Appendix A

Interview Transcript Expert D

OK. Recording is on. OK. I will first tell you something about myself. I'm Tom, 24 years old, Dutch. Currently I'm doing a master data science and entrepreneurship. So it's like combining the business perspective with the technological perspective. That's why I like data mesh so much because there is some intersection between them. Before I was doing a bachelor in industrial engineering, so that's why I also have a business background. And yeah, nowadays I'm working for Bain Company, the consultancy firm. It's not actually working. It's more like an internship. And yeah, they are really curious about a possible data mesh implementation. So yeah the subject of my thesis is architectural design decisions related to data mesh. And now I'm creating my first framework for the data as a product principle and there will be two more. The other one is about the self-serve platform and the last one is about federated governance. We're missing the domain-oriented framework perhaps, but yeah, my university supervisor told me without data as a product cannot be observed without the domain oriented psychology. So that's why we somehow merge them and. We also think that Evans in his book on domain driven design has already captured a lot of that knowledge and data as a product is, yeah, there is some kind of lack of knowledge over there. So that's why.

Expert D
Yeah. OK, perfect. So, so obviously, my name's **Expert D**. I'm out of Toronto, Canada. I've been in the industry mostly financial services for probably 35 plus years. So that'll give you an indication of how old I am. More than double your age. More than double your age. But I'd like to think that gives me a little bit of wisdom perhaps. And I like I said, I've been doing financial services for many, many years. I've been an executive at large banks, I've been an executive at insurance companies and most recently I've started my own consulting firm for last 6-7 years give or take. And we kind of do one or one or two things really, really well. So we focused on data mesh, data as a product. Really in general we call it data ecosystems because you know when it's all said and done, a data mesh is just a whole bunch of data products that interact in a consistent fashion. Yeah, so we build data ecosystems using the data mesh principles, the data mesh designs and such approach for large financial services firms. So I've implemented global data mesh at a global insurer. I've done strategies for banks. I've consulted to some, you know, workshops and such with European banks. And right now I'm working with one of the largest technology firms in the world. That's the focus is on payments. And I'm building their payments data ecosystem with my interest. So, yeah, so all of this, like I said, I call it payment or sorry, I call it data ecosystems. Because it's a combination of many, many things that actually allow the data mesh to, you know, to actually realize that the principles that are the kind of the foundation of it. So anyway, that's my background. I have had a chance to look at some of your material. Ohh and I will let you drive the discussion and you can tell me where you want it to go because the questions I can answer.

Interviewer
I've read a lot of articles of you and I'm yeah they really inspired me. So you have seen perhaps some similarities between my frameworks and your articles. So let's just deep dive into it and start right away. Can you see my screen right now?

Expert D
Yes, I can

Interviewer
Great. All these models or all these frameworks are created by Python. And the reason I did this in Python is because I expect some iterations before the framework can be considered perfect or complete and everything can be smoothly changed instead of making the framework again and again by things like draw.io or PowerPoint etc. So for that reason I did everything in Python. And what we see over here is like the high level overview of all the decisions I found in a Gray literature. And what I mean with Gray literature is everything that hasn't been published, so. It hasn't been published in an academic journal. Like Medium posts, YouTube videos. Podcasts, etc. Things like that. So these six decisions are the decisions I have identified and I will now quickly go through each of them and at the end we can get back to this high level overview and yeah, check if there are some decisions missing etc. So the first one is about, yeah, identifying your data product type. After that, there is a decision on migration. So which approach for the creational for data product do you need? Do you want to start from scratch like Greenfield Development or is there a current legacy architecture where you want to migrate from? Furthermore, there needs to be some kind of infrastructure layer around the data product. So how does it communicate with other data products, with the self-serve platform management layer and consumers etc. If we deep dive even further into the data product, we can think of the anatomy. So what's happening inside the data product? What components does it need? Yeah, there is the data product interface. So it's like the perimeter of the data product, what kind of ports do we need, etc. And of course, the last step in software development is how to deploy your data products. So let's first have a look at the data product type. I identified 3 main differences. So the data product can be exposed as raw data and then also some practitioners do not agree with this because raw data can be really confusing for the consumer etc. But another part of the practitioners mentioned the value of having one source of truth and exposing your data product as raw data. Another one is exposing your data product as an algorithm and we can specify this even further in optimize optimization based decision support systems. So like BI tools, business intelligence. We can expose the data product as an AI / machine learning model. And the third option over here is to expose your data product as derived data and we can also combine different yeah different methods. So for example a hybrid product would have two ports. One for exposing data product as raw data and another one for the derived data or algorithm and the decompose product is. It's like merging. They're all database derived data etcetera. So that's what I have identified and I'm really curious about your opinion if you see some similarities.

