Episode 45: Jean-Michel Lemieux
Episode Transcription

Mik (00:00:06):
Hello, and welcome to the Mik + One podcast where I sit down with industry leaders to discuss the Project to Product movement. I'm Mik Kersten, Founder and CEO of Tasktop and bestselling author of Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework.

Mik (00:00:27):
I am thrilled to have Jean-Michel Lemieux joining us on today's episode. Jean-Michel is an author, investor, and advisor and former CTO and SVP of Engineering at Shopify. He has been working in tech for 27 years, initially building software, and building teams, and companies. He was a founding member on the eclipse platform and open source team and has led the engineering organizations at Atlassian and Shopify. I have found Jean-Michel's wisdom to be second to none, and I have learned so much from him over the years. So I'm just thrilled that he could join us today to talk engineering, architecture, scale, organizational design, and flow. So with that, let's get started. Jean-Michel, welcome to the project to podcast. It is so great to see you after all these years of us having had a chance to work together in parallel universes, how are you doing?

Jean-Michel Lemieux (00:01:19):
Great to be here, Mik. Yeah, I think it's been 20 years that we've known each other and haven't hung out like this. So thanks for inviting me and a great idea.

Mik (00:01:28):
Yeah, no, I was reflecting on it and I think I probably spent two years getting to know you entirely through your code, before I ever saw your face, I ever saw you in person, and it was all of that code in eclipse itself where I think as you remember, I was trying to make your code do things it wasn't exactly meant to do, while I already... There was so much beauty and elegance to what you created there. It was probably some of the most extensible and cleanest code I saw in all of Eclipse and across those 60 million lines or whatever it was at that point, and as I was trying to implement, as you remember, Mylyn in the aspect of programming stuff and everything else. So just tell us a bit about how your journey with software started.

Jean-Michel Lemieux (00:02:04):
Yeah, well, that's a great story. I mean, I do remember probably through Bugzilla as much as in code, because I think in the early days of eclipse, as we scaled and opened up and open sourced, I remember one of the interesting things, maybe a bit of a tangent to start our conversation, but as a software developer, you spend so much time trying to make things work. You're like, man, like you're in school, you're like, "Oh man, I've got to make this search algorithm work." I've got to... You got to make things work. And then, I think a really big lesson around Eclipse which was interesting is, when you build

software with others, a lot of time, it's just figuring out how do you communicate? So I think we spent 60% of our time in Eclipse once we open sourced it in Bugzilla, talking with people. And I remember your team were... and you were doing something really interesting where you were trying to use our APIs in ways we hadn't imagined. And I think that's why I was so fascinated with what you were doing with Mylyn. And I think wasn't it called Mylyn at some point [crosstalk 00:02:55] Mylyn-

Mik (00:02:56):
It was but then the Eclipse Foundation realized there was a trademark issue with DuPont.

Jean-Michel Lemieux (00:03:02):
[inaudible 00:03:02] I love Mylyn, it was great. And I think my fascination with what you guys were doing back then was, I always liked people doing crazy things with our software and you were probably one of the craziest in terms of just trying to do things that were like, we had actually didn't know how to do it, right? And I think that's why we ended up hanging out a lot back then.

Mik (00:03:20):
Yeah, exactly. And I think the... it was, it was amazing trying to exactly change the interaction model to actually be more around the collaboration around those Bugzilla bugs, those tasks, and connecting that up to code.

Jean-Michel Lemieux (00:03:32):
And I think for those who don't know, what you were trying to do with Eclipse, so Eclipse IDE platform was, you're like, when you're working on code, you should have a lens onto your code that gives you context awareness into the tasks you're doing, not just into the global, right? Because I think people get overwhelmed when you're coding with just having too much craft around you. And then, so basically, if you think about what Mik was doing with Mylyn was, hey, can I apply a filter on every part of Eclipse that I can change on demand? And for that, I think it was a great idea. I think you were probably onto something, I think we just don't know how to implement it yet so. But anyway, that was a long time ago but it was fascinating. Anyway, that's where Mik and I met, in Bugzilla and in Java development 20 years ago.

Mik (00:04:18):
Yeah. And all our collaboration was through those Bugzilla bug, right? So what today are Jira issues and other issue tracking issues for people, that's where, I think it was probably a couple years of collaborating through that before we actually met face to face at a conference, so.

Jean-Michel Lemieux (00:04:29): Exactly.

Mik (00:04:30):
And it did definitely, I mean, for me, it was super interesting because I think at the same time, we were very focused on understanding software architecture, you and Jeff McAffer, I read your book at the time. I thought it was wow, this is how you build these extensible platforms. Some of the extensibility back then that the two of you created are I think a lot better than some of the frameworks we see today, and really established some of those foundations on extensibility, on modularity, on components. So having be reusable. I think that, the real instinct to me at that time was just, it was not just about the code, right? How much we were actually collaborating and how much the... what we now called value streams had to do with that flow of work, that collaboration, through the issues, through those back then Bugzilla bugs, how that was a first-class structure. And I think we've seen a lot of that, right? Because after Eclipse, your work took you actually, I mean those Rational and IBM, but then to Atlassian where you were very much working on defining how issue tracking and how that collaboration across teams works, not just at the level of the code, which a lot of us grew up around, but actually at that level of collaboration. So tell us a bit about that, what you learned through your days of Eclipse and IBM, but then actually, what took you to Atlassian to become the VP of engineering at Atlassian?

Jean-Michel Lemieux (00:05:44):
Yeah. Maybe just to back up a bit, so, I had a pretty, I guess, boring entry into software development like, went to university, told my parents I'm going to computer science, because my guidance counselor told me to, and I didn't have any better ideas at the time. And back then, my parents knew no one who was a software developer, they didn't exist. So they're like, "Okay, I guess." Wrote software for a while, and I think in the early, I could say early days, but I mean, it's not like we're that old, we're 50 now. It was actually really interesting software we're building. And maybe there was a bit of a luck, but graduating university and then in like '93 to '96 was both the invention of the internet. But also, more importantly was the invention of telephony as we know it, right? The digitization of the telephone networks was probably the first place that I guess modern software was being deployed and people were paying for it. There was a bit of an arms race to digitizing the phone networks. And if you think about the kinds of software you need to run in phone networks, it's actually pretty complex, real-time systems distributed. I'd say, I'm going to use the word cloud now, but there's computers every like... Right out school, I was using everything I learned, right? Like distributed system, TCP/IP, non-blocking sockets, moving, doing fraud detection on phone networks where, because as soon as they were rolling out new technology, reminds me a bit of crypto, like first thing that happens-

Mik (00:07:11): Really?

Jean-Michel Lemieux (00:07:11): ... is people [crosstalk 00:07:11]-
Mik (00:07:11):
I didn't notice you were doing fraud detection too, wow.

