Episode 8: Dean Leffingwell
Episode Transcription

Mik Kersten:
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 best-selling author of Project to Product: How to Survive and Thrive in the Age of Digital Disruption with a Flow Framework.

Mik Kersten:
Today I'm joined by someone who's recognized as one of the world's foremost authorities on Lean-Agile best practices. It is none other than the author, entrepreneur and Co-Founder and Chief Methodologist at Scaled Agile, Dean Leffingwell. I have personally applied Dean's ideas to the way I build and manage software delivery throughout my entire career, and I find his contributions to the industry to be invaluable.

Mik Kersten:
Most recently, I've been very impressed with the work that Dean and the SAFe framework team have done around SAFe 5.0, as it provides organizations with key blueprints needed to shift from project to product at scale. If you're enjoying this discussion, I'd encourage you to sign up for the upcoming virtual European SAFe Summit on the 10th and 11th of June, which is run by Scaled Agile, and where I'm honored to be a keynote speaker. So with that, let's learn more about succeeding with software at scale from Dean himself.

Mik Kersten:
Welcome, everyone. I am so thrilled to have here with me today, Dean Leffingwell. I've followed Dean my entire career, when I first got into software and Gail Murphy introduced me in a 100 level course, she introduced us to a tool called [RecPro 00:01:39], which I got to actually use and it made me realize that software, maybe it was not as invisible as I thought as she was teaching us how real world large scale requirements work. I then followed the amazing work that he did that Rational because I think some people know that I'm actually a fan of the Rational Unified Process. Even though I saw many organizations butcher its deployments, I thought there was some amazing ideas there that we're now just recreating, as product value streams and does the end to end traceability and visibility for those sorts of things.

Mik Kersten:
So, I think there are a lot of historical guides for me personally there and I think for the industry as a whole. Then of course, Dean became Chief Methodologist at Rally software, where he created one of the, well, maybe the first enterprise grade, large scale portfolio level Agile tool out there, with Rally. He's author of the Agile Requirements book, Software Agility and Managing Agile Requirements, as well as the SAFe Reference Guide. Which I still have to say Dean is, I think, one of the most dense, beautiful and potentially underappreciated books that I've read cover to cover because there's so much gold in here in terms of the art and the practice of software delivery.

Mik Kersten:
I think most recently, the latest inspirations have come from what Dean and the framework team have done with SAFe 5.0. Because I think the way that they've encompassed product thinking and customer centricity, design, thinking and so on, really helps everyone move forward in this direction of innovation. So, with that, welcome Dean.

Dean Leffingwell:
Oh, thanks, Mik. Thanks for having me.

Mik Kersten:

So can you please just give us a summary of what's in 5.0, why it's key. To me, I think it's a very major version update of the framework and why you're excited about it?

Dean Leffingwell:
It's clearly a major release. It's 5.0, is an upgrade, so we have major minor releases. We need to bring our constituents along, SAFe program consultants, they teach the courses associated with SAFe, so we obviously want them competent in the latest version. So, in the case of 5.0, in order to teach a 5.0 course you need to be 5.0 certified. So that's what we mean by a major release. That means that our practitioners, the people that do the work and do the training, need to know the latest and greatest.

Dean Leffingwell:
5.0 started of course with 0.8, it's been a continuous evolution. This is actually the seventh or eighth release of the framework. As you know I'm a huge fan of whatever works. I'm probably one of most practical methodologist’s if you can call me that, you'll ever find. And if it works, and we can figure out how to fit it into SAFe, it's going to go in there.

Dean Leffingwell:
About a year and a half or two years ago, the pressure started to move from just technical agility if you will. Not as if we've mastered that, and everybody's great at it. But let's just say you get good at it. And we see people that are good at it. And they're going, "Well, where's really the impact on our business? Are we increasing market share, are our marketing teams along for the ride? What are the sales teams think about the fact that now you want to have a release every month and their SLS for their customer, the service level agreement says every year, you're going to buy an upgrade?" So that's really business agility, and we were told loud and clearly by our key constituents and customers, that yes, SAFe is great and technical agility is awesome, and we can't do business agility without it. But if we don't have business agility, also, we're still not going to be in a mission.

Dean Leffingwell:
So that comes in a number of dimensions, areas that weren't particularly technical in nature, continuous learning culture. I personally think I know you're a fan of Carlota's work. You showed me that work, and she talked about the various stages of technology development and the management paradigms that go with it. And if you think about something like mass production, you think about Taylorism, and the manufacturing engineer is the smartest guy and told everybody else what to do.

Dean Leffingwell:
Well, I think the prevailing paradigm for the next decade is going to be continuous learning. So continuous learning culture is an example of things that we've integrated in the SAFe. That's not a technical practice, this is a cultural practice and things that we didn't have before. Also, the ability to measure how you're doing. As more and larger enterprises adopt SAFe, especially those that didn't start early, they want to know how they're doing. Are they meeting their benchmarks for adoption, are their outcomes improving, are the KPIs different? So, this need for business agility was expressed clearly. So, we extended the framework to do that through a couple of areas, the organizational agility competency, the continuous learning culture competency, the ability to measure and grow.

Dean Leffingwell:
In the center of all of that, however, is I think the thing honestly, that I'm most pleased about, which is I've been a fan of design thinking for least five years maybe more, I don't know exactly. And there's another great knowledge pool about who your customer is and how you come to understand them. And that's one of the things that in your book, Mik, Project to Product, I asked the question of the teams that I'm training,

who's the customer for a project? And they kind of scratched their heads. So how do you do an empathy interview for a persona that's a customer that you don't even know exists? It's just kind of nonsensical.

Dean Leffingwell:
So the addition of customer centricity and design thinking and tools around that, simple things like empathy mapping personas, story mapping, Jeff Patton's work is great. I mean, story mapping relates the job that a developer does to the job to be done that the user does. So, incorporating that for me is ... I'm relieved that we've managed to kind of incorporate that in a way that makes sense.

Dean Leffingwell:
And of course, if you're doing business agility, and you don't understand the customer's journey, and you don't really understand your customer, and you're not really solving the user's problem, you're not going achieve business agility anyway. So that kind of aggregates together and what we released that SAFe version 5.0, just recently.

Mik Kersten:
Yeah. And I think that's really what got me excited about as well is that we're seeing ... I think we're seeing this as an industry wide move this does that relate to Carlota's work and on the fact that these organizations need to get through the Turning Point, need to become software innovators, but that for business agility, there needs to be a new way of thinking. And I think you and I have both been shocked by this fact that every time, "Oh, you asked your framework team?" I've asked countless IT leaders is, who is your customers? Can you identify the customer for each product value stream? And of course they can because with this product structure that's been completely insulated and isolated away. So there's no direct line from what the teams working on the Agile release trains on the backlogs are back to the business. And so that just makes to cite the three ways of DevOps of flow feedback and continue learning it makes that continuous learning piece impossible.

