EPISODE 880

[INTRODUCTION]

[00:00:00] JM: Data engineering involves numerous tools; a data lake, databases, data warehouses, numerous APIs, streaming systems and microservices. There's no shortage of ways to interact with data and manage data. But many companies are struggling to figure out design patterns and best practices for how to manage data and build data infrastructure.

Zhamak Dehghani is a principal consultant and portfolio director with ThoughtWorks, where she works with enterprises to improve their software systems and workflows. Zhamak is the author of an article called How to Move Beyond a Monolithic Data Lake to a Distributed Mesh? And she joins the show to discuss her perspective on data infrastructure as well as the data mesh concept that she has coined.

Data mesh represents the movement away from having a centralized data lake that all teams interact with and towards having different data products and individual data management systems for individual domain teams.

[SPONSOR MESSAGE]

[00:01:16] JM: As a software engineer, chances are you've crossed paths with MongoDB at some point. Whether you're building an app for millions of users or just figuring out a side business. As the most popular non-relational database, MongoDB is intuitive and incredibly easy for development teams to use.

Now, with MongoDB Atlas, you can take advantage of MongoDBs flexible document data model as a fully automated cloud service. MongoDB Atlas handles all of the costly database operations and administration tasks that you'd rather not spend time on, like security, and high-availability, and data recovery, and monitoring, and elastic scaling.

Try MongoDB Atlas today for free by going to mongodb.com/se to learn more. Go to mongodb.com/se and you can learn more about MongoDB Atlas as well as support Software

Engineering Daily by checking out the new MongoDB Atlas serverless solution for MongoDB. That's mongodb.com/se.

Thank you to MongoDB for being a sponsor.

[INTERVIEW]

[00:02:37] JM: Zhamak Dehghani, welcome back to Software Engineering Daily.

[00:02:40] ZD: Hi, Jeff. It's good to be back.

[00:02:43] JM: You wrote a piece about software architecture, and the approach that you outline in this article was called a data mesh, and we'll get into what that means. When I was reading your article, there was a classic anecdote that came to mind, and that was the story about when Amazon moved to their service-oriented architecture. This is a pretty notorious story, where back in the early 2000s, Amazon had grown to be a really large ecommerce company and it was starting to have a lot of trouble communicating among internal teams, because those teams needed to start interacting with each other and they needed to consume each other's services.

So, the shopping cart team needed to interact with the shipping team. Those two teams had services that interacted with each other. They needed a common way for those services to communicate, and Amazon had to put a lot of work into forcing teams to have standard APIs that each other could consume, but this ended up being really productive for the company. It was a very scalable pattern.

Your article about data mesh, one core idea is that we need to take a similar approach to sharing data among teams as we take around letting services interact with each other. Could you draw a comparison between these two architectural challenges, the challenge of one developer wanting to consume another developer's service versus the challenge of a developer wanting to consume another team's data?

[00:04:17] ZD: Yes, absolutely. I think you made a very good observation there. Maybe a little bit of just background. I came to the world of data from years of working in distributed computing, distributed architectures, big enterprises where these problems are quite hairy and difficult to solve because of the size of the organization, the richness of the business domains.

Because coming from having seen the problem of scale in an operational domain and having seen how the likes of Amazon and other organization have tried to tackle that by decentralization, decentralized ownership. Any distributed system to work needs to have standardization around interoperability, right? The standardization that we got around microservices, with HTTP and REST and other particles bringing interoperability within the services, those were the foundational building blocks for us to be able as an industry to accelerate building systems and at scale, that by scale I mean serving a really large segment of the industry. Scale of your operational throughput and then scale of your operation, like how many number of teams you have, but in parallel can work.

So, I think those – I kind of came less [inaudible 00:05:39] to the data with that point of view. Then I saw the challenges of the big data architecture at scale and I thought, "What do we think about decentralizing data? Why data has to be in one big place?" Then to your specific question, "How can we optimally decentralize ownership of the data? Then if that happens, what's sort of interoperability need to be put in place? So different teams with different datasets can actually work? We don't end up with the data silos that we're in today.

So there's definitely a parallel between kind of distributed services design where your ownership of a domain is given to a team, and that team is responsible for operation of that domain and providing services to the rest of the organization through some standards. The idea of the distribution of the data ownership to domains where they actually understand where this data comes from and what it means and providing an abstract that I call like data as a product, or data product, around that data so that it can be consumed securely and easily by the rest of the organization. There are details that we can go through as what that means.

But I think if you step back for a minute, to me, the interesting point, the crisis I guess that I had to get me to think about this and to write about it is probably the crisis that maybe Amazon had at the early days, which was the crisis of scale that led them to organization that they designed themselves differently. I think I see crisis of scale as well as other types of crisis in the data space that are very similar.