Expert D
Yeah, I think it looks very good. I think the key thing that you're highlighting in which I strongly believe in is any data can be wrapped into a data product. And I think you kind of correctly highlighted I think in the previous slide perhaps from the previous thing. But it's important to understand what a data product is and in your context here it can be raw data, it can be data that has an algorithm on top of it. It could be data that's been combined, transformed, refined and is now, you know, exposing derived data. All the above I've seen and

they're very common. So I don't know too many practitioners that would say that raw data is not suitable for a data product. It absolutely is. Now the key thing about the data product is obviously there's the raw data or the data. There's a clear boundary around it and there's an owner associated with it. So from a technical perspective, one of the single most important parts is that interface or contract that you mentioned. Now maybe we're going to get to that later, but I'll just kind of talk a little bit about that and why that's so important. So in any enterprise I've been at and done data mesh and data products before. There's not one, there's not two, there's, there's an inventory of you know 10s if not almost 100 data products that can be out there. So as you highlighted, you use domain driven design and you're gonna do, you're gonna identify some of your core domains. Like I said, there's depending on the granularity that you go in the size of your business and the regionality, whether it's global and it's different countries have different regulations and how hence different domains. It is very common at least in when I was doing this at a global insurer there there's hundreds of domains. OK. The question is which one will we bring into the data make as a data product. That depends on the business value, but ultimately there's lots of them. So, the key epiphany in this is if you're going to build, if you build one or two of them then you can have bespoke type interfaces. If you build 10 or 100 or you know more than you know more than five or six, you want to have a consistent interface, a consistent contract across all data products and that is that is what I'm finding is that the piece of the puzzle that almost everybody is forgetting, everybody starts and says I'm going to build my first one or two data products. They build some, almost all are around how do I ingest data and how do I consume it. But the interface has to have all the other parts and I like the fact that, I think is in the previous slide you mentioned some operability that you need to focus on, so the key thing that I see is, is that interface should have, let me trivialize here a bit, it should have like a /discover endpoint so every data product can be discovered. What is the metadata associated with the data product glossary of it. And now I can start to feed that into a catalog so people can find these data products. So first you're going to have a lot of data products. You have a standard interface. If you have a standard interface and you introduce things like /discovery, you can actually have a catalog so people can find these many different data products. So I think you're spot on. I think you've considered this quite appropriately. And I think. Yeah, like I said, you've done a good job here.

Interviewer
Oh, great. Thank you so much. Yeah, yeah. Because I think this is the basic step you need to do before even implementing the data product. You first have to clarify for yourself what type of data product you want. And you need to ask yourself the why question. Why? Do we want the data product and how does it align with the business perspectives?

Expert D
Yeah, absolutely. I think that might be covered elsewhere. But have you touched on something that's probably perhaps important is. And one of the decisions I think on if I'm not mistaken, previous slide is like which domains do we actually build data products for? And I think it's important that the phrase you mentioned, which I think is also very critical as you look at this the right answer to which you know which domains do I build or build first is all about the business value and the fact that you want to assign with. Like data mesh is relatively new you have to convince folks of it. That means there's no Big Bang. What you want to do is pick one sponsor that has a very big you know a material problem that you can solve and then we use data products, data mesh actually solves that. And that addresses some of the other comments where a lot of folks think data mesh, data products are technical things and there absolutely are some technical components. But it is largely the blending of how do I use this particular technology, self-serve boundaries, ownership, local autonomy to actually solve

practical business problems and that's the key to success.