Dean Leffingwell:
So many of the practitioners in a large shop or somebody building big systems deliver solutions to internal customers. And it's not natural to wake up and think about that and say, "Well, my customer for this is the person that's processing shop floor orders." It's true that they know they exist. But do we apply customer centricity to design thinking that person? Or is this always about this external person, this patient walking through a hospital or somebody logging into a website. So I don't know what the percentage is. But most people develop things that serve other systems, which are also customers of your system, or people that are internal, not just the external one. So I like the way that you've described the project to product move. And this is also supported by team topologies, Skelton and Pais work, because they're going, "You've got a platform, who's your customer? Oh, it's a developer. Now, these shouldn't be brilliant insights. But we know we're solution providers. We love the technologies, we go down and dirty and, "Oh, yeah, somebody is going to use my API. I'm pretty sure because it's so cool."

Dean Leffingwell:
Well, I think that the contemporary view is now evolving to say, wow, I don't care if they're internal or external, there's a person there, and I need to know them, and I need to understand their pains, I need to have empathy with them, I need to understand their jobs to be done. So I think re centering on that customer, it's not always the guy logging into a website, that's kind of the trivial case, it's somebody about to do an upgrade for a big SAP system. And you're building the tools to support that or that your user in that case, your customers developer, and their customer is somebody using SAP to do purchase order processing. So that whole chain of understanding the customers I think, is well supported by your book.

And by the ways, thank you for stealing the title of my next book, by the way, and totally destroying my plans in that area.

Dean Leffingwell:
But yeah, I think I'd say 5.0 knowing there's a real person, sometimes that's a system. So sometimes it's a little more abstract than that. And understanding their needs is key part of application development. And maybe we lose that as we get bigger but we can't afford to lose it for very long.

Mik Kersten:
Now, you and I thought this was clear, that you had different types of customers. It's actually one thing that I've noticed from the many meetings since publishing Project to Product over a year ago, is that there's an over focus. And again, I kind of wish I spent more time going through this in the book, there was an over focus and thinking that customers that value streams are only external. No, you've got internal ones. Large organizations, they can have more software being produced for internal customers than external ones. Your developers are a customer of your tools, your value stream network, your methodologies, your processes, your training, and so on.

Mik Kersten:
So this is such a key thing is that ... I think this is why we like about 5.0, everything there actually applies to the internal customers. One of the main questions I ask is, okay, so it's internal shared service, it's a data pipeline, it's an API. What's the release strain? What's the value stream? Where's the backlog? Do they have a roadmap? It's the exact same concepts apply. And it is amazing to me how often that's an eye opening or mind opening question is oh, wow, those things are first class value streams. Those are our internal products. That is the way tech companies think about their developer tools, that is the way car manufacturer thinks about the plant. Those are actually maybe even our most important products, because they make every one of our developers productive, or they make the business applications productive, because they have a common view of a customer, rather than every single business application, duplicating the logic and the data storage and all the fun that entails.

Dean Leffingwell:
One of the more kind of advanced topics in our world as you can probably understand is value streams. And they're not like the exit path on an airplane when the lights go out. They're there, but you can't really see them. And SAFe supports development value streams. That's what we do. We help IT practitioners and developers and other architects and technical people build better systems. But guess what, most of those systems are used by operational value streams that monetize in some way or another. Some discrimination between development and operational value streams isn't clear until you make it, when you make it, you then ask yourself a question, "Who is the customer for my SAP system? Who's the customer for this little actuarial table I built in Excel, that's being served."

Dean Leffingwell:
So I think the value stream thinking is another good hook for that, because you're forced to ask the question, how does the enterprise make its money or deliver its value? And how do I help those people do that? So development streams actually only exists for a purpose. And it's to make systems that help the operational value streams more effective and more efficient. So with 5.0, I think we're clear about that. They've been in since 4.0, but when you come to business agility, and we added the competence around organizational agility, we were compelled to call that out much more clearly. So that if you're supporting an operational value stream, who are the systems? Who are your customers? And oh, by the way, what other support do you need internally to make your solution support that operational value stream. That

could be security or additional help from Ops or customer support, that's going to be necessary to help those users.

Mik Kersten:
And it's interesting that you mention that Dean, because having read and studied 4.0 or 4.5 in the early versions, I actually thought this all was clear on value stream thinking this kind of thinking in 4.0. That's a lot of the basis and some of the ways of augmenting that with measuring flow and so on of putting in business results for internal product value streams like API's, well guess what you just count adoption of the API. That is a great way of measuring outcomes. That's what thankfully, we're starting to see customers do.

Mik Kersten:
But I think that these things are not first class. I'm now again, understanding how important these kinds of taxonomies are. Because the fact that you've got design thinking right in the middle, there actually does ... and customer centricity, actually doesn't make people think, well, if we really want a great customer experience, maybe we can have 18 different views of the customers as they flow through our different web properties.

Mik Kersten:
But what was really eye-opening to me hearing you talk at the SAFe summit, and just seeing how these things are being applied and misapplied, is that while that was there in 4.0 the business people were having trouble ... just organizational structures, were having trouble wrapping their heads around it. So I think the blueprint for where they need to go was there, but the organizational structure, the way that intakes work, happen still, was not.

Mik Kersten:
So for me, the biggest aha moment from conversations with you from the past year, is what you introduced to me with Kotter's second operating system. Because I think like you have just been bashing my head against this problem of the fact that it doesn't seem that complex to shift to the structure that SAFe provides you with, to measure flow, to understand your external, your internal and your developer centric value streams. But somehow the structure that's in organizations today doesn't support that. You and I are both pragmatists, so we don't want to wait until there's been an four year long reorg that's now perfectly horizontal or whatever the-

Dean Leffingwell:
165,000 people at wherever HP or whatever adopt [inaudible 00:14:43], I don't think that's going to happen in my lifetime. So that's an interesting idea, but it doesn't really apply to our current case.

Mik Kersten:
It doesn't and so what you said to me was Kotter's second operating system and the way that we're actually leveraging that in SAFe can catalyze the shift without requiring a reorg tomorrow. And this is not clearly SAFe 5.0, you want to make a fundamental shift in the way that the business learns and plans work and the workflows and the feedback loop from that. But can you just tell us around that, because I really do think and I've run this by many customers myself, there's a key thing here that's SAFe is a catalyst for.