Jean-Michel Lemieux (00:07:15):
We're doing fraud detection. We're basically doing, yeah, because people were doing reverse call collect, fraud and [inaudible 00:07:21]. So, because it was just, everyone who's imagining software, just enabling all these new use cases, the problem is you enable them and you just don't know all the back doors you left open, right? So there was a bit of a... it was just a fascinating time. And also, the phone networks were busy. So the software you're deploying right away, you're getting millions of phone calls. Anyway, for me, it was like just, the Netty was really, really interesting software to write. I found it intimidating, it was like... I was in Bell Canada's central office, I had between 12:00 and 4:00 AM to upgrade our software. That was what it was like, and it was like fascinating. Maybe I got lucky, people don't talk about...

Jean-Michel Lemieux (00:08:03):
People talk about Netscape, and web browsers of Microsoft. But a lot of us who were building out, I guess, more of the telephony space, it was just really, really fun software. I'll get to why that helps Shopify because it does, there's a connection to Shopify later. So I guess, a bit boring, writing software doing that. And then, I think because the kinds of software were running in telephony space, it was complex. And I think, at the time, in Ottawa like Nortel and BNR were like, why are we always searching for different... there must be a better way to write software because it was hard. It took a lot of time, a lot of money, and I mean, we weren't that good at it so it was pretty buggy and this was mission critical software. This was not a social network where, if you have too many likes, you just stop accepting likes, these were phone calls.

Jean-Michel Lemieux (00:08:49):
So I think there was a bunch of really interesting spinoffs that came out of the telephone industry around how to write better software. And one of those was in Ottawa called ObjecTime. And I don't know if anyone remembers ObjecTime, we were a small starter of a hundred people and Rational Software ended up buying us. So that's how I got into like the, I'd say almost a software methodology space of Rational buying this startup in Ottawa because we came up with a... it was like realtime. It was called ROOM, Real-Time Object-Oriented Modeling, which is a way of defining realtime systems, and modularity, and encapsulation, and just making it easier for you to build distributed systems, which was, again, the hard part in telecom. We got bought by Rational and then you get sucked into just thinking about how to build software for a lot. So I think it

was really in some ways boring, but also interesting of just writing software and then being bought by a company, Rational Software, who was supposed to be a bit of the teacher of how to write good software. And then, ran a bit of time in there. And then, Eclipse was a really natural evolution to, okay, we've been doing this for a while, let's help developers write software. So, that's how I got into Eclipse having, I guess, written a lot of software, knowing how hard it is, and I think trying to make it... trying to maybe advance the space of just making it easy, the right software. That's up to eclipse.

Jean-Michel Lemieux (00:10:10):
And then, you're with me with Eclipse, which was both humbling, because I think, I guess for the first time, we wrote a platform that was open, and we built a community around. And I think that was really good, but it was really good software. We had really good principles and I think that was fun. And I think you participated in that, and I think after that, at some point I was like, I always found with Eclipse that we could have created our own company. We could have... SpyNote had a amazing... built a billion dollar business out of this. Because I think we, it was probably one of the funnest teams I've been apart. And I think me going to Atlassian was, I was like, "I'd love to build great software and be a businessman how do I do both together?" So I think going from a bit academia of software and open source, I was like, "How do I build a business?" And then for me, that was my next chapter of like, it looks like it's really hard and all the telecom companies that I grew up with, Blackberry, Nortel, they had all died because of other things. And for me, there's a curiosity of going, man, it feels really hard to build companies. And so, from software to building companies is that transition of why I ended up at Atlassian.

Mik (00:11:15):
Well, I actually think we would've been better off if you had SpyNote, a company. And it was back then OTI, Optimization Technology International was doing this for a while around small talk as well. But if we'd had that for the Java ecosystem, fundamentally we'd have better tooling today. But I guess notably what you started seeing, actually what Rational was seeing is that the way that we manage work, the way we track work, all this issue tracking was becoming a really important business model and then system of record in its own right. So, spent all this time coding and learning about these issues at scale. I'm sure we'll get back to when you talk more about Shopify, that helping others code and building this extensible platform. And then yeah, tell us more about how then that shifted into moving to Atlassian.

Jean-Michel Lemieux (00:11:56):
I think I was just... It's not that I knew how to write code, but I think one of the interesting things around Eclipse is, I became a manager, we built some other technologies. And having seen businesses die around me, I was like, "The combination of building great

software and building great teams together, I think is the hardest thing to do, right?" And how I think the best companies end up flourishing, and the best communities end up flourishing is those who can do both. So Atlassian was a company that I think we're kind of competing against a bit, like they were basically our competitors to Rational who had been the big developer tool company out there with bug tracking, and version control, and all that. And I saw these two young dynamic guys in Australia doing it. It was November 12th, I still remember it. I was having dinner with my wife and I said, "I'm going to send an email tonight that could change our lives. I'm going to send an email to Mike and Scott who are the CEOs of Atlassian and say that I think I can help them out. And it might turn out, do you want to move to Australia?" She's like, "Yeah, whatever, just..." Anyway, she thought I was joking. I sent the email out and before you knew it, we moved the kids to Australia and was leading the engineering team at Atlassian, and way over my head.

Mik (00:13:08):
I remember those days. So, yeah, tell us, it'd be great to hear more about that. How did it start? What were the main things that you were trying to achieve at the start and yeah, how did you get over your head?

Jean-Michel Lemieux (00:13:20):
One of the things that I'd not appreciated as much about the Eclipse team was that we kind of grew up together. We went from a small team to a bigger team, and I think when that happens, you end up having these super high bandwidth conversations with your peers because you have this shared context of just so many learned experiences that you don't write them down, you don't... But you build this shared context that's like so valuable. I land in a new continent, a new company, like the tech stack, I kind of like, "Oh it's Jo and Java, but it's more around the people, the culture, the history. And next thing you know, I'm like, "Why can't people read my mind? Like my old team, and I see that in a good way, like, we'd read each other's minds. We'd fully finish each other's phrases. And I think landing in a new company where I don't have that context and they don't have the context on me, was real... Honestly, it might sound really simple, but it was like, it threw me for a loop for like the first year of going, "Holy shit, I've got to build all this up again." Build trust, build context on technology, build context on culture, before I can get back to having these high bandwidth conversations that I just left because I'd worked with the same team for 10 plus years.

Mik (00:14:33):
Right. That's so interesting because you knew that tech stack, I'm sure you got your head around the architecture really quickly. So it really was the team structure and dynamics and leadership that was [crosstalk 00:14:43]-

Jean-Michel Lemieux (00:14:43):
Absolutely. And I'd said that the culture around the company, just what they've been talking about that I wasn't purview to, what decisions that they made, that I haven't met purview, and how do they work? And when I showed up at Atlassian, I think, one of the biggest projects that was ongoing that had just started was moving to the cloud. So this is 2010, moving to the cloud, how do we do that? And that was obviously fascinating back then I think, was a really good decision which is, you can either change your entire tech stack or you can put VMs in the cloud in a very efficient way. And I think at that time we decided to put VMs in the cloud very efficiently.

