Beyond the Commit, Episode 4: Ed Addario - Transcript

In this episode, we explore how today's technology leaders can stay sharp, innovative, and effective in a world transformed by AI and rapid organizational change. We look at what it really takes to remain close to the craft, how to judge AI tools by their actual business impact, and why the right team structure is essential for experimentation and long-term growth.

Paweł Dolega talks with **Ed Addario**, CTO at Kuda - a seasoned technologist whose passion began at age 11 with a TRS-80 and evolved into a career spanning mobile, cloud, AI, and over 15 years in fintech. Ed co-founded Navro and previously served as CTO at Currencycloud and Salary Finance. Today, he's helping drive Kuda's mission of "banking for every African on the planet." He's also an active member of IEEE and ACM.

Ed shares his perspective on staying relevant as a hands-on leader, building organizations that encourage innovation, and navigating the journey from startup chaos to scaleup discipline. He also reflects on the realities of attracting top engineering talent in a competitive global market.

A concise, practical conversation for anyone shaping technology teams and products.

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

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

Paweł: So we are up right now and we can start with the topics. Hi Ed, it's a great pleasure for me to have you on this episode today.

Ed: Good morning, Pawel. Thank you very much for inviting me.

Paweł: So we have plenty of different topics to discuss today. So I'm already excited about those topics. So things that I had on my mind that we discussed before, you know, before these episodes, there's a bunch of different topics. So with you having a lot of experience in the senior engineering or senior leadership roles in the engineering departments in various organizations, I thought that, you know, we could talk about broadly speaking the life of a CTO—so responsibilities, challenges, trying to reconcile the individual contributor technical roles with the management responsibilities. We also could talk about how to manage the team dealing with or handling the innovation and work on the project, on the more business side of the project. And then obviously, since we are right now in September 2025, we could also talk a little bit about AI and the impact on engineering in general.

Sounds like plenty of topics. And I think we could start, given that we typically aim this podcast to our viewers or listeners in the senior engineering roles. Maybe we could start with this angle of the CTO role. And as I mentioned, you've been serving as a senior technical leader in many different companies over the years. Could you tell us a little bit about your perspective of the role of the CTO?. You know, what are the responsibilities? What are the challenges?. You know, the sort of discussion between technical skills, management skills, soft skills, and things like that.

Ed: Well, I think, I mean, ultimately, the role is going to take different shapes and dimensions based on the stage of the company and the key reasons for the company to have a CTO, right?. So, for example, in the case of a startup, the role of a CTO, but first and foremost, usually it's not—the title is not a CTO, it's just a senior leader, could be a senior engineer. And the expectation is that you are going to be a little bit of a jack of all trades and you're going to be wearing multiple hats. One day you're going to be going with your co-founders or the founding team, you know, cap in hand to try to raise capital. So you need to provide a little bit of context on how the engineering organization of this newly founded company is going to be growing, the challenges that you're going to see, how you're going to do retention. So you need to be able to talk from the perspective of what an investor is interested to know, which is: What is your target market? How are you going to develop the technology? What is going to be different?. What are you bringing to the market that is different from your competitors?.

So if the organization is a little bit larger, it's not necessarily a startup, let's say the scale of the dynamic changes. Sorry, let me just go back to complete the startup case. So as I said, you are wearing multiple hats. And the next day, you are going to be the tech guy that is going to have to code a potential prototype. On the next day, you may be the infrastructure guy. So you need to wear multiple hats, right?. And you also need to recognize where and at what point you want to start bringing additional people to the organization. So it's a tough job. I think that being the technical lead for the startup, it is a really, really demanding job because it forces you—it's going to be stretching you in so many multiple directions, especially at the beginning. My advice is the sooner you can start building a team around you, the sooner you can start delegating some of these activities, the more you're going to have time to think about what happens next.