Dean Leffingwell:
Well, about the same time I don't remember exactly as I was finding your book, Kotters's Accelerate book came out, and I can't say I read all of his stuff, but I mostly do. And this was a big shift from the normal

kind of OCM pattern and leading change to the problem of the organizational structures. And he showed that picture and I borrowed it frankly, this is brilliant of a hierarchy and how difficult it was for hierarchy to operate like the network that was formed. I see behind you, there's lots of people there, you're probably a pretty sharp little network. When you double or triple or quadruple your size, the tendency is going to be to add the structure you need to survive. Recruiting and hiring and comporting with the laws of the land, and making sure that you're in the right facility, and you've got the cash flow, and you've got the right funding model. All of those things don't come out of the network side.

Dean Leffingwell:
So when I put your book juxtaposed to him, and I said, Project to Product, and I looked at the network operating system on the left of his view, and I looked at the hierarchy on the right, what's the dominant way to get things done on the right, the hierarchy? It's a project. That's what we did. So to me, it was like two different views of the same problem. So the aha light went on for me is that, number one, it's not clear to me that you should tear those hierarchies down because they do lots of things. My expense reports need to be approved, we have a budget for this year. This is not an arguable construct, and in the larger enterprise, they exist and they have served us well.

Dean Leffingwell:
And I think Kotter called out one of the greatest inventions of the 20th century, well, 21st century now and it's time to move on. So I saw this ... what leaked out of those two books in my reading after reflection, wow, this is the same problem for two different perspectives. Mik is basically saying we have to move from this project mentality to flow. And flow means we want to dedicate people on a long life basis to a product thinking. It doesn't happen in the hierarchy side. The hierarchies, we need to talk about how we're going to react to the Coronavirus. How are we going to manage our expenses, the flow side says I just want to build a better solution. So that to me, rang true. And it reflected also, I think, some of the major challenges people have moving to Agile, which is very egalitarian.

Dean Leffingwell:
Let's just say in a cultural standpoint, if my team doesn't argue with me, there's something wrong. And if a new dev comes on, and they're sharp, and maybe the ... I remember one young person, that was a really good modeler, and he went into a crap team that couldn't model whether he was a hero, because he could express in whiteboard UML like models of their solution and they will go, "Oh, that's it," because they couldn't see that. So that part is tougher for a hierarchy, because your rank has value and adds, "Oh, that's really egalitarian." And frankly, your ability to help has the value props and the value propositions are different.

Dean Leffingwell:
So we express that in SAFe and saying you know what, if you need a network operating system, and I said, in hindsight using Kotter's words and your view, "Well, that's what we've been doing all along. We've been going in as large enterprises that can't get stuff out the door that don't have an organized backlog, that everything's a project, everything's in emergency. Teams are multiplex to the 10th degree" ... They're multiplex to the point where absolutely nothing new happens.

Dean Leffingwell:
I remember one of my first roll outs after a few iterations, nobody got anything done. And when we looked at it is nobody worked on the budget. They all worked on whatever leftovers they had from a previous life. So I think the dual operating system is a good frame. Now, is that less complicated than a single one? No. But guess what, we're solving the most difficult problems that industry today faces. There's a 100,000

companies that if they don't make the transformation to digital are going to be waxed, we're all going to be working for Zuckerberg, if we're not careful here in another decade, I don't want to do that.

Dean Leffingwell:
So I think our job is to help companies who did not start with 100% Agile DNA, who have big legacy systems, who need to innovate both at the edge with their platforms. And also at the core of digging, while we need to move this whole thing to the cloud, and use Azure, DevOps or whatever your tool of choice is to make your development go more quickly.

Dean Leffingwell:
So it's a good expression, I think and helped me ... I was in Japan a couple weeks ago, and they get hierarchy for sure. And when you say this exists, and is real, but it's not efficient for this other thing we need to do, then they start nodding their heads, and they go, "Yeah, you're right. Let's talk about that or the thing." And then all of a sudden, we're talking about SAFe. We're talking about value streams, we're talking about [inaudible 00:19:51], we're talking about teams of Agile teams, and you've got the hook. But what they envision is, oh, these boxes go away, my job goes away. And everything is this weird egalitarian thing where Agile hippies rule the day and code whatever they need. That's not the vision of Agile that's not Scaled Agile, and it's not real Agile either.

Mik Kersten:
Yeah, this is ... I think we have the same mission, right? We want those 100,000 companies that the vast majority still hopefully around 80%, but getting smaller each day of the economy to thrive. And-

Dean Leffingwell:
I think it's important. You and I think it's important for big time, macro-economic reasons. I don't think having 30 companies 20 years from now, it's going to be healthy for our society.

Mik Kersten:
No, exactly. And I think there are people within those companies who want to make these changes. But I've seen I think what you've seen, which is these wholesale changes, like let's make a holacracy, let's make the whole organization Agile. I think, you and I have been both entrepreneurs. We've helped structure and build organizations. And I think we know from experience, it doesn't work at these scales. There's a reason, and I think this is the only thing that you pointed out in Kotters, work, than I then read was that there's a reason those hierarchies exist. People need their salaries, their pensions, that health care plans and all the other things that come from managing that many people effectively, those HR structures, but we can't so shift into holacracy, even though it's been tried at modest scales. Whether it's Zappos or GitHub, it's failed. Interesting [inaudible 00:21:20] devolves into very political structures that are not egalitarian. That's a whole other body of work. But I think the interesting thing here is that you're providing an incremental path for organizations today, to take steps to business agility, rather than thinking that they have to reborn as a digital native.

Mik Kersten:
And I'm absolutely seeing this danger of people thinking that no, let's start from fresh. Let's go back to green fields. Let's resurrect our business as a whole set of two pizza teams, which can't possibly work when you've got an existing demands, customers, markets, equipment, and there this-

Dean Leffingwell:
[crosstalk 00:21:56] lines of legacy code via the[inaudible 00:21:59] or via the back office IT system.

Mik Kersten:
Yeah, exactly. So if we don't provide organizations with a path through that, if you don't provide today's leaders, today's teams and today's organizational structures with a path to organizational agility, a roadmap ... Because I'm a big fan of the SAFe road map as a concrete set of steps that you can do, implementation of maps to do this, then again, they're going to continue down the current trajectory. And a current trajectory is not viable for the health of the world economy.