Mik (00:15:22):
Yeah. What was it, racks, just VMs running on Rackspace single tenant back then when you started?

Jean-Michel Lemieux (00:15:26):
Yeah, it was single tenant. I forget what was... There was these micro VMs, it wasn't VMware, it was like our own kind of managed VMs that we could run and spin up. And actually, it was a small colo provider that we were partners with. It wasn't in Rackspace, and yet we'd spin up single tenant instances for, probably we did that for like five, six years.

Mik (00:15:47):
Wow. I know single bays still doing that today, but you saw [crosstalk 00:15:52]-

Jean-Michel Lemieux (00:15:52):
I mean, it's fascinating because I think the... Let's go to cloud software because Atlassian was probably my, in some ways, a bit of an introduction to me, because we've been writing software, but I guess not in the cloud shape as much, but you have to write software very differently, right? To be multi-tenanted and all that fun stuff. And I think the entire planet now has... We wrote software in a way that it wasn't cloud native and I think AWS has been really good and they realize really early that people moving into cloud, aren't just moving. It's not moving, it's basically rebuilding their platform. And helping people do that is extremely valuable. At Atlassian, I think a mistake I probably made was delaying that architectural change.

Jean-Michel Lemieux (00:16:40):
I think we were in scrappy, this is way pre-IPO at Atlassian, we're like, "Let's get this thing running." Hard to say if it was a good call or not. But I think, if I had to go back in time with a time machine, I probably, I think we should have taken one or pieces and split them out, like cloud native built them. Not the whole stack, but say, "Hey, why don't

we make search?" Okay. Take a feature and go, "Hey, we could build search in a cloud native way, cross-platform, multi-tenanted, search really is, usually has a good API towards it. It's like you can suck the data into it, index it, multi-tenanted, and then bring it out. And even we can even still using Atlassian search as we've seen, but we could... I think I would've taken a couple of those components and made them multi-tenanted, made them cloud native, have them plug into the single tenant version of Jira, Confluence, whatever we're running, and then I think bring that culture into the team earlier versus I think we... It took a bit longer to do that, so.

Mik (00:17:39):
Yeah. And I think when we get back to what you did definitely at Shopify, I think let's get back to this topic because I think that is some of the key learnings. It's great getting these out right now for others to hear. And of course, we'll send resources to... I know you've been documenting some of this through Twitter, through this book that you're writing. We'll get back to that, but I'd love to just dig somewhere into that because I think so many people out there are now faced with similar decisions that you were making back then, right? Which is how quickly to move their application portfolios, their workloads to cloud. I'll just never forget. I remember actually, where we were standing when you told me, "Oh, I'm going to actually fork the code base of Jira and beyond. And I thought, this was such a crazy thing for me to hear you say, because all our... and this is many, many years ago now, but all the mutual work we've done, were really around the extensibility and modularity. And I thought, wow, like that just seems so way wasteful for Jean-Michel to be splitting the code base into the cloud code base and the multifamily code base. And I feel like for the following years, I actually learned why that was such an important move. And of course now, you've got more approaches where you've done that in a pocket and split out some services like search and gotten that DNA into the operational DNA, to the teams and so on but-

Jean-Michel Lemieux (00:18:50):
And that's what the fork actually ended up being, is building some cloud native services and saying, "We don't have to make them on-prem, just like a lack of a baggage. I think there's a really good saying, which is, duplication is better than neuronic abstraction. And I think that's probably at the core of why the fork at that time, was the right step to take, I think, for the long term architecture of the Atlassian platform.

Mik (00:19:17):
Yeah. And I think, I still see this as being very counterintuitive to so many of the enterprise architects out there. That duplication is better than the wrong abstraction. Because again, it's based on the lot, what we learned through computer science, or through our training or coding, is it just seems wasteful. Take us through that, through how you did that, through what you learned and maybe what you would do differently.

Jean-Michel Lemieux (00:19:35):
Well, I wasn't there when it ended, I think Atlassian just end of lifed I think their server platform, which was the single tenanted version of [crosstalk 00:19:44]-

Mik (00:19:43):
But that, by the way, just blows me away, that the decisions you were making 10 years ago actually had impact on news I saw yesterday relating to Atlassian stock price and how they're handling the fork of server. So the impact of these architecture decisions and how they're executed have these massive consequences in terms of actually the value, the growth of, and the way that companies can finally support their customers.

Jean-Michel Lemieux (00:20:07):
Yeah, I don't know why it's that surprising? I mean, if you think about it, like as an engineer, you really don't know if you've done a good job until five years into the future. Which is like, if you're writing the code is like, did it age well or not? Basically, did it get us closer to where we're trying to go? Like, you don't know those right away, right? We're building things that are into the future a lot more so I think that's what's hard being an engineering leader is, you're basically making it bunch of bets into the future, and you won't know until for a while, but I think the bet that at Atlassian we were making at the time was, a lot of things you're trying to do technically is about improving your velocity of evolution, right? Like softwares, it's almost like a biological system. It's not fixed that it evolves, so we're like, what's the quickest way we can evolve this thing? And we'd realize that, as I said, we did VMs, we did single tenanted, we tried to evolve it, all Comix together and it was too slow. We had to go build some cloud native things. And without the handcuffs of saying, we're going to have to build two versions of it right away, or figure out how to make it work.

Jean-Michel Lemieux (00:21:13):
There was a cultural burden that was away and it's sped up velocity, right? And then, so I think that... I mean, that was the crux of it which... I think, and if you think about it, every time you evolve your software, you end up forking something. Like at Shopify, we're going to add a new tax system into it. Well, we keep the existing one, we create the other one in parallel, we route traffic to it for a while, we make it do 20% of the use cases, we test it out, and at some point, we merge it back. Forking and emerging is how we develop software forever. And I think the Atlassian was just a bigger one that we had to talk about publicly because just managing expectations and their customers who are like, why am I not getting features? So I think, as part of building a company, I think selling the vision of where you're going, ended up being important. That's why I think there's a lot of emphasis on it of just letting people know Atlassian, we're building for the future. And do you want to... it might be bumpy, but come on the ride with us.

Mik (00:22:06):
Yeah. Well, I think that's what's so fascinating about the Atlassian one was, it was visible, it was so significant. And the decisions that you made resulted in billions of dollars worth of value being created. And if they'd been made differently or you'd wait another three years, or something of that sort, it could be in a very different spot.

Jean-Michel Lemieux (00:22:23):
Yeah. I mean, a lot of people... This was not my decision really. I mean, it was all of us together. And the founders were, Mike and Scott, really pushing us. Mike and Scott, if I learned anything from them was their incentive questioning of like, could we be better? And how do we get better all the time? And I think they made the environment to this could be possible, make these big bets possible, so.

Mik (00:22:51):
Awesome.