[00:07:23] JM: What does the crisis of data look like at the average enterprise that you encounter?

[00:07:32] ZD: So, I had I guess a fortunate position to work across multiple clients both globally and within the west coast of U.S. and the Bay Area. So what I have observed, the [inaudible 00:07:47] modes or the crisis of big data falls into a few categories. You have the bootstrapping problem. So a category is that they thought about, “Oh, data lake.” Like they had a bunch of data warehouses really didn’t work and then thought about, “Okay, the data lake is now a new solution.”

So they’ve been stuck in how can they get data into the data lake? How can you actually make it useful [inaudible 00:08:12] organization and how to bootstrap that process? So they’ve got organization go through months of designing the data lake. Maybe they have some POCs, have a little bit of a data in there. Not a whole lot of use cases actually using that data. So it’s kind of a bootstrapping problem. Being stuck as where – At one end of this hairy problem we should get started? Shall we get all of our domains pouring the data first? How do we go about bootstrapping it?

Then the organizations that have bootstrapped themselves and do have sort of a data platform or data infrastructure, is that there are two friction points that doesn’t let the system scale. One is if you’re thinking about a retail business, there are hundreds of operational systems from designers thinking about what to design? From merchandise, to planning, to pricing, to selling in the shop, selling on the ecommerce. All those operational systems are generating and emitting data.

So, data is ubiquitous across the organization. How do you get this ubiquitously available data and very large different types of data into a manageable, like big data platform, the data lake? So there’s a problem scale of ingestion and there’s a problem of scale around consumption of that data, because you’ve got different categories of use cases. You’ve got your business intelligence producing reports and sales KPIs so that business can make a decision. You’ve got your machine learning use cases that needs data for training and testing. Price optimizations as an example or personalization. You’ve got your analytical use case.

So I just want to explore to see what I discover. What sort of insights I discover about my customers that I didn't know in the past? So responding to all those diverse set of use cases through a centralized platform is another failure mode of scale. So there are two points of, I guess, failure mode around scale ingestion and providing different types of access.

Ultimately, I think the other failure mode is ignoring the technology itself or the organizational structure itself is how do I change my culture? How do I change my organization so that we make decisions based on the data and fact and observations? We have a cycle of experimentation built into our operation rather than intuition or whoever has the highest position in the organization.

So it's a cultural shift in terms of – And operational shift as well. Like how operationally we change our function so we can empower our employees to use data to do their jobs more efficiently? So that organizational kind of operational model itself as well that's kind of a challenge. But I haven't tried to tackle that one. That's too hard, people problem.

[00:11:03] JM: The frequent approach of building systems around data is this data platform approach. This idea that let's say we're an enterprise that's been around for seven or eight years. Our operational business is going really well. Maybe we're a shipping company or an oil exploration company and our day-to-day business is just fantastic. We've always had this exhaust data. We've had some logging data, and we would love to take advantage of this exhaust data. The approach that we take towards trying to take advantage of that data is we centralize it all. We put it all into a data platform, and then we put data discovery tools around it. We put data cataloguing tools around it so that different teams can access these different datas, data sources, from a centralized data lake. Maybe we can pull that data into a data warehouse and do large scale operations on it. But the data lake is the point of centralization. What are your critiques of the centralized data lake approach?

[00:12:13] ZD: So I think I personally, I have to give, I guess mention this, that my mental model is biased towards decentralization, because successful platforms that I have seen in my career, they've been built with decentralization at the heart of them. So I have to just put that out there.

That, historically, looking back, I cannot find many successful, large scale implementations of technology solutions that had a centralized solution.

But that aside, I think the multiple problems that comes with centralization. The first problem is that once you have – And this is going to sound a little bit academic at this point, but I'm going to take it to data in a minute. Once you have a centralized piece of architecture, your architecture and your organizational structure are going to mirror each other. I think everyone's heard of Conway's law, that your organizational communication structure and your architecture is going to mirror each other.
So in that case, you end up with a centralized kind of platform and a centralized team responsible for it.

The second problem, the moment you have that problem of centralization and you need to scale it, you need to think about, "Okay. How am I going to break this apart into its pieces?" So I can scale this out. I can have multiple teams working on different aspects of it. The paradigm that we have adapted from data warehousing days is that, "Okay, let's break this big platform apart at the first level kind of decomposition access around its functionality, around this mechanical functionality." So you will end up with – And I've seen this actually implementation in really large tech organizations here in the Bay Area. They say, "You know what? We're going to build a team and a bunch of services around ingestion, an ingestion of all kinds of data that can come to the organization. Then we're going to build a set of services and tooling around transformation. So all of the pipelines that we need to put in place to transform any kind of data to any kind of data."