Dean Leffingwell:
It's a huge risk. And well, as I just said, it's a macroeconomic risk and we take it seriously. So part of our charter, part of our mission now we're both commercial companies, I've never believed that you had to be a philanthropist to solve problems. You can solve problems by selling products that really help people do a better job. So we're commercial companies, but we share that mission. And we share that customer base, we want them to be successful. Not that SAFe and your methods and tools aren't used in some of the big time Fang companies the Facebook's, etc. We know that they actually do, but in general, it's that 100,000 enterprises that didn't start out with pure DNA. Or maybe they hustle DNA maybe ... Nokia was a very strong technical company for sure. And you know those people, they had good software teams, but they weren't able to make the transformation quickly enough as the market changed.

Mik Kersten:
Yeah, not to mention the other side of this is, that the tech giants, the effective ones, they actually have these structures as well. It's just that they've made these two operating systems work together. The one that's all based around the customer around flow and the flow of value, and the one that's needed to keep a large, often multinational company running and its staff happy and productive and growing.

Mik Kersten:
So can you tell us a little bit about why you see these things failing? Why organizations are having trouble adopting this thing, and maybe if we can just start because I think that, especially for the last few months, I've been seeing this desire to, again, scorch the ground and re implement things with DevOps with two pizza teams and so on, without ... and the interesting thing that I've seen from that Dean, and this is what I'd like us to dig into. It's not something we've had a chance to talk about yet. Is I actually feel like anytime that I see that happen, it isolates the business. It basically creates IT for IT, we're creating lean ... I'm a fan of Eric Rhee's work. We've applied it all along and Steve Blank as well. But it creates this basically bubble of the new set of four or 10 Pizza teams for working the new way that it's actually isolated from the business.

Mik Kersten:
There's the hard problem of how work intake comes in, how you map from potential requirements for mixed physical software systems or other complexities, highly regulated systems these large organizations have into Agile backlogs and release trains. So I think there is this gut reaction I'm seeing like we need to start over to become like a tech company, not realizing that Amazon actually has these hierarchical structures above their two pizza teams, product value streams, and then these basically portfolio program levels-

Dean Leffingwell:
[crosstalk 00:24:56].

Mik Kersten:
... put in place.

Dean Leffingwell:
Awesome product managers as well.

Mik Kersten:
Exactly, exactly. So these principles are actually all in play there. But I feel like there's been this trend of a knee jerk reaction, like, oh, let's start over and be like a startup.

Dean Leffingwell:
Everybody wants a simple answer to really hard problems. I do too. I always want to find this [inaudible 00:25:14], I think the simplest thing that can possibly work. And we wouldn't support two operating systems if you didn't need them both. But I'm reminded that DevOps really triggers a thought number one, it's integral, the value stream, so it's just part of SAFe. But I think it reminds me when we started SAFe, so I went down and dirty kind of poster up, I was independent. I could do whatever I wanted in terms of methods and practices, and I discovered Extreme Programming first. I hadn't even discovered Scrum yet. I didn't really ... honestly I didn't yet have an appreciation for Scrum. I have one now, because I've used it over time.

Dean Leffingwell:
And when I went in these larger enterprises in 2005, and 2006, RAP was two big iterations were too long. People were using inception for requirements, and elaboration for design, the old patterns. I don't want to do that anymore. So I worked in some really small environments literally on two pizza teams, or maybe four or five pieces, couple teams together in the room. And the first thing I did is we just went down and dirty. We threw out everything. We threw out the notion of architecture, we threw out PMO governance, and we went down and dirty and started that in the large enterprise.

Dean Leffingwell:
But it's only eight to 10 weeks later, when you discover wow, you've thrown out bathwater, bathtubs, the entire bathroom here. And what happens is you start to replace autonomous with autonomy. Or vice versa, you start to replace autonomy with autonomous and you create autonomous, super powerful, really fast, localized, locally optimizing teams that can create one thing well.

Dean Leffingwell:
Well, you kind of see the same thing in DevOps right now. It's like, well, if I just did DevOps well and brought the ops teams to bear and we provide autonomy to the DevOps teams, they would be fine. Well, like you said, Amazon has great product managers. They have great architectural infrastructure, they have great strategies around evolving their technological platform.

Dean Leffingwell:
So it simply doesn't work to take a component level view and say everything is a component of the same type. They're all same, we'll just cookie cutter them out and bye good Joe, they'll figure out a way to build a great MRI system.

Dean Leffingwell:
It doesn't work that way. You've got to have strategy, in particular, certain central properties of centralization, we've got to pick a strategy and go, we can have six teams think they're doing single sign on differently, even if they use the same protocol, anymore we can ask 60 teams whether or not we should be in the MRI business. So we have to marry the autonomy we want to grant with the guidance and governance that puts the business in the right place at the right time.

Dean Leffingwell:
And that's why I say that I like to think of my team that we have autonomy. We're the framework team. We could do a lot of stuff, but we are not autonomous. Everything we do impacts other people and it impacts the way we do course work. It impacts our partners, it impacts the enterprise. What we do affects our P&L. So let's autonomy. Yes, autonomous? No. That's a subtle distinction. And it's not a trivial one to me. But it's absolutely critical to understanding we want, largely. We want our teams to have autonomy, but they're not autonomous. They have to operate within a system that works to the larger end.

Mik Kersten:
Yeah, and I think it's interesting for me, it was actually Extreme Programming as well, [Hiltzik 00:28:23] book '99, our team at Xerox PARC embraced it 100%, I was working out like for two years. And it wasn't till we started working within a much larger structure of Eclipse itself, the entire development environment from IBM Rational at that point in time, and Object Technologies International, that we realized that additional layers that needed to happen on top of that for us to effectively manage what back then was a 60 million line source base. It was all open source. But we recreated at that point structures that are effectively identical with only slightly different names, probably 30% different terminology that we've gotten SAFe 5.0.

Mik Kersten:
In the way that we work to manage that larger code base, fully and open source, and with a level of effectiveness that we were seeing there at the time. So I guess what I'm seeing people doing is now reliving that journey that's been 20 years or 21, at this point. And the frustration for me is that they just don't have the time.

Mik Kersten:
So if you going to ... now go back to kind of the benefits of that free for all on the full autonomy of XP, and again, amazing work that's being recreated by consultants all over the place now, you're missing that entire journey that we've all been on over the last two decades of how to effectively scale that. And again, connected to the business. Where you've got sophisticated lines of business to figures, customers markets, and of course, these massive legacies of internal IT systems and products.