Jean-Michel Lemieux (00:22:51):
And they kept at it right. My replacement at Atlassian, who actually just left, Shree. We didn't overlap, I think he started a year after it, but I know we talked a lot when he joined about keeping that going and, the in sets of investment we have to make. And maybe there's a lot of people who aren't in this situation, you have a lot of paying customers, there's a lot of pressure internally to just keep going the way you're going. I think Microsoft's probably a really good example of a platform level change that they waited 20 years too long to make, right? Which is, think about how does every Microsoft product store files on the file system? Cool, okay, the cloud doesn't work like that. So I think like Microsoft have been rewriting their, I mean, their client server architecture for a while. And I think one of the things that their current CTO told me was, again, their biggest mistake was delaying that a bit too much. They had to fork, they had to go, "Okay, we have to build cloud native again." and cloud native is storage UI. How do we do UI? And where do we put the business logic? Those are the three things you always have to think about.

Jean-Michel Lemieux (00:23:52):
And in a very fat client server app, a lot of that was on the client. And we're like, "We have to rewrite that." So actually, the conversation I had with the CTO of Microsoft at the time was just very impactful in that, as engineering leaders, we're all in this spot where we're like, "When do we... How do we evolve our software? And how do we know when it's the right time to make a big fork versus a small fork?" And I think that's... I mean, there's no answer, but I think great engineering leaders at any point in time have a bit of a map of going, here's where I could be looking for a step change and how to invest in that. So, and I think that was a big, I guess that was a lesson from Atlassian

that I took into Shopify. And that was very, very high in my mind right now, which is what architectural changes we have to make. And I think my brain now is very wired to doing them sooner than I would've in the past.

Mik (00:24:42):
Right. And so I think it's interesting the way you put it, where the CEO's of Atlassian created the environment to empower those changes. I love what you said that is that, the changes were about improving the velocity of your evolution, right? So to me, one, I think one of the biggest learnings and it gets into the Flow Framework is, we have to architect for that velocity for each value stream. We can't create this one single platform that just becomes a dependency on everybody, is that duplication's okay as long as we're increasing velocity, right? I think it's a lot of what we've seen. And one of your colleagues, Geordie Henderson, he and I talked about this on the previous podcast, right, is that, is just optimizing around that velocity and the velocity of the evolution, how much quickly you can get better. And of course, being able to measure that, are we getting better? So, and I think that fundamentally is what I've seen helping people breakthrough the brick walls they have on, never allowing duplication. They're always trying to centralize everything. And again, they are trying to overspecify things rather than providing that autonomy to each of the value streams, to go faster and figuring out how to go faster. So-

Jean-Michel Lemieux (00:25:41):
I mean, to that point, there's a bit of a mental model that... I forget who, I'm probably influenced by a bunch of people on the internet around this. But to your point of, when you're writing software, what kind of software do you write for what features? Do you have to sweat the details of every line of code do you write? How do you decide? And I think the best Bender model I've come up with is, you have to think about all the projects that you're doing around. Either it's a platform project, it's a feature project, or an experiment. There's three kinds of things that you're doing generally. And I'm going to take out all the nuances, but just three buckets, right? The platform work you're doing, I'd say is 50% of what your roadmap is. And those are things that you've extracted from writing a lot of software, and you've found out, you found some primitives, right? You've found some shared primitives, you've found some abstractions. You've found some places to put it into your code, where you're going to enable a lot of things into the future, and that's half of your roadmap. And those are things that, again, like platform work has to enable new things to happen in the future.

Jean-Michel Lemieux (00:26:42):
They have to... I'm adding this concept into our core thing of [inaudible 00:26:47], that's going to be able to be used in all these different ways. Cool, that's a good investment. And that code there has to age really well. But then 40% are features where, you know

what, like duplication is probably fine, right? It's probably time to market that matters, it's probably... The risk there is actually understanding a domain, actually building things that care, that matter, right? Like in commerce, actually building a checkout that's going to be useful or building whatever, like a loan system, whatever. Those features there, you can't overengineer them because you don't know enough yet, and they're providing value. And then 10% are experiments, which is, you better be ready to throw it away. And so I think if you don't identify the kinds of work that you're doing, every piece of software you're going to write's going to look the same, or you're going to put the same effort to, which is not what the value you're getting from the business. And I say value from the business almost, I sound like a point here manager, but every code that you write has a purpose, and it's never the same. And you have to know what your purpose is. And then when you know what your purpose of that software is, then you can apply the right tools to writing it. And I think the biggest mistake I've seen people make is, they think all software has the same purpose so they apply the same tools to every piece of software.

Jean-Michel Lemieux (00:28:01):
And then it creates slowness, frustration, and even the worst thing is, you've got different, I'd say, types of engineers who are good at different types of writing software, and they put the wrong people on the wrong software system. In a nutshell, I completely agree with what you're saying, and I think you have to recognize you're writing different software. And then as a team, as a company, give names to those things and have dialogue around what are we writing this for? And then that's going to help you decide how you write it.

Mik (00:28:29):
Yeah. And I think Jean-Michel, what I love, and I think you have documents on your notion of platform investments in the book that you're writing. And [inaudible 00:28:38] found on your Twitter feeds, I want to make sure people don't miss this. People would be shocked I think that in terms of a lot of the way that portfolios look today in larger enterprises, the fact that you're saying 50% of your self portfolio, those team's efforts should be going into these platform parts, right? Where one of the main things that we see by the way, when we measure companies' value streams, is that there's some notion of greenfield, and experiments, and everyone's... Enough people [inaudible 00:29:05] startup up and they're doing bits and pieces of that. So I think that's understood somewhat. The vast majority of where the development is focused, where you've got all the thousands of staffs at these larger companies actually have to do on those feature parts of things, right?

Jean-Michel Lemieux (00:29:21): Correct.

Mik (00:29:21):
And the under investment that we see in-

Jean-Michel Lemieux (00:29:23):
The typical distribution is 10% experiments, which I think is... People do it because they've read all the books they're afraid of failing if they don't find the next moonshot or... And then it's 80% features.

Mik (00:29:34): Yeah, exactly.
Jean-Michel Lemieux (00:29:35): And the 10% platform.

Mik (00:29:36):
That's right.

Jean-Michel Lemieux (00:29:36):
And then, they complain developers leave.

Mik (00:29:39):
Yep.

Jean-Michel Lemieux (00:29:40):
Because the platform work actually, a lot of it has to do with making sure the developers can actually write code in your company. And probably 15% of the platform work is understanding what kind of software you're writing, build tooling infrastructure, all that stuff that developers need to actually write code is completely under invested in most companies. And so it's nutrition problem. And then the other thing is, they don't get any long term velocity. They get short term velocity because you learn a lot by building those 80% features. You learn about where you're duplicating code, where... And sometimes, it makes sense, sometimes it doesn't, and then you're learning about abstractions that can enable more things.