The last one is, "Okay. Now, we have to serve this data. So we've got to put a bunch of tooling around search and discovery and APIs and whatever you need to put in place to provide access." So you end up in this pipeline architecture that is allowing you to somehow breakdown this monolithic into its pieces.

But the challenge with a functionally pipelined architecture decomposition is that the change in organization, the change around data actually happens in an access orthogonal to your functional decomposition. So if you need to create a new – Give access to the new types of data, create new use cases, you end up actually needing, "Oh, I need to make some changes to the source. I need to ingest new sources. I need to create new transformation. I need to create new access points for it." So you end up having this still fairly slow moving end-to-end feature development with a lot of friction in between. So that one problem is decomposition of a centralized platform.

The other challenge with centralization, let's look at – One is the scale. How do you scale out your systems and how you break it – Somehow break it apart within that centralized piece we talked about. The other challenge is this silo that you create with your people, right? So a very common problem and frustration that I see with organization is that they have said, "Okay. These domains, these operational domains, yeah, we figure out domain-driven design or domain-driven organizational design." They have their clear boundaries and running their ecommerce business or the management, whatever those systems are.

But the data, let's put that aside, where you've got this big data platform and let's throw a bunch of data engineers at it to solve all the data problems that we have. So they've created this silo that doesn't allow exchange of knowledge around what the data actually needs to be. So the concept of the domain and the language of the data is really lost, because the people who run ecommerce system intimately understand what is the exhaust or what is the mission as a user interaction looks like? What is the shopping cart state transitions look like?

They intimately understand that data. But the data engineers that have been siloed, just because they know how to use big data platform tooling, they don't have that organization. But they need to ingest it. They need to transform it and make it useful, again, for a set of use cases that really they don't understand who's going to use their data. So they're stuck in the middle, and that creates a point of friction, because they are a team that are – They're under a lot of pressure. They don't have the knowledge of the domains intimately. The data on them changes without them knowing what actually got changes continuous break that can happen. Then they're serving people that are fairly frustrated, because they need to make – They need to have access to new types of data or new projections with new modeling and new aggregations, but they're depended on the central team. So that creates that silo.

I think another problem, and this is going to go now industry-wise and me ranting about data engineering as a whole, is that by creating this siloes, which we had in the past. We had the dev and ops silo, right? Developers didn't know what writing your application look like, and the operation folks had no clue when an application broke down and how to fix it.

Creating these siloes creates this kind of skillset silo as well. I see a lot of [inaudible 00:17:55] engineers that they haven't really, I guess, adapted the best practice software engineering. I mean, we live in the Bay Area. We work with the best. So my view might be a little bit biased towards seeing the best of the best. However, globally, data engineers not having working very closely with software engineers or generalists in general has led to, I guess, lack of best engineering practices. Like CICD for data, what does that look like? There's not a lot of good practices. Like versioning of the data just coming up. We see some movement around that.

So a lot of good software engineering practices get lost because the data engineer is being siloed. Conversely, I think like software engineers, I think knowing how to use Spark and a bunch of other data management tools should be part of the tool belt of every software engineer. But siloing their personas and their work, there's no cross-pollenation happening for software engineers to know how to deal with data and data engineers who know how to write good software. That's a challenge in itself.

I think in the industry, we have a huge gap, skillset gap. If you look at LinkedIn, for example, analytics and information. I think 2016 might – I don't have the latest, but I don't think it's any better. But in 2016, LinkedIn was indicating that there are about 60,000 data people in the world that claim to be data engineers in their platform. Only in the Bay Area we had 65,000 jobs open. So there is a huge gap in skillset. How does that happen? By cross-pollenating your people.

So that's I guess another problem with centralization. The other one, the most important one, I think we missed the notion of the domain. I mean, we forget the most important part of this, it's actually the data itself and meaningful data around the domains of your business. Not the technology under it. Why focusing on the technology and data platform as a first-class concern. We actually forget what matters, which is the data or domains.

[SPONSOR MESSAGE]

[00:20:07] JM: A thank you to our sponsor, Datadog, a cloud monitoring platform bringing full visibility to dynamic infrastructure and applications. Create beautiful dashboards. Set powerful machine learning-based alerts and collaborate with your team to resolve performance issues.

You can start a free trial today and get a free t-shirt from Datadog by going to softwareengineeringdaily.com/datadog.

Datadog integrates seamlessly with more than 200 technologies, including Google Cloud Platform, AWS, Docker, PagerDuty and Slack. With fast installation and setup, plus APIs and open source libraries for custom instrumentation, Datadog makes it easy for teams to monitor every layer of their stack in one place.

But don’t take our word for it. You can start a free trial today and Datadog will send you a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started.

