Transcript provided as a free resource by 

Data Mesh Radio Episode #202: Creating a Balanced, Sustainable Approach to Your Data Mesh Journey

Interview with Kiran Prakash

Listen (link)

Transcript provided as a free community resource by Starburst.
To check out more Starburst-compiled resources about Data Mesh, please check here: https://www.starburst.io/info/data-mesh-resource-center
To get their Data Mesh for Dummies book (info gated), please see here: https://starburst.io/info/data-mesh-for-dummies/

List of all previous episodes (including links to available transcripts) is available here.
The folder of all previous transcripts is available here.

0:00:00 Scott Hirleman
The following is a message from George Trujillo, a data strategist at DataStax. As a reminder, DataStax is the only financial sponsor of Data Mesh Radio, in the Data Mesh Learning Community at this time. I work with George and I would highly recommend speaking with him, it's always a fun conversation.

0:00:19 George Trujillo:
One of the key value propositions of a Data Mesh is empowering lines of business to innovate with data. So it's been really exciting for me personally, to see Data Mesh in practice and how it's maturing. This is a significant organizational transformation, so it must be well understood. Empowering developers, analysts, and data scientists with downstream data has been part of my personal data journey that reemphasized the importance of reducing complexity in real-time data ecosystems, and the criticality of picking the right real time data technology stack. I'm always open and welcome the opportunity to share experiences and ideas around executing a Data Mesh strategy. Feel free to email or connect with me on LinkedIn if you'd like to talk about real time data ecosystems, data management strategies, or Data Mesh. My contact information can be found in the notes below. Thank you.
LinkedIn: https://www.linkedin.com/in/georgetrujillo/
Email: george.trujillo@datastax.com.

0:01:11 Scott Hirleman
A written transcript of this episode is provided by Starburst. For more information, you can see the show notes.

0:01:18 Adrian Estala
Welcome to Data Mesh Radio with your host, Scott Hirleman, sponsored by Starburst. This is Adrian Estala, VP and Field CDO at Starburst and host of Data Mesh

TV. Starburst is the leading contributor to Trino, the open source project and the Data Mesh For Dummies book that I cowrote with Colleen Tartow and Andy Mott. To claim your free book, head over to starburst.io.

0:01:48 Scott Hirleman
Data Mesh Radio, a part of the Data as a Product Podcast Network, is a free community resource provided by DataStax. Data Mesh Radio is produced and hosted by Scott Hirleman, a co-founder of the Data Mesh Learning Community. This podcast is designed to help you get up to speed on a number of Data Mesh Related topics. Hopefully you find it useful.

Creating a balanced, sustainable approach to your Data Mesh journey. Bottom line up front, what are you going to hear about and learn about in this episode? I interviewed Kiran Prakash, who's a principal engineer at ThoughtWorks. Some key takeaways or thoughts from Kiran's point of view. First one, potentially controversial, you must have exec sponsorship to move forward with your Data Mesh implementation. You need the top down push for necessary reorganization when those times come. A Scott note, personal note here, I don't think this one's actually controversial controversial. I just think it's one that a lot of people don't want to hear and that more people need to have this put in front of them. It's often ignored. Number two, another potentially controversial one, Data Mesh, if done well, doesn't need to have a huge barrier to entry. That's a misconception. If you think about gradual improvement and evolution, you'll be on the right track. So this one is about that people think that you have to have your entire vision laid out and you have to have buy in from everyone and that you have to have everything kind of teed up before you get moving. You can have it be pretty small and you don't have to be like, this is our new data strategy. It's like, this is what we're using to try and accomplish these few goals to start.

Number three, "the curse of the data lake monster." This was an article Kiran had put together in a concept and it was like the data field of dreams. There was this expectation that if you build a great data lake, value will just happen. If you ingest and process as much data as you can, the use cases will just happen. They'll just emerge. And it really wasn't the case. So we should apply that product thinking to focus on what matters. So he was thinking about this in 2019 and then Data Mesh kind of came along as well a couple of months after he put out his blog post. Number four, this curse was a manifestation of Conway's law. The strong separation between IT and the business led to mismatched goals and subpar outcomes. With microservices, that started to be much less of an issue on the operational plane. So why not try to do some of the same things with data?

Number five, it's easy to lose sight of Conway's law and aim for distributed

architecture first, but the organizations doing Data Mesh well are changing their architecture and cultural approaches and patterns together. Don't try to do the architecture first. You really don't know what your key challenges will be just yet. You don't know what you're going to need to tackle via your architecture. And so a lot of people are making decisions that aren't really going to help tackle their specific challenges. Number six, it's very important to have a target operating model and get clear on your organizational vision and purpose around data. How will you actually use data? How will this be part of, you know, like I talk a lot about data practice and you need to create kind of that data practice concept of how does this work in day to day operations? Number seven, once you have an organizational vision and purpose, domain should start setting goals aligned to that vision and purpose. You know, don't have them set goals before you have that vision and purpose. So once you start to do that, then you can start to do that around data.

Number eight, as others have noticed, don't get ahead of yourself. Work in thin slices for your Data Mesh implementation. Stay balanced at an overall level between the Data Mesh principles as you add more and more thin slices. But don't try to solve all problems up front. And if something doesn't require a ton of governance, that's not a red flag, right? If all of your first 20 use cases don't require very much governance, that's a big red flag. So, you want to stay that kind of balanced, but each one thin slice may be kind of differently balanced. Number nine, if you modernize your legacy software, but don't change the organization, expect to do the same modernization in about five years. Data how we're trying to approach data in Data Mesh is about using software as you know approaches and it's the same thing here. If you don't change the organization, expect to keep trying to do this same thing every, you know, three to five years.

Number 10, to really get to a scalable approach to Data Mesh, you should look for organizational and process reuse as much as you look for tech and architectural and data reuse. Number 11, move from measuring outputs to measuring value outcomes. Sounds simple. It isn't. And it's crucial to changing your mindset around how you approach data. Number 12, potentially controversial, a really key way to look at your progress is to use the four key metrics from the DORA, right, around DevOps to measure how well you are doing in your software engineering practices in general. A key aspect of Data Mesh is about applying good software engineering practices to data after all. Number 13, if you want to measure the value of your data work, you need to break it down into tangible objectives. Ask the owners of those objectives to provide the value of meeting the objectives. Then look to measure how much data work contributed to achieving those objectives, right? You need to think about this and break it down when you think about the value because otherwise people say, what was the value of this? What is the value of what you're trying to achieve? You tell me, and then we'll talk about how much data work actually helped you achieve

those or tackle those.

