sense.

So, yeah, I don't know if that answers the question particularly well.

It doesn't matter. It's not important because it opened up something that's more interesting to me, which is this muscle of being open to changing in your mind, as Marc Andreesen would say strong opinions, loosely held. How do you think about that? Why do you think that's important? How have you trained that muscle? Because it's not easy.

Yeah, I don't know. Myself I'm not really ego-driven in a way. I don't need to have the best opinion in the room. I do have an opinion and a strong opinion normally, but generally, I'm more open to collaboration and I think that's what a lot of smart people talk a lot about and rightly so. And I think that's probably where our educational system should start going towards. We're always going to arrive at better solutions by working together from what I think anyway and what I see, the end result.

So for me, opinions, they shouldn't be the end goal. I think it should really be trying to find a middle ground or the optimum solution and the right decision and doing decision making in the optimal way as well is more important to me than having said the right opinion or being proved right or proved wrong. It's just the way my mind works, I guess, or my personality.

That's the perfect note to end on. Chico, thank you so much for your time. This has been a fascinating conversation.

Thanks so much, Gonz. Really enjoyed it.

EPISODE SEVEN

Håkan Holmberg

About: Håkan Holmberg is the founder and CTO at Vinter.co, an ESMA-regulated crypto index provider based in Stockholm. Before joining the team at Vinter, Hakan worked at the Swedish Government, helping the National Register system develop processes for managing confidential data of Swedish Citizens. Hakan holds a bachelor's degree in mathematical statistics, a master's degree in mathematics and spent several years researching blockchain technology at the Stockholm School of Economics.

vinter

To listen to the episode, **click here**.

Gonz: Hakan, welcome to the podcast. It's a pleasure to have you here.

Hakan: Thank you. Likewise.

So let's get started. What's the two-minute version of you? I have to ask, you went from academia to government, and then a CTO of a crypto company. So what's the two-minute version of you?

So I started to study pure mathematics, stuff that is very hard to apply in the real world, but a lot of fun to do. Went over, studied statistics and probability theory, got a job as a quantitative psychologist, in fact. So I did quantitative modeling for the psychology department at the University of Uppsala. And then I got a job as a statistician within the government register system. The Swedish government has huge databases on all kinds of data on their citizens, and then they do regular analysis of that.

And then I started to slowly go into data engineering instead. So I became responsible for the platform that was in charge of standardizing the index and indicator that the government uses in evaluating different health care welfare systems.

So that's what I was doing. But at the same time, I did research at Stockholm School of Economics when it came to blockchain technology. And I started to realize that this would be a force and a very important system in how we do business in the future. So gradually, I started to talk to friends of mine, for example, Jacob, the CEO of the company. We studied math together, and we decided to start a company, and it became an index provider.

So we developed pricing algorithms and performing strategies for the cryptocurrency. And that is where I am at the moment.

That is absolutely fascinating. And before we dive into what you're currently doing and the role of the CTO in a blockchain technology company, I love to know, what's the difference between working at the government and working at the private sector?

Huge, especially if you work in a startup. The government has, well, hundreds of employees, thousands of policies, so you're really a part of a large organization. I was technically, primarily only in charge of driving the algorithms, the functions that calculated the specific indicators. The databases and all of that was, of course, taken care of by its own separate department or unit.

While when you go over to a startup, I mean, you start out as the sole tech guy in charge of developing everything that has to be developed. So yeah, you learn a lot. And it's a lot of fun. A lot of work, but a lot of fun. And your job is to implement a lot of really new technology. Every technology that you want to implement, you technically can.

So that is a lot of them, because you can't do that in large, older organizations. I mean, there's a lot of things you just going to have to work with.

So you went from the government, hundreds of employees, thousands of policies, to co-founding a startup. Huge, huge change. And as co-founder, you came in as the first technical employee and CTO. Talk to me about that, that transition, and how was your role and how do you think that will change over time as the team grows?

I mean, as for now, we are very much in the startup phase. We're a very small team. So everyone talks to everyone. In one way, that is easy to solve because everyone gets the holistic picture, you can all jump in...You say to the person, I mean, if you want to call the CEO on Sunday, I mean, you can do it.

That I don’t know, but at least you can call at later time during the weekdays.

And that’s, of course, a lot of fun. And people not to refuse...Important part of the project, the team. And I think that one of the most important part to move forward is to maintain people’s understanding that they’re a part of something bigger, and that they can maintain a holistic understanding of what they’re doing.

But except from that, I mean, it’s everything, technically. You’re going to have to implement all the important infrastructure that is necessary when you start to get a team. Yeah, that’s going to be big. I mean, right now, it’s been very much, I sit, I decide how the infrastructure should look and I implement it.

And then now, we are three in the development team right now. We have data scientists and data analysts that works together because we’ve decided earlier on to implement our platform in Python. And one of the main reasons for that was because it’s a language that both data scientists and developers can share.

And it means that we have quite a good of information, so to say, transfer between the different teams. But yeah, definitely the infrastructure or the organization has to be more thought through and developed. But I do think that one of the benefit of working in a large organization that is very focused on policies and infrastructure that I came from, I do have quite some ideas of how you can do it. But yeah, that’s it.

I think it’s going to be something you just continue. And one thing that I really look at now is of course, we look at trying to hire senior developers that also have a lot of that experience so that we get that experience with that employee probably solve some of the issues.

That makes perfect sense. So let’s double click on that a bit, sort of hiring. I’m very curious, how do you think about recruiting and retaining talent in a whirl or sort of COVID essentially made talent markets global? So that’s the first complexity. And the second complexity for you, for you and your team, would be the industry that you play with. Do you think about hiring people with specialized knowledge? Or do you just train them as they come in? So walk me through some of your hiring frameworks for recruiting and retaining talent.

The question on the global job market is of course, that we went global. So when the job market went global, our company went global. All our meetings are online with clients and with our colleagues. Today, almost everyone in the team works remote. We have offices in Stockholm, but our colleagues are all around the world. And when we’re hiring, we’re also hiring all around the world.

So now, I’m talking to a guy in Ukraine, a guy in Malaysia, a guy in India. So we’re really trying to get that working. So one of the main things that I’m looking at now is to try to really get a good digital environment working for the team. So you can come as close to a physical experience. Because I do believe that the physical experience has some advantages.

And so the important part for me now is try to create a digital environment that has almost those features of natural interaction between people, and just the simple fact it’s high, really high, audio quality and video quality is super important.

So that is some part we work with. When we work with looking at the competencies, I mean, in general of course, it’s of course very important that when you have an interview with someone that you feel that I would be able to work a lot with this person because we work a lot.

And I mean, if you go into a startup, you have to be truly dedicated to the task. It will not be a 9:00 to 5:00 job. That’s just the case.

So you're going to have to be motivated by both your mission, what you'e trying to offer to the world, but also by simply learning the technologies, becoming really good at what you do. So to see it also as an education. So that is of course, what you look at. Someone that sees the job as more than a job, as a part of growing and developing and so on.

Now we're still early, so we do also look at specific experiences within Python and Jango framework. And Redis and Postgres, and BlueOne, the technology that we use. Technically, that is yes, because we really need it immediately. We don't have the resources to train someone at the moment. I think that in the future, that will become less important. And character will become more important than a specific experience. And I have a strong tendency to go on character even today, even if I know that what I do really need is competence.

And then how do we find the talent? Everything from job ads to reaching out on LinkedIn to people that seem to work on competitors, going out on Upwork, like throwing out a couple of parallel tasks or see who's doing the best job. And then it's like, do you want to be hired? Something like that. I mean, we're trying a lot of things.

And do you look for people who have industry and crypto experience, or who are at least captivated by the topic?

Of course. I mean, you always get excited if you meet someone that is excited about what you're doing, and that has already figured it out. However, I have had interviews with a developer, who said directly to me, "I don't believe in crypto. I think this is not going to work out. I applied to your job because it's cool. It's data distribution, large amounts of data. It has to...You have a high speed, extreme requirements and accuracy and so forth. So I think that the technology and the techs that it uses is exciting, so I applied for that. Crypto is not my thing."

And I haven't seen that as a disadvantage at all. I get intrigued by that. However, I do tell them that they will end up in some fierce debates within the company. But as long as they can accept that, I'll go for it. So it's both ways, to some extent. If you're really engaged and have great arguments for your position, I like it. And then I'm less interested in what your position is.

That's real, real cool. And I didn't expect that. But essentially, if someone is interested in the technical challenge and is able to have reasonable debates about things, that's some of the core traits that you're looking for in hires.

Yeah. And I mean, I know that I got to win the debate. So I'm not concerned about that. It's just a matter of time and they will then invest in bitcoin. So that's not the issue.

One thing you said is that you have infrastructure or an operating cadence that tries to mimic a lot of the real world interactions, because you think that's...even though you went global, that is valuable. So what sort of practices or what's your operating cadence to make sure that your team is high performing? Some of the tools that you use, maybe.

I mean, as I also said in that statement that I'm very much still looking for those tools. But, I mean, as for now, I mean, we use Google. We have tried Discord. I do like Discord. I've had some issues with the quality there, but I know you can buy better quality. So I'm thinking about doing that.

We have tried Telegram, Signal, Skype, really a lot of the different tools. I know that there are better tools out there that create more of a virtual office environment, and I want to try them. I hadn't had the chance yet, so if anyone has a suggestion that they know is just awesome, send it to hakan@vinter.co. And I will try it out. But that's a challenge, technically. We haven't solved that.