Mik Kersten:
So how do we stop people from reinventing this wheel because what I'm seeing happen all over is ... and it's fun. And so I want to be clear, it's fun to reinvent the wheel. I like reinventing the wheel. I used to love writing issue trackers. It's fun to rewrite tools like JIRA or Azure DevOps. As a developer, it's fun to recreate Agile frameworks if you're not focused on delivering value to a business at that point in time, and you've got some latitude. But I've definitely seen now just year after year of trying to recreate something that gets us to Agile circa 2005, within a large organization and large government institution that's providing citizen services or SAFety to a nation. And I just don't think we can afford this right now where we've got other parts of the industry or other nations, we're now moving much more quickly.

Dean Leffingwell:
I think it's just too late. Reminds me of a couple things. A few years ago, I was in a bank that got rolled up into another bank. And I met their Agile working group and they were really sharp and they had developed for that bank, I won't name the bank, their own Agile way of working, the scaled Agile way of working, and it was honestly pretty good. And we started looking at it. We said, "Well, how are you going to maintain this?" "Well, we're going to evolve it." "Well, who's going to do the training, how are you going to roll this out?" They started talking about well, Agile doesn't really have much treatment for intentionality of architecture. So we've got a module for that. I said, "Yeah, we've had to address that too."

Dean Leffingwell:
So I'm sitting here kind of on with SAFe in my background, in my back pocket, talk to these people that are honestly doing a fantastic job of taking this bank down a primrose path to a sole custom proprietary invented, scaled, Agile model that I think, technically was fine. You might argue nuances or whatever, but he argues those points, what developers make their code is better than somebody else. But there is no way in that industry, or in that bank to replicate across what then became to roll up 50, 000, 60,000 different people. And when I looked at their executive management, they said, "Well, we really need a standard here, we need a common taxonomy." We can't call that a story, a user story, a backlog item, or a work item. You got to know what that is. If that's an epic, we've got to have meaning for that.

Dean Leffingwell:
So they focus on a couple of things. At least executive management said, "A common taxonomy matters, we do multinational development, we have to understand a word when we say it."

Dean Leffingwell:
And secondly, even if this was better than SAFe, like we get in a vacuum to create a thing better than these people that [inaudible 00:32:15] their entire careers in the whole company around it, we couldn't scale it or support it. we would have branched ourselves and we would then recruit people to the bank X Agile way of working, as opposed to saying, "No, let's take one of the standard models."

Dean Leffingwell:
So I think the root of that is this, I think Deming's famous quote is that it's a disease of management. And his actual quote at the time was a disease of Western management, that our problems are different. So since they're different, we have to solve them ourselves. But he goes on to say the problems are different to be certain, but the principles that underlie improving quality of product and service are universal in nature.

Dean Leffingwell:
So when you ask how you would avoid that spinning, or let's just say a poor application for us, that's the principles of SAFe and there's only 10. And we focus on those. And there are things like working process limits and small batch size. They're like organizing around value. They're taking an economic view. So if you look at those and say that's the real underlying core, if we can agree on those, then what instantiation do you want? One that is commercially and readily available and largely free? At least from a public framework standpoint, or should we invent our own?

Dean Leffingwell:
But I think it's natural. I think every developer thinks they're methodologists. And everybody thinks I can invent a framework better than everybody else's, that gig is up. That's like saying, let's start a new search engine company now. You really need to move on and figure out what you can do that's unique and different and I think inventing your own scaled Agile framework, is not it. So if SAFe doesn't work for you, for some reason, find another one. For goodness sakes, don't branch your entire company into a corner of the universe that they'll never be able to recover as methods evolve.

Mik Kersten:
Yeah, so I think two key points there ride us. I've only got a few hundred data points from the global 500. But the problems are not different. It actually amazes me how similar in terms of moving towards business agility, how similar the problems are, whether you're ... whichever sector you're in, whether you're pure software or like financial services, mix hardware software, car company, federal agency, the problems are

starkly similar, which is I think why, by the way, I think Dean you mentioned the word taxonomy. Although it's fun to invent your own frameworks and things, there isn't time. And I think people mis appreciate two aspects of taxonomies. Successful taxonomies have driven scientific progress and technological progress. I'm stealing your line now.

Mik Kersten:
But the amount of work to create this quality taxonomy to gain community feedback on it, and then the key thing that you just mentioned, which I think is really hard to miss, is to evolve it over time to incorporate the best practices. I think I've said this before that I think that the intellectual property of the scaled Agile framework will be one of the most lasting contributions to our industry, last specific methodologies and practices. And as long as it continues being evolved, it'll incorporate new ones.

Mik Kersten:
Be that DevSecOps or the next trend. So can you just speak a bit about this journey? Because what impresses me is that the journey of getting to this point in the taxonomy.

Dean Leffingwell:
Taxonomy is critical. And frankly, I under appreciated it. I understood it kind of intuitively, but not necessarily scientifically. If you can't tell a cat from a horse, what do you do when you call a vet? It's just ridiculous. Taxonomy is the basis of science. Science moves forward by saying that's a wood door or it isn't behind you rather than a molded door or a plastic door. So when we started down this path, I was forced because I was trying to make this one image the original big picture I drew when I had a rolling wave of 25 people PMO people in 45 minutes slots over a full day at one of the world's largest enterprises, you could probably guess where that was. I said, I'm not going there to teach people Scrum, they don't care. I need to show them the after case.

Dean Leffingwell:
So I said in the after case, the teams are aligned around a common cadence. And there's this lightweight governance, this architectural runway, et cetera. And then they got it, because they knew how they work now, and they knew they weren't getting the job done. But they couldn't just say, "That doesn't work, let's try a wild experiment." So when I showed him the after case, I said, "That's the way you would look," and that won, so that was the original big picture.

Mik Kersten:
That's when you drew the first big picture?

Dean Leffingwell:
Definitely. I drew it on a whiteboard, and then as the PMO people walked through it, I continued to elaborate it and said, "Okay, here's the governance portion of that, or here's the portfolio piece." And then they understood the future case, future state. Then we had a dialogue. Otherwise, is if your [dog 00:36:53] doesn't hurt, PMOs suck, requirements documents are terrible, you don't get it. You should really be agile. Give me a break.

Dean Leffingwell:
Really, I can tell these people, they don't know how to ship whatever they think they are. Well, I can't, but I could show them the future case. In that future case, you have to label these things. So I made myself a simple rule. There's never going to be more than an eight by 11 sheet. If I can't express it in PowerPoint, it's too complicated. And then it says, okay, I have to have a label for the icons. And we struggled mightily with some of them, in a large enterprise, does the product owner own the product via Scrum? No, they

contribute a really valuable piece. Is a product owner a perfect term there? No. Well, am I going to fight gravity and try to retrain the world or go to battle with a Scrum alliance over ...