Number 14, think of a use case as a value hypothesis. You are making a bet that something will have value. It's okay to be wrong. That's the nature of betting, but limit the scope of your mistakes so you learn and adjust towards value instead of making big mistakes and then all of a sudden there's this big, you know, hullabaloo because this thing didn't go well, right? Make it so that these mistakes are the learning opportunities so you pivot towards the value. Number 15, another potentially controversial one, if you don't have a culture where it's okay to fail, it will be very hard to do Data Mesh well. Personal note on here, I think it will essentially be impossible to do Data Mesh well if you don't have a culture where it's okay to fail. Number 16, many times what people consider minimum viable product is neither minimum nor viable. This is often due to a culture where you can't test things with users when they're still very rough. If you don't have this, that will limit your success with Data Mesh. However, most people are reasonable. So if you read them in that this will be a rough sketch or first iteration, they usually are on board to help you iterate towards the good.

Number 17, "architecture is about the important stuff, whatever that is." This was a quote by Ralph Johnson that Martin Fowler likes to say. Number 18, always think about necessary capabilities and build to those. The most important are those capabilities you need now. Don't get ahead of yourself, right? Don't lock yourself in so you can't meet future capability needs. But as well, people are trying to focus too much on getting to all of the capabilities, because this is how we've always built data is to have all of the features, you know, all of the data platform, all of the data offerings, it's about the features, it's about the cool tech, it's not about the capabilities, flip that, think about the capabilities. And finally, number 19, the data platform is really a misnomer. There will be multiple platforms, users care about their services, not if you have one platform or five or more, right? Don't have platforms brawl where you have, you know, 20 different platforms. But don't over centralize. That usually leads to scaling and flexibility challenges. That's part of the thing that a lot of Data Mesh likes to kind of approach and help us solve. With that bottom line up front done, let's jump into the interview.

Okay, very, very excited for today's episode. I've got Kiran Prakash here, who's the principal engineer at Thoughtworks. Yes, another person from the Thoughtworks Germany team, but I really like talking with them. They've been putting out a lot of great content. We're going to cover a lot of different things, but kind of one of the things that we're going to talk about is this, where a lot of people have gotten burned of that kind of the curse of the data lake monster. And then when you start to say, we're going to do a new approach to data, then people start to go, didn't we already get this? Wasn't this a thing that we already tried and, you know, how that

difference and how we can think about that with Data Mesh. When you really are looking at a Data Mesh implementation, you need what Kiran was saying is a balanced stool, right? You can't go overly technical, you can't go overly organizational. You have to kind of have that people process technology and that you don't try to overcorrect in one way, but that it's also okay to go a little further in one at one point as long as you're catching up and that you're not getting super, super unbalanced because then the stool falls over. How do we constantly think about getting out of your own way and shipping value early and often?

This is something where a lot of people are treating Data Mesh not as an incremental build. They're treating it like a data project instead of data as a product and that you're building and that you get a minimum viable product out and that it's not, I have delivered the project and it is done. And then where should we look at building out the necessary tech for Data Mesh? Where to start, where people should actually look at it, how people can actually think about a data product and how you can kind of measure your progress as to whether you're doing well or not. So before we get to that, Kiran, if you don't mind giving people a bit of a background on yourself and then we can get into the conversation at hand.

0:13:30 Kiran Prakash
Yeah, sure. Hi, Scott. Nice to have a chat with you. And my name is Kiran. I'm a principal engineer from Thoughtworks based out of Berlin. I've been with Thoughtworks for close to 18 years now and started out as a full stack engineer. But over the last three or four years, I've been exclusively focused on data. I'm one of the senior leaders in our data and AI service line. And over the last two and a half, three years, exclusively on Data Mesh, I've been helping different clients sort of implement their Data Mesh consulting, consulting them on how to start their journey. And yeah, recently I've been playing the role of technical principal on arguably one of the largest Data Mesh implementations we have at Roche. I can't even take the name because the story is out there. Probably we have written extensively about it. Yeah, so that's my brief background.

0:14:23 Scott Hirleman
Yeah. And we had Ammara on as well and we've had Omar at Roche on as well, so.

0:14:28 Kiran Prakash
Cool. You had our full team here.

0:14:31 Scott Hirleman
Yeah, pretty much. So why don't we start with kind of your concept of what you've been calling the curse of the data lake monster and what that is and kind of how that manifests, but also like what does that mean? Why is that important when

people are thinking about implementing Data Mesh?

0:14:51 Kiran Prakash
Yeah, so the curse of the data lake monster is a blog post I co-published with one of my colleagues in 2019, even before Data Mesh was a big thing, right? Because one of the things I was seeing often done in many of our large clients was building data platforms, right? And their idea of building data platforms is install Hadoop or whatever the latest big data technology out there, just connect it to all your sort of operational system ingest terabytes of data and then sort of hoping that somebody will come and make use of it, right? And the second step of somebody coming and using the data to do something productive almost never happened. It rarely ever happened, right? And this is where in that blog post, we make an argument saying that, hey, why can't we apply the same product thinking principle to the world of data? Why don't we sort of build this in thin slices instead of starting it as yet another platform initiative and just measuring how many terabytes of data I'm ingesting and how many tables I'm creating and what not. But what about the value? How much value are you generating? How much business value are you generating? Is there anybody even using it to build cool stuff within your organization?

And this is what we call the curse of data lake monster because just we saw this pattern happening over and over and over again in many different organizations. I can think of a few reasons why this happened, right looking back, and one of the main reason is at least in large organizations, this separation between sort of business and IT, right? And business had their own problem and then sort of IT was a bunch of technically minded people and for them, everything looks like nails, right? So it's just one more platform away from solving all the problems within the organization. And for engineers, it was an easy thing to do, right? It was cool thing to do also like this new technology, which can process all this data and scale, horizontally scale and what not. And it was too much of an hassle to talk to business and talk about are they finding this useful? Are they using it to build anything, generate value, right? It was too much, right? It was basically Conway's law playing it out. There was this artificial separation between and business and platform, which I think is probably the primary reason.

That sort of changed with around 2014, 2015 with coming of sort of microservices, right? The organizations start thinking about having smaller teams aligned with their business domain who is sort of quickly responsive to the needs of that particular domain. And somehow like the world of data kind of stayed oblivious to it all along until very recently 2019, until someone coined this term and started talking widely about it. I think that's also because maybe the area of data engineering was sort of too specialized and it was a bit removed from the world of general software engineering, which was kind of embracing this practice of domain driven design,

microservices, working closely with the business domain, shipping sort of things faster, using cloud to sort of do all of this. And I think I'm glad that this is changing. I'm glad that people are kind of embracing this domain driven design within the world of data as well and starting to think about, okay, not just like building more platforms, but how do I work with the business to sort of shift value, create business value, right? I'm really glad that this is at least happening right now.