Absolutely. That makes perfect sense. And how do you deal with time zones? You mentioned you were talking to candidates in India, for instance.

I think this will be a bigger problem in the future. As of now, I think that I do like the fact that we have very separate time zones because it becomes easier to have 24 monitoring [inaudible 00:15:55] person on all our servers. So I really like that. I mean, with developers, we really try to have as few meetings as possible. That's also a challenge, in order to get that in depth concentration time that you need in order to solve a problem in a good way.

So I only tried to have one hour meetings where we go through exactly what has happened during the day and what will happen the next one or two days. We also use Asana at the moment as a task manager system. So I really try to work with that one in a structured way. I also constantly are asking for structures. So it's a very important topic when I discuss with [inaudible 00:16:40] how do we improve the structure of our communication? If anything is unclear, tell me that. And also tell me if it's some kind of a systematic problem here that we can solve.

Or is it just the one time?

But when it comes to know, and honestly, I do work quite many, a lot of hours. So I have a lot of hours spent. But sometimes, you have to take one meeting with one developer a little bit early and the other one a little bit later. But yeah, definitely it could become a problem in the future. But as of now, we have, in fact, been able to work around that.

Totally. I think that is a problem that most companies are going through at the moment. I'm curious, at this stage, do you feel the need to measure your team's performance in any way? Or you're such a tiny group right now that it's something you can just get a pulse on every day?

As of now, we get a good pulse. I mean, I can track the task assignment. But I mean, I have such a great relationship with the ones I work with, so I mean, I work quite a lot, and they work more than me. So I'm not in the position of challenging them when it comes to productivity, for sure.

But in the future, it's going to be maybe more of a challenge. But I truly believe that you should hire people that are extremely motivated, extremely hardworking. And so the responsibility of motivation in the end, ends up on you, in the sense that they should be of the caliber that they shouldn't be too hard to motivate. So if you do not succeed with that, I mean, you have to go get what do you do.

It all goes back to who you hire.

Yeah.

I'm curious, so startup, small team, looking to grow. How do you balance the short-term growth and revenue requirements that come with being a start up? Again, sort of the long-term tech development or in a way, in a space like crypto, which is like there's a ton to be explored and to be done? How do you balance those conflicting tensions?

The straight answer at the moment is we don't. I think that we're extremely client driven. So what happens is that we start a discussion. I mean, our sales process are very long. So to that extent, it is a little bit easier to be very client driven, but still has some kind of more of a long-term perspective.

So our sales process is quite complex, B2B process. So when they start out, they normally start with backend testing of different ITs. And then in collaboration with the potential client or the assigned client, you set up some kind of product that they want. And then slowly, you can see what kind of technical requirements you would need for that.

But there are, of course, projects that we have and that we really believe would be great to implement, and that simply are lagging behind now because we have a lot of client requests, which is a good thing, to some extent. But yeah, in the future, I think that, of course, we're going to have to sit down and have more of a proper discussion about how the budgets are being spent and how much we should really focus on long-term projects. But as of now, it's very much client driven.

I'm curious, so you said you have this client-driven environment, which at that stage is super beneficial, particularly with the longer sales process. Is your team, and you the CTO, the one who prioritizes that? Or do you work with a product team for that?

I'm very lucky because our CEO that is in charge of the sales process is a statistician and started mathematics with me. So he likes tech a lot. And he also, even if he's data scientist, he knows how to program Python and all. So he has quite a good understanding for what the tech needs and require. And that helps us to the communication.

So I would say that when we get requests, the prioritization becomes automatic, simply because we have both a good understanding of what takes time, what needs to be done, what kind of resources do you have to put it on the different tasks?

So I think that that is an absolute strength of the company that you have not only a sales guy that knows tech, but also he's interested in it and likes it, and has then a passion for the product, how the product is being developed. So I'm lucky there.

Luck usually plays a role.

Yeah.

I'm curious, what's keeping you up at night over the next 12 months or so?

Everything that has to do with the core quality of the product: accuracy in the sense of will people be able to manipulate the values, are our validation processes that we have in place enough? Of course, uptime state access and also constantly ensuring everything because we are a regulated entity. So to some extent, our development process started a little bit differently than others, in the sense that we started to write a long application to the financial, the European security and market authorities, specifying exactly all the requirements that are needed from a regulated benchmark provider, which is a bunch.

So technically, I have to deliver an entire infrastructure with all the requirements and security rules and disaster recovery plans and all of that have to be in place. So we started with all that, which is most often maybe not that common for startups to begin with that end. But that also, of course, makes me constantly aware of what I need to fulfill in order to be a regulated entity. And that also keeps me up at night.

But honestly, I have some great guys in different time zones. They do a great job of monitoring, so I do sleep pretty well.

That's the dream. Well, you have to deal with regulation, but you have a great team to support you.

Yeah.

What sort of writer or book has got the greatest influence on your career, and why?

Oh, yeah. I mean, first of all, if you've dedicated your life to the promotion of cryptocurrencies, I would say it's maybe not a book, but a paper. And that is Satoshi Nakamoto of course, who had a huge impact on me, given that it, to some extent, made me decide on a career path.

In the day to day work, in fact, before I started mathematics, I studied physics. But then before that, I studied philosophy. And I studied what's called the "Swedish Theoretic of Philosophy." So it's a main focus on logic and metaphysics. And those courses, and especially a book by Susan Haack, a professor at that time at the University of Florida, on the philosophy of logics has an immense impact because it taught me that you can be formally logic of all about more or less everything you can think, in logic in your everyday life, you can structure things together with metaphysics where you set up all your assumption on premises. You really can get a good structure on reasoning in most questions.

And that does help me. Because when I studied those courses, I realized that quite often, I wasn't that logical. So it's very boring and dry, and I don't even know if I would recommend it to anyone. But to me, it had a [inaudible 00:25:51]

Let's start with the bitcoin paper then.

Yeah. I think so.

That is a perfect note to end on. Thank you so much for your time, Hakan. It's been a pleasure.

Yeah, you're welcome. Have a great day now.

EPISODE EIGTH

Arseniy Vershinin

About: Arseniy Vershinin is the co-founder and CTO of Personio, Europe's leading HR platform for small and medium-sized businesses. Before starting Personio, Arseniy co-founded nuclinio, the team collaboration tool, and was an iOS developer for Freeletics, the mobile in-demand fitness startup.

Personio

To listen to the episode, **click here**.

Gonz: Arseniy, welcome to the podcast. I'm super excited about this conversation. I'm a big fan of Personio, so here we are.

Arseniy: Awesome and super happy to be here. Yeah. Super happy to talk about some of the challenges and things that we experienced at Personio. Hopefully useful for your audience.

Yeah, absolutely, absolutely. Looking forward to it. But before getting into Personio, I'd like to get into you. What's the two-minute version of Arseniy? Let's start with that, set some context for the audience.

Sure. Happy to do so. Yeah. As you pronounced my name correctly, my name is Arseniy. I'm Chief Technology Officer and Co-founder. We are building a holistic HR platform for small, medium-size companies in Europe with the goal to enable better organizations through automation, making sure that the HR people are focusing on us, people. Super short about my background, so I come from a small Siberian city deep in Russia, and that's where I developed a passion for programming for a computer.

So I studied them afterward in Moscow and got in touch with the local startup scene, and worked with some startups there. There was a christening into the startup road for me. It was a great experience. I learned some great people, but I wanted to actually work in a more international and diverse environment.

So that's where the road took me to Germany to continue my studies. So I got into the Technical University of Munich here, and also this program called CDTM, or a Center for Digital Technology and Management. A lot of great people I got to know from that program, including my co-founders, Hanno, and Roman.

And in 2015, when we finished with the program, we decided to unite forces together to build Personio, and since then it's been an amazing ride. A crazy fast-scaling journey, as fast as it gets, and I'm happy to be contributing to the technology vision to the architecture and working with our awesome teams on making this great company.

I bet it was a crazy ride from starting in 2015 to raising a big [inaudible 00:03:37] earlier this year in January. What I'm very curious about is how do you think your role as a CTO started and how it changed over time? Right now you are about 900 people on the team if I'm correct?

Close. Close, yeah. We're close to that. It's a great question. I think I experienced probably a pretty wild ride in terms of starting as a founding member and coding on the first prototypes, lines of code down to now, managing together with our VP engineering, Sebastian, and a pretty sizable team. I think overall, as you very correctly, underlining your question, the role of the CTO throughout those periods of time changes radically.

But I think the common denominator to this role is that I think a CTO should really focus on enabling the organization, whatever this organisation is, in whichever face of existence it is, to resolve the most important and strategic business problems with the help of technology, either directly or indirectly.

So if we talk about the initial phases of the company founding, this happens directly. So the most important task that a CTO together with a founding team obviously has, is to create MVP. So to basically prove that the model that the idea, hypothesis, actually has traction on the market.

And then I think the role shifts into making sure that you have a well-running team to make sure that this product that hopefully found initial traction on the market actually is able to scale, reliably, securely, performantly, addressing the most important customer needs. And then, at later stages of company development, I think it depends on what a particular CTO wants to focus on and what a particular set of business challenges are.

So what I, for instance right now, focus on what my mission is to establish a compelling technology vision and architecture, which enables us to develop the first-in-class product, but also to hire the best technology talent.