Thank you to Datadog.

[INTERVIEW CONTINUED]

[00:21:15] JM: So the advantage of the data platform that I can see is you know that all your data is in one place. It may not be entirely correctly, but at least you only have to set up one permission system, one set of best practices on top of that central data lake. Of course, the consequence is many of the things that you enumerated, I think most architecturally, just the idea that there is this one monolithic thing that we all pull from. This intuitively feels like there might be something wrong with it.

But at the same time, we can look at a company like Uber, or a company like Lyft. We’ve done shows on both Uber and Lyft. As I understand, they have a central data lake essentially that they all pull from. The way that technology has evolved over the last 10 years or so is these startups tend to be at the leading edge of best practices that then are followed by the rest of the industry, the banks and the large enterprises and so on.

You're effectively saying that this might be an anti-pattern that we as an industry have all centralized upon. Why do you think that is? Why do you think that even though the kind of top startups have centralized around this centralized data lake approach? It may not be the best approach possible.

[00:22:48] ZD: So, I give an analogy. When a startup starts their business, do they have to go and build microservices or they start with monolithic application? It's easier to build a monolithic application and it's the right thing, right?

So for the scale of your business, how many independent teams you are running? What's the size of your engineering organization? There are a lot of factors that decide where is the right point to stop breaking down your monolithic solution?

So I guess I want to say two things about what you mentioned there. One is there is an inflection point where you feel the pain of a centralized solution, whether it's a monolithic application or a monolithic data lake and you have to think about what is the right way of breaking this part? It's not the first day. I wouldn't suggest somebody starting today to start thinking about decomposed or decentralized solution, because there is an overhead that comes with it.

So, for many companies, it's probably the right thing to have one data lake, one team responsible for it and provides self-serve access to the rest of the organization, and there is a size and complexity argument that would lead to, "Okay. This is now – It's the right time to breaking apart. What is the best way to break it apart?"

But they also mentioned there that you're creating – By having one thing, one platform, one data lake, you have a standardized way of operating it, right? Standardized way of getting to it. I think that standardization is absolutely the key, the self-serve nature of it is absolutely the key whether you have a decentralized solution of whether you have a centralized solution.

So I was envisioning and implementing with our clients around data mesh is that the only the ownership of the data itself lies with different teams. However, for a data to not just be the emission of your operation system, but also be an asset that you are responsible for providing to the rest of the organization, you need to impose some form of a standardization around access to it, around what sort of metadata we are providing. What sort of essentially siloes? Like the level of the quality of the data, the quality that this data has. It has to be registered somewhere. It has to be discoverable. People can discover it.

So there is a level of standardization. Also, there are other aspects to standardization. For example, harmonization of your identity systems so that you can stitch data from different places through a joint identity. So that standardization need to be applied to these distributedly owned datasets that are owned by the domains themselves.

I actually know kind of the unicorns in the Bay Area that they have reached out to me and say, “Oh! You’re actually doing that. Let’s have a talk.” So there are companies that are doing this. Maybe they haven’t really talked about it much. So the standardization key.

Then the second piece that is absolutely key, which is whether it’s applicable. Whether you’re building a data lake or you’re building a data mesh, is some form of infrastructure that is shared in common. So, in fact, maybe all of your data from different domains do land on Amazon S3 if they fit that model, or they do land on Kafka topics for the streams. That essentially is being managed by my data infrastructure.
So, physically where the land lands, that comes down to your shared infrastructure that is provided as a service to these domains that are now serving their data with some level of standardization and equality to the rest of the organization.

So there’s definitely some centralization there around infrastructure and there is a standardization around the seams between the data, like how the data is exposed, discovered, securely accessed, which is applicable whether you’re doing a lake model, a mesh model. Obviously, with a mesh model, you need a more sophisticated system, because you’re providing those capability self-serve to multitude of teams. Not just one team.

[00:27:00] JM: So, let’s get into your data mesh approach in a little more detail. Your idea around the data mesh is that if I want to consume data from another area of the organization, I should go directly to that area of the organization and should be able to hit some kind of API or like a data tap of some kind rather than going to a data lake that that team has exported their data to. Am I articulating your data mesh approach correctly?

[00:27:40] ZD: I think so. I think that's correct. One little, I guess, addition. Let's play a role, right? I want to consume data. I want to provide data. So start with the case that you're suggesting. I want to consume data.

If I want to consume data, I still probably want to go in one place that give me visibility to all of the data products, and I want to really emphasize on the product one, but maybe as a separate topic later. All of my domain data products that I can find. I don't really want to go to every single team and have a chat and find out what they have. So I think there is still a need for a data registry, data catalogue, data discovery tool to give me all the data that is available to me.