And if there is a single thread of what a CTO has to do, no matter the type of organization, no matter the size of organization, it is going to be that. So ultimately, the CTO or the technical leader has to have a clear idea or at least an intuition of what happens next. The organization is trying to go from A to B, and definitely you have a responsibility to ensure that that journey happens. But before you reach B, I would say you already need to know where and what is C. And ideally, you need to have a line of sight of what you want B to be. Because it's all about the journey, right?. And that will be something that the technical lead will have to do, no matter the size of the organization. However, if you now are in a scale-up situation, the role changes a little bit.

Paweł: Mm-hmm.

Ed: In that the key transition between startup and scale-up is that when you're a startup, most likely the decisions, the thinking happens in one single room. You have everybody next to each other. You can discuss. Everybody has full knowledge and visibility as to what's happening. But as the organization starts to grow, you have additional teams. You're potentially going to be operating in multiple regions. You're going to have more than one service, probably you're starting to enrich your products and your services and therefore trying to keep all of that in one brain, it becomes much more difficult.

Paweł: Just one question before we go: you say startup to scale-up, could we put some sort of numeric value to it?. When you define scale-up, is it about the size of the engineering team or the revenue?.

Ed: It's tricky. It is very, very tricky, especially nowadays where we are forecasting that very soon there is going to be a billion-plus company—a billion in revenue, a billion-dollar size organization—that is only going to have one or two people behind, two humans and a lot of AI. But we'll touch on that in just a little bit. So no, don't think that it's—I mean, there is definitely an element of headcount, I think, that determines the transition between a startup and a scale-up. But I don't think that it is a cause; I think it is more of an effect. So to me, the way that I gauge where in the spectrum, in the continuum of evolution of an organization, at what point a startup is starting to transition into a scale-up—to me, has to do more with: in a startup, you rely more on the sheer will of people wanting to make things happen. However, in a scale-up, you have to start transitioning from single heroics into relying on the process, relying on an established culture and established ways of working. Because what you're looking at doing is, as the name suggests, scale up—scale up your operation.

Paweł: That's a wonderful definition, by the way. I very much like that and I agree with that from my experience.

Ed: And by the way, that is also the reason why I think many companies struggle with this transition point because you need to start approaching things differently. As a matter of fact, I think that one of the symptoms is—and I heard this many times, I also do some consulting on the side—a typical question that I get from CTOs is, "We feel that we need to start putting some processes in place.". That is a sure sign that you are already in that scale-up phase, in that it's not about what a single person or a small group of people can do. This is now about how you can get that magic, that engineering magic replicated across different countries, different teams in a way that it all works together seamlessly. And I think that's kind of the inflection point for scale-ups. It's about recognizing that humans are important—don't get me wrong, they continue to be important, you still want to have the sheer will and the drive to make things happen—but you want that to be somehow codified in the process that you're going to lay out for the organization.

Paweł: So in your experience, I know that you said that it's difficult to assign a numeric value in terms of headcount, but in your experience, are we talking about the transition happening more like when the company becomes 20 people, 50 people, or 100 people?.

Ed: It's hard to tell because it's not really the headcount. It's about the inflection point in which you realize we are growing. The demand for our products or our services is growing; we're having some success, some traction in the market. Therefore, we just need to keep building on the momentum. But indirectly, I think it is connected to size, definitely. So let me put it this way: for a startup to claim they are a scale-up once they hit the 150 to 170 mark—and I'm talking about the whole company, not just the engineering organization. I think once you start crossing the 150-170 headcount, you are already in a scale-up situation.

But again, the wording is not directly linked to headcount. It's linked more to your ability to keep delivering products, enhancements, or additional products at a pace where you're going to be able to meet your customer demand.

Paweł: So we talk more about this mythical product-market fit?.