For such a thoughtful answer, I want to double-click on one thing you just said, which is hiring top technology talent on the market. So you have four offices right now. It started in Munich, you're opening one in Amsterdam, but now the world or the talent markets are going global, especially with COVID and all the new software that's been built around that. So what do you think about this problem? What do you think it takes to recruit and retain talents in a world where COVID made talent markets global?

I think for us, personally, as a company, global talent recruitment has always been a priority from the get-go. So that's why our talent acquisition strategy has not changed significantly under the influence of COVID as we've been recruiting everywhere from Brazil to Europe to Asia, starting from the first year of us starting to search for a team and hiring for the engineers. In fact, our first engineers who still work with us come from Vietnam, right? So it just shows how global the talent already is, right?

I think there are two particular developments, however, that I can talk about. The first one is that on the recruitment side, there are two major developments that are underlining the market. Firstly, it's both the demand from the talent for more flexible and remote-friendly working options.

And secondly, which is more particular for the pandemic period is that in certain regions it became hard to actually get talent from their countries to whatever the location that they actually prefer to go. So for instance, we experienced major problems in relocation projects from Latin America in certain cases, as embassies got closed down and so on.

But that triggered even more of the need for us to actually embrace the remote strategy. So overall, in that particular aspect, we started the pandemic by making sure that our employees are as best prepared for a long remote period as possible with grassroots efforts such as the Home Office Task Force.

So the set of employees who wanted to think about what's the best possible experience we can offer to our colleagues, whatever the equipment maybe they're missing to actually make sure that their offices, their home offices can be actual home offices rather than some corner at a lunch table.

And now actually past this pandemic is over or soon hope to be over, we're offering what we call PersonioFlex. So this is a possibility for employees up to 50 percent of the time to work remotely, not basically tied to one of the offices to a particular location, but overall globally in the world.

We recently also released an article about that so feel free to also, whoever is listening to the podcast and interested to read about the details of how we implemented that. But overall, I think that's the major development on the recruitment side and also on the internal talent side that we've undertaken.

That's super interesting how you've been able to just weather the pandemic, and now build strengths on top of that. What I'm curious about is, what are some of the core traits or personality traits that you look for in new hires? Not the hard skills, but from everything I've been able to read and answer the conversations I'e had, the Personio culture is a huge reason why people join the company.

That’s a great question and you’re absolutely right. Hard skills are definitely the baseline at which we look for during our recruitment process, and they differ from role to role. But as you mentioned, what is absolutely critical for us is the alignment with our cultural values. And here we have a written down and very well lived through what we call Personio Code.

This is the set of operating principles and core values, where operating principles for us are the behaviours we really care about in our people, in our talent. And we both foster those behaviors or the talent that is already there in the organisation and develop our talent along the lines of these operating principles, but also look for those traits in the new recruits. And we also have the core values. So those are some of the things that we really aspire to be as a company.

So talking about operating principles since that more aligns with the question that you asked, I can give you some examples. For instance, we’re looking for a high level of diligence. So overall, Hanno, our CEO likes to compare the teams that we built to a sports team. So it’s a very aligned and very highly performant set of individuals who work towards the same goal, but at the same time who have fun on their way.

And that’s for us super important character traits to test for. So again, high level of professionalism, being diligent, having a solutions-over-problems mindset. We’re still scaling extremely quickly. Even by the end of this year, we’ll grow our engineering organisation and the overall company by approximately 2x.

So we’ll double from the time of when we started the year and that means that there are a lot of things, a lot of challenges that will arise and that we will need to be resolved. And that’s why we encourage all the people to really challenge the status quo, bring in the best experiences that they learn about outside the company and really start thinking about implementing them.

There are some other traits such as communication is key, or also seeking to improve, also personally, that we look for. Again, all those things are wrapped up nicely into Personio Code and we check for them in all phases of the recruiting process.

There’s this saying that every time a company doubles in size, everything breaks, and I definitely experienced some of that. So I’m very curious, how do you plan to evolve and optimise some of these principles over the next 12 or 18 months?

Yeah. In fact, we’ve been just recently through one of those waves of optimisations that are tweaking of the Personio Code that resulted in the current version of it. So that’s why I think over the next few years, we will obviously regularly look at it, but I believe that at the moment it actually constitutes a pretty stable set of values and operating principles.

What I think will happen over the course of the next months is us firstly, using the opportunity to really meet together and really discuss and to work on the culture, also in person. So that’s why I’m super excited to again be at the time when the offices could be opened up.

In fact, we’re also planning a massive amount of get-togethers, probably closer to September with all of our teams. We will be able to fly them together and then both talk about our culture, about how we can live it together. Also, there are more things that we’re planning again on the cultural side, such as the company hackathon that is happening over the course of the next month and will take the whole week actually, which I’m also personally very excited about.

It is one of those things that fosters. Again, people getting together, applying the solutions-over-problems mindset, having fun. So hits all points on our Personio Code and our culture. And I’m just happy to leave this Personio Code in more of an in-person environment over the next month.

That’s incredible. So you’re saying you’re going to take a week or the entire team is going to take a week off to work on this hackathon together in person? How did you guys come to that decision? Because it’s really unique.

I think by now we see that there is overall. It’s a pretty common practice in the industry. And this time we’ll see how much it can happen in person. I think August is still going to be a bit of a challenging time so this one is likely still going to be in a hybrid mode. But the tradition of the company hackathons, in fact, started with a proposal of one of our employees who has seen how successful this concept can be. And she worked with us to make it a reality, making it the first hackathon, which was back then limited to only product and engineering teams and departments.

However, we saw how many awesome ideas really came out of it and a lot of them actually made it into production, into the day-to-day experiences that our customers have that we wanted to scale this idea further out. And in the next iteration, it was already... I think it was a couple of days where all of the company took time to actually collaborate and solve some of both day-to-day challenges but also things that they saw long-term customers might experience.

And by now, we scaled this idea even more. And it’s going to be this week where we all together in the new setup, also collaborate on this. And also, one more thing that I think is interesting is during even the pandemic time, so the last hackathon we had of that proportion was last August, I think. It was still very deep in the pandemic. We still decided to do it, and we did it completely remotely, which was a great task to our collaboration abilities, and I think it went extremely well.

So I’m happy that, again, applying our solutions-over-problems operating principle, we were able to overcome this challenging situation and came out with an extremely fun and productive event for everyone.

That’s incredible. I definitely wasn’t expecting that when I came into this conversation. Something I’m very interested in is team development. So I know you have about 1,500 Euro annual development budget for each employee, so I’d love to know a bit more about how you think about that, but also something that’s very unique to engineering teams as career development. So I would also love to double-click on how you think about career tracks for your different tech teams.

Yeah. And that’s a great question. Overall, on the principal level, when developing our career path or career framework, it was essential for us that the growth of expert individual contributors in this career track is considered as much as the managerial growth. I think that’s really the key because we would love to see great engineers actually do what they love, which is engineering in a lot of cases. And obviously, we also enable people, if they want to, to grow into a managerial track. But since our individual contributor track is so much developed and aligned in terms of the growth progression, that’s where a lot of engineers also prefer to stay on this expert track.

So that’s an overarching principle that we took into the career framework development. And I think by now there have been several iterations of this. In fact, we introduce more roles, more levels with various types of impact, as you classically reduce the organisation scales. And we rework it or adjust it almost every half a year to a year to reflect both the type of talent and behaviours we want to see in our teams and also to reflect on the scale and growth that we experience as an engineer and the company organisation in general.

That’s great. I often see technology companies. So this regarding this idea of... And I see being as an equal to a manager. Everyone thinks of managing as a default path for ambitious, and police who want to level up. But it’s great that you’re thinking about this equally.

It’s definitely important to contribute to this thinking pattern. We see that. I think it’s a pretty common way to think about it in the Valley or in the US startups. And that’s why we would love to see more of that. It also happens in Europe, where I think overall, from what we see from the market, this mindset is a bit missing. And I think it’s a huge mess because those extremely talented people can contribute with a major impact to the hardest problems that a company can have.

But for this, they need to be enabled to actually grow within their expertise track as much as possible.

You mentioned impact. So I’m curious, what indicators, if any of course, do you use to measure your software teams like performance?

I think there’s a variety of indicators and metrics that all start with people. So that’s why we regularly look at how, firstly, happy people are within our teams. This is normally one of the highest correlating factors to how both a software team, but in general, an organisation performs. So what we do for this is we send out regular employee pulse checks and look at the engagement score of the organisation broken down also by teams. So that’s one of the first things that we look at.

Then looking at the more classical indicators of team’s performance and also customer happiness, which I think is also another important angle to look at how well a team is performing. There are sort of different clusters. Firstly, since we use this concept of mission-based teams, that means that a team is owning a piece of the product end-to-end, and is able to both define its vision and develop it end-to-end. We’re looking at the customer happiness associated with this particular piece of product that a team is managing.

So does a metric such as UX satisfaction. So you can ask a customer, Hey, how do you find the experience with this particular feature part of the product and also reflected back to the team?

Those are a set of metrics on the team level that more correspond to the business-as-usual metrics such as performance quality, reflected in how many buckets have been reported with that particular and other more engineering metrics. But also another aspect of it which we started to get more and more into is this actual engineering performance. And this is always a very hard, over-debated question. But we believe that so far what starts to work fairly well for us is to track such metrics such as a deployment frequency, such metrics such as change of failure percentage or mean time to recover.