0:18:49 Scott Hirleman
And do you think then when we're looking at something like Data Mesh and that... Yes, it's applying some of the microservices practices and I love when somebody just says, "Oh, Data Mesh is just microservices for data. And it's like, "Oh, my sweet summer child." No, no, there's a lot more. But yes, that is an important pattern to take into to account. But like we've seen decentralization not work in the past because people have gone to data marts and there wasn't any focus on, you know, decentralize all the things instead of decentralize where it makes sense and centralize where it makes sense and have coordination or kind of have that federated approach where you have a centralized governing body of certain aspects, but that you're giving as much of the power as you can. You put it in the people's hands who actually have the context to make the kind of proper day to day decisions, and so that you're not having to... If you have to ramp up on context for every decision, then that's why all the decisions take so long and why often they're bad and why they're often defensive because you don't actually know all of the potential issues.

So it's like if I just feel like I don't know what are my unknown unknowns or whatever, then I'm just gonna say no. But when you think about what we went through with data lake, and you're talking to companies about Data Mesh and you start to say, "Okay, we're gonna decentralize this stuff after data lake was promised as this new thing where it's distributed but not decentralized architecture. And how is this so different? Are you finding that people are getting it because they just didn't see the value from the data lake or are they just frustrated from yet another approach? And it feels like... And this is why people think that Data Mesh is a technology or anything like that. Are you finding that it was a tougher sell two years ago and now it's not as much and people are really starting to get it? Or like I'm just trying to figure out like if somebody is considering it in their own organization, where do you think there is the kind of the failures of Data Lake in a lot of ways? Is that going to help or hinder them in these conversations?

0:21:17 Kiran Prakash
Right, right. First of all, I don't think... We are still very early on in the journey, right? I still don't find it sort of easy to convince somebody on Data Mesh. It's still... People need to sort of reflect on their own experience and then sort of be willing to try it and then sort of learn from that, right? So it's still not a given or an easy sell. We are far

too early in the journey. Having said that, wow, how this thing is different is if you look at the first two principles of Data Mesh, right? Which is about domain driven sort of domain aligned teams and data as a product, it has very little to do with technology. It is about organizational change. It is about basically aligning your data engineering team closer to the business where they can take the call.

And data as a product is primarily about applying product thinking to data is am I building the right thing which is valuable? Am I thinking about the consumer who will be using it? Am I thinking about the various modes of access that they'll be consuming this data from, right? And am I building something that makes them happy? All of this is sort of product thinking, and only later it is about technology change, right? It's about once you federate those teams, do you have the platform and the governance capabilities to sort of make these federated teams sort of successful. I think that's the biggest shift which we try to sort of communicate over and over again. And some organizations have started getting it, right. And this is why I think it could make a big difference.

And this is how it is different from, let's say, previous approaches to just sort of distribute your architecture in terms of like data lakes, many different data lakes on the cloud and whatnot. So this is... Data Mesh we say it's a sociotechnical paradigm. It's not enough if you just sort of distribute your architecture, but you also need to change your organization to continue on sort of evolving your technology, right? Otherwise you'll succumb to the Conway's Law, right? If you don't change your organization and if you only change your tech, pretty soon you will see that the tech will start reflecting the communication patterns of your teams. And that means that if you have like one of teams far remote from business, you'll soon fall back onto something that looked like the centralized data lakes or data warehouse.

And I see that people are sort of getting that message. And at least I, whenever I speak to new potential client, I try to sort of reinforce this aspect saying that, "Hey, adopting Data Mesh has implications beyond just technology and you need to be ready for it." Which also means that it... Like one of the implication of this is in at least in large organization a successful Data Mesh transformation needs some kind of a top down push or a top down mandate because it's not easy to affect these sort of reorganizations in terms of domain aligned teams and maybe changing your reporting structure for having your engineering teams not just be accountable for IT but also for business. These are all not sort of easy change. It needs some blessings from sort of top management to sort of make it happen.

0:24:42 Scott Hirleman
Yeah, there's not a magic wand that you just wave and this happens. But I think Vanya Seth was on recently a colleague of yours and she was talking about kind of

selecting your blast radius and being like, "Hey, we shouldn't have everything. We shouldn't put all of our eggs in one basket, we shouldn't try to completely rearrange the entire organization in just a very, very short period of time or anything like that." So how do you think about that and how do you think about like what a lot of people have talked about is thin slices. And maybe if you could define that 'cause it comes up a lot but I don't think it gets defined very often. But then talk about that kind of balanced stool approach as to you're adding your thin slices on, and if you keep having like tilted slice and they're tilted in the same way, you're not gonna build that balanced stool as you're going, but that you're also not trying to say, "Okay, we are going to need to reorg when we think about the way that we're doing this." But I'm not going to reorg the entire company to do Data Mesh, and I'm not gonna say, domains you now own all of your data versus enabling domains to own their data.

So if you could talk about that kind of early days and how this approach and thin slicing really works and how people can actually think about, "If I'm early in my journey, how should I think about it?" And maybe even as they progress in their journey, how do you keep adding that it's only thin slicing that you're not all of a sudden trying to go into big scale project mode instead of kind of that incremental product mode.

0:26:25 Kiran Prakash
Yeah. Yeah, I think that's a really good question and I'll take my time to sort of probably answer this because one of the things I'm seeing there is people sort of perceive Data Mesh as a huge barrier for entry, right? You need to do all of these organizational change, change your reporting structure for it to be successful, right? While we keep saying the opposite, right? So you need to be clear about your target operating model. You need to be clear about where you are going, right? So we tend to use an operating model, which is based on a model called Edge, which basically the main concept there is that you need to have your organizational sort of vision and purpose clarified. Like how do you want to use your data to do what and what is your strategy to sort of get there, right? How do you want to leverage data in your organization? Sort of have that purpose vision sort of clarified.

And then sort of tell your domains to come up with sort of goals, which is aligned with your broader organizational vision. And also come up not just goals, but also value hypothesis in terms of how do they want to achieve these goals and how do you measure when you sort of you're moving in the right direction. So how does it look if this hypothesis sort of comes true, right? It could be things like, "I reduced my lead time from X days to Y days, or it could be that I reduce my inventory sort of holding cost by X dollars." So it could be like, "This is sort of the hypothesis you're making and the measure is this thing like lead time or holding cost or customer satisfaction, anything that you can actually measure, right?

Once you do that, you can give the domain sort of autonomy to shape their approach. Like what platforms do they wanna build on? How does it align with the broader sort of data strategy of the organization and what is the first thing they're going to build? What is the first use case they're going to address? And then set up teams who will go about sort of realizing it, right? And this is the target, this is the target operating model. This is how you want your organization to function in a, let's say mid to longer term. And you're not gonna get there on day one. And we say the way to get there is via thin slices. And this thin slices is not just the thin slice of software you're gonna build, but it's the thin slice of what is the first governance practices you're you're gonna put in place? What is the first thin slice of platform you're going to build so that these data product team can function?