Ed: There you go. Which is not mythical, I'll tell you that, Pawel. I mean, you have to have a product fit. Because it's just a fancy way to say, "Do people like and want what you are producing?". If the answer to that is yes, then you do have a market fit right there. If the answer is not so much, you need to find what the angle is. A huge challenge that I see nowadays, especially for startups or newly formed companies, is that competition is brutal. And I can go back in time—I mean, I'm old enough that I can go back in time to when the internet was still an unknown thing. And I'm looking at opportunities to compete compared to what it is today. And today is really brutal. So if you don't have something that makes you different from your competitor—whether that difference happens to be your product, your services, or the way you position—your advantage could be purely marketing. You could be making more noise than your competitors and therefore attracting more customers, not necessarily because you have anything that is materially different, but because of the way that you're presenting. That is still valid, but it gives you that uniqueness. So you need to be able to have that "unfair advantage," whatever that means.

Paweł: It's very much in line with my experience. I lived and I still live in Poland, and I think in 2010 it was relatively easy to make any kind of software business because there were so many gaps. Even talking recently with some venture capital companies, they also say that 10 or 15 years ago, in general, there were many different niches or areas of the market that were not covered by software. And today, when you operate on the forefront of the industry, obviously there are always new things. But in the past, there were many gaps in businesses that were already existing. Today they still exist, but the niches are getting smaller and smaller. And that actually affects the potential return because those gaps are not big enough in many businesses to justify investments which would need, I don't know, 100 times returns.

Ed: I agree unless there is a transformational event, right?. And I just want to make the point because something that I definitely do not want your listeners to get is a sense of doom and gloom that there is nothing else to be done, that everything has been discovered under the sun. That is not the case. All I'm saying is that competition is brutal because now you have much more. There are more smart people nowadays that realize, "Hang on a minute, there is no constraint to what I can achieve.". It is different nowadays than it was before. So there are more opportunities. Yes, the market is a little bit more brutal in my opinion, however, the opportunity is much higher.

Technology follows a cycle. I think that the people at Gartner, when they introduced their innovation curve, codified it incredibly well. But pretty much, there is always a rhythm and a flow to transformational technology progress. And every time that there is one, the entire paradigm—actually, let me use a better word—the entire way of working gets reset, and that creates a tremendous amount of opportunity. And I know there is a whole section in the podcast where we're going to be dipping a little bit more into AI. So I'm not going to get ahead of myself, but the message is: it's definitely not doom and gloom. It's actually getting even better. And the opportunities to create something out of nothing right now are much more abundant than they were five, maybe six years ago.

Paweł: Yeah, so we started—and actually I interrupted you there—but you started about this definition of the role in the startup. And then we went into this digression, but you were moving to the definition of the CTO at the scale-up. So maybe we could continue that.

Ed: So for companies that are in the scale-up phase, the role of the CTO changes a little bit. You're not the jack-of-all-trades anymore. You're not the one consistently trying to drive the team to succeed technically. Now your role is more about aligning the technology operations to, one, get efficiencies. So you want to be able to do more with the same. There are going to be cases in which you're going to have to do the same with less. Every company is going to go through economic cycles—a time to grow and a time of readjusting its size for a number of factors. So the role of technical lead really is: how can I continue delivering at the same pace with less resources, less people, less room?. Hopefully, those are not going to be that frequent, because it's not necessarily a comfortable situation to be in, but you still need to drag through. But as you keep growing, you need to start looking at conserving your cash. Efficiency, cost efficiencies, doing more with the same—it is going to be a constant theme that the CTO has to be always at the forefront of.

You also need to be a partner to the rest of the business. In startups, success is very much driven by your technical capability. In a scale-up, your success is very much dependent on your ability to consistently deliver your products and services at the quality they need to be and at a cost to remain competitive. So the dynamic changes. As a technical lead now, you have to be the partner to the rest of the business. You need to interact with finance, with the product teams, with compliance if you happen to be in a regulatory environment. And you need to be able to translate their needs and their agenda into what the technology organization has to do and implement to underpin, to enable, to leverage, and to help grow the rest of the organization. And it is a two-way street. You also need to be able to translate the technology needs and your roadmap into a language and a value proposition that your head of compliance can understand, that your CFO can understand, so that they themselves can also adjust and realign their own agenda to help support yours. So it's about bringing technology and translating that in a way that the business can understand the value, the need, and the merits.