So this is a classical set of metrics that are talked about in this accelerated book. So it’s pretty much a table stakes book in quite some teams around how to organize, around the classical DevOps metrics and how to run the team successfully. So that’s on the team side. And it all, obviously, wraps together as the whole mission of the product and engineering department. It’s to build the best in-class product with more aggregated, more lugged, but still very true metrics such as Net Promoter Score. And the Net Promoter Score then reflects on, Hey, how’s the whole department performing against the goal of making our customers happy and productive.

So overall, those three buckets of metrics are regularly looked at. And also, we regularly develop them and challenge them.

Yeah, it’s a highly debated topic, but I think you have some pretty good systems in place for it. I’m curious, do you worry that something like deployment frequency might create the wrong incentives for two logic teams?

I think that is really a question of how do you work with those metrics? Right. Because, for instance, what definitely is not allowed to happen is where you translate deployment frequency into individual performance and then create individual performance-based metrics or conclusions out of that. I don’t think that this is something that makes successful teams and motivated teams.

As typically those metrics, they start to game how you actually do development. I think what really works well for us is looking at those metrics on the team level and other high-level aggregates.

For instance, as we are growing more than two weeks per peer in terms of the number of engineers added, it's then important to track. Hey, does your deployment frequency stay approximately proportional to the number of people that get added and onboarded, so that your organisation doesn't tip over in terms of productivity. And that's the type of things that we do or also start doing. Also, it's important to look at the trends and that's precise. Also, I think that goes in the direction of the example that I made.

What's keeping you up at night over the next 12 or so months?

Yeah. I think by now we're running a pretty tight ship, so it's nothing that particularly keeps me up at night. But I can tell you that the majority of my time right now I spent with teams, working on our replatformification and monolith duplication effort. So we're migrating to the new architecture powered by micro services. And there's a lot of technical discussions happening. We are discussing how we can tackle it effectively as an organisation. And that's where I spent, probably, the majority of the time, which is not spent on other business-as-usual topics such as interviewing or one-on-one, and so on.

How do you ensure that your team is fully involved and contributing their best to these conversations? Because what usually happens is this, some of the sessions or conversations are just top down. So how do you make sure that your team contributes?

Yeah. There's those discussions about technology, strategy, vision, and tactics that happen on a daily basis, continuously, in all kinds of levels at the company. Because it's precisely the pattern that we don't want to create where only management thinks about the grand visions, and then those are not rooted into the reality of the team's problems. So there are several things that we do, talking about this monolith duplication effort. We have a task force-based setup where engineers get together on the regular basis and work together with our engineering leads across all of the teams to both come up with some of the solutions to architectural challenges that underline the migration, get their experience in and to disseminate the information and the solutions that they actually created.

And overall, that's the pattern we pursue across a lot of other similar discussions. There are groups of people who are working frequently in such units as a group across various types of teams that discuss architecture and strategy and technology on a regular basis. Those discussions often result in an RFC, a Request For Comments document, which is then exposed to the whole organisation, and where people can receive feedback from whichever stakeholders that they want, that being management stakeholders or their peers across different tribes, across different teams or specialties.

And that's how the proposals are normally being discussed and travel from this idea phase to something that can get closer to the implementation. So overall, it happens to no levels and it typically starts pretty bottom-up, actually.

This one is my favorite questions. So which writer or book has got the greatest influence on your career or just your life, and why, obviously?

I can tell you that overall, I'm a pretty big book nerd. And I think books are something that I found a tremendously helpful guide to any type of extremely quickly paced growth phase, and we've been at it for six years, counting in August, also. And there are several books that I've read, several times actually, or we read several times, or sometimes we listen several times. Two of them I want to mention here in particular, the first one is a Radical Candor from Kim Scott.

Yeah. I think it's a pretty overall well-read and popular book. For me personally, I think what I really found exciting, she gives a lot of those really great example stories from her times, the various stages of development of the teams. And this concept of Radical Candor, overall, very much ties back into the culture that we are building at Personio. And seeing how actually this culture can be operationalised, in reality, overall, made me very inspired. So I came back to those books several times, especially during our performance.

When the performance cycle hits, I often take a look at it again to serve as a guide as to what's the best way to actually structure and formulate feedback.

You mentioned you had a second one.

Yeah. So the second one it goes, it addresses a similar topic, which is how to give the best possible feedback. And this one is Crucial Conversations from Kerry Patterson. I think there, what I found particularly interesting is the in-depth guidance towards how to, again, structure the feedback in terms of the purely, purely framework-based approach to this. Also, especially in the very first phases when we started to scale the team, proved to be, at least for me personally, a tremendously useful tool to really get some of the points across that I wanted to get for some of my peers and coworkers.

And overall, those are the two books that helped me quite a lot on this journey — you asked from the beginning — of transforming from the founding member to CTO now of a pretty sizable work.

I think that's a perfect note to end on. Thank you so much Arseniy. It's been a fascinating conversation.

Same here. I'm super happy chatting to you. It was a great conversation. And yeah, looking forward to more conversations maybe in the future.

EPISODE NINE

Adam Renklint

About: Adam Renklint is a co-founder and CTO of Pitch.com, the collaborative presentation software for modern teams. Before starting Pitch, Adam worked as CTO for Wunderlist, the cloud-based task management application that was acquired by Microsoft in 2015.

Pitch

To listen to the episode, **click here**.

Gonz: Adam, welcome to the podcast. How are you doing today?

Adam: Thanks. Yeah, I'm doing well. I'm very excited to be here.

Perfect. Let's dive right in. What's the two-minute version of Adam?

Two-minute version is that I'm a suede who came to Berlin to join a small startup called Wunderlist about 10 years ago. I have a background as a mobile and later front-end developer. And I guess you could say I'm in love with a web platform as a playground and application delivery mechanism. And I can get really nerdy about user interfaces, and in particular, how we create delightful moments in areas of software where status quo or the frustration of the status quo and there's not enough fun, so make software fun.

That can really be reflected in the page product, right?

Yeah. I mean, we've really put a lot of emphasis on that and the whole company, in a way, as a reaction to our time working at Microsoft after Wunderlist were in a more corporate environment. So we really tried to inject a lot of fun, and in a way, also remove a time waste in there. So you can really focus on the parts that you enjoy about creating a presentation.

Yeah, absolutely. And we're going to dive into the transition from startup to corporate to startup in a few moments. But first, I want to take a step back and start thinking, or I would ask you, how do you see the role of the CTO, but most importantly, how it evolves over time? You or Pitch started a few years ago. You've been growing pretty quickly, so I'm very curious about your perspective.

Yeah, that's definitely interesting. At the start, the role of the CTO is basically the lead programmer from building the first prototype to hiring the first engineer. And in that time, I was very much involved with every line of code. But as we grew the team and hired other really smart and passionate people, there is a need to both show the future direction to make sure we're all walking towards the same goal, but also keep scouting and looking for talent that can broaden the team, have new perspectives.

So basically I was going to say, today, I have four main objectives, set the technical direction with our business goals, higher amazing people and teams with experience and ambition, create an environment in which these people can be creative and do their best work and have the incentives to do so, and finally, I think, reinforce or observe and reinforce the culture that exists through values and principles in just talking a lot to people, which is what I do a lot these days.

That is incredible. And I would have double clicked on all of them. But I would start on the second one, which is you mentioned recruiting and hiring an incredibly talented team. So I think we could both agree that post-COVID, the talent markets really change. First, you have to compete for talent locally, but now it's globally, right? So how do you think about recruiting and retaining talent in the world where of COVID made talent markets global?

Yeah, that's a tricky question. I would say that now remote work or work from anywhere, has in the last 12 months, become table stakes, and it's not really something you can use to differentiate anymore. So any organisation that doesn't offer remote work in some meaningful way will really struggle to grow their team in significant ways.

So I'd say now being well-funded and able to pay above market salaries is definitely something that helps you to attract people initially. But in the end, I think that most people are motivated more than money to find a team in which they can be constantly challenged.

No one likes to be the smartest person in the room.

So creating groups or pods of people with lots of diverse experiences so people can learn from each other and then put interesting technical challenges, products, opportunities in front of them and have them refund that. The way that we do that at Pitch is through experimentation a lot, through iterative development, we co-develop Pitch with our users. So that's also very much a conversation between two parties.

And I think for a lot of engineers and designers, that kind of work where you're very tightly connected with the person that would use it in the end is much more rewarding than putting some software in a box and throwing it over the fence.

Absolutely. I'm very curious. What are some of the core trades that you look for in new hires, not hard skills?

I think in its essence, I believe that developing software is a team sport. So I think great communication skills are super important or also a bias towards teamwork and tight collaboration rather than any kind of superheroes or 10x engineers who want 10x teams that can be greater than the sum of its parts.

Things I would also look for is, although it's very tricky to see in a CV, in a conversation, but demonstrated ownership and accountability, being able to take really complex topics and break them down because everything is really hard when it's a ball of mud.

But if you can take it apart into smaller pieces, then most of us are able to solve those smaller pieces. And then when we put them back together, we can build something that is really impactful and powerful, but by reducing the complexity of how we need to maintain it.

It's something that's also super important for us because of how we develop products. As I said before, where everyone is involved is having a sense, an intuition for what is good, simple, and useful, and finally, delightful experience. That's something that everyone at Pitch, to a certain degree, is a trait that we all share.

That's great. And the last one is not one that I was expecting. So it's very interesting to see how you think about that and how it reflects in the product. One of the most interesting things is that you broke down your role into four parts. It's this thing about creating the right environment for your team. So what are some of the practices, cultural principles, operating frameworks that you use to create a high performing environment?