However, where that does – Who owns that data? Where it comes from? I probably shouldn't care. But really as a consumer, if I double click into it and learned who owns it so I can need to have a chat with them or have a chat with them, I realize that that data is actually owned and served by the specific domain where it comes from. So, I know that if I go to them and have a talk to them, they intimately know about that data, because they're generating it essentially.

Sometimes these datas, they more aligned to the source. They're coming and being captured and served at the point of origin, where the system, the operations system, like if you're in the health system and you're a claim processing system. The history of the claims or the events of the claims that members of your organization might have submitted is provided by that claims domain, right?

So I go to that particular team to know about that. So these are domains that are – Data that is very aligned to the source of origin. I call them native data products. But also I might find, as I search this catalogue, go, "Oh! There is a really awesome aggregation of historical medical records of a health patient that I have or the clinic visits that somebody has aggregated and provided as a data product that I can go and find out about. If I double click it. Who owns it and who served it? It's probably a separate team.

I shouldn't probably get to that point from a consumer point of view, because from a consumer point of view, I have all the information that I need to consume that data. I know what it means. So I have documentation and schemas and information about – A lot of information that removes the friction of me going and searching deeper through that discovery.

I think the paradigm shift is really around how did we get to that point? That's the implementation of a lake versus a mesh that matters here. That's the role play of, "Okay. If I need to provide data, then what's the journey there?

[00:30:32] JM: Has the data mesh approach, have you seen it employed at any companies or what would it look like if it was employed?

[00:30:42] ZD: Yeah, pieces of it. So, definitely, we have done aspects of it. But I can't point in like one golden implementation of it that I say, "This is what it means at scale." So I have to put this out there as part implemented, part hypothesis, part being tested and implemented right now at my clients. So what we're doing is you have data infrastructure as a bare minimum on demand as a service provided to you to the domain teams.

So, for example, if I'm the team in the claims world, I have all the tooling that I need to make the claim data sets in a reliable fashion and trust in a secure way available to whoever has security access to it. So the tooling that I would need. So then the next step is as a domain, as a claims domain operational business unit, I have certain KPIs. I have certain – Maybe my boss has a bonus attached to it even, or OKRs that articulate I'm not only responsible for processing claims really well, but I'm also responsible for providing information about the process claims or the claims events to the rest of the organization so that they can use that data, right?

Then I have data engineers in my team and they have data engineering skillset as part of my team. I have a technical product owner similar to the technical product only that thinks about the claim system operations, the things about the claims dataset that we need to provide to the consumers who wants claims information.

That technical data product owner thinks about the lifecycle of the information. Shall we provide both events or historical snapshots and what are the consumers we have? Do we have data engineers? Do we have ML engineers, or a data scientist want to use this, or do we need to provide some analytical tools? So that's a job of that guy.

The next thing is the implementation of that data product. So then I would use the tooling of the data infrastructure. I will register my dataset with this kind of data catalogue to give access to the rest of our organization. I build whatever pipeline I need to build to get the data out of my database or event streams or whatever infrastructure is suitable for my claim system, and I would have that data in a timely fashion refreshed, and I'm responsible as a team for reliability and trustworthiness of that data.

Then that data becomes part of the ecosystem of this mesh that is available. If I'm curious as a provider, then on the other side of it is that now we have claims data as well as all sorts of other data that domains are providing available to whoever wants to use it.

So if you have a downstream system that wants to join claims information, with member information, with clinical visits information to make some projections or some recommendations to the patience as when to go get care, and there are plenty of good examples around that. That's a downstream domain, that its job is patient care, right? The critical moments of intervention in a patient's care, for example. Detecting when is a good time to go and tell the patient that they need to go and visit, get help or do their annual check or whatever communication we can do.

That's a downstream domain that needs that information. So they will require some sort of an aggregate form of data, right? Whether they decide that within our domain, we're going to use, again, the data infrastructure tooling to create the aggregate that suits us or whether we say, "You know what? Actually, that information is a reusable information that we should give it an ownership and make it a reusable data product."

So there will be someone in the organization that will own that aggregated view, and it's very likely that patient care domain would own that, because they are the most closed use case or fit for purpose use case for that data. So then they use the data pipelining solution that the infrastructure provides and we go from there. They create the joins and the aggregates and provide that data as part of ecosystem, as part of the mesh registered with the catalogue.

They're responsible for its liveliness and accuracy and so on. So that's how you kind of scale out all of the possible use cases and all of the possible sources where the data might come from.

[SPONSOR MESSAGE]

[00:35:25] JM: SpringOne Platform is a conference to learn the latest about building scalable web applications. SpringOne Platform is organized by Pivotal, the company that has contributed to open source technologies, Spring and Cloud Foundry.