So it's a thin slice of all of these aspects, like thin slice of what is the software and data product you're going to build? What is a thin slice of platform you're going to put in place? And what is the thin slice of maybe the governance and organizational change you are going to affect in the direction of your target operating model? How you eventually see your organizations sort of work. The way to arrive at this thin slice, at least in the what we say is like, "Work backwards from your use cases." You made those value hypothesis, right? Which is like, if I build this data product, it'll help me achieve this goal, right? And you can sort of elaborate that into use case which is, I would want to build this data product which helps my back office team to make this whatever decision. Take this action based on data, and then from there, work backwards to identify what data products you need to build to satisfy this use case. What source data products do you need? And then identify like the thinnest possible slice you can build. Which you can put in front of that back office person or whoever your target user is to sort of play around. It could be the beta users, it could not be the final thing you'll end up building, but it could be something that the customer can have a look and give an early feedback on, right?

And then think about sort of what platform capabilities I need to put in place. It could be simple things like storage, compute and maybe some access control sort of mechanism, right? Those are maybe the only three things you need as part of your platform to begin with. And as part of governance or and operating model, you start kind of defining what a data product owner role looks like, and you start identifying people and giving them that responsibility, right? So that could be the first thin slice. It's that you can start your journey on and you almost want to sort of evolve your Mesh one thin slice one domain at a time. You don't want to sort of build it all at once, but you want to evolve with one thin slice at a time as I defined just now.

0:31:00 Scott Hirleman
So two different things that I'd like to go down, but first let's start with I wanna get to

the thing where you talked about kind of measuring that value and kind of thinking about it ahead of time and then measuring, did we meet that and was that the data's fault? Was that our hypothesis fault or what, you know? But let's talk about how you are when you are adding these thin slices, you can get unbalanced between people, process and technology. And even when you're doing thin slicing you can. So can you give some examples of what goes wrong when you're too focused on organizational versus technology or technology versus organizational? Like what does that start to look like? And then we can talk about like how do you measure if you are staying balanced and how do you correct yourself if you find that you're not balanced?

0:31:53 Kiran Prakash
Sure. So maybe let's look at the most common pattern we see, which is framing this Data Mesh as like a purely technology initiative, right? So if you do that, what you might find is... I have this like famous quote which I like, right? Not famous, but something I read which is, if you... This was in regards to sort of legacy migration, and it applies to maybe the Data Mesh context also which is, "If you change your legacy software and do nothing to change your organization to sort of operate in the new mode, expect to do this legacy modern organization again in sort of five years", right? Because those sort of old habits is hard to let go. If you only sort of introduce this new architecture and do not sort of introduce this domain oriented data teams and product thinking, you can surely expect like in few months or few years down the line the same old habits of sharing data over a network drive and doing shadow ITs and building more platforms instead of building useful products, you'll start seeing this pattern recurring, right?

So you need to pay attention to this also. Like yeah. You are building... This is a technology shift, right? You are kind of moving from centralized architecture to kind of a more of a decentralized architecture. But once you do that, you need to change your organization to make sure that these changes kind of stick and your organization sort of doesn't fall back to the old habits. So that's the one thing. Focusing too much on technology and framing this as a technology shift. But if you only sort of frame this as a organizational or a governance change, then your federated data product teams, they may not have the necessary tools to operate in this new federated environment, right? So that means that you may find different data product teams solving the same problem over and over again, right? Or each of them sort of independently figure out what is the best way to compute, do computation over the data, right? Like to use X tool or Y tool and how to do access control. How do I monitor my data products and pipeline, right? So they'll start sort of refiguring this... Solving these problems all over again and that's a waste, right? You want to provide them with a platform which makes it just easy to build and operate this data product and the product teams can only focus on creating value and not

just solving hard engineering problems over and over again, right? This is what you want to achieve.

So basically if you kind of federate your teams and don't give them the ecosystem and tools to operate in this new environment, you'll see a lot of waste. You will see basically, yeah, people solving the same problem over and over again, or they build data products and each, but each one of them kind of mean... When they say data product, they mean a different thing. And you'll find that these data products are not interoperable, which kind of defeats one of the main purpose of Data Mesh which is that you're building this reusable data product which you can interconnect with one another and to create this network effect of sorts, right? So those are the risks. So you need to pull on both these levers at once to see that the Data Mesh is kind of a successful transformation sort of effort within your organization. You can't just sort of focus on one or the other.

0:35:40 Scott Hirleman
Yeah, it sounds like a lot of what you're talking about is almost capability silos of, if everybody is doing the same things over and over, they're gonna do them in different ways because they're trying to get to their end outcome and...

0:35:51 Kiran Prakash
Correct.

0:35:52 Scott Hirleman
And it's not even... It's waste. Think of reuse from capabilities and ways of working and things like that. It's not just the... This is the thing that's come up more and more is that people think of reuse just being about the data itself and just creating data products that are usable for multiple different use cases. And it's like, no, it's everything. When you think about product ways, you think about scaling, you think about you don't have everybody doing the same thing over and over. You think about specialization, you think about that as to within that domain, but you have those centers of excellence to kind of think about how do we enable people, how do we train people? How do we give them the capabilities to do this? How do we give them the capabilities to do this from a technical like technology standpoint that we hand them to do this easily and so that they are inherently interoperable or that it's somewhat difficult to make something that is not interoperable and that you provide them the ways of thinking and the capabilities and put that in front of them. I did wanna then say like how do you measure, how do you think about if you are unbalanced? How do you think about like seeing, "Hey, we've gone too far one way, or we've gone too far the other, and how do you course correct, right?" Like how do you measure that and how do you actually get yourself out of a bad situation?

0:37:20 Kiran Prakash
Yeah. And so how do I know what do I measure? This is a tough one because I'll tell you what not to do. And then I'll probably sort of think of what my area of what to do. What not to do is sort of move away from measuring outputs to value outcomes, which is, it doesn't matter how many terabytes of data you're ingesting, it doesn't matter how many data products you built, it doesn't matter how many lines of code you wrote, or it doesn't matter how fast your data ingestion pipeline is. What matters is, is it having the desired business outcome? I remember like one of the, basically, I don't know who it was on LinkedIn, one of the data leaders, he was saying his theme for 2023 is that, where's my money? That's the ultimate measure. All the engineering effort that your data engineering teams are putting, is it having the desired effect on business? If not, maybe then you need to do something better. That means that maybe it could be that the teams are not clear about what is a valuable thing to build, which is that communication between what is clearly important for the domain and the business, is not sort of distilling down to your engineering teams, you gotta fix that. You need to make that very clear and put in place to sort of incentivize them to sort build what is actually valuable.