Yeah, so I think you can summarise that as creating a room with a lot of trust and then get out of it to really put together teams of individuals and give them a nice challenge to work towards a problem to solve and then get out of the way and trust the process. And I think a really important part there is that no one should be flying blind. So developing something in isolation is not good. We do it very closely with our users in many different ways. But I think shipping features very early with feature flags, but also be doing diary studies with users or just user testing sessions and really trying to put everything we do blend together two perspectives.

One is our vision of how we think that things should work, which is important, and the other is what the users need from us right now and in the future. And then when we merge these together, we come up with a compelling and delightful product. And either of these two, if they're too strong, we end up building something that people probably don't like, or we build something without conviction that is just a collection of random features. So that balance is something that we look for when we hire people and we train and we keep on practising.

And with that, other practices that go in line with that a little bit is a constant evolution of our practice, trying to be reflective of what you're doing and how that's working in the form of postmortem analysis if something went bad or retro if something didn't go so bad. So we try to really analyse things and make smaller tweaks the whole time can help teams remove the parts that they don't like very quickly, so that they can really focus on building impactful things and doing that in a fun environment, where they feel challenged by the right things

I want to go back to this idea of creating a room, building trust, and then getting out of it. What are some of the things that you do to create trust, especially in the remote environment?

Yeah, this is actually, I think super tricky. I don't think we or anyone else have fully solved this yet. But I think you have to do it even though you would want to have a process and tools that are somewhat aligned, you also have to carve out room for every team to be different in whatever way is good for them.

And finding some tools or more importantly, finding practices that work for them. Some examples of something that most or many software teams are doing is some kind of planning and some kind of retro. We don't really prescribe how that should be done. We ask them to plan ahead in some way so that they know what they're going to ship, and we know what they're going to work on.

But the format of this is not required in any way. So I think recognising that you've hired experienced and motivated individuals that are playing at the top of their league and then putting them in front of well defined problem statements so that we don't try to solve the wrong problem or build in the wrong direction, I think, can really unleash a lot of creativity.

And with that also, I think there has to be a little bit of room for conflict or friction as well. We're trying to build something out of nothing in many cases. In some cases, it's like iterative development trying to improve something, but for the most time, we're building something out of nothing. And there's a metaphorical rumor talking about, there needs to be space to be also vulnerable, be humans, and create a culture of trust within there.

That doesn't necessarily need to exist in the same way between teams. But within this team, they need to be a really well functioning pod tha's going in one direction, but with the ability to have a standard storm in there to create a new idea and also recover from that.

I would have switched lanes a bit. I mean, having all these conversations with incredible CTOs all over Europe, and this is the first one in which the word product keeps coming over and over and over and over. It's different. I'm not saying it's a bad thing, actually. That's why I want to double click on it, but it's very, very interesting. So how do you think about the role of products in a technical organisation? How is Pitch set up right now?

Yeah, so I think there's an instance, but we are a little bit different. I would call myself a product CTO. Our CEO, my friend Christian, would also consider himself a product CEO and also head of product. So this is really at the essence of what we do at Pitch. Actually, product is more a standing word for experience. It's really about how people are interacting with it. That is at the core of what we do.

So, like I said before, everyone is involved in it. We have, at Pitch, quite a limited number of product managers that are working mostly to facilitate the process of inventing product, and maintaining, and iterating on it more so than prescribing what we should do. So I think the culture at Pitch is really that everyone is co-developing the product and together with our users.

And I'm not sure if I could do it any other way, to be honest. That is how we work with Wunderlist as well. Also the reason why maybe some of us didn't fit in so well at Microsoft, where it's more traditional lanes, everyone should stay in their lane. Yeah, but it's working out really well.

Wunderlist, Microsoft, Pitch, again, like everyone is. So I think we can both agree that it's more fun and everything to work at a startup, right? Fast growing. We don't really like to state our legs, as you said. But I'm wondering, what's something that you would export from corporate to startups, something that is actually sort of worth discussing, something that's actually worth keeping.

I mean, I think there are some things we like to bash at the corporate experience sometimes. And like I said, starting Pitch in some ways a reaction to what we experience there. But for sure, there are a lot of things that are working well in a corporation that's well or like that. Like, one thing is, I would say that they have pretty clear paths if you're okay with staying in your lane for growing and growing your influence.

And I think in a company that is a little bit bigger also that has multiple products, many different teams, one exciting thing that we really can't offer yet at a company like Pitch with 130 people and one main product is jumping around from very different areas. And one day, I'm a machine learning engineer. And next, I'm writing some policies about something. The truth is, in a startup, you have to do that all the time. And you're kind of not able to stay in one lane.

So if that becomes your wish to specialise in that way, like some of the bigger companies are good in a way there. And as a startup, we're kind of growing towards that. Eventually, we adopt a lot of the good parts about a larger company as well. And I think it's every start up stream to not implement any of the bad parts as we grow. But growing is hard. I think we have to be humble about what we're doing.

Growing is hard. I think that definitely captures it. What's keeping you up at night for the next 12 months?

That's a good question. I would say that actually, that unless I have an important decision that, for some reason, I'm hesitating to make, I'm procrastinating, then I sleep really well at night. I'm not so worried about things.

I think our challenges in the next 12, 18 months will definitely be interesting and, at times, intense and stressful. But my method for dealing with it is the same as with software engineering. You take the big ball of mud and you try to decomplex it, to break it down into smaller pieces, and solve them one by one.

So this way we can go from a feeling of, it's overwhelming. The world is on our shoulders. Everything is out of control. That is mostly an illusion. So we can break that illusion by breaking things down into smaller pieces and finding the essence of each problem and solving that, and then zoom out.

And of course, you also need to look at it holistically, I think, but that's more a second pass afterwards, an optimisation, if you will, to the actual solution. So, yeah, my advice would be to not worry and break problems down.

One thing you said is that sometimes you procrastinate on a decision. Usually it happens to me when the decision is not super clear or when it's not obvious. So how do you actually get to a decision? Do you have a framework for that?

Yeah, not a framework per se, but methods that I think I've stolen live here and there. What I like to do is jump into Notion, write a lot of bullet points. At some point, they grow into a few different sections of trying to describe the opportunity or the problem, something that I learned from who used to be our VP of Engineering and now our CEO is risk analysis and really thinking multiple steps of what is the risk of doing or not doing something. So that's also something I'm incorporating in decision making a lot.

The simplest thing is usually to not do anything. So what is actually the cost of not doing that? And is that smaller than the cost of doing it? And I think at that point, if I'm sitting on a decision and unable to actually get anywhere, one of the best things to do is just talk to someone else, not pull the decision in their corner and try to force them to climb over that hump, but just get feedback and have a rubber duck to bounce back your thoughts at you.

Because usually the answer is there. You just have to dig a little bit, or there's not enough information, and then you need to go out and find that. Yeah, so I guess, in essence, trying to structure it and breaking it down again into smaller pieces. I guess this is actually a trend with me.

That makes perfect sense. And I'm actually going to steal that, if you don't mind.

Of course.

We're going to wrap up with a fun one. So what writer or book has got the greatest influence on your career or your life? And why, of course?

Yeah, reading, I mean, I would say I'm not great at reading books. The Earliest 37 Signals books, so 37 Signals books. They were quite influential to how we approach product development and try to be lean and learn from iterating. But I honestly prefer short form content front loaded with insight rather than anything sped out over 400 pages.

Life is really short, so give me the gist and I'll move on. I'm much more of a reading papers, research papers and essays, and following the discussion of that. I'm actually one of the people who really enjoy discussion threads online, maybe not on Twitter, but things like Hacker News and Lobsters, where there's really reasoned and nuanced discussion about things, which is often more insightful than the material in itself, because it combines many different experiences, and you can have a more well-rounded view on the topic.

That's a perfect place to end on. Adam, thank you so much.

Yeah, thank you. It was great. It was great fun.

EPISODE TEN

Georg Schroth

About: Georg is a co-founder and managing director of NavVis, the indoor mapping and positioning startup. Before founding NavVis, Georg was a Ph. D. Candidate and Post Doctoral Researcher at the Technical University of Munich. Georg also served as a graduate visiting researcher at Stanford, and worked as an Engineer at the university's patented GPS integrity algorithm: DLR.

NavVis

To listen to the episode, **click here**.

Gonz: Georg, welcome to podcast, super excited to have you here.

George: Thanks a lot, Gonz. Happy to be here.

Let's dive right in and let's give the audience some context. So what's the two minute version of Georg, and what you're working on?

Definitely a dangerous question to ask in two minutes, but I'll give my best. So yeah, I'm Georg. I'm one of the co-founders of NavVis and the CTO of NavVis. And NavVis vision is really to bridge the gap between the physical reality in the digital world. That sounds a little bit abstract, so let me try to be a little bit more concrete on this. So you certainly know that logistics, real estate, all of this has been truly transformed through GPS and digital maps, right? So none of this would be really possible without this, and we take it for granted, of course.

But it's unfortunately limited to outdoors, right? You don't have maps. You don't have GPS indoors. And NavVis vision is exactly to make buildings digitally accessible. So if we focus on factories, construction sites where the pain is the biggest. We do this in three steps. First is that we capture the building in unprecedented speed, like maybe a satellite would usually do this outdoors, even in a plane. We go inside, of course, around and around the building, and we are able to do this in millimeter accuracy.