Paweł: This pursuit of efficiency and consistent development ties very neatly to what you mentioned earlier about this transition to more procedural work, which you need to think about as you approach the scale-up phase. This transition from individual heroism to more management responsibilities is something many people struggle with. I think there was even an article written by Paul Graham about the "maker's schedule" vs. "manager's schedule.".

Ed: That's it.

Paweł: So yeah, that was about the requirement for the individual contributor to have deep focus and long stretches of work versus management work, which is a lot of interruptions—30 minutes here, one hour meeting there—and even a half-hour meeting that could destroy your state of flow or focus for half of the day. How do you reconcile those two sides of the CTO role?.

Ed: It's not easy. And I'll probably rephrase the focus element of it. As an individual contributor, you have the benefit that you can focus on a narrow set of activities and deliverables. The key difference between individual contributors versus people leadership situations is that for an individual contributor, it's about you and the change that you directly can exert. However, as a people manager, the dynamic changes because the influence that you exert is about aligning, empowering, and delegating people to do what they need to do. So your role becomes one of: "I'm going to find the time, the money, the resources, and the people that you need for you to be able to work and deliver. I will provide direction in terms of what the rest of the business feels is important and critical.". How you get there is on you; that's where the personal agency comes in.

Now, the focus in those two approaches is actually exactly the same. You need to be very focused whether you are an individual contributor working on a particular piece of work, or a manager developing a budget spreadsheet for the next year or working with a product colleague. That focus, intensity, and purposefulness translates equally well. The key difference is that when it comes down to how many contact points or human interactions you have throughout the day, the individual contributor will have fewer than a people leader. Your job as a leader is to keep an eye not just on your own backyard, but the rest of the house. What is the marketing guy doing? What is the product guy doing?. How can I help enable or steer from my perspective?. But yes, there is intensity and focus on both.

Paweł: Do you have a specific way of approaching this? Like Thursday or Friday being "no meeting" days for deep work?.

Ed: I personally don't believe in that. Maybe I just haven't done it correctly, but the biggest issue I find is that the day designated as a "no meeting" day often becomes the day where all of the meetings happen anyway. There's this perception that "Pawel's calendar should be wide open today, and I have this very important thing to talk to him about.". I just don't believe that works for me. However, time is absolutely your most precious resource. If you are not protecting your own time fiercely and ruthlessly, people are going to take chunks of it and you won't be able to do what you need to do. Rather than designating specific days, I manage my calendar ruthlessly. Especially when joining larger organizations where you have a PA, usually the first week is a discussion of: "Thank you, but I will manage my own calendar. I will decide what goes in.". Because I know better than anybody else what is important to me. I keep lists and I have regular cadence interactions with people I manage or work with, and each has an agenda. You have to be ruthless and keep track of that on a daily basis.

Paweł: Yeah, so what it means is that if there is no agenda for the meeting, there is no meeting.

Ed: Exactly right. As a matter of fact, I try to bake in a culture of "defend your personal time.". If you are invited to a meeting and you do not know why, what it's for, or if it's for information, brainstorming, or decision-making—reject the meeting automatically. Sometimes I even give a little email template to people to copy and paste: "Terribly sorry, I would love to join, but I have clear instructions that if there is no agenda, I have to decline. Please let me know what the agenda is and resend the invite.". If a meeting has no agenda, it's out of my calendar almost immediately.

Paweł: I think it's important to say that this only works if you have support from someone in your role—a senior leader who supports this approach. Mid-level or senior engineers might find it difficult to answer that way because it might seem rude. It's part of the organizational culture.

Ed: Which is why I do communication to the business saying: "Please understand that if somebody declines a meeting, it's not because they are being rude. They are declining because as a company, we decided that if a meeting doesn't have a purpose or agenda, we ask people to reject it.". We give them the executive "air cover" to feel comfortable doing so. But you're right, not everybody feels assertive enough. If you are a manager and you are issuing meeting requests without being upfront about the objective, you are already missing a key element of being a manager: providing clarity and direction. So, be kind to your team and yourself. Be specific. If the purpose is just to "shoot the breeze" or have a chat in a remote team, make it explicit. "No specific agenda, just to understand how you're doing." That is still valid; people just need to know what to expect.