And this is a problem which doesn't happen in startups, it mostly happens in organization because of this separation I was talking about. And you don't ever hear startups talking about how much data, how efficient is the data pipeline or how big is that big data. You'll only hear, are they profitable, are they... How fast are they growing, how well are they doing sort of targeted marketing and how well are they using data to drive new use cases. That needs to get into the front and centre, and then basically you need to question, that's the ultimate measure, but the problem is in large organization, the engineering teams may not see the direct connection. They see this, if you put the business metric in front of them and say that you need to do whatever it takes to achieve this, a lot of the time they don't see how their actions directly contributes towards it, because there is usually a lot more things needs to happen before you achieve that business outcome. Maybe there you can provide certain metrics which tells them are they doing job as efficiently as they can. I'm a big fan of these four key metrics from that DORA research, which talks about how fast is your pipeline and how often when you deploy do you introduce bugs. When something fails, how long do you take to recover from it.

And what is the... On an average, of how many bugs do you introduce for X number of commits. Those are the things which will tell them, are they doing their software engineering as well as they can. And we now have well defined research which says that if the teams are doing this well, they'll also be in a high kind of these contributions of high performing businesses and engineering teams. Maybe you can use that, like the engineering teams, they can use these metrics to make sure that they are functioning as well as they can, they have all the necessary tools. For

example, if their lead time is too slow, it could be because maybe they don't have the necessary capability on the platform to deliver the data product as fast as they could. Or if it takes far too long to get in to production, it could be because they don't have a good monitoring or test automation, you need to do that and you need to put your focus on fixing that, and also make sure that every other data product team has that same capability to adapt. That's sort of my long winded answer, it's like try to see if the ultimate measure is the business outcome you're expecting, but if you think, or you should also think about your engineering teams, are they being as efficient as they can. And you could use something like Four Key Metrics to measure that and fix things where if something is missing.

0:42:18 Scott Hirleman
Yeah, the only issue with that is that nobody has any benchmark right now. It's like, I'm benchmarking against, huh? But I did wanna go into that, I had a really good episode with Pink Xu at Vista on measuring value of data work. And her perspective on it, and this is the right perspective, is you can't, the data team or the platform team can do work, but they can't actually measure the value of the work, they can measure the impact and somebody else... Or that you create a framework for even measuring that impact and then somebody else places value on that. And then you ask them to place value on that ahead of time, and then you can measure whether that value of the impact actually made sense. What you would say is, okay, we're going to do this work and we expect it to reduce cycle times for a new data product from four weeks to two weeks. What is the value of that? They're not the ones that can say, and that creates X amount of value, they'd go to their partners and the business aspects, the people that are deploying data products and saying, How much value does this provide? Should we prioritize it? What is that?

And then you look at that and you say, one, did the data work do what we thought it would do, and if not, why? Oh, we reduced the cycle times from four weeks to three weeks, but it's still three weeks, and the reason is that the people just aren't data literate enough, and so we have to go and actually do one on one with them to onboard them every time, and even though the platform is ready to onboard them, it doesn't work. Is it the data works fault or is there an externality there? And then, but you would go to the partners and say, we're gonna provide you a way to measure what this impact is, but you're the one that has to provide value. How much value does it create to increase revenue 5%? Well, what are the other aspects of that 5% increase? If it's at negative operating margin, if I'm selling at a loss, maybe that's okay, 'cause I'm trying to take market share and that's what we're actually aiming for, so that 5% increase, yes, it was worth it. And so, yes, we wanna pursue that or no, our gross margin from a company standpoint, because this thing shot our margin up, our revenue up 20%, but it's at 50% margin. And the rest of our business is at 80% margin. All of a sudden our gross margin goes from 80% to 65% or 70%, or whatever

the math says. And that's a really, really bad thing.

How are you actually having those conversations? And historically, when people have tried to say, justify your data work, it hasn't really been... It's just kind of been handwavy because they haven't been able to say, "I can't measure the value of the impact, but we can help you measure the impact." How are you finding those conversations actually going because the finance person wants, what was the return on investment on this? And you're like, Well, I've got stories for you and you can go talk to people and say, this is what the value of that was, but I don't inherently have a dollar figure for you, and that, as a former finance person that can frustrate people, but you have to have some sense about that. Long long question, but how do you think about those different things?

0:46:13 Kiran Prakash
Yeah, this is a serious problem. As I said, having this lofty goals, the engineers on the ground may not sort of connect to it, they don't see how every commit they're doing contributes towards it. That's a serious problem. And I don't have a silver bullet answer for it, but what do we do? We try a few things. One is that, Okay, your business, your organization may have that longer large goal, but can you break it down into tangible goals for these domains. For example, let's say, let's take retail for example, of course, the ultimate goal is to sort of increase your customer base, sell more goods, increased revenue, profitability, what not, that's the goal of the business. But every domain within say retail, like the customer growth, they have a specific goal saying that, hey, this year we are going to concentrate on this demographic, growing our customer base within this demographic by X percent, which is a very specific goal, contributing towards an organization goal. And maybe your fulfillment department is just focusing on maybe decreasing the returns and the mistakes made in delivery and fulfilment because that is gonna contribute towards profitability again. And your let's say supply chain domain is focused on just making data-driven decisions on which suppliers to prioritize, let's say.

Here, each of your domains have a specific goal which is playing towards your organizational goal, and hopefully your domain is small enough that the data team can directly connect to it, and they don't have 100 things to worry about, they have one thing. Okay, I'm building data product which allows my marketing team to do targeted campaigns, and the measure is that, are we sort of increasing my customer base in that demographic? That's sort of easy to measure. Of course, even there, it is not a single straight line because you might be building the brilliant data product and generating insights, but what if your marketing campaign is not working? What if the content is kind of missing? Those things are there and there is no perfect answer to it, but hopefully breaking it down to sort of smaller domains and modules and having your smaller teams, the problem becomes easier to spot and unfix. And

even if something fails it only fails in something, a local domain and it doesn't spectacularly blow up, this goes back to that blast radius, you're kind of limiting the blast radius within a particular sort of domain.

That's how we think about it, which is, yeah. You still need to think about business, but maybe create smaller goals which are tangible enough that the data engineering teams can also sort of connect with and, yeah. And see... Do these experiments. Also make sure that it doesn't sort of, the feedback loop of once you do something and to know whether it has worked or not, it shouldn't take an year or two years, you should know in let's say a matter days or weeks. Try and shorten that feedback loop. Those are the things we can do, I think.