Jean-Michel Lemieux (00:30:17):
And if you don't go and bring those learnings back… and I think have a diagram in that chapter of the book around those three types of work like platform, features and experiments have feedback loops between them that are fascinating. So the feedback loop between platform features and experiments is platform enables more features and features enables more experiments. And then in the inverse, experiments informs the features and the features informs the platform. So the minute you have an imbalance there, I hate the word debt, but you're just creating like a long term non-efficient software process, right? Because you're not using the feedback loops and you're not learning, and you're not taking those learnings and doing something about it.

Mik (00:30:58):
Yeah, exactly. I think to your point, the 80% being on features as what we're seeing, because of course we get to measure all these flows, right? So the two biggest flow dysfunctions that we see across the enterprise customer base that we have is, first of all, too much load on the teams doing the features because they're always behind, and because they're always... The bottom line is if we actually look at their flow efficiency where they're blocked, it's from, they're waiting on platform components, they're waiting on common views of customer data, common pipelines. And you think, the cloud services that they needed to layer over that aren't there, there's other issues with infrastructure. So I think again, you nailed it and really, the thing you speak about so clearly is shifting to that is actually going to make everything go faster. And getting to that right kind of balance, and it's a really big difference if you're talking about thousands or tens of thousands of developers, the difference between 50% and 20%,

Jean-Michel Lemieux (00:31:49):
It's a big number. And I've had to, I guess in the past, I did not protect it enough. And now, I mean, Tobi knows. I mean, Tobi's a... he was the CEO, who's still the CEO of Shopify. He's a developer, he's an engineer, but he also runs a company and he would often... He'd go, "Hey, those people are working on platform. Can I move them to this?" And I was like, I just want to make sure he understood all the things that this is enabling in the future. And so there was always a good... You have to kind of... and show the value. I think, you have to run the platform like it's a product as well. Like it is a product, and I think one of the things I learned as well, just in terms of lessons, as an engineering leader in a bigger organization, I always found myself as the head of product for the platform. And that meant that I have a responsibility to manage a product, which is, I've learned from product managers that they're really, really good at being organized. And I actually wasn't that organized. And I wasn't that...

Jean-Michel Lemieux (00:32:42):
Well, I wasn't that good at measuring outcome. So in the platform I'd say, five the most important features of every product are owned by the platform teams, uptime, scalability, performance, extensibility, and maintainability almost, right? Those are the five features. So I would make sure I had a product team within the platform that we're measuring those things at all time, and showing those as value, and actually showing this to our marketing team going, "By the way, you're only looking at that 40% bucket of work to promote as features, but we've got these five other features here that have to be inherently built into the platform to be successful and they're useful. And it's funny because once I actually had to re-bring this mindset into Shopify, because I think I ended up on the backseat of just doing what products said. I was just actually, no, like I personally own these features now and it'd be funny... And you do interviews like, what

do customers care about? They care about uptime, they care performance, they care about extensibility, about especially in the enterprise, they care that your software can be malleable and molded. And I was like, you have to invest in, a bit like Eclipse, this all came from [inaudible 00:33:56]. You have to invest in primitives to make your software sensible.

Jean-Michel Lemieux (00:33:59):
A lot of teams I find, as you said, if you measure their velocity, you'll see it, they're all blocked on learnings of like, if I only had a workflow engine, man. Well, then build one. Ah, but who builds one, and it's hard to, it's hard to figure out who's going to do that, and I guess as a CTO, what's great is, as a product owner of the platform, it's a fascinating and fun job than to just, to take it with that discipline, right? And take it seriously as not just technical work to keep things running. But it's actually like it's probably some of the most valuable features of your entire business are owned by careful investments in those areas.

Mik (00:34:37):
Yeah. So this is what I'm finding so fascinating right now. Some leadership in these large organizations is recognizing that the digital platforms they're creating are the most important assets, but that's not common. I think you've had the privilege of working for CEOs who were developers, right? Mike and Scott were sick of Bugzilla, and so they reimplemented [crosstalk 00:34:57] some of that working style. Tobi, as you mentioned.

Mik (00:35:00):
It'd be great if you could just dig in it a little bit. I do want to get back to this point of the platform as product and ownership of that. But, you've effectively made the economic case for how leadership, the executive team needs to think about these platform investments. You've actually come up with a very prescriptive number, which I agree with, it's I think it's the same number that Jeff Lawson put on Ask Your Developer in that book, right? He actually said, 50% around 50%. And you've successfully made the economic case to the companies that you've worked with, to create that kind of investment. So how did you do that? Is it mostly through to the outcomes? Is it mostly through accelerating philosophy, accelerating learning, closing the feedback loop? Is there anything more? And for all those people who are now at 10, 15, 20% of platform investment, who are going to listen to this podcast and are going to go talk to their CEOs, any more tips you have for them?

Jean-Michel Lemieux (00:35:50):
It's a three-step... well, not a three-step process, but I think there's three themes that are going to help you. I hate the word, make the argument, but just promote that this is a worthy investment. This is like investment, like anything else. Right. And the first is a

mindset of, you're doing this work, the platform work is because the business needs it. Not because you're an engineer and you like doing this work, right. This is work the business needs, right? So everything you do has to be in the context.

Jean-Michel Lemieux (00:36:18):
Step one was always, I have to make the best use of my engineers. We're hiring these, there's not a lot on the planet, and we're fighting for them, and I cannot have a leaky bucket. I have to make sure at any point in time that my developers can get their work done, right? Okay. Step one. And no one's going to argue with that. They're like, "Yeah, okay." Well, how do you know if they're getting work? Well, you've got to put some things in place where they work waiting for code reviews is the... The biggest thing I see is just development environment set up, right?

Jean-Michel Lemieux (00:36:47):
How long does it take for a new developer to get the environment set up? And they're like, "Ah, yeah, but we have to stock, and we... ah." No, it has to be, you have to have basically a drop dead number that you never change at your company. And if it does, it opens a can of worms. And I've seen hundreds of companies that are like, "Yeah, we never," and then now it's actually two days and they have to do this, and they've got to plug into this other thing. And it's like that [crosstalk 00:37:10]-

Mik (00:37:10):
You mean the speed of [crosstalk 00:37:13]-

Jean-Michel Lemieux (00:37:12):
Can I get endeavor environment set up? Can I get a code, some change production. Like I said, that has to be like pick, I think we picked 10 minutes, right? Pick it, it has to be 10 minutes, okay? Bam. And then, okay, maybe it's 28 or, but it has to be [inaudible 00:37:27] and you have to keep it there every year. Okay. So that's theme number one of platform work. And it's really hard to argue against that, given that much money we're all spending on engineers. And how we know it's the hardest resource to get.