And we do this in hours where it just took weeks in the past, so that's what we call it reality capture or some people may call this 3D laser scan. So the second part is to bring this data, this powerful data into a web-based platform, so that people, pretty much anyone can access these buildings, like search, annotate, collaborate as if you were on-site. Then the third part is that with our application, you can tell where you are once you are on-site.

So if you're running around in a factory, you want to know where you are. This is what I can do in addition to providing you with this digital reality or digital copy of that building. So there's actually where many years of research went in where we developed an AI that provides these blue dots sometimes, it's called like that indoors, but without the need to bound beacons, any kind of infrastructure inside the buildings. And in fact, this is also where the whole thing started when I was working in the Stanford GPS lab back in 2007. It was pretty apparent how powerful GPS and maps are.

And I was so convinced to bring this value to the buildings in industrial applications. And the positioning technology was the first thing that we wanted to build, and so I decided to form a research team at the Technical University of Munich under the supervision of Professor Steinbach, which ultimately led them to the foundation of NavVis in 2013. And today, NavVis is already a team of over 200 people that truly disrupted the industry. We are adding more than 5 million square meters of building space into the platform every month, and serving really the largest enterprises, so it's been all right, I would say.

I bet. I always want to ask a bit about your time at Stanford at the Technical University of Munich. You've been sort of, I want to say, obsessed, but you've been focused on positioning, mapping, and special recognition for the past 13, 14 years. Why?

Yeah. I think I really got inspired. I had the honor to work together with the fathers, if you want the founding fathers of GPS. And it truly took me away to see the power of it, right? It's just immense on how much this has impacted not just our daily life, especially during Corona. Imagine having a food delivery, close delivery, all these ecommerce stuff, and doing that by someone opening a paper map and looking they need to go, right?

The creation of that map by itself requires an immense amount of technology to keep it up to date and to know where you are.

Then imagining where we are today, even almost seeing the diverge of autonomous driving. And then in a building, many of us just go into buildings that are a few hundred square meters in size. But think about a factory logistic center, all of these large buildings as well do not have a map of how things are actually looking like, and also don't have any way of positioning inside this building. Seems odd and it's something that is very unlikely that will stay like this. And I thought, okay, someone will solve it. Let's try to contribute to this, right?

Yeah. Yeah. Crazy. We are used to doing something that looks as simple as Google Maps, but we forget the paper maps.

[crosstalk 00:06:03] like how many satellites up in the sky to help us with this. And even the stuff that Einstein said was necessary to bring this system up, and give us positioning, right? It's just incredible how much technology you need to come together. Yeah, not so much about... For me, at least, the thing that we need to just enlarge the amount of technology, but to enable these things that we take for granted inside a building. And it's something that we are always doing today to a fairly good extent for professional industries.

Something I'm very curious about is, so you started NavVis about seven years ago. And now, you're about 250 people. You raised about 70 million, so it's been right as you said. So what I'm curious about is what was the role of the CTO back then? And how did it evolve over time as the company grew? Especially for a company that sort of deals with hardware, which is kind of unique.

Yeah, true. Yeah. I think in the beginning it is probably with almost all founding teams on a deep tech company that you're kind of a little bit like on Earth. You must be an Earth. And I'm probably not even the guy in a team that has the deepest technical knowledge. And so in a sense, you really must laugh at technology, for sure. And I think that is definitely true. But then, as always kind of transitions into more and more, providing more and more directions versus concretely solving things by yourself.

So yes, the role is drastically changing over time. And from working on algorithms and finding solutions, it's becoming more and more like directions and empowering of people to solve these problems. And in my view, it's quite important to the side at a certain point in time, whether the CTO should be more focused on the engineering world or how much, at least on the product management side of things.

Because I truly think that these two things have to be very closely linked to each other. There should be no question about that. These two things must be working hand in hand to solve something. Honestly, if you look in the Silicon Valley, what made really successful farming teams, then it was that mostly those farming teams had at least one member that actually were able to connect the business with deep technology knowledge.

And these people typically don't come without any technical background. You have to have a good technical background, and then you have to grow into this or develop into this role. But still, it is also hard for me to just be good in both of these dimensions. So for me, it really means today that I am really trying to bring these two roles together if you want.

I'm responsible for the development, but also for the building of the product, vision, and strategy. So I try to do these two things. And in such, I try to orchestrate the teams, ensure the product management, product design, product engineers work really hand in hand. And for me, probably the biggest task is to develop a product, vision, and strategy, and then empower the people to provide the right team set up, the right direction, the right concept to execute on this.

So you are in a sense, the bridge or the connector between those roles. What I'm very curious about is how do you translate your ideas or your vision in terms that both technical teams and the rest of the organization can get behind?

So I think it is extremely important that you provide to the team as much context as possible. Obviously, I love to go back and say like, hey, you want to just go and do this and that in the software in our architecture, even in process development processes and algorithms. But then I realized that this is not... I can sometimes do these things, but this doesn't really scare me. How could it?

And so for me, it is more important to really, on the one hand, really provide the context of what is the field or is the landscape in which we want to solve problems. And then also help to identify the biggest and most valuable problems that we want to solve with knowing how... At least having intuition of how much effort it is to solve them.

Yeah. And then basically, empowering a team by making them as autonomous as possible to solve them without too many other strings attached. I think like what I need to do to help them to have an impact.

One thing I love to sort of spend some time on is hiring. You could argue that since COVID. Talent markets have gone global. You don't have to compete just with people for companies in Munich, but also in the rest of Europe or just the US, big tech, small startups, big startups. So things are starting to get interesting. So how do you think about hiring on retaining talent in a world were COVID made talent markets global?

Yeah. I think this is, of course, getting tough. No question about that, and we're relocated, you could say, fortunately. I would rather see it as a benefit. In the same street, Apple just started to build the campus for them. And so does Google come closer to us? Can imagine it's coming from all the sides. And then, of course, this is not the only thing. Of course, we have tremendously cool startups. So many of these people are actually friends of mine that really just enjoy doing that.

This is really becoming more and more also a melting part of the best talent. And I say first of all, it's great that we have it here, at least in Germany. Those companies are there, and that we actually have the chance to build something together and just revolve the world. And then, of course, yes, it will be not fair to say that I'm just happy for the others. Of course, we also want to play a big role there. And in such, on the question, indeed, how can we stay competitive with them?

Because there will be no way... And we should not outperform Google in terms of salaries. Actually, we truly don't think that is a good idea. We want people that basically... Don't want to be small. I think you said there's more clog in a big machine, right? We want people that actually what they do really makes a huge difference to the outcome of that company. And in contrast to maybe other companies, we are not trying to win a race of amenities or nice to have things.

We have more muesli and more coffee and maybe even more flexibility... What do you call it? Like working hours or wherever you want to work. We, of course, do that as well to the degree that it helps our people and our company to achieve, and that is actually what is most important to them, an extremely nice or extremely strong impact into this world. So I think, like these people that start with NavVis, these are talents that really stand for what they are standing up for, a vision that they truly believe in, an impact that is noticeable, and that they personally grow themselves.

And these three things are things like... Let's say, if you ask someone, what do you prefer? At least our people at NavVis, right? Even more flexible work or working for a product that is truly inspiring, impactful, and a hundred percent sure they take the second. And this is exactly what we need to do. We need to pitch our provision. We need to show what we're really able to change or have the chance to change to build in this world. And that is, I think, what differentiates us from other companies.

Speaking of what people would prioritize, what are some of the core traits you look for in new hires? You describe yourself as a team, as curious innovators, as proactive doers. So I'm wondering, how do you think about that? What sort of commonality?

Yeah. Actually, a very difficult question I'd say, because every role is of course, quite different. But as a common I would say what I'm looking for is the ability to very quickly exchange with this person to be able to really quickly understand what this person is telling me to learn from this person, and the other way around that I can basically tell this person what I know, what my thoughts are, and to really exchange quickly iterate on what we are thinking.

And I believe this is very important these days, much more than in the past, because the world is moving so fast that existing knowledge is important. No question about this. And an even more important experience. But what's even more important is to be able to quickly iterate and develop something new, and this comes in a team usually, right?

And so that exchange is a very, very high priority. And second, I'm definitely looking for energy drive and passion, because this is something that you cannot teach, right? No way that you can teach everything, but not energy drive. And this is really tough. And so if that is paired with the willingness to take ownership, like full responsibility and the willingness to make this ownership successful, then I think I found a very excellent new member of the team.

So what are some of the practices, principles, methodologies you use at NavVis to create a high performing environment?

So I think to some extent, bring this into relation with our values. We call it rather guiding principles. And we took quite some effort to understand what they are, because you kind of feel that, but you need to kind of put it in to work to tell this. Also, people that are recently just joining all this. And he said, look, this is valuable for us. Right. How do you reflect on this? You don't have to agree to everything, but you should know what is valuable to us.

And part of this is, for instance, that we have hopefully the culture to iterate fast towards world class results. We don't try to just live in the hopes of making a huge impact, and only then show it to the outer world. We rather try to slice it down into smaller segments, realize that we can't know everything, and rather than we have to bring it through the world to learn an Iterator. And then we also understand that we will make mistakes in this way. But these mistakes should have an impact smaller by not going out with one big chunk.

So that is one part, I think an awesome development culture makes a lot of sense. The other part is about focus. It comes to high performance. Then I would say it is about reducing work in progress. I think everybody knows this, that this is something bad if you have work in progress. And I would not claim that we are perfect in this either, but I'd say this is something that at least I try to remind myself all the time that we are just not getting faster by putting more in parallel. And this is much easier said than done.