Paweł: Sure, bonding in a remote team is important. But different people operate differently. I am not usually a fast thinker; sometimes I need 5 or 10 minutes, or even a few days, to think about something. If I learn something during a meeting and have to give feedback immediately, I'm going to give lousy feedback. If I receive a one-pager two days earlier, I can be prepared.

Ed: Spot on, Pawel. Exactly right.

Paweł: But it requires that organizational culture.

Ed: It does. It is a necessary condition, but not sufficient. You cannot ignore personal accountability. As a leader, you can set the rules and create the space, but if people do not step up and take the option to decline when they lack context, then it doesn't matter how much process you create. It's a two-way street.

Paweł: In your case, joining established organizations makes it even more difficult. If you were a founder for 10 years, you set the culture. But as a new CTO, people might think, "This new guy has this fantasy about agendas, but I don't care." Is there a recipe or game plan for that?.

Ed: It's a fact of life that happens. I struggle with the concept that a person can somehow create culture by "dictum" or directive. I have seen companies fail miserably when they say, "As of tomorrow at 9 a.m., our culture is these 10 bullet points.". The recipe for cultural change is that the culture of a company is a reflection of the behavior of people in leadership. You have to lead by example and be relentless. For example, with remote working, it's common to have sessions where everyone's camera is off. I want to talk to the real you. So I lead by having my camera on. I might make a joke about it: "How do I know I'm not talking to AIs?". You coach, you mentor, you cajole. Once people understand the reasons why, culture becomes contagious. It takes time—usually a good year and a half to two years to really "flip the needle.".

Paweł: And as with many things, it's difficult to spot shifts at the beginning. Switching topics slightly: how much time a week do you spend on technical work? Like building software or reviewing code?. I know you do that because I've seen your contributions on GitHub, like llama.cpp.

Ed: That is how I keep my sanity. It depends on the stage of the company. At the very beginning of my previous company, it was just the three of us, so I had to code prototypes for investors. But the larger the organization becomes, the more you have people who are way better than you. You have to delegate because a manager's role is to influence outcomes through others. I keep close to technology for three reasons. First, I love it—I wanted to be a brain surgeon until I got a summer job at a computer store and it was love at first sight. Second, I do it for my own sanity; it's my go-to place on weekends or at night. Third, I do it to maintain awareness of what it is to be a software or DevOps engineer. You need to be able to put yourself in their shoes when discussing pipeline issues or resource investment.

Paweł: Do you also do that during regular daily work? Like fixing a simple thing to see how the pipeline works?.

Ed: Yes, I do, for two reasons. One—and this is uncommon—is to send a clear message that I have awareness of technology, so don't try to tell me something inaccurate. Every technical person has, at some point, used three-letter acronyms (TLAs) to obfuscate a concept to sound more impressive. If I fix a PR for a failing pipeline secret, it sends a message that I can dive in if I need to. The second reason is better: if I have an idea with many moving parts, creating a prototype or a few lines of code helps the team "grok" the intention faster because we are talking a common language.

Paweł: So you are talking about prototyping an idea?.

Ed: It could be a prototype or a change in how we manage clusters. For example, we were looking into "Karpenter" for our Kubernetes cluster on AWS. The easiest way was to have a quick spike. I had more experience with Karpenter than my DevOps team, so I did a simple version to prove if it was the right choice. Retaining that coding capability is helpful. Having said that, it's not a "recipe for success"—I've known incredibly successful CTOs who would struggle to write two lines of code. It's just another tool in the toolbox.

Paweł: That's an interesting angle because one might think you need a certain understanding of the job to manage it.

Ed: A hundred percent. But there's a difference between understanding the *practice* of software engineering and being able to write a polymorphic C++ class. One is the "overhead," the other is dropping into the wires.