Dean Leffingwell:
I know one large company vendor actually looked at Scrum Master and said, "Well, number one, you're not a master, just because you took a course," and they called the iteration manager. Well, that's probably okay. But what is the point of that we fight gravity and this universal trend, to at least have some common definitions around some things.

Dean Leffingwell:
And then we got in places like the thing above a story. It was an epic. There wasn't enough surface area there. I've been a frustrated product manager my whole life, its features and benefits. So the thing about the story needed to be features, why, it didn't really happen then, that was a term that was there for the taking. Then Epic, we had to define it differently. Because even though that was an Agile term, it was too close to story. It was still too small thinking. So one by one, we argue about these terms.

Dean Leffingwell:
We are having a discussion right now, whether a team that is not co located is dispersed or distributed. We care because[inaudible 00:38:35] how to work with a X team is that dispersed or distributed? We can't say, be dispersed or distributed.

Dean Leffingwell:
One of our teammates worked with the team for a while and it didn't work out for a number of reasons. But as exit interview, he said that those framework guys, they sit in a room and they argue about words all day. True, because we pick a term and we put value to it.

Dean Leffingwell:
Now a value stream for us may or may not be what somebody else calls it, a product manager may or may not be like CPO, Chief Product Owner, it doesn't matter. What matters is that we provide consistent definitions and we associate the right responsibilities with those people. So I'm not a fan of program increment. Am I stuck with it forever? Probably not. But in any case, we know what it is. And we can say that's a planning interval. Well, why didn't you call it a planning interval? I guess we weren't smart enough, seven years ago to call it a planning interval. Well, maybe we'll get there. Maybe we won't.

Dean Leffingwell:
But the point of the matter is, these words have to have meaning. And they have to have definitions that people kind of apply, and especially when it comes with responsibilities and people doing the work. I'm working on another project right now, just to clarify certain responsibilities in certain areas. Yeah, there's shared responsibility. There's also shared responsibilities where everybody's responsible, therefore nobody is. So the SAFe taxonomy includes events and artifacts, and roles and the roles have responsibilities. And you might argue the title of the role, but hardly anybody argues the responsibilities of product manager and you read them. You say somebody needs to do that no matter how good my Agile or DevOps teams are.

Mik Kersten:
Yeah. And I think my team's, for example, and it's really interesting that you say that program increment, because we use SAFe terminology, but people didn't like that term, program increment. We don't have programs. We know the size of Microsoft. They have programs we don't, and someone's like, "It doesn't matter. It's a common terminology." Our job is not reinventing these terminologies, a whole bunch of I

think some of the smartest methodologies or industries agonize over regularly. I didn't realize we were still agonizing on this one,[inaudible 00:40:36].

Dean Leffingwell:
Honestly, I never liked it. But it's a problem. But the label is a problem. What it is, is not a problem.

Mik Kersten:
Exactly. What it is, is not a problem.

Dean Leffingwell:
Which I've heard over time.

Mik Kersten:
Yeah. And I think my appreciation from this comes from the programming languages side, when you're designing programming languages like we were, we would agonize over the keywords. This is how people are going to express their ideas and code for years to come, we can never change them. Because you [crosstalk 00:41:05]

Dean Leffingwell:
Domain models.

Mik Kersten:
Yeah, domain model. Exactly.

Dean Leffingwell:
What's the domain model [crosstalk 00:41:08]

Mik Kersten:
Exactly. Exactly.

Dean Leffingwell:
[inaudible 00:41:08].

Mik Kersten:
Yep. So we would come up with the best thing that we could. But we put such care and thought into the taxonomy. And there were endless discussions of people itering on those things. And so I think, again, the ability to tap into all of that, and the fact that it is evolving is just so critical, because that common terminology means that you can stop worrying about that, stop having discussions.

Mik Kersten:
And you and I've seen this over and over where, during a meeting where people are supposed to be discussing the value they're delivering, they get caught up in terminology discussions, and what's an epic. And I cannot stand a what's an epic discussion when I see customers doing. We've gotten over it long ago. So I think it's just about getting out of your way and moving ahead.

Dean Leffingwell:
This was reinforced by me some years ago where we were working at a big German bank and we were meeting with them and a bunch of other executives off site. We were just approaching the localization issue. And we talked about the potential to localize SAFe. And we've not done that. The VP stood up and

says, "Whatever you do, do not translate these iconic terms into other languages because they will lose all their meaning. Lean portfolio management has no obvious translation in Mandarin and whenever you put into it is going to create the equivalent baggage that we have with a word program increment. So make that stay. Now if you want to translate the definition of that, or in case of Japanese use katakana to describe how to pronounce that, great, but do not give me a different word for an epic in Vietnam, China, India and Germany. It's going to be an epic, that's what we're going to call it."

Dean Leffingwell:
So we got a stern lecture about how valuable it was for them to be able to be on the phone with their teams in Vietnam, saying what's the status of these stories? Or what are the features in the backlog? What do you say ... what are the things in the backlog that are in some cases or features and maybe they're not, maybe they're stories, maybe they're initiatives, you got to have a way to approach the problem to be able to communicate. If you get that out of the way, now we can figure out what the features should be not what to call it.

Mik Kersten:
Yeah, exactly. That is a fascinating point, right? It's actually similar to programming languages. You don't internationalize them those words like class or method, they happen to be pretty important in a universal way, and took that much effort.

Dean Leffingwell:
Remember your first efforts, and all what you doing had to do the domain model?

Mik Kersten:
Yeah.

Dean Leffingwell:
All of a sudden, it's like, what kind of report is that?

Mik Kersten:
Yeah, that's right.

Dean Leffingwell:
It's the same thing. It's just the basic for communication across people that need to work together to a common purpose.

Mik Kersten:
Yeah. So Dean, the terminology and the goal of this is to get people to think in terms of outcomes and delivering value and business agility. And so I'll just tell you a quick story. I hope this one's not too contentious, but I'm going to just go straight to the times that I hear someone blaming SAFe for the rollout not being successful, or transformation not being successful.

Mik Kersten:
And this one's somewhat representative. So what happens is they start the roll up, and I'm being told that it looks good, it's working, they're getting some training and so on. And then the practices start, people change nothing, all they're doing is call it, painting it SAFe. And they start using some of the ceremonies. Of course, Agile deployment evolves into this, but let's just blame SAFe, they start blaming the ceremonies for a lack of focus on results. And of course, they're still arguing but the ceremonies as they're doing this.