Jean-Michel Lemieux (00:37:39):
The second point is, and this is where I think CTOs or VP of engineerings have to put a bit of their product hat on, which is really understand where the product's going and promote, and again, look at all the features and go, "Man, we're seeing some patterns here, we're seeing some patterns and some of these patterns are more defined than others and let's explore them." And at a certain point in time, you're going to figure out for us to implement these well, there's some common abstractions, right.

Jean-Michel Lemieux (00:38:09):

And I think that that second bucket of what abstractions do we need in our platform, like new technology evolutions, or abstractions, that we need that are going to be hard. And as a CTO or VP engineering, you have to again, understand why you're doing it. And the tool I love to use there is literally writing up a bit of an RFC on, hey, we're going to be adding, let's say Shopify as an example, right? We're going to add subscriptions to the core data model of Shopify. And we don't have to put it there.

Jean-Michel Lemieux (00:38:41):
I remember I actually wrote a draft of the RFC just because it was a big debate, because it was a hard thing, but we're going to add it at this spot in the platform. I had a team with me and we built out a time horizon of like, if we do it this way, what are the pros and cons, and what are all the features in the next five years that will be able to be built off of this. And then so basically, it was almost like a feature investment plan based on some of this platform work being done or not. And that took about six months of work, of just like, I had one or two architects trying to really understand it. We looked at... That was a really good dialogue because then I was like, "Hey, this is value that we're going to get." Now it's going to be hard to put it in this spot, but it's the right spot. And then the third bucket is measuring those five product features that your platform enables, uptime performance, scale extensibility, and maintainability. Maintainability is mostly just looking at your support queue and just the inherent quality of your software. So I think that that last bucket is having metrics and treating your platform as the main accelerator of doing a really good job around those five product features.

Jean-Michel Lemieux (00:39:53):
For me, that's my portfolio. That's literally like, every year I'd go to my teams, and I'd obviously be working with the senior developers in my company that put a plan together that had investments in those three buckets. And I treated it very seriously. Again, you have to put yourself in a mindset where you're treating this work as importantly as your CPO is treating his roadmap.

Jean-Michel Lemieux (00:40:19):
But what I found most engineering leaders are too blase about this. They don't have a plan, they think the product plans and you just do, and then you end up in a spiral of making, I'd say the wrong balance of investments. You're making good ones, but you're making the wrong balance and over time, It's going to slow you down. Anyway, in a nutshell... Sorry, I'm kind of pretty highly [crosstalk 00:40:41]-

Mik (00:40:40): Exhausted.

Jean-Michel Lemieux (00:40:41):

... around this, because there have been so much learned experience about me screwing up in this area, but that's my instruction manual, I think for making that possible and making those events possible.

Mik (00:40:53):
Then in terms of the people, because in the end, you've had to put people in charge of it. And we've seen different models for that, where these platform owners, the people doing that, and effectively at some point the product management for the platforms, right? There's more the single-threaded owner style of things, there's I think more of what you did in terms of putting two or three people in a box. How did you think about who's got the platform ownership, who's driving to these three goals? Who's making sure that developers are happily and effectively able to contribute, and driving the retention from your first point and someone that's another benefit, as you mentioned previously of this?

Jean-Michel Lemieux (00:41:27):
It's hard to [crosstalk 00:41:28]-

Mik (00:41:28):
How do you bring those people in, and how do you structure the actual leadership of the teams?

Jean-Michel Lemieux (00:41:31):
My recommendation is take like 30% of your engineering team and put them in the platform bucket, just explicitly. Just go, "Hey..." And you give them some of these features. They're obviously building all the devtools, they're building pipelines, they're doing a lot of the infrastructure work like make it easy for people to spin up services, like all that fun stuff. But the reality is a lot of the platform work you're doing are in code bases where the features are happening. It's not necessarily like there's just one magic API that all the abstractions go to.

Jean-Michel Lemieux (00:42:07):
So the other work has to happen is where as a technology leader, you're influencing the roadmaps of the other teams around the org. By doing that homework and having a really good guild of senior engineers and all these groups who... their job is to come up with the plans for their areas that fit into the free buckets I just talked about. We'd bubble up and we'd share that, and they would have those tools to work with their product teams on the things that they do.

Jean-Michel Lemieux (00:42:35):

But I think having that 30% just I almost call it like a platform slush fund, where I didn't... All the time I have to maybe debate the one or two engineers, like I had a bit of the JML slush fund of engineering stuff that I didn't have to go and necessarily convince everyone out was. And then there's another 20% where a bit more strategic and we had to work in different product areas of I guess, deciding the order of what kind of work do we do. For me, that's worked really well, because one is it gives you a team that can be led by engineers that are... most of their customers are engineers. You have a bit of a freedom to move around without maybe some organizational complexity.

Jean-Michel Lemieux (00:43:12):
But then on a flip side, keeping 20, 25% of the platform roadmaps in the actual product teams forces you to be highly collaborative, and explain value really well so that they buy into it. And they help contribute and make it better over time. I think I I've had some exceptionally good PMs who once they get it, they get super jazzed up. Platform work's not complicated. Often, it's just where do we put this and why? And I've had some amazing PMs who've really matured their thinking about understanding how these investments pay off in the longer term, and having those throughout the org is way more valuable and having them all in one org.

Mik (00:43:49):
Yeah, I think that this makes somewhat sense. So for that 30%, you actually... it was more entering heavy leadership in terms of how that work was being done. And then you had more-

Jean-Michel Lemieux (00:43:56):
Exactly.

Mik (00:43:57):
... PM in the 20%.

Jean-Michel Lemieux (00:43:58):
Yep.

Mik (00:43:59):
Okay. But were you still using like the two or three in the box model for the 30%, for the JML slush 30%?

Jean-Michel Lemieux (00:44:07):
Not as much. It was probably.... We had some technical product managers in that group, doing a lot of collaboration. We had like share a project with Google Cloud doing container security models. So, we'd have some technical product managers who'd really

help, I guess, roll out some of the platform changes across the company. Because I think a lot of the platform work success is actually adoption, so it can't be built in a silo. It has to be built and adopted, and success is when it's been used and we're seeing value from it. So I think having some of the TPMS in those orgs have been great. Then we actually started hiring UX people in for just the UX of the tooling we're building. The user experience of development environments set up [crosstalk 00:44:51]-

Mik (00:44:50):
Oh you did, Wow, that's awesome.

Jean-Michel Lemieux (00:44:51):
We had a small UX team, not great. They were great, not big, just the user experience of using the platform. And one of the tools we built, which was... You're like what kind of platform investment makes sense is I think the entire tech industry have spent billions of the dollars on developers of writing code, like in IDEs, and like quick fixes, and running tests locally and all that stuff. But there's so little money that's gone towards what happens when you press deploying your codes in production? Like, what happens? So I think we spend lot of time at Shopify and this was because probably a lot of history of seeing how screwed up it was and how lack of investment was, as an engineering team you're running a bunch of things in production. And as an engineering organization, when I go to my GitHub repo, I see code that's stale and static, and isn't running. But as an end org, what matters more is what we're running.