Interviewer
Yeah, I think Zhamak has a really good term for it. She calls it sociotechnical approach. So there are some social elements and of course some technical elements as well.

Expert D
Yeah, absolutely. I think people underestimate these socio part of it. And then to be honest with you, I spend almost, you know, on any given data, data mesh engagement probably spent 75% of my time on the social part, not the technical. Ohh, OK, that's a different discussion.

Interviewer
Yeah, people don't want to change. Exactly. It's hard to change. This also leads to my second framework I think, because this is more about your current architecture or perhaps you don't have a current architecture and you want to start from scratch with Greenfield Development. And in your article about solution patterns and accelerator patterns. I saw these four and I also saw, uhm, I don't know the name anymore, but besides the CQRS pattern, there was another one. But I didn't see that in other literature, so the Strangler-Fig, master data management, zero trust and CQRS were mentioned a lot by other practitioners as well. But do you know which one I mean?

Expert D
There was the CQRS, and there's a whole bunch of them. There's. Yeah, there's live replica. Yeah, real time replica or something. I can't remember. Yeah. Live replica was the thing. Yeah, that's awesome. Yeah. So these are all really about the interaction patterns and the zero trust is The only exception, but they're really about the interaction patterns and how data products work together. So there's other patterns that are around, how do I actually most effectively transition from a legacy environment into a data mesh or data product environment. So we'll go through a few of these real quickly. Obviously, Greenfield Development, you have some new stuff. Fantastic. And there's you know there's not a lot of difficulty in doing Greenfield development. So I won't talk too much about that but you know no enterprises is an island, they always, you know, almost by definition they have lots and lots of data and it happens to be in lots of these older legacy systems, sometimes enterprise data warehouses which may be very centralized and very slow and bureaucratic. The opportunity for data mesh which is where I spent a lot of my time, is how do I actually fit my legacy environment and actually create data products out of my mainframe systems or data products out of my enterprise data warehouse so they can participate in a broader data mesh and take advantage of data products and localized autonomy, which gives you the speed and agility which is so crucial. We'll go through a few of these very quickly here, the zero trust one is very interesting because if you think most organizations, as I mentioned have a legacy environment every single large. Now I'm in financial services, but I suspect this is across all industries. Every single enterprise is moving to the cloud, you know, different speeds but almost all of them as quick as possible. And they want to take advantage of, you know, the speed and agility that comes with the cloud, the ability to focus on more and move up, up, up, the stacked into business value as opposed to low level technical infrastructure. So a lot of organizations are saying how do I get to the cloud? Now here's the problem with the cloud, it gives you a tremendous amount of speed and agility, but you really, really have to think about how you're going to do your security. I mean there's been lots of articles. Capital One had a huge data breach several years ago, Equifax. And those two are, you know, the ones that are public. There's many others, but those are the big ones. And they had a huge effect, if I'm not mistaken. You know, independent of the fines