This year's conference will take place in Austin, Texas, October 7th through 10th, and you can get $200 off your pass by going to softwareengineeringdaily.com/spring and use promo code S1P200_SED. That code is in the show notes.

Attend SpringOne Platform and work hands-on with modern software. Meet other developers and software leaders and learn how to solve and find out solutions to your toughest scalability problems. Go to softwareengineeringdaily.com/spring and register for SpringOne Platform in Austin, Texas and get $200 off with promo code S1P200_SED. That's S1P200_SED, and that code is in the show notes.

Thank you to SpringOne Platform.

[INTERVIEW CONTINUED]

[00:36:46] JM: Let's contrast this in practice. So, if I am taking the approach of the centralized data lake for my architecture and I want to consume the data from another team, my way of accessing that data from the other team is going to be I like at the data catalogue. The data catalogue is going to tell me where in the organization I can find that data, and it's probably going to – If I'm looking for logging data, it's going to tell me, "Oh, the logging data is in this S3 bucket in our S3 data lake." I'll go there and I'll find the data and the data catalogue maybe tells me the schema of that data, and so that I can pull that data into a data warehouse perhaps and so some operations on it or pull it into a streaming framework and do some operations on it. That's the centralized data lake approach.

If I was taking the data mesh approach, then maybe there would be a logging data team that would have all that data in their own data system. They wouldn't have had to export that data from their own data system into the data lake. I would just go directly to them and take their data or engage them and figure out a workflow for interacting with them and pulling their data on ongoing basis. Am I understanding it correctly?

[00:38:16] ZD: Half of it. I'm just going to try again.

[00:38:19] JM: Okay. Sorry.

[00:38:20] ZD: No. It's good. It's good. Actually, it's a test for me to know how to articulate this. I think in terms of – The consumer experience should be more or less the same. Whether you're operating in a mesh world or in a data lake world in a happy path.

Happy path is what you just described, where the data is actually there, what I'm looking for. Somebody, somehow, cleansed it, made it available, made it discoverable and is available. That's the happy path. Whether it's coming from a note on the mesh, which also might point me to S3 bucket. It will give me some documentations about it, because the point of it is this self-serve. Give me some documentation. Give me the address. Tell me how I need to – what sort of role maybe I need to have to be authorized to access it and all that sort of good stuff so I can go find it.

From the consumer point of view, you're in a happy path. When the data is there, the scenario is more or less – Actually, the experience probably more or less is the same. The problem is with the unhappy path. The unhappy path is when you – On the consumer side, the data is not quite what you want. So then it's kind of the same data, but it's not all of it. So maybe missing logs from some places or it's not quite in the format that you need it. Then you are reliant on – Usually, you are reliant on a central team to give you the data that you need and go and find these data and change it and modify it and then make it available. That's the point of friction, because a centralized team is usually not only serving you, but also serving a whole lot of other people.

The other part of the unhappy path that is with a centralized data lake is not so much the consumer said. But if the data isn't there, the experience of getting data – I think we are trivializing actually the complexity and the organizational challenge around making data available and understandable, harmonized way to the rest of the organization. If there is one central team that its job is for months to figure out what's the right data source? Because this data has been copied across multiple systems, or like five systems almost capturing the same data, but not quite.

So there's a whole lot of exploration that we have to do in discovery by this centralized team to find out where the data comes from and then what actually it looks like. What sort of ETL jobs or the tools that I need to do? Change data capture or whatever it is that I need to do to get that data. It's very fragile, because the system that is emitting the logs, they can – The lifecycle of that system and the shape and the type of the data that they can emit, that is changing closely to the lifecycle of that operational domain. You change your system, the shape of your data changes, because that data is not supposed to be used by mass consumption. It's a byproduct [inaudible 00:41:30] operation. That's another problem. It's not a data asset. It's not a first-class concern. It's just a byproduct of your operation. So, who cares?

So then that fragile connectivity to get these data from all of these sources in one place by one team and make sense of it, harmonize it and have this kind of fragile system. So I think the experience of the consumer is not so much different from the happy path, is the experience of the provider is very different. The team that is working to get this data in a way that you can consume, their experience is very different. Whether you have one team trying to stitch all of these pipelines together and create some sort of a harmonized view of all these logs, or whether it's actually the team who are responsible for different parts of that log through. There has to be a global governance and global interoperability.

So following a global governance interoperability saying, "You know what? I'm actually responsible for providing a consistent log with some sort of a standardization around what logging should look like to the rest of the organization," rather than have one team that its job is to get logs from all over the place and think about how to make sense of it and store it. So it's more how the data gets provided or gets served. That's the part that changes. Hopefully the experience of the customer or consumer is delightful regardless of data [inaudible 00:43:03] one place or owned by multiple teams. There's a level of consistent access and level of product thinking that goes through how we provide data to the consumers.