0:49:52 Scott Hirleman
And I'm trying to pin you down and I don't think, you said you don't have the silver bullet, you don't have the magic wand, but like, okay, the marketing team wants to increase conversion rate, they're gonna run an A/B test, and they're gonna try these different things, and you've got, you gotta measure your baseline, Sadi Martin was the one that was like, all these people are doing A/B tests and they're never measuring their baseline, what are you doing? But you do an A/B test and your conversion rate starts at 7%, and A is 6.85% and B is 6.7%. What was the value of not changing without running that experiment? The value of keeping it as it was and understanding that that data work had value, even though the answer was stay the course, don't change. This is where people get into, not experiments, they get into low hanging fruit, and you're only looking for low hanging fruit so that the only wins you're getting our sure wins and you're only going for very sure things, and usually those aren't the big, big wins, maybe you go in and you find a few big wins initially, but then after that, you're just playing small ball, you're playing uninteresting data work.

Again, I'm trying to figure out, have you run across something where the data work was of significant value, even though the end outcome wasn't of significant value? Where you go, our hypothesis about the end outcome was wrong, but that doesn't mean the. Itself was invaluable, or was not valuable. That's a frustrating English word that invaluable also means extremely valuable, but that it was not valuable, and so how do you communicate that because you're getting into this, the value of the work is in making good decisions, and so then how do you add value to or how do you attribute value to making the right decision, because when you have finance people, again, they wanna talk numbers, and you go, well, the value of not going into this market was that we didn't spend a bunch of time and money and effort because it would have been a bad market for us or to go after this segment, it would have been a very low margin segment or a very high churn segment or whatever. How do you get people on board with that? Have you seen that people are on board with that, or

that we're still too early in how we communicate that, that people really aren't communicating that fact that well?

0:52:54 Kiran Prakash
Yeah, of course we see that the time. Earlier, if you remember, while I was talking about Lean Value Tree and the Target Operating Model, I used a term value hypothesis, and we often use that interchangeably with use case. And the reason why we say it's value hypothesis is, of course, you're making a bet, you're making an assumption that if I do this, something good will happen. And when you're making a hypothesis, there is a good chance that you're wrong, but the trick is to make small mistakes, not to make one big mistake, but if you can make many smaller mistake and learn from it, you're in the right direction. Your process of making decisions is right. Your outcome, it's not in your control, it could always go wrong, your hypothesis could be wrong, or there are other externalities that is affecting the outcome, but you should do what is in your control, which is about making good decisions, and you also hinted at that. And last part of it is also, again, culture, it should be okay to sort of fail, and as long as your failure is limited and it's a small experiment you're doing, that should be fine, and you're learning from it.

That is also another culture shift of doing these smaller experiments within a stable product team. You're not just doing one big bang project, and then in the end you get something. It's like you're setting up product teams which are working on this hypothesis and doing experimentation and adjusting based on what they learn, or based on the feedback loop. And that's the best we can do, because you will be wrong a lot of the times, it's just that you should sort of... When you're wrong, you should make sure that your downside is limited and you're not spectacularly blowing up. That's the best we could do and that's also, for me, the core of what product thinking is, which is you're making these smaller bets, iterating faster, learning, putting software in front of your user, learning from it, and adapting.

0:55:14 Scott Hirleman
Sports betting and stuff, there's, you say, Okay, do I think this team's gonna win or lose, or here's the spread between these teams or whatever, but you think about a big, big underdog and you bet on them to win instead of win against the spread and you go, okay, they're gonna win, it has a big payout. And it's okay to make those bets if you understand, hey, I'm not gonna bankrupt myself by going for this crazy payout, there was one that I saw that was just kind of a meme thing of somebody had made this huge, huge bet, like a million plus dollar bet to win $12,000 because this team was so favored and it didn't go their way. And it's like, why are you making that big of a bet for something that's just not gonna pay you off that much, and has a huge downside.

If I can sum up a lot of what you're saying it's, so much of this is just communication, it's not that you say, here is the framework, I can just go into a company, that company, the people that are understanding the... That are asking for the ROI and things like that, of like, why are we making these bets? They have to understand here's the value of what we're doing and why these bets that didn't pay off are still of value. It's because we didn't make a big decision that then cost us. So you had two other things that I wanted to make sure we did cover as well. One was getting out of your way and shipping value early and often. I think a lot of what you've been saying kind of circles around this of just make small bets, get to things like you don't have to... I think of project versus product. The project, you're trying to deliver all the value at once. Instead of product, you're trying to build into incremental value. So how have you found that conversation around getting people out of their own way to ship value and ship early and often and to get to that incremental value and build?

0:57:30 Kiran Prakash
Yeah, I mean, this is the only way I know how to build software. I have always done this. And it still surprises me how uncommon it is when we talk to huge or big organization. And I think the last part is also this concept of, what's that, minimum viable product. I feel that's one of the most misinterpreted terms out there, which is it's not, especially we see that in a few of these big companies, especially if this project is IT driven, they don't want to put something in front of the business where they lose face. We keep hearing that so often. And the minimum viable product is neither minimum or a lot of the time not viable just because they made the wrong assumptions. And they didn't manage to test it in front of the user before it was too late. And yeah, and that's why we keep, at least, I started using this first thin slice because this conveys the term better. Especially in the context of Data Mesh, it's not just the first thin slice of the product you're building, but of the platform and also the governance changes you want to affect.

And we keep reiterating that, yeah, you want to evolve your Mesh one thin slice at a time. Yeah, it is a difficult message. But I feel like people do get it now, right? It's not a revolutionary thing anymore, especially when the product ownership and those things becoming more popular. It's about how do we apply the same principle to the world of data? How do you imagine the first thin slice, which is valuable? It's primarily a design thing, right? It's very easy to imagine this finally big thing, which is perfect and polished. But it takes some practice to start imagining what is the first iteration which goes end to end, which derisks a lot of the assumptions that we might be making. And how does it look? What does that first thin slice look? And who are going to be my users for it? And what is the first thin slice of the supporting platform and the governance I need to put in place? It takes some imagination. That's what we do. At least that's what I like doing, help our clients come up with that approach and consult them on what is a viable thing that we can build, which is still useful. But

derisk a lot of assumptions we might be making and put you in a path where you're able to ship those incremental value frequently and iteratively.

1:00:30 Scott Hirleman
So Andrew Pease had talked about, to get to actual data quality, you have to put something in front of your consumers, your customers, or whatever of the data. But it has to be of good enough quality where they're not going to go, well, I can't ever trust this. So are you finding that some of this is also upping the data fluency, the data literacy of the consumers so that you can say, this is not fully baked. We need to collaborate. If you are not ready to collaborate, we will not work with you. Is that the line that you have to draw in the sand where you go, look, this is how data works. It isn't a 1 or a 0. We have to be collaborative on this, or it won't work. And if you're not going to agree to that, that's okay, but we're not going to work with you. Is that as drastic of a line in the sand as you have to draw? Because otherwise, again, people go, well, this one thing's wrong, and so I can never trust this again, right?