that, for example, Equifax had or had the money they had to give back to some other clients, the impact of the stock value was tremendous, billions of dollars. So here is the thing is if you're going to move into this new quote data mesh data product world and you're going to try and take advantage of the speed and agility of the cloud? You have to get your security right now. As I mentioned earlier, you're not building one or two, you're building 10 or 20 of these things. Which means not only should I have a cookie cutter or a pattern for my interfaces, but I should also have a cookie cutter for my runtime environment. So if security is an issue on the cloud, why don't I create a cookie cutter or a pattern that allows all of my data products on the cloud to be safe and secure? So I call that my zero trust architecture. So what we do, what I've done passes is we provide on the cloud. We provide. Usually it's through a combination of Kubernetes containers and a few other things. Every single data product out-of-the-box when we create it, it you know has TLS data encryption, OAUTH2 for access. You know, in a variety of other things in the entire container cannot be accessed by default unless you're explicitly given permissions. So actually in all the clients that I work with, we try and create these runtime patterns out-of-the-box, not just the integration patterns, but the runtime pattern. Zero Trust architecture is one of the key ones because obviously as we're moving data to the cloud, data products to the cloud, there's sensitive data you have to get your security right, and if I'm going to build 10 of these things or 100 of these things I ought to have a pattern to do it properly. So that's a big deal. The Strangler-Fig pattern really says, and I've used this a few times before, where I have a mainframe, a legacy data source, and most of my clients have mainframes. And what they're saying is, you know, they're very bureaucratic, very slow, hard to change and costly to change. Can I wrap something around them and recognize the fact that they may not change, but can I move the data to a place where it can be used more effectively? So many times what I've done is we moved legacy data, we've included things like change data capture, so anytime, something changes on the mainframe IMS database or DB2, any change there gets captured, gets pushed onto a Kafka topic and then there's lots of listeners to these Kafka topics and those listeners may be an AI/machine learning data product. It may be an analytics and reporting data product, it may be a customer engagement system that changes the format of the data from maybe third normal form to you know NoSQL. You know one record has all the attributes in it, optimized read and customer engagement. So now the question really is how do you do that? Well, obviously we talked to the zero trust environment, so at least we know where we're putting this stuff is safe. But how do I actually gradually first off connect the old system legacy system with the newer ones. OK, that's step number one in the Strangler-Fig pattern. The second thing is how do I gradually move more data over and take away data from the legacy system? Because by definition if I can move it over real time and I can capture all the relevant data that is on the legacy system and capture it and move it. I should have a real time duplicate alive replicas one pattern in the mission and if I have the live replica I can actually start to think about how I can take stuff away. Turn things off of the legacy system and you know use the newer system and point those interfaces from the old system to the new system and gradually strangle if you will the older system and eventually deprecate it. Now I should be straightforward that is a multi-year. If not sometimes a very long experience to do that. But like with any long journey you have to start and the Strangler-Fig pattern is a well-known pattern that is established in the industry but it works perfectly for building and establishing your data products and figuring how to move from older non-data products systems into to other ones. There's a few other things I could go. I want to be respectful of time so I can go on longer Tom, if you want I can talk about. But it's up to you. I could continue for quite a long time.

Interviewer
Yeah, I think the master data management is really relevant for me because at Bain they have

the Snowflake interface as well as the AWS. And from my perspective, Snowflake is really a master data management tool. So everything is organized really centrally and for my feeling this really goes against all the data mesh principles of being decentralized. And I saw that you also mentioned some kind of master data management data mesh so that there can also be a pattern in that. I'm really curious to hear more about that.

Expert D
Now I'll talk briefly about master data management. So the typical approach that's used today is I have my book-of-record and I have, you know, I have Tom's account here in my customer system. I have Tom's, you know, banking accounts in my account management system. I have Tom's preferences in my digital system and I have you know the loans that Tom has, the credit cards that Tom has and the checking accounts, they're all in different systems so. There's a very common requirement, especially recognized in Europe as a little further ahead than Canada and the US in terms of privacy and the right to be forgotten. You can't affect privacy, let alone this right to be forgotten unless you know when all of Tom's data happens to be master. Data management tries to say that this is, you know, this is who is Tom and the problem with data. There's a bunch of different ways to solve that problem that you know to find out you know who Tom is and what that data Tom has. One approach is in very small organizations they kind of centralize it. Sometimes they put it all into Salesforce.com, but the problem with that is they tend to move a lot of data. Now the second approach is, is we're going to only move a few pieces of data that what they call tombstone information kind of sounds wrong but it's the key elements of data that define Tom and all the information around Tom. But what ends up happening is we move you know these and usually by the way at least in the financial services firms to define Tom in a master data management system is usually. Anywhere from 50 to 75 attributes, and what they do is they move those from all those various different systems and they move it into the master data management system. Now here's the problem. OK again you're moving data and if you're moving data you know you run the risk of it being out of sync. You know something the address changed for Tom, he moved from Amsterdam to Berlin and the core system says you know he changed from Berlin. But the loan system says you still live in Amsterdam and that's a very common scenario by the way systems grow organically duplicate data but the problem you have is even that core 50 to 75 indicia attributes can become stale, OK, and out of date. And the moment they are stale or out of date, you have customer experience issues and maybe even some regulatory issues. If you can't, you know, if the data is out of sync and you have that right to be forgotten and you can't delete the data that you are obligated to, you're running some real regulatory challenges. So here's the thing, master data management to varying degrees today is predicated on moving data to a central location. My proposition is you can do that with data mesh. OK, without moving any data. OK. And in fact all you need to do is you need to have a directory of all your data products, OK, and the metadata that's in those, the metadata rarely if ever changes. There's not very frequently that you change, you know, you add columns or change columns in a database. It happens, but it's very rare. And when it does happen, the metadata can easily be affected and moved if necessary. But what I actually do is we actually have these references to all of the data products, OK, and we do live queries, OK, live queries of their metadata. OK. And we do that through what we call a data registry. So this data registry says, I know all the data products in the enterprise, and all I have is conceptually URLs that point to them all. They're all accessible at REST interfaces. And then when I need to say: tell me about Tom, OK, I'm trivializing, there's a lot more sophistication. Tell me about Tom. I go out to all the relevant data products and I say tell me what you know about Tom and then I can go and gather that information and I can begin to reconcile it using master data management filters and reconciling match merge type functionality. But what we end up doing is, the functionality