[00:43:14] JM: What are the responsibilities of a data provider in this architecture?

[00:43:21] ZD: Yeah. So their responsibility is to serve data based according to set of standards or governance to the rest of the organization. Their responsibility isn't having the most reliable infrastructure, having the most scalable store. Those domain-agnostic infrastructure pieces should be dealt with by the data infrastructure folks. But their responsibility is as a domain that is representing, let's say, logs. Their responsibility is provide logs with a good documentation.

Let's rewind. If I say the success criteria – Let's describe it with the success criteria. The success criteria for a data domain team, a data team that is around the domain concept, is the reduced lead time for anybody to come and find and use their data. So if you use that as a success criteria, then everything around – The tooling around – Sorry. The underlying tooling comes from the infrastructure. But how do you increase lead time to discovery of your data and to usage of that data plus by providing good documentations?

Examples that datasets where people can use to apply in sort of a standardization that is globally organization is deciding on, for example, using some form of a federated identity management to describe the identity of entities in your domain. So, some sort of a [inaudible 00:44:57]. If you're representing member information in your dataset, internally your system may recognize that member by some internal system I.D. But externally, to provide that data, to serve that data, you're using some sort of a global unique member I.D. and you do some adaption there and the infrastructure provide you the tooling to do that.

So, standardized data that you define the security around it, you are responsible for providing. If there's PIA information and non-PIA information, how do you provide those datasets differently through different access controls? So all these security around your data.

Really, do whatever it takes to get that data in an easy to consume way to people who are authorized to access it. But then you might say, "That's a lot of work for a lot of different domain teams to do." That's the beauty of infrastructure, of self-serve infrastructure, because you abstract away a lot of the tooling that folks need as self-serve API-based kind of tooling. Not everybody builds a data catalogue. There's one data catalogue with an API that say, "Oh, you're a data product? You want to register yourself with that catalogue? Here's the API. Go for it, or here's the GUI tool to go for it." So the tooling would facilitate how to serve reliable data to the rest of the organization.

[00:46:12] JM: If you were the CTO of a company that was trying to figure out its approach to data and you decided you're going to go with the data mesh approach. What kinds of best practices are you putting place? What kinds of suggestions are you making to the teams throughout the company?

[00:46:33] ZD: Yeah. The first and foremost, and this is an exercise that we actually do with a lot of our clients. We take them through – A lot of our clients that come to us, because they've stuck or they couldn't scale or they fire the whole data warehousing department or they couldn't really respond to the needs of the organization.

So the first thing I would do, and I do that practice with my clients, is that go through what we call the cycle of enterprise. The cycle of intelligence, as in go through a couple of use cases in my organization that I can connect the dots between the operational system to the data that needs to be served by those systems to the real use cases of that data, whether it's for business intelligence or ML-based optimization of my operation all the way back to my users. Whoever the users use, it might be an end user or maybe another operation system. So close that loop. Think about a few of those use cases, end-to-end use cases that close the cycle of enterprise intelligence.

Once I discover that, then I would start with identifying for one of those use cases, what are the data domains that I need to unlock and make that part of the mesh? Some of it might be sourced. Some of it might be intermediate aggregate ones that I know is reusable by other people. Some of them might be very fit for purpose for that particular consumption or particular use case.

I go to those domains and I put a team together that's responsible for my data infrastructure. So those folks are usually my infrastructure people in my organization that now they need to know how to store and serve data at scale. So I put, I guess, a process in a place or a few initiatives in place that my data infrastructure people are enabling a few of these domains.

I assign the ownership, the product ownership, the technical product ownership for data to those domains. So they can think about, "Okay, what is the data that we need to provide?" Then I would have teams that probably they're already in place or maybe they're not, that really are going to be the first consumers of those data products for that particular use case, that cycle that I talked about, and make sure that as part of the implement. The first iterations of implementing my data infrastructure as well as my data domains, I am delivering value. I'm showing that the data is actually being used to deliver real value. Then I will repeat that a few times. I have to scale this up, right?

So, overtime, I will have more of my domains to join the mesh to be kind of activated as data notes on this mesh and I would put – I guess, ask my data domain teams and data infrastructure team to come up with their success criteria, and I'll probably ask for some success criteria. Like for the data infrastructure, one of their OKRs should increase lead time for any data domain to be activated on this infrastructure.

So, I would keep people accountable and ask them to create ways of measuring their OKRs and their success criteria and really try to not serve data just for the purpose of serving data, but try to show that for every served data, there is a use case that we serving, and enrich the platform kind of overtime.