1:01:40 Kiran Prakash
Yeah, I don't think it's that hard a line. I think most people are reasonable when we talk to them and start how this is done. As I said, people have difficulty imagining what it might look like. What is the first iteration of it look like? And it's about expelling that distrust or fear that, okay, this will be set in stone. No, this is not set in stone. It's probably set in water. Software is malleable. We can respond to your asks faster. And it's about giving that confidence. And once they start seeing it, once they start, the team is able to iterate, incorporate feedbacks, and change stuff fast. Most people do come around. It's about doing it the first time, which takes a lot of effort and convincing. But yeah, I think it is possible. That's what we try, at least, to do. Even if there are distractors, once you start shipping the first iterations, most people do come around when they see that this is a reality.

1:02:57 Scott Hirleman
Yeah, I think it's explicit implicit, right? If you get explicit with someone and say, you must understand that this is not set in stone, and we're putting it in front of you so that we can get it to a better place, like this is... Again, I'm not delivering a project to you. It's not, this is your meal. I have delivered it to you, and I'm going to go away. It's like, hey, we're working to make this something that's where we're cooking together, so. I did want to hit on the last point that you wanted to talk about, which is early in your Data Mesh journey, how do you start to pick out your tech? What's necessary? What are the capabilities that you see as necessary so that people can actually think about what is actually a data product? How can you make this so that it is easy to create and maintain the data products, that you're focused on that actual product lifecycle and things like that?

So you've seen a lot of these early journeys. I would love to just turn it over to you, and you can just kind of wax poetic about what are some good things for people to actually think about, and what maybe what are the things where you go, you know what leave this for later, or leave this on manual mode. Don't try automated access control via APIs and things like that. I think people try and do that way too early, because it's one of the things that Zhamak is coming from the operating side of. We got to a place where these APIs do have automated access control, and there's privacy by design in them, and we need to get their data. But we're nowhere near, we don't even know how to really do analytical APIs very well. Do you want to paginate through, hey, I'm going to pull 100,000 records, I'm going to paginate through and grab each record individually, or can I run the filter and grab that as one? We don't really have a great way often of writing that in API form, so. But what are the things that you recommend people focus on, and maybe what are some antipatterns? And just kind of turn the floor over to you of when people are really thinking about early in their journey, what's your recommendation? What's your kind of advice? And where have you, what antipatterns can you help people avoid?

1:05:17 Kiran Prakash
Yeah, I think that's an interesting question, which goes in the direction of basically what is architecture and what kind of architecture decisions you need to make early on, right? I like this loose definition, or definition that by Martin Fowler, where he says architecture is about the important stuff, whatever that is, right? So you want to make these architectural decisions, and the good decisions are those which are basically, which are... So good architectural decisions are those which are kind of harder to sort of reverse. Like if you take, if you decide on something and it's sort of harder to decide later on, so you want to make those decisions now. Some decisions are easy to reverse. You don't have to worry too much about them, right? So I'll give you an example. So for most large organizations who want to sort of going down the lane they sort of, going on the Data Mesh journey, you need to be on one of the public clouds. I think it's very difficult to build something like this if you're in a completely sort of data center that you're managing on your own. You need a lot of, I think it's, I would say like close to impossible, right? You need to be on some sort of a public cloud, which gives you this horizontal scaling capabilities, infrastructure as code, and security by design. So that's the first thing, right?

So you probably, it's a good idea to be on some public cloud if you're a large organization, because this decision is going to be sort of harder to reverse later on. You may have to sort of rebuild a lot of things. Once you're there, don't sort of go too big about, okay, putting the perfect platform with all of the capabilities all at once, which you're hinting at, right? So let's say if you have one data product team, like your first data product team in your domain. So what do they need to ship value? They probably need some storage on your public cloud. They need some way of

computing things. And then they probably need a way of exposing the data to its end users. And the end users could be other engineers who are building sort of APIs, or it could be your data analyst who would want to do some ad hoc analysis on the data, like data scientists, or data scientists who would want to sort of build some ML models on it, right?

So what kind of capabilities you need? You need storage, compute, some way to sort of publish it and control sort of access on it. And if you're doing ML, you need some way of basically building those models and publishing them and putting them to sort of use, right? So build that capability so that you kind of build that end to end use case. Or you need probably some capabilities to ingest your data, especially if it's coming from, let's say, your legacy on premise systems. You need a way of securely sort of transferring the data from your operational systems into the cloud. So those will be the basic capabilities you need in the first place to ship maybe the first use case. But once you start to scale, once you start having more domains on board, you then run into a problem where the teams need to discover what other data products are there so that they can build on it. And you also need a way of establishing sort of consensus such that the teams, when they say something is a data product, they mean the same thing, which means that you need some maybe a centralized catalog where you can represent what this data product is, where you can publish what this is about, and people can sort of read up on it and consume it on their own sort of basis, right?

You also probably, once the people start using this data product and it goes into production, you also need capabilities where the teams can monitor what is happening and react on failures. So you add that, right? You see where I'm going. In the beginning, you don't need all of it. You start with basic compute storage, some ingestion mechanism, and maybe a workbench for your data scientist and your data analyst. And then you sort of gradually build. You introduce a catalog once you have more domains team. They need to discover. And then you kind of harden the definition of what a data product is. So you need some kind of a metadata management tool. And then once your consumers sort of grow and they start using your data products in sort of mission critical sort of use cases, then you need good monitoring and alerting system so that you can react faster to the failures and fix them, right? And then as you grow, you need to sort of put together all of these things about fine grained access control and the way to automate it, right? And I'll be also honest and say that not all the technologies you need to have this federated team work seamlessly, it doesn't exist. We are kind of building it as we go along.

One of the main areas is how do these teams sort of share data, especially if they're on some sort of a polyglot storage? Like let's say one team is on Redshift, another team is on Snowflake. How do they sort of seamlessly share this data product

without having to physically copy data from one cluster to another? That's a gap that currently exists, right? And there are no easy answer for it. There are workarounds, but there is no sort of very good answer for it. And also things like how do you do sort of very fine grained policy as code, like access control and defining policies as code? There are a few tools out there, but sort of nothing is perfect. All of this is evolving. I'm pretty sure as this technique becomes popular, these tools will start to emerge. That's what we saw happen with microservices, right? Like in the beginning, there was nothing. We were just deploying these microservices on whatever the web server we had. And then the Kubernetes came along, which made it easier to sort of manage this cluster and deploy microservices as a spec.