Paweł: Technology changes, and as a CTO, you wear many hats. With roles like data engineering, data science, and design, it's impossible to have experience in all of them. You'll have to manage work you aren't truly experienced with.

Ed: That's where humility and honesty come in. You aren't expected to know every single thing; you're expected to know how to find the answers. It's incredibly refreshing when a senior leader says, "Actually, I do not know the answer to that, but I will find out.". Really powerful leaders know their limits and are comfortable disclosing them.

Paweł: That's a great thought for young leaders who might try to hide that they don't know something, which backfires.

Ed: Don't try to project an image of knowledge if you don't have it. It will destroy your personal credibility. People are much more willing to help when you are transparent.

Paweł: As the famous quote says, you can't fool all the people all the time. Do you believe you can be an effective technical leader without *any* background in software engineering?.

Ed: It is possible, but much more difficult. Leadership comes from making the right decisions and gaining the support of people. If you lead technology experts and aren't one of them, you have to overcome human resistance: "What do you know? Why should I follow you?". I know a leader who was a lawyer by trade—he was a good listener, a fast learner, and open about his limitations. He turned that into a strength by saying, "I know what the business is trying to do, but I don't know the best technical way. Help me understand.". But you have to prove yourself more. If you go in on day one talking about "sigmoid functions" and "Kubernetes," the shields come down a notch. If you're not perceived as "one of the team," those shields stay up.

Paweł: Let's talk about the first 100 days of a CTO joining an established company. What is your game plan?.

Ed: It's unlikely you'd join without an idea of why they are looking for a CTO and how they will measure success. If you don't know that last bit, clarify it. You want your focus geared toward those things. When you join, all you have is hearsay—the CEO's laundry list of disasters, or everyone telling you what is wrong. I take that input, but I categorize it into "the three Ps": People, Product, and Process. I want visibility into the people dynamic—are we supporting their careers?. For products, I look at challenges and quality. I also use a framework—some will be familiar with DORA. If you aren't, Google it (but specify software, because in Europe DORA also refers to digital resilience regulations). The DORA study looked at what makes technology companies successful and recognized about 15 dimensions for high-efficiency delivery. I use that to guide my thinking and inspect all the corners of the "house.". You need first-hand experience of the "as-is" situation to come up with a "to-be" plan that addresses concerns.

Paweł: This "first-hand experience" ties back to being hands-on—walking the floor or checking how the deployment pipeline works.

Ed: Exactly right.

Paweł: Moving to scaling the organization: what qualities or skills do you look for when hiring engineers?.

Ed: The qualities are the same as they have always been. When expanding, I typically focus on entry-level positions. For senior roles, I want to look internally first to create career progression. If someone in the group is close enough, we use coaching and training to bring them up to that expert level. That creates a cascading effect of promotions. Of course, sometimes you need to go out if you don't have the tech stack in-house—like if you're an Android shop starting with iOS. You're looking for people who can do the job, but also people who are "human.". If you hire someone whose attitude is toxic, that is a bad hire. I focus on skills, but if I have to choose between someone with skills but a bad attitude versus someone transparent and human, I go for the latter. Elevating skills is something you can do; changing someone's perception of their own self-worth is incredibly hard. Don't let a toxic player in your team.

Paweł: You mentioned you often hire entry-level positions. It's a weird market right now—layoffs in some places, but growth in others. People claim it's much harder to find entry-level positions today.

Ed: That approach only works once you have "critical mass.". If you are a startup, you need people who can hit the ground running. But once you have enough senior levels, you want to create promotion opportunities to retain them. It will be painful—you won't get the same velocity as with an experienced person—but you gain retention. Regarding the entry-level market, AI is already having an impact—some of those positions are being replaced. If it hasn't happened in your organization yet, it will within a year, or the organization will be phased out.

Paweł: What do you look for when hiring or promoting to a technical leadership position?.