Mik Kersten:
And so this executive at a large financial services company is saying, "SAFe to me that it's not working." I asked him, I said, "Well, the is SAFe not working. Why is SAFe not working? [inaudible 00:44:38] frequently I see it not working sometimes. And it sounds like it could be similar reasons. It works for us. Why is it not working for you?" And so he says, "Well, because I think you and Dean think that people are too outcome oriented." And I think what we've got people here is that they've taken the very surface of it. And then of course, he says, "Well, there's nothing actually wrong with the SAFe, the concepts are great, but the problems are people. They're taking the surface concepts of it. They're celebrating the ceremonies, they're only measuring the ceremonies. And they haven't actually made anything more agile in the process." So how, how do you think about this? How do you, again, get that kind of mindset over to the true SAFe the business agility mindset?

Dean Leffingwell:
Well, it's a common problem, I think, with frameworks in general. I was involved in a roll out of Scrum for I don't remember how many of you like 1,000 or so and it just simply wasn't successful. So then enterprise says, well, Scrum doesn't really work. Well, we didn't change Scrum, Scrum is Scrum and it freaking works for sure. They couldn't apply it. That was really a different issue. You can do that with anything. You could do waterfall badly, waterfall well, you could pick any branded thing, wrap any of the other frameworks in our world and do them badly.

Dean Leffingwell:
I agree. We call it kind of painting the place safe. And I think especially the danger as you get into the middle market, and the later adopters who need to go Agile, fast, they really need to make progress fast. So they start calling things safe and hoping for the best.

Dean Leffingwell:
Well, there's a few things that I think can correct that. Number one, if you focus on the principles, you're going to get better inspect and adapt. You've got to make sure that you do that corrective action loop.

Dean Leffingwell:
In addition, we do have some really outcome oriented measures associated with SAFe there's portfolio metrics, there's Agile release train metrics, as well. And that includes one that you might think isn't really an outcome. It's called predictability. Well, guess what, if your teams and arts aren't predictable, you're gonna have a heck of a time achieving any other objective. So I would focus on the outcome metrics, not the outputs and the processes.

Dean Leffingwell:
In addition, it's maybe a minor thing, maybe it isn't. We've never had what I consider to be a great treatment of KPIs at the value stream level. We've actually added that since 5.0 was launched. So a couple things to look at, I would look at the Agile portfolio metrics and I would look at the KPIs and say if we measure our business that way, those are clearly outcomes, not outputs and not process steps.

Dean Leffingwell:
Now, having said that, just like Scrum, you don't go into a lot of theory as to why you do a daily standup, you just do it. And those ceremonies have value. So I wouldn't say that measuring whether or not you got PI planning done on time is a terrible thing is just an intermediate step. And you have to get the muscle memory for that. So PI planning is a good example. Not easy to do that with 100 and 150 people. First couple of times, it's 50%, or 60% process and 30% contact. The fourth time, it's like, well, the process is

no longer concerned the taxonomy is out of the way. It's now just reasoning about the system that were being built.

Dean Leffingwell:
So the ceremonies are important and Inspect and Adapt that's an important thing. The system demo that's critical. But how do you measure the effectivity of a system demo by the number of people that came? No, that's just a data point that should help eliminate some of the other reporting you have to do.

Dean Leffingwell:
In that regard, I had another recent experience that I think really highlights a challenge, which is that hybrid models are probably the worst of all. So when you start doing things like SAFe and you say we have a system demo, and we have PI planning, and the teams do backlog refinement, if you don't throw away and stop doing all the other things you were doing to address the problem, you double up. And you can end up with more meeting overhead than you had before.

Dean Leffingwell:
But if you do an analysis, and one of our SAFe fellows did in one enterprise. He counted the time spent in meetings and the number of meetings before SAFe and after, and made the case through objective evidence that their meeting time had dropped by about half if, I remember right. And that's because they threw away the stuff they didn't need to do.

Dean Leffingwell:
A system demo, for all practical purposes is 80, 20 of the reporting problem. How are we doing on GDPR? Come to the demo and ask that question. How are we doing on this getting ready to launch a satellite? Come to the demo and ask the question. And if the demos are too frequent, come to the PI boundary because you're going to see a demo there.

Dean Leffingwell:
So if we focus on objective evidence of the system, not the ceremonies, we focus on the principles and we focus on KPI like outcome measures, I think you can get past that. Again, people want really easy answers really hard problems. I ran into this maybe only 24 months ago, a Scrum team of 30 requirements analysts.

Mik Kersten:
Sorry. Taking a moment for that to sink in. Never heard that before.

Dean Leffingwell:
I know, this is the podcast that you can't see Mik shaking his head. This was 10 o'clock in the morning of what was to be a two day consulting engagement. And I'm thinking maybe I could catch a one o'clock flight, I can't [inaudible 00:49:44]. And this was promoted by their Agile working group. So it's easy to be wrong, it's not easy to be right. You have to have the right training to be right. You have to have the mindset. We reinforce that you've seen the Agile Manifesto. Yes. 19 years old. That's still really good. We reinforced the principles of flow. That's a new decade of thinking. We have reinforced systems thinking, we have reinforced decentralized decision making. Those are the things that are truly universal in nature and can get you out of the just painting it SAFe box. But I agree it's a risk and people fail to say, "You bet."

Dean Leffingwell:

Can they fail with any method or practice? Of course, you can, that's really ... but it's the poor workman that blames their tools. It was a poor company that blamed Scrum, I think it's a poor company that blamed SAFe, because they're not getting the outcomes they need.

Mik Kersten:
Yeah, exactly. I think you hit on all the ... if I look for the success patterns, where they're getting rapid time to value, I think you hit on all three of the most common ones I see. As not doubling up, so not getting rid of your EVM and PRDs and project management if you keep both-

Dean Leffingwell:
This is reporting.

Mik Kersten:
Yeah, and reporting. Exactly, this is meant to make things go away. If you keep both of course your teams and your people will react. That doubling is a disaster. And this is I think ... you said it Dean in the way I haven't heard before is, from the roll up, you should be able to reduce the meeting time, reduce the ceremony, reduce the reporting.

Dean Leffingwell:
In PI planning, one of the nice things about having some experience of this like we both have is, we remember JAD, right? Joint Application Development.

Mik Kersten:
Yeah.

Dean Leffingwell:
Where every month or so we get everybody together, and we would do the requirements and the design and the implementation and demos that we didn't mostly do demos and mostly design sessions. But PI planning session is not just planning, it's just really hard to describe what happens there. It's requirements, analysis, communication, design, architectural, program planning, outcome measures, goal setting, et cetera. It's basically everybody that's involved in building the system meets together for a day and a half to two days to figure out what to do next.