I think most importantly is the two culture aspect change is that the data domain is a first-class concern. So the data that's been emitted from different domains are first-class concerns in an organization. They're not just the byproduct of my operation. People need to care about their quality and be accountable for service level or data level objectives, or whatever we want to call them. I'll also think about how to actually change my business to be data-driven.

We think about, like there are a lot of opportunities to do ML base and data-driven functions in a business, right? So as a CTO, you have to think about where are those opportunities hidden? Where are we writing a whole lot of if and else code and rule based applications and imperative coding to make a decision maybe correct, maybe not, or where can I change that data-driven, model-driven, AI-driven type decision making? You think that's fuzzy decisioning land itself to that so that I can close the loop. So I can close the loop between the data to information and insight and intelligent decisions and intelligent actions.

[00:51:24] JM: What's been the feedback in response to your piece on data mesh?

[00:51:31] ZD: Surprising. It's admit. It has been quite surprising, because I first talked about it at an O'Reilly Conference. It was a relatively intimate setting with 40 something audience. When I talked about it, I wasn't sure if I'm going to get sharp objects thrown at me or it's going to resonate. I was really surprised. Everybody came after the talk and started giving me feedback. The pain is real. The underlying root cause seems right. It seems like we are really ready for a paradigm shift that we have inherited from 30 years of data and data warehousing, could have been more.

So that was the initial feedback that I kind of tipped my toes in the water and started putting the ideas out there. Then I wrote the article, and I've got three different kind of categories of feedback. First, a group of people have come and said, Oh! You know what? We are building actually solutions applications. So the tool makers of the world. Don't ask me who they are, because I won't remember their names from the top of my head. Tool makers said, "Oh! We are actually building kind of applications or solutions or tools that solve this sort of – Works with this distributed model."

The second class of response has been really positive in terms of it makes sense and curious. Where have you done it? Have you done it? What are the first steps? What are the tools? A lot of curious, and I feel responsible for writing more and advocating more and talking more as I go through this journey myself. As a company, we go through this journey.

Then there has been a group of people that actually have come and said, "We are doing this," and we just haven't talked about it. We haven't written about it, and many of those are in kind of ten minutes walk away from me. I can go and talk to them.

So it seems like it has – I'm getting inbound calls every day for companies that want to come and work with us and learn more about us. Interestingly, a lot of them from U.K. and

Switzerland. I don't know, but there's this huge influx of folks in the U.K and Europe in general that it seems like they have a lot of data problems for some reason that they want to know more about. So I'm really excited.

[00:53:41] JM: Hopefully not GDPR related.

[00:53:43] ZD: Yeah. I don't know. Not GDPR related as far as I can tell. But maybe GDPR has been a driver for change, and the change has been hard. Now I'm looking in the crystal ball. I'm not really sure, but maybe that has been focused around data. Has been pushing the companies to think about their architecture differently, and maybe that's just one of the drivers. But I haven't directly heard about GDPR. I have heard a plenty of like fun stories. I mean, they're not fun stories. They're painful stories. Around the different failure modes and why they have reached out and where they have been stuck. It has been, I guess, confirmation of what I had observed in a smaller set of clients that I had worked with.

[00:54:30] JM: Zhamak, thank you for coming on the show. Is there any closing thoughts you'd like to give around the data mesh approach?

[00:54:37] ZD: Well, thank you for having me. It's always wonderful to talk to you. I love your show. So it's great to be back.

[00:54:43] JM: Thank you.

[00:54:43] ZD: In terms of closing thoughts. I would just love to see data mesh take into the next stop. I love to see industry kind of working together and really establishing the internal implementations more. One last thing I would say is that if you're doing a data platform and it hurts and it's hard and it hasn't given you the promise – It hasn't implemented the promise. Stop questioning that and start doubting that, because there might be other ways of approaching the problem. Think about some of the principles of distributed system design that has worked really, really well and has proven themselves to address the problem of scale, such as domain-driven distributed architectures, such as product thinking, such as platform thinking, and they're all packed into – Those ideas are packed into this approach.

[00:55:38] JM: Okay. Well, Zhamak, thanks again for coming on the show. It’s been a pleasure talking to you.

[00:55:41] ZD: Sure. Thank you, Jeff. It’s been wonderful.

[END OF INTERVIEW]

[00:55:46] JM: GoCD is a continuous delivery tool from ThoughtWorks. If you have heard about continuous delivery, but you don't know what it looks like in action, try the GoCD Test Drive at gocd.org/sedaily. GoCD's Test Drive will set up example pipelines for you to see how GoCD manages your continuous delivery workflows. Visualize your deployment pipelines and understand which tests are passing in which tests are failing. Continuous delivery helps you release your software faster and more reliably. Check out GoCD by going to gocd.org/sedaily and try out GoCD to get continuous delivery for your next project.

[END]