Ed: Same thing: human qualities and the capability to get the job done. But for senior leadership, I don't always promote from within. Sometimes you need a particular skill set or experience from someone who has operated at that scale for many years to bring that "what good looks like" perspective. That is a good reason to go out, and the team won't feel bypassed because they recognize the need for that outside perspective.

Paweł: In our organization, we announce every role internally first. But some people don't want to be leaders. There's a tendency to think a good engineer should become a manager, but that can backfire.

Ed: You're touching on a critical point: "it takes two to tango.". Never force a management role on someone who doesn't want it. You will lose a great person and end up with two vacancies to fill instead of one.

Paweł: Andy Grove said there are only two reasons someone isn't doing their job: they "can't" (no skill) or they "won't" (no motivation). It's a simple way to look at it.

Ed: I'm going to steal that quote, Pawel!.

Paweł: How do you assess if an engineer will be a good leader?. First-time leaders often don't know if they will enjoy the role because they've never tried it.

Ed: Let me put you on the spotlight, Pawel: in your experience, what are the key differences between a good leader and a bad leader?.

Paweł: In a single sentence: an individual contributor thinks about their individual impact; a leader thinks about how to increase the impact of the team.

Ed: A hundred percent. That's the first thing you use to measure them. Second, a leader must be able to gain hearts and minds. You can tell when someone has credibility—when they talk, people listen. If people already go to Pawel for suggestions or input, he has leadership qualities. You can elicit feedback from the team. Once you make the appointment, you can't just let them be; you need a "social contract" to give them coaching and support to develop those new skills.

Paweł: It's a significant transition. Do you see people going back and forth between leadership (people management) and individual contributor roles?.

Ed: Definitely. Senior individual contributors are leaders in their own right, as they exert influence on the outcome. 30 or 40 years ago, there was a stigma if you "stepped down" from management to be an individual contributor—people thought you must have been terrible. Nowadays, I don't see that. People have reached a level of comfort with what makes them happy. They might do people management for a few years, learn what they want, and then repurpose their career back to an individual contributor role. It's a perfectly valid option today.

Paweł: That's also a way to keep technical knowledge up to date. How do you keep a culture of experimentation alive while balancing the business need to build what is required today?.

Ed: Companies like Google or Meta do this well by ensuring innovation opportunities are available to everyone, not just an "innovation department.". There is always a tension because value is often measured only by features released to production. As a senior technical lead, you have to diffuse the "we should be doing something better than playing with toys" way of thinking. You should link innovation to real business needs. For example: "We're experimenting with React Native because we think we could consolidate our Android and iOS teams, but we don't know how it will integrate.". You link a business outcome to the investment. Also, make sure your roadmap reflects that. It shouldn't just be "playing with gadgets"; it should be a time-boxed play with a specific charter to see if there's a better way to achieve an objective.

Paweł: I love the measurement of outcome. Innovation naturally involves failure. Is it difficult to sell that to an executive team that might treat engineering as a cost center?.

Ed: There is always a challenge regarding "opportunity cost"—what are we *not* doing while we experiment?. You need to explain this in a language they understand. If I mention "on-the-air updates" for React Native, I might lose a non-technical board. If I explain that we are currently duplicating effort across two codebases and want to consolidate to become more efficient, they can relate to that. A POC (Proof of Concept) allows us to measure the potential cost savings.

Paweł: No conversation in September 2025 is complete without AI. What is your experience with AI tooling in software engineering?.

Ed: I remember when the internet was new—that was transformational. Companies that jumped in succeeded; those that didn't disappeared. Cloud computing and smartphones were the same. AI is the next truly transformational technology. We finally have computers that show a human degree of intellect. The downside is the hype—people claim it will solve everything, and not many people fully understand how it works or how to use it in their organization. But specifically for technology, I am seeing real changes. Modern LLMs can go through code and pinpoint bugs or performance bottlenecks. I use them for open-source projects like llama.cpp. It fills me with wonder and concern. It *will* displace human beings from some positions, just like the Industrial Revolution did. We all need to learn how to use AI so we don't end up displaced. Many companies that tried to replace their entire customer support early on had to "eat humble pie" and backtrack, but the technology will continue to improve.