Jean-Michel Lemieux (00:45:52):
So we built this tool that's helps us basically show everything that's running at Shopify. Who owns it? What's... We actually had quality metrics, we had uptime metrics, we had costs. Just really understanding the things that as engineers, we think about when our software is running to put that into a tool, so that as your org scales, we're not just talking about the COVID writing where we're talking about the COVID writing. I guess that that tool probably, again saved us having to hire another 150 engineers.

Mik (00:46:26):
Wow.

Jean-Michel Lemieux (00:46:26):
And it's fascinating, we added a bit like we did in the Eclipse and Vs Code, like quick fixes. So, let's say I want to create a new app at Shopify to do advanced taxes for whatever, some country somewhere. You can go and you can create an app, and you can go, "Actually, I want a PagerDuty setup, I want this set up and a Splunk, and I want a Datadog dashboard created. So, what we did is we automated the development setup of the things you need to run code-

Mik (00:46:53):
Oh, no way. That's so cool.

Jean-Michel Lemieux (00:46:54):
... the same way that your ID was doing that, when you... You know like ID scaffold like stuff-

Mik (00:46:59):
Yeah.

Jean-Michel Lemieux (00:46:59):
... I was like, "Why don't we scaffold what it takes to actually run something? And then the other thing we scaffolded was a culture of whenever you're running something in the company, everything is a different value. Sometimes it's an experiment and sometimes it's a tier one server. So, putting into this tool, going, "What's our tier one services, and then what culture do we have at one how do we run those? And that's very different than tier four. And again, that's not in GitHub. I can't see it.

Jean-Michel Lemieux (00:47:24):
Again, that was a platform investment that I think paid off again over to.... We built that maybe eight years ago, nine years ago, and I think kept our engineering team sane and fast over the years. But at the time, it looked like we were had maybe a of not invented here syndrome, or... But that was one of the biggest pain points is who's running what, who owns it? How do I decide what onboard people that to explaining how do we care about running things? So, that was a platform investment that was like, I'm extremely proud of and I think is paid off in spades.

Mik (00:47:59):
That's amazing. And I think just the clarity with which you think and describe the different types of work and teams between the platform forms that features the experiments, but then actually digging to the platform and how much of this... I guess, publicly, I'd seen so much of what you were doing on the platforms wasn't supposed to Shopify his own ecosystem, who are building on your APIs and support developers. But just how much of it is actually driven by the productivity and the happiness, and the success of the internal teams as well?

Jean-Michel Lemieux (00:48:29):
You do usually hit two stones at the same time, you know?

Mik (00:48:33):

Yeah.

Jean-Michel Lemieux (00:48:33):
So it's like we built this tool to help us run pro software, it was like, because we wanted uptime to increase for customers. We wanted performance to increase because... and you have to build a system to do that. The same thing at Shopify, one of the biggest projects we had going was we had an API for our partner developers and we'd an API for Shopify. So, like the UI of Shopify, like the mobile app and the web admin was all based on an API that partners weren't using. We almost had like a partner API. And I think as we evolved the platform, we're like, "We need the... It has to be the same one." Because we have to increase the quality of it, because we're going to have more and more developers building on Shopify. We can increase the quality of it if we don't use it, and we can't let our partners find the bugs before we do. And we had ideas of making it better, it had to be more performing.

Jean-Michel Lemieux (00:49:19):
Again, we moved to GraphQL and then we're like, "We're going to have one API for everyone." So again, it's like, we got a lot done with that, I'd say almost infrastructure project for the longterm. Now we end up having an API that was powering our UI, our mobile apps that was faster. And in the same one that the partners were using, we had better feedback loops, better monitoring. But that was like a multiyear platform project, again, that had some product input. Obviously, it made things faster and made developers faster, but it created feedback loops for the future, which... It's hard to explain that on paper why it makes sense.

Jean-Michel Lemieux (00:49:57):
But I think every CTO or VPN out there is like, "You need a list or two or three of these." Well, you probably have the list of 10 and then you should strike seven out, because they're probably going to be like waste of time. The timing's not right, but there's always a time for one or two of those things that are going to set you up for the future. And that, I think your gut's going to say it's too early and I'd say start, because you know what, it's actually easier to stop at some point than it is to start three years too late.

Mik (00:50:22):
Yeah. And at the end there's is a roadmap of what you're going to productize in your platform next, right?

Jean-Michel Lemieux (00:50:26):
Yeah.

Mik (00:50:26):

And just how much internal and external benefit you can get from that. So, Jean-Michel, to me, one of the things that was so interesting about you moving into the CTO and that VPN role at Shopify was you came into an even bigger organization, an even bigger code base. So I can only assume that overnight, you turned everything into microservices and this beautiful architecture. But can you tell us how you really approach it? Because I think again, so many people out there working with such massive, existing portfolios and code bases, and under investments and platform, and too much tech debt, and all those kinds of things. I think the pragmatism of your approach where what you've told me about thinking of how you can actually work with this tribute models and the like, but get to some of these platform goals. Just take us through that at a high level.

Jean-Michel Lemieux (00:51:09):
Shopify had a very decent architecture when I joined. And actually one of the things I was so excited about was it was actually pretty simple. And Tobi, and Cody, and like the team that was there before I joined had a couple of good values of just using a few set of technologies, but being experts at it. And at that time, I described Shopify is at that phase was a product, like a really good product, but they kept the surface area, pretty good, like not overly complex. So I think what I brought to the table was really the transition from a product to a platform. And I think that's the biggest change we're going... So, we've got a tax service, we have this, and they're all like... How's this going to evolve over time?

Jean-Michel Lemieux (00:51:56):
I think all hands I gave at Shopify, I was like... And at that time, all the developers were redding in the same repo. It was on GitHub was Shopify slash Shopify was the thing we ran. And I was like, "Over time, there's a lot of things that Shopify's going to do that not all our merchants are going to need at the same time." We're going to need to build Shopify like an operating system, where we've got the kernel and we've got a lot of applications that are built on that. And that's going to give us, one is longterm velocity because not everything has to be in CORE. It's going to give us a great platform. It's going to help us focus on what actually is in the kernel, what's in the CORE, and what matters, what do we get the most leverage out?

Jean-Michel Lemieux (00:52:31):
So I think that's where I spent most of my time was turning Shopify to a platform from a technology perspective, from a culture perspective, and making it more sensible, making it so that we can have a big community of developers, both within Shopify and outside that can contribute to Shopify without all the code going in the same space. But I think there was a great foundation because we were simple. Shopify's architecture was simpler and Tobi and Cody made it very cloud-native, which was great. I think the speed

of evolution of the architecture of Shopify was.... I think we might have been one of the fastest teams on the planet earth, in terms of what we've been able to do.