The third is actually that we want to work with the customers for the customer, so we call it empowering our customers. I think this is also... Frequently sad, but you really, truly must believe that you are not considering the customer as someone that is just finally should finally buy our product, and you try to convince them with all means, but you really are deeply caring about what you create as a value to them. Is it really helping them? Or are you just trying to fool yourself on the customer at the same time, right?

Because it’s probably not taking forever until either one of them finds out that it’s not working. And maybe as a fourth one, what we call it, we own what we do. So it’s very much about ownership. I think if you want to build a company that’s growing 100 percent year over year, then you have to let go of things, right? It has a lot to do with trust. And if you need to empower the people, that is basically a very tough exercise sometimes. On the other side, we need people that want to be empowered.

This is also not for everyone, but I want to have ownership and understand that ownership means responsibility and that this is really your own. You must care about it. This is yours. You don’t let this be ignored. You will definitely raise their hand if you feel you can’t make it happen, right? And this is something that also very much reflects into engineering problems, because then the people who not just do it IC, even though they don’t believe it’s the right thing to do, they will only do it if they think, Yes, this makes sense and we will actually deliver something that has an outcome.

How often do you revisit or evolve these principles?

I think to some extent we try to rather remember ourselves, that these are principles, rather than to completely try to reflect on them, and shouldn’t change them. Because to some extent, it’s a little bit like the many methodologies. There are times when you want to reflect. There’s times where you want to just take what we have, even in a scrum rate after sprinkler, you rather want to execute and not constantly rethink if the planning was done correctly, and I think that is a little bit the principle here as well.

So the last time we did that was like just before Corona, when we kind of really finished on that. We actually had to send our then or present it in the way like, after all the workshops we have done during Corona, and I think it was a great exercise as well to kind of see. Hey, this culture is not only dependent on us physically meeting each other, and though I find this is very important. But it’s really something that we feel these are common terms we want to rely on to each other and in such.

I think I would not feel the necessity to review it again, but I think it has to be coming from... When I realized, I mainly come into a different stage of a company. We reflect on some things that have maybe proven to be not good for us, right? It should be not just reflecting where we are, but also where we want to go. Then I’d say this is something where we want to reflect, but it’s not like we want to reflect on this every half year. If it’s still true, it’s rather like we want to feel that there’s something that we need to improve.

So speaking of a high performance environment, what indicators, if any, of course, do you use to measure performance at software teams?

Yeah. That’s a very difficult question. Generally, I think on metrics, you can’t do it right, first of all. I think you can just build your best dashboard, and you will not know anything about your company. And I think metrics can tell you a story, right? Only the team can. So I think first of all, I want to understand what you want to improve and why. And then I think metrics can be used as an answer to very specific questions, very detailed questions.

So to test the type of fees that I think like what you can do with metrics. And usually, those types of fees should support a specific business code, right? So you want to go in a certain direction. You should be only done as long as the questions and answers help to drive positive change. I think it’s a precondition for anything. And then I think, like, literature I would say lead time from idea to deliver software, cycle time. So how much time would actually take you from changing software to delivering software, team velocity is certainly something, or open close rates from issue to resolution as well as question failure rates.

But these are all things. I think you should only really implement and do if you actually have hypotheses that you would like to falsify or verify. And then it actually makes sense. In our specific company, it's very tough to compare teams against each other, because we have built product teams. Since every product is different, they have different technologies, different software stacks. And yes, there are, of course, synergies in many dimensions. But it's impossible, I would say, to compare the velocity of one team against the other.

The pure idea of it sounds silly, because they have their own definition of story points and such. If it all analyzes the trends, so relative to one squad, but not trust them. So I think usually what you want to do is to first of all, understand where you would like to be in terms of product outcome, and then should try to bring this down to certain pieces that need to work together to actually make that happen.

And then if at all, you can maybe compare against competition. If you have direct competitors that have similar products, and see how much they have actually achieved and what term of a time, and then where you believe, you have been stronger product management engineering in focusing the business and so on. So again, I think there is no easy answer on which metrics to use.

No, but that's what makes it interesting. So thank you for sharing that. What's keeping you up at night over the next 12 months or so?

Yeah, definitely on how to grow this company a hundred percent year over year. So I think that is just extremely difficult, right? I mean, everybody, I think knows this. But obviously, it's easier with the first millions of revenue. It's becoming incredibly harder the more revenue you do to grow. And you are just required to constantly rethink our strategy. You can't just continue to do what you have found to be successful last year.

You constantly have to challenge this again. And I think, overall, this strong vision is of great help to stay on course and not to just jump for short term opportunities. But also realize that sometimes, you just have to invest into longer term things, right? Because if you want to grow a hundred percent, you need to take every straw that is on the way. And just put it on a big [inaudible 00:23:50], and hope that it's twice the amount of straws that you had last year. But that's definitely not going to work, right? And you kind of need to reflect that in phase of all the pressure that is put on you. And, yeah, that's what definitely keeps me up at night.

That saying, how does it go? What got us here, won't get us there?

Yeah, exactly. I think this is unfortunately not the full thing that you need to have, right? You have to have new concepts, new ideas, and realize that you can't burn your team, right? You have to know what you have, and then rather take risks, right? You can't grow a hundred percent without taking risks, and you always have to accept that protecting what you already have is less important than building what you want to have.

That's a perfect place to end on. Georg, thank you so much. It's been a pleasure.

Thank you so much as well. Great talking.

EPISODE ELEVEN

Daniel Krauss

About: Daniel is the co-founder and Chief Organizational Plummer (yes, you heard me right), of FlixBus, the smart and green mobility company. Since 2013, FlixBus has been helping people experience the world through sustainable and comfortable travel, and they now boast more than 350,000 daily connections to 2,000 destinations in 29 countries.

To listen to the episode, **click here**.

Gonz: Daniel, welcome to the podcast. This is exciting. How are you doing today?

Daniel: Hey Gonz. Thanks for having me. Great. Not really. It used to be great until 10 minutes ago, and then it started raining in Berlin. And I have to admit we had better summers in Germany, not only due to Corona, but also the weather is really... I'm hoping it's better at yours, and Argentina is sunny, though.

Yeah, it's sunny and nice. I miss Berlin though. That's where I want to relocate to. So let's dive right in. So to give you context to the audience, what's a 60-second version of Daniel? Let's start with that.

60 seconds of Daniel is: I'm a techie at heart. That's what I used to study. And somehow people let me know in the past, hey, that works. Do that. What my passion is is more building organizations, and really making sure people work well together at functions.

And when I had the chance to bring both together was co-founding Flix almost 10 years ago. And I'm responsible for all of our tech department, customer service, as well as people in org. And I love doing that, since it started, 2011. And hoping that now we'll really get our second bite after having survived that Corona pandemic, which obviously, for mobility, has been challenging.

So normally, what I would ask is, what is the role of the CTO? And how does this evolve as the company grows from a couple of guys 10 years ago to over 1,000 people now? But your LinkedIn says Chief Organizational Plumber. So I'm going to ask, what's the role of the Chief Organizational Plumber?

So the Chief Organizational Plumber is responsible to make sure energy flows. And I think these fast-growing startups, most important is that you have enough people doing the right job. And yes, we're a platform company, and therefore a tech company. And that means most of our people, or a good portion, about 200 to 250. It fluctuates a bit. I think about 230 currently, are techies. So there is a flow. And what I think I have to do, and in a later stage a CTO has to do, is to make sure the foundation works well.

And it's not that easy, because agility is a broad spectrum, and just copying the Spotify model wouldn't work. And you have to balance it between the product and engineering people, to the business people, and setting that ground. And at the end, making sure we continuously work on value creation, on one hand, satisfying the customer's needs. And that is what a Chief Organizational Plumber does. And if it comes to an emergency, well, it's literally like Super Mario. You come and plumb it out.

That's a great analogy. So I've been talking to CTOs for the past couple of months. And you have something unique, which is Flix has this interface between the world of atoms and the world of bits. Normally it's just software. So do you think there's something unique to your role that comes from that?

Oh, I think so. It's the heart which keeps the company going, which beats, is obviously our platform, which not only takes care of sales, but also of the entire inventory management, the planning, the pricing, everything. But the truth is our relationship management between us and the mobility partners, customer service, as a real mobility company, which is very much as it would be for any airline or for any train company. That's so much different from a pure SaaS company.

And obviously, I have to make sure that our techies understand this real world. And it's not B2B SaaS, as I said. They're actual people. And if shit hits the fan, they need to be brought from A to B. Because that's the service they took care of. It's cool, and I love it, so it's easy. And it's very customer-centering, and we have a cool app. But at the end, they pay for being moved from A to B. And that's something which I need our engineers to really understand and to buy into.

Now, on the other hand, I also need our business folks, that they understand how tech and product development works, because this is complicated. No, it's even complex. And I keep on saying to them, you guys want to create tons of revenue. You guys want to move hundreds of millions of people. And you ask us, as artists, to paint the greatest picture you can sell to your customers, in order of bringing them from A to B, and in providing mobility. And if you ask an artist, you have to have faith.

You can pretty much say what it has to do. Is it just a painting or a sculpture? But at the end, you wouldn't sit there and say, hey, draw this round, draw that line.

And managing that balance is important. And I think that comes with a passion of working with people and loving to build organization. And it's also important to have some foundational knowledge about technology and architecture to build that bridge. Eventually, I am even the bridge.

What are some of the tactics, maybe communication packs, or just organizational principles that you've set in place to bridge this gap between business and tech, which is a fairly common problem?