Paweł: I see many engineering managers coming back to coding because AI makes it easier to tinker. Software engineering became very complex—complicated pipelines, Kubernetes, containers. 15 years ago, you just copied a PHP file to an FTP server and saw the change instantly. AI removes a lot of the "accidental complexity" or boilerplate, making it easier to get to the core of a prototype. But as you approach a "scale-up" with a larger codebase, do you see those productivity gains diminishing?.

Ed: Part of the problem is overinflated expectations. Gartner's "hype cycle" is real. AI isn't replacing all developers tomorrow. In 2022, the results were laughable. In just three years, it has gone from a "toy" to something that can code with human-like depth. Technology evolves—it gets better and cheaper. Whether we see AGI (Artificial General Intelligence) is a topic for another day, but we are seeing exponential improvements. There is some concern that we won't have enough data to move to the next "plateau," though.

Paweł: Training costs are skyrocketing—GPT-5 might cost $2.5 billion, and GPT-6 could be $20 billion. That's a country's budget!.

Ed: But look at "DeepSeek"—they claimed to produce a model with state-of-the-art capabilities at a fraction of the traditional training cost. Who is to say there isn't another breakthrough on the horizon?. And who cares if it costs $20 billion to train if the return is measured in trillions?. It's a no-brainer investment. Right now we are only looking at the cost in money, but there is also a cost in electricity and resources.

Paweł: Everyone agrees that future models will do the work of software engineers; the question is "when.". Do you anticipate the work being affected in the next 10 years?.

Ed: As we know it today, definitely. Does that mean the need for humans in engineering will disappear?. I don't think so. When tractors were introduced, they replaced the person with an ox, but created a whole industry of people building and maintaining trucks. The field will be dramatically impacted, but it won't disappear because these models cannot "invent" yet—they are sophisticated mathematical algorithms. Genuine discovery or that "inspirational spark" isn't there. The difference between having a job and not will boil down to how well you use AI as a tool to increase what you can do.

Paweł: Some people say generating code is easier now, but reviewing and integrating it is just as hard. Do we need different processes?.

Ed: A few months ago, the joke was: "It's great that I can 'live code,' but now I have to 'live debug.'". The practice will definitely change. We will need to approach problems in a way that makes them more accessible to the machine so it can give the best result. That process of "framing requirements" is where the transformation will happen.

Paweł: That's what software engineering is: translating a problem into a way a machine can understand. Software engineers are best positioned to handle AI.

Ed: Exactly. Before, we knew we were superior because the machine was "stupid" and only followed precise instructions. Today, a non-technical person can interact with an AI agent to create a decent application. 15 years ago, it took days to code "Flappy Bird"; today, I can do it in 10 minutes. It's exciting, but painful because of the human impact—it is displacing people.

Paweł: You mentioned the "billion-dollar company with only one or two humans.". We see smaller teams reaching high valuations, mostly in the US. Why is that?.

Ed: It's a tricky question without touching on politics. Part of it is access to capital—success feeds success in the US. Also, the US has a culture of risk-taking—the "go big or go home" Americanism. The narrative that success is up to the individual is woven into their fabric. That attracts smart people from India, Europe, and China. Europe has more concern about risk-taking, and the cost of capital is higher. Until that changes, the favoritism toward the US will continue.

Paweł: In the US, you make more money, but the cost of living is also higher.

Ed: Yes, you could make $600,000 in the US, but you might spend $700,000 to live there. With remote work, the dialogue has changed. You have to look at both compensation and cost of living before deciding.

Paweł: We have plenty of material. Any key thoughts for a CTO joining a new organization?.

Ed: Be humble, be human, be purposeful, and be collaborative. See technology as a means to an end. That end is driven by product, marketing, and finance as well. Establish deep relationships with your colleagues, because together you rise, divided you fall.

Paweł: Thank you for the conversation, Ed. I hope we can talk again in the future.

Ed: I would love to, Pawel. Thank you very much for having me.