Jean-Michel Lemieux (00:53:06):
We wrote our mobile apps and react-native, introduced a new GraphQL endpoint, hyper multi-tenanted backend distributed around the world. I'd say we had a friendly monolith where our philosopher splitting out new services was if they scaled on a different axis, then we would split it out, right?

Mik (00:53:29):
Okay.

Jean-Michel Lemieux (00:53:29):
So if you look at Shopify like the back office where you log in as a merchant is very different than the online store. The online store's going to have 100 million visitors and then you're going to have 20 staff. Okay, that's, they scale differently, so we'd split out the online store into a Renolink pipeline, into different things. So that was how we ended up, I'd say, modularizing Shopify along scaling accesses, and then making it so that we could have different UIs on the same back off. So we knew we were going to have multiple mobile apps that were going to build right for different functions.

Jean-Michel Lemieux (00:53:59):
So that's where we ended up having a really good API and saying, "You know what, there's going to be a bunch of UIs of commerce. There's going to be a button on your website, there's going to be a mobile app, there's going to be a shop app. Let's make the UI of Shopify a lot more plugable so that we can have different commerce experiences. So that's probably the crux of what I did over the seven years there was just that transition from product to platform, and then having a couple of the clear goals of why they're valuable to the business and shipping them.

Mik (00:54:26):
Awesome. Wow, that's amazing, Jean-Michel. So, tell us, we've got to start winding down here. It's so great to hear you and see you capturing these experiences, in your book and your website building from the right side. So, just tell us a bit about where that's headed. Please keep doing it, and thank you for sharing some that here. And just tell us some of the key things that you're now getting across to others following in your footsteps.

Jean-Michel Lemieux (00:54:50):

Well, I found that a lot of us again, we're leading engineering orgs, we never took a course on this. We went to school for something completely different, although it's related, it's actually pretty different.

Mik (00:55:00):
It is really.

Jean-Michel Lemieux (00:55:00):
So I think the next phase of my career is I'm helping as many aspiring CTO, VPN engineers as I can, I have to share some, like not just tips and tricks, but just ways of seeing things differently. Because I think we're not trained to see kind of patterns or see... and how I guess, large software organizations come together.

Jean-Michel Lemieux (00:55:20):
So I am writing a book called The Building from the Right Side, which is inspired by a book called Drawing from the Right Side. I did kind of a fine arts undergrad in high school, and I think drawing of course, he does look at things and look at patterns. I'm really inspired by having your eyes open and looking at things, and putting pen to paper. So I'm writing a book that's completely free on the internet with things I've learnt. And hopefully going to help everyone who's trying to do this look at their teams, looking at tech stacks, and ask the right questions at the right time, and hopefully make it enjoyable to lead software teams.

Mik (00:55:57):
Amazing. [crosstalk 00:55:58]-

Jean-Michel Lemieux (00:55:57):
And have as much fun as I did, you know?

Mik (00:55:59):
Yeah. So we'll link the book on the website because I think it's such a key topic, and it's come up on the last few podcasts I've done. Just give us a little bit of a teaser. I think you talked a lot about the platform investments, but the alignment and autonomy. One of the most recent things that you've written that I think had just a massive reception out there on Twitter.

Jean-Michel Lemieux (00:56:18):
It's a bit embarrassing because it took me forever to figure out this, but humans are trained to be autonomous. So in school, our job is to go away and come back, and get an A+ and not ask, talk to people about how we got there. Our parents are like, "I can't wait till you're autonomous and you're out of the house, and you fly out of the nest." Our

entire life is about autonomy, and then people have written books to remind us. They go, "Hey, I talked to all these humans and they really value autonomy." And I'm like, "No shit, that's what we grew up on."

Jean-Michel Lemieux (00:56:47):
But then when you build a company and a team, you realize that alignment is way more important than autonomy is. And alignment is what gives you a shared language of where you're trying to go and why. And it's hard to build, it's hard to measure. If I ask you, "Mik, are we aligned on this?" You're going to go, "Yeah." I'm like, "Give me a number. Are you happy about alignment?" "Yeah, it's an..." I'm like, "That's all BS..." But if I tell you, "Are you autonomous?" You're like, "Yeah, because my boss isn't bugging me or like..."

Jean-Michel Lemieux (00:57:13):
So, autonomy is this like sacred cow, and every company that values autonomy over alignment is going to die, because you're going to have a company with a lack of clarity, a lack of culture of actually even discussing things at the right level. And I think we're all allergic to talking about the things that you have to get aligned on are actually a lot more in depth than you think they are. And books are trying to again, like get out of people's way versus why don't you just... You should be brainstorming on things.

Jean-Michel Lemieux (00:57:42):
And as a boss, as a leader, a CEO, a CTO, it was really important that I get aligned with my team, and that we share a set of values of how we're going to build things. And in like every project, I would ask a lot of questions and I want my team to be asking themselves. So for me, alignment has to come way before autonomy does, but it's hard and it's hard for me. And I value my own contribution as being the most autonomous CTO ever versus be the most aligned.

Jean-Michel Lemieux (00:58:07):
I got told time and time again by my bosses, and it took me way too long to realize this. And then I hired so many people that I had to have that conversation with. Again, I'm going, "Hey, we're going to be talking a lot." And I know it's not normal because you read all the books that says we're not supposed to... I'm supposed to get out of your way. But we're going to talk because what we doing is so hard and complicated and changing, that we're going to build a really close relationship of alignment.

Jean-Michel Lemieux (00:58:32):
So I think that topic is underdone. And I think as you can tell, I'm a bit excited because it took me so long to get that realization, you're so much going against us wanting to value

that. So that's the crux of my books. I'm trying to find these nuggets of things that are maybe counterintuitive. And once you see it, you can't unsee it. It changes how you live and how you work with people, and hopefully how fulfilled you can feel in what you're doing.

Mik (00:59:01):
That's awesome. That'd be another title. That would be an amazing book and stuff that alignment greater than autonomy sign. That's a-

Jean-Michel Lemieux (00:59:07):
I know it's a big topic, but that was 32nd [inaudible 00:59:09].

Mik (00:59:09):
That was amazing. So, Michael, thank you so much. Just the wisdom that you've shared and I think there's people be listening to this over and over, because there's just so much depth behind what you've done. And thank you so much for sharing it so clearly with the audience.

Jean-Michel Lemieux (00:59:23):
Well, it was my pleasure and thanks for teasing it out.

Mik (00:59:27):
Awesome. Well, till the next time. Thank you so, so much. Thank you to Jean-Michel for taking the time to join us today. For more, follow in my journey on LinkedIn, Twitter by using the hashtags, Mik + One or Project to Product. Jean-Michel's Twitter handle is @jmwind, and he's got some great content and he continues posting. You can also reach out to him on LinkedIn. I have a new episode every few weeks, so hit subscribe to join us again. You can also search for Project to Product for the book, and remember that all of proceeds goes for women and minorities in technology. Thanks, stay safe, and until next time.