It is communication. So in the meantime, we obviously have reached a certain size that I have a very good VP engineer, who is, from an IT architectural perspective, even much better than I am. And it's true for our VP product in terms of the product engineering processes and all that.

So my daily diary is almost only communication. And I try to not only focus on strategic communication, talking with my VPs, but I really, on a regular basis, dive in, providing space that people can ask myself, literally, everything. Run around the office. Now, I keep on nudging people via Teams, because the office is pretty empty.

So you set tactics first, and it's purely communication. It's over communicating. I'm sometimes even annoying, maybe. But I don't care. I make sure that people get the big picture and how dots are connected. And as things change, also, as a management team, we really improved or increased the cadence of all hands, to really make sure everybody knows where we are currently at. Because mobility in Corona means safety first, on the one hand. On the other hand, we really have to match the demand out there. And that means it changes quickly.

And that's what I do.

And I, on the other hand, try to listen quite a bit. And here and there, hang around in one or two meetings more than I would have had to. Sometimes I'm muted and I do other stuff. I will tell you in advance. So it's not impolite. But why I do that, to listen also to certain discussions just from a business perspective, to have an idea what is going on, and to eventually be able to catch up and explain the other perspective.

So that is what I'm doing, basically, all day long.

I'm going to switch lanes a bit. Because you said two things. The first one is about having enough people and the right people. And the second one you said was the offices are almost empty right now. So I'd argue that one of the second-order consequences of COVID is that it made talent markets global. You just don't need to compete with people in Berlin, but with people in the US, and in London, and Paris, and Vietnam.

So how do you think about recruiting and retaining talent in a world like this?

In terms of recruiting, it’s not a question any longer if people would like to move from A to B. It’s just a question of finding the right talent, independent of where in the world. Yeah, there are social security challenges, freelance versus just regular employees, you have to figure out. But in general, it’s not as relevant as it used to be where people live. Therefore, you have much more focus on what the culture is, and what it means to work together in such a hybrid environment. So you don’t have loners. I think this view is better.

And I really had to also reconsider our strategy in terms of product teams. Because really, we’re totally keen on colocation. And I still love it. But the truth is, this is somehow past. I also laughed, quite some cool stuff in the ‘80s. But at some point in time, V12 combustion engines will be forbidden. That’s what it is. Deal with it.

And therefore we really have to make sure people get trained how to work in that environment. So you must not have loners. This is psychologically not good. And it’s also not good from a productivity perspective, first. And there we really have to improve our recruiting processes.

Regardless of that, I always recommend it, if young startups ask how to do that, you have to have dedicated recruiters. That's the only way. There is no other way around to find a good talent.

And in terms of retention, that’s particularly important in general, but particularly important to our model during Corona. People were insecure because it wasn’t clear how mobility, Corona, what will happen to the company? And to make that clear, I really was very open and transparent to our leaders, and said, hey, if you feel uncertainty at your people, let me know.

So what I did, and I keep on doing, to retain people, I engage. If I realize there is a little bit of uncertainty... And people love the company, love the job, but they’re not sure if everything is safe enough, and they can feed their families also over the... They do not have to go to IBM or another US blue chip. We are stable. It’s a super-healthy company. But it’s eventually not obvious. So I just got feedback where these things appear, and then I engage. I’m a retention guy.

And because that’s obviously very close to my role as a founder, that I’m trustworthy. And that’s good.

One thing about retention is giving people the ability or the opportunity to grow. So how do you think about team development? What sort of career tracks have you developed for your tech teams?

Truth is, if you only look at salary and compete with the market, and you do not invest quite a lot to make that a meaningful factor. I’m not saying you want to underpay. You want to pay market average. But you want to give people enough opportunities to grow and to be appreciated. And it’s about roles. It’s about salary. And it’s about the content. That they feel comfortable and are not open to be nudged by the other tech recruiters. And there, personal development is a crucial game.

And I think it can be supported by training, and coaching, and all that stuff. But it has to be much more on-the-job, to give them the opportunity to play around with new technologies, if we’re in engineering.

So there is one rule we have. As long as there are three people... And three is the reason because, if just one guy or a girl comes with a new technology and implements something with a new language, and he or she has a piece of loner code, that I don’t want. If it’s three, then they basically can also... And it’s a micro-service or a [inaudible 00:13:36] service, they can just play around with anything. And I didn’t watch closely, deep into our code base for a while, and then I was like, whoa, we do stuff in Scala and Golang.

See, you didn’t know that.

But it’s fine. As long as I can make sure it’s maintained, I give people the freedom to really try out the new shit, even though sometimes it’s not obvious on the first view. So giving people room to grow, giving them their own budget to also look for conferences and all that is really important. Because that gives people a certain perspective. And obviously, we put that... We have a very flat hierarchy. We have almost no additional leaders in FlixTech. And that means you still have to... It’s all a game. It’s like playing Diablo.

You need to gain experience. And that has to be somewhere in inventory.

And therefore we have career paths. We have a techie career path. And that’s junior, mid, senior, principal, distinguished. And the occupation is more or less techie. And the role can be different. It can be a product owner, it can be a tester, it can be a data science role, or the standard engineering role. And then we have the process and product role. And then we have the leadership role, that’s people manager, people engineering, people manager, who are more the facilitator. And in that role, particularly, is, by the way, also attached the responsibility to develop their hood to grow.

And there people just can not only gain experience, but they obviously can also reconsider.

We had a very good principal engineer who recently decided to become a people engineering manager. Because he's like, I've seen so much, and I'm in my mid-30s. It's hilarious. But I now want to take care of others. And not only as a sidetrack, because anyhow it's expected from a senior engineer to do that, but mainly. And then he just became a people engineering manager. And now sometimes complains that he must not commit stuff any longer, but make sure that people work all together.

Diablo is a great game, by the way. I haven’t played in years.

Oh, yeah. Looking forward to there’s, I think, Diablo Two remastered ahead of us. So that will be great.

That should be fun. That and Age of Empires.

Oh, right.

So you said giving people the opportunity to try shit. What do you think about top-down versus bottom-top decision making in such a large organization?

The top-down decisions are, to me, a framework, some guard rails. So we can afford a certain level of money, cost, for us, as Flixtech. That’s dependent on the topline. Because obviously I want to maintain a healthy growth, and also a healthy bottom line in terms of EBDA. I’m not saying it has to always be profitable, but it has to be healthy. So that means there is a ratio. How much is the tech organization okay to cost in terms of pieces of revenue?

And I explain why I think about that, and that we obviously have to have economies of scale. So that’s the explanation sessions I continuously do. And if things change because we grow faster and the topline goes up, or sometimes Corona, topline goes down, I’m like, that’s the frame, it changes. And that’s something which are guardrails. I decide top-down. And how to achieve all that, with what technologies, even backlog, people tend to just arrange that properly so they find the best way to move within those guard rails.

And that’s almost always a bottom-up, or rather from the customer towards the inside, in order, as I said, to achieve what we, annually, usually set as our budget. And that’s the balance between both.

Yeah. So it all goes back to communication. Because you set the frame, and then communicate it, and then people play within that sandbox.

Yeah. One of the key challenges, especially, and continuously, if business people and tech people work together, is that business people tend to tell tech people, this is what you need to develop. And tech people, really... and that's fine... just come back and say, hey, can you state the problem, and then I come up with a solution? And that's exactly the same, usually, in that specific guard rail exercise.

It's not a real problem. If you then talk about a problem, you come close to sharing the value, and why there is the dogma of growth, blah blah blah. It's not a real problem. But it's more what my responsibility as CEO and founder is. So I just set the guardrails. But I can't just shoot a number because then it's like, why is it?Why is it only this piece of the cake and not more or less? And if you are not able to come up with a problem, but really ask people to do something without stating a problem and letting them find a solution, then it's your fucking duty to pretty much explain it as long as everybody says like, ah, now I understood.

What's keeping you up at night over the next 12 months?

Over the next 12 months. Well, that's pretty clear. My wife and I have ETA for a little daughter in September, so -

Congrats.

Emily. Thanks. And I'm pretty sure most of the time in the next 12 months she'll be keeping us up at night.

That's the perfect answer. And I'm going to switch lanes and wrap up with my favorite one, which writer or book has had the greatest influence on either your career or your life?

That's a good question. In terms of career, there are so many. One of the ones which I can always... What I would have read in any environment, startup, high growth, not a classical corporation, is, for sure, "The Hard Thing About Hard Things" from Ben Horowitz, because it really is eye-opening. If you're a new founder or you're joining a new, younger company, it's eye-opening. It's not all true, but it's funny to read, and it's eye-opening, on the one hand.

And the other one, which I think from a life perspective is... Actually, let me Google the English word. I can't find the English title of the book. Well, then people have to choose their own Rushdie book. The only thing, which I have to admit, is that all of them are pretty thick. So you need to have time.

Well, that's usually not a problem.

When I've found the English title, I'll shoot you afterwards, and then you just put it in the subtitle of the podcast.

Yeah, perfect. Cool. We'll make it work, as we usually do. But I think that's a perfect place to end on. Daniel, thank you so much for your time. It's been fascinating.

Very welcome. Thanks for having me. And it's a cool format. The only thing I really have to enter about Seedtable is, if you hit Munich, and the one in 20 startups to watch in 2021, I'm not sure if I found Flix.

You guys are too big, I think.

All right, all right. That's the perfect answer, though. It's a pleasure talking to you.