And then we saw that the service mesh came along to kind of handle a lot of the common concerns, like crosscutting concerns with these microservice team, things like monitoring, logging, and access control. So you saw that once the technique became popular, that the tools evolved to fill that gap. And we are yet to see that with Data Mesh. It's still very early on. We are kind of fashioning the Data Mesh on the tools that we had for traditional data engineering and data warehouses. But things are changing. As these kind of techniques become popular, you will see we have already started seeing the tools that are specifically designed to cater for these sort of autonomous data products, which we say is kind of the fundamental building block or an architectural quantum of the Mesh. How does it actually live and transfer data between different data products? We are seeing different kind of solutions and technologies emerge for it. But it's still sort of early days, which also means that you don't want to over engineer your platform. You want to stay flexible. You want to do the bare minimum you need to solve the problem at hand and keep your options open for these new technologies that are emerging, which makes this much easier to build and operate data products.

So yeah, that's my kind of approach. So don't imagine one big platform which is going to solve all your problems. Stay nimble. Just build enough capabilities to cater to your existing data product teams, and then keep your options open because this technique is only now becoming more popular, and we are seeing tools emerge to cater to that need.

1:13:43 Scott Hirleman
Yeah, and one thing that I would extract out of there, and I want to make sure that I'm not putting words in your mouth, but that you talked about not one big platform. And so you think about we're providing services, right? The users do not care if it is one unified platform or if it is 15 smaller platforms that work together to offer them services. They care about their user experience, and can they achieve what they're trying to do? So trying to overengineer things and create the one platform to rule them all is so counterproductive and such a bad antipattern, but it is the way we've

thought about things because we've had the one data lake to rule them all. But when you're thinking about decentralized data, you think about not just distributed systems architecture, but you think about decentralized architecture as well, and that you go, okay, if I am providing the self service platform to a data scientist, that's going to look different than if I'm providing it to a business analyst. And it's going to look way different than if I'm providing it to an exec, right? They want to go in and say, okay, I've got these questions. Okay, I've got pre canned answers that are already here in my dashboard. That's what they're self serving from.

And if they've got other questions, and you give them the capabilities to potentially answer those, probably pairing with somebody rather than being like, I want all of my execs to be able to write complex, multithreaded SQL queries or something like that, where it's, "Oh okay, I need to be able to join from 15 different sources and understand the difference between left join, inner join, right join", all that fun stuff. But that it's so much about don't solve for all of your problems until you come across those problems. And even then, don't necessarily try to solve for them. Try to have a workaround until you're going to be able to easily fix later and replace with something when you find something that works for you, but that it's not you overly tie yourself to point solutions instead of as a product that's supposed to serve the broader set of use cases. Is that kind of, if I were to sum up a lot of what you were saying, is that a lot of what was kind of coming through there?

1:16:19 Kiran Prakash
Yeah, absolutely. Absolutely, yeah. And I especially like the point you made, which I didn't say, but I like that I want to reinforce it, which is about basically, is this just one platform to rule them all, right? It is not. Especially when you're a large organization, you are going to probably have a few platforms which cater to different needs. And the user definitely is not going to care. At the same time, you don't want every product team to build their own platform. And that's also a wrong approach, because platform don't ship value. The product ships value, right? So it's in a spectrum. It is kind of, it's not one, and it's not like one per team, but it's somewhere you need a handful, which kind of covers all the different engineering needs of your data product teams, right? Some domains are sophisticated. They have the capability to kind of deal with infrastructure as code. Maybe some other domains are smaller. They want some canned solutions or a low code platform to run with, right? So you need to take all of that into account. And you need the minimum number of platform which caters to the different ways your data product team wants to use and build and ship data product.

1:17:43 Scott Hirleman
Yeah, well, and it was something that you mentioned earlier about reuse, right? Like that reuse on the architecture side is crucial, because if you don't have that reuse,

then what are you doing, right? So, well, we've covered a whole heck of a lot of things. Is there anything that we didn't cover that you wanted to or any way that you'd want to kind of wrap up the episode?

1:18:02 Kiran Prakash
No, I think we spoke about a lot of things, right? Maybe just sort of reinforce. I tell this to all my clients is that, yeah, Data Mesh has these four principles. And you need to sort of make progress on each one of them, because these principles are interconnected and kind of they reinforce each other, right? And if you just drop one of those principles, let's say you drop the domain orientation or you drop, like say, federated governance, the others are not going to work so well, right? You need to make progress on each of those. But at the same time, you need to sort of lead with one or few of them, right? You need to sort of get that right. And I often tell that if you had to sort of get one of those principle right, sort of aspire to get data as product, right? And I think the rest of the principle kind of tend to fall into place if you kind of aspire at excelling at data as a product, because there's no way to do that right without proper domain orientation or without sort of creating a self service platform that enables you to scale these data product teams, have many of those within your organization. So maybe that's one thing to sort of wrap up and reinforce, which is that, yeah, if you want to sort of pick one, like pick data as a product and aim to get that right, and the rest will follow.

1:19:28 Scott Hirleman
There's a reason that this is Data Mesh Radio is on the Data as a Product Podcast Network. I think Data Mesh is a very, very big concept, but data as a product is also a large concept that isn't just inside Data Mesh. And I think just shipping data products is not data as a product, so. Well, I'm sure there's going to be a lot of people that would love to follow up with you. Is there any particular place you'd like them to or anything specific you'd like them following up about?

1:19:56 Kiran Prakash
I'm active on LinkedIn and Twitter. I'm not on any other sort of social network, so you can reach me on both. I'll provide the link later. You can probably look it up in the transcript. Yeah, I love talking about this. Data Mesh is what is occupying my mind of late. I'm a big sort of fan of bringing all of these good software engineering practices we've been following in the rest of the software engineering world to the world of data. This is what I'm interested in and passionate about and spend a lot of my waking time thinking about these days. And yeah, if you like what I said and want to chat more, please do reach out to me on either of these channels, and I'll be happy to chat.

1:20:44 Scott Hirleman

Yeah, and we'll drop links to those in the show notes as per usual. Well, Kiran, thank you so much for spending the time with us here today and kind of sharing your expertise. And as well, thank you everyone out there for listening.

1:20:56 Kiran Prakash
Thanks a lot, Scott.

1:21:00.1 SH:
I'd again like to thank my guest today, Kiran Prakash, who's the Principal Engineer at Thoughtworks. You can find a link to his LinkedIn and the blog post we mentioned about the curse of the data lake monster in the show notes as per usual. Thank you.

Thanks everyone for listening to another great guest on the Data Mesh Learning Podcast. Thanks again to our sponsors, especially DataStax, who actually pays for me fulltime to help out the Data Mesh Community. If you're looking for a scalable, extremely cost efficient, multi data center, multi cloud database offering and/or an easy to scale data streaming offering, check DataStax out. There's a link in the show notes. If you wanna get in touch with me, there's links in the show notes to go ahead and reach out. I would love to hear more about what you're doing with Data Mesh and how I can be helpful. So please do reach out and let me know, as well as if you'd like to be a guest. Check out the show notes for more information. Thanks so much.