Dean Leffingwell:
Is that an overhead ceremony? I think it's requirements and design and planning and governance all rolled into one and they're doing that ... their work has to be done anyway, the question is how many separate breakout meetings it takes, to interrupt your schedule to do what could be routine. The system demo is a good focus. Every two weeks in system demo, try to do that without DevOps. You can't. You demoing from people's laptops.

Dean Leffingwell:
So if you do the demo, if you do Inspect and Adapt at the end of every increment, you're going to get better. And if you don't, you're not.

Mik Kersten:
Exactly. So Dean the last question, you hit on this one, a lot of people are asking about this, it's how to think about DevOps and SAFe. I think the industries realize how important DevOps is, it's everywhere, but alone, you're not actually getting to its principles, right?

Dean Leffingwell:
Yeah.

Mik Kersten:
You're only solving the dev to ops problems, the CICD pipeline-

Dean Leffingwell:
Absolutely.

Mik Kersten:
... you're not actually getting the flow and feedback and continue learning. So, thoughts?

Dean Leffingwell:
So the label DevOps, isn't that old. I don't know, maybe, I don't remember if it was a phoenix project that pioneered that or not. DevOps is about primarily code to deploy. And if you can't do code to deploy, how agile are you? Well, the answer is not agile at all. What I started seeing the industry is people saying, well, we're going to do scale[BATs 00:53:00] or we're going to do DevOps. Well, kind of a dah! What you're really doing is simplifying the value stream. And DevOps is front and center. And if you think about on a slightly higher level view, how do you do innovative product management work in design thinking if you can't get feedback. And I don't want just feedback from a storyboard or a wire frame, I would like to actually deploy that thing in some portion of the market.

Dean Leffingwell:
And in the unicorn project, you saw that with the AB testing of large scale data systems. Well, if you don't have DevOps, you actually can't get feedback. So you're going to have a fun time doing all your innovation design thinking, create some ideas, create a big batch of stuff, hit a firewall where you can't deploy and get feedback. But if you can get the DevOps pipeline smooth and lower the transaction cost of deploy, then you can run smaller batches through deploy and the economics start to work. But if the testing and overhead of testing a deployed system is so high, you can't afford a smaller batch, the economics are against you, and that forces you into really big batches of requirements.

Dean Leffingwell:
So DevOps is just integral to the value stream. It's integral to what we do. For me, it goes back to XP. So DevOps reinforces continuous integration. Well, XP didn't invent continuous integration, but it demanded it. Well, that's just a piece of DevOps. And maybe, yeah, maybe I was integrating into it at dev branch and not in the ops. But it was the same sentiment, which is, if we can't get fast feedback from the small batch, we don't know what we're doing. DevOps, I think is a brilliant movement, we're lucky to have it. Because it says creating a big batch of largely on delivered Agile code is no more fun, maybe even more painful than creating a big batch of waterfall code.

Dean Leffingwell:
So DevOps is actually integral to value stream, it's integral to SAFe. It's been there one way or another for ever, calling out specifically ... I don't think we did that until version four. I don't really remember exactly when we said we got to put some surface area on that, to make people understand that the [inaudible 00:54:57] slavery pipeline is part of SAFe. And it's what people have to do to shorten the time not from code to deploy, but from idea to deploy and feedback.

Mik Kersten:

Exactly. And I think my introduction to Theory of Constraints was actually from Kent Beck's Extreme Programming. He had an amazing chapter on that. And the whole idea was you invest in the constraint. Maybe your constraint is CodeCommit to CodeDeploy. If you don't have a bigger view, you don't know that. And chances are, as we've seen, oftentimes, as Dean and others have pointed out, it could be the architecture, the software architecture, it could be the work intake, it could be upstream bottlenecks, upstream of CodeCommit and CodeDeploy.

Mik Kersten:
So I think, again, having that operating model actually allows you to discover that, measuring these four metrics and these KPIs and outcome metrics. While you're putting in place this new process that makes all of that much easier and gives your organization the right structure and taxonomy is the very clear path forward.

Dean Leffingwell:
If look at SAFe and go, "There's a lot of stuff there." I asked the question, "What would you take out?" Well, we probably don't need that customer centricity thing. Well, maybe we don't need DevOps. Maybe we don't need to combine for flow. Maybe we don't need a demo. Maybe we don't need a spec in that, maybe we don't need lean portfolio. We kind of need all that stuff. So yeah, it's apology and an immediate retraction. You got to do that stuff if you really want to compete with the companies that are good at digital, you got to do it all. I wish it was easier, then again, I guess maybe you and I wouldn't have so much fun in our careers.

Mik Kersten:
Yeah, no, I think we thrive on making complexity easier. But it doesn't get fundamentally ... This is fundamental complexity, in terms of what it takes to build software at scale.

Dean Leffingwell:
And you said, I don't think we're going to invent a programming language to solve this problem. And maybe [inaudible 00:56:37] and others think that maybe AI will make this programming obsolete, but I'm living in a different world where there's 20, 30, 40 million lines of legacy code. And we've got to do new value through that stuff fast. And that requires some pretty serious thinking, that requires a framework that's got some mass.

Mik Kersten:
Yep. no, I think you might have another role here of people doing models for ML and SAFe 10,.0 or 8.0, but for now that this is around the way people work and the way they deliver value, so I think AI is not making this go away anytime soon.

Mik Kersten:
All right, Dean, thank you so, so much. I learned lots of things. Whenever I talk to you, I think I learned more on this discussion than ever. So-

Dean Leffingwell:
And the reverse is also true. I enjoy our learning together and challenging is what we do. That's what makes you grow.

Mik Kersten:
Excellent. All right. Well, thank you, everyone.

Dean Leffingwell:
Take care.

Mik Kersten:
And until next time, Dean, where can people find you online?

Dean Leffingwell:
Yeah. LinkedIn.

Mik Kersten:
Excellent. Thank you so much.

Dean Leffingwell:
Okay. See you later.

Mik Kersten:
A huge thank you to Dean for joining me on this episode. For more, follow me and my journey on LinkedIn, Twitter or using the hashtags Mik + One or Project to Product. Dean's Twitter handle is @deanleffingwell. He's got some amazing content that he's posting regularly as well. If you want to learn more about flow metrics and SAFe go to flowframework.org.

Mik Kersten:
I have a new episode every two weeks, so hit subscribe to join us again. You can also search for a Project to Product to get the book, and remember that all the proceeds go to supporting women and minorities in technology. Thanks. Stay safe, and until next time.