of the match merge is actually the toughest, it needs to be done no matter what you do. OK. But what we're saying is we're going to go and get that data and do it in, relatively speaking, real time. OK. So what does that mean? It means that we can get the result in near real time and, and in today's world, this is definitely doable to get that information, match, merge it in real time. OK. But what happens now is I leave the data in the data products. They're never stale. OK, and I can always get to the most relevant information immediately and if I needed to go and for example delete it. Uh-huh. Because there's no data in the master data management system. All the book-of-record data is in the various different data products. I can just issue again trivializing and conceptually I'm just issuing a delete command and they go and delete the relevant information. Or I can go and get the as I mentioned previous scenario, get the information about Tom, I can go get it and I can do real time reconciliation of addresses or whatever. And here's the beautiful thing about that is, is if I do real time reconciliation of addresses, for example, there's lots of different match merge in tools to actually do that, we can apply it and now we can actually make changes in real time, or we can actually direct the other data products to make the changes to their system and give them the right information. And what ends up happening is, is the whole, the whole thing. And the problems with centralized data and data duplication largely go away. Now that's not a trivial endeavor to set that up, but I will say that I've never seen any master data management system go live that hasn't had huge amounts of problems, hasn't taken a long time for us you know, 10s of millions of dollars. So, so my take is I think I can spend that money a lot better to get better results.

Interviewer
Yeah. Also see also see a lot of common ground with the event streaming background now absolutely with real time data management. So perhaps we can have a look at the infrastructure layer. This is a more extensive framework. This is what's happening around the data products. Not inside the data product. So I have identified in your article, for example the schema registry, which is connected with the central data product catalog and the central data product catalog enables some kind of central monitorization and centrally governed the data, etc. So for example for your global policies and to check for data lineage at the same time, we have the event streaming backbone. And some articles mentioned, uh, the shared storage. But I can imagine that the shared storage can also lead to some versioning problems. When we have a shared storage between two data products, and one of the data products is getting versioned, well, it has to wait for the other data product to yeah, to being version, to getting versioned as well. Otherwise, there are some problems with the shared storage. So they kind of depend on each other. In my perspective, this leads to some problems with being autonomous as a data product. So I'm not sure about shared storage, but a lot of practitioners they are, so we can perhaps discuss that. Furthermore, we have the API invocation, so there can be multiple access points like gRPC GraphQL, REST APIs. And these two are related with the shared storage. We need to have a SQL access point. This was actually mentioned the most cause, yeah, lots of data analysts nowadays require some kind of SQL access and on the bottom we see more some kind of nonfunctional criteria. So data product policy enforcements mechanisms where we see the security controls for example in in-memory cache and the query catalog. So the query catalog is really the sample data some kind of guideline where all the possible queries are defined for the data analyst. And fine-grained access is related to the security controls, but not really important here it's mostly about these options.

Expert D
OK, so so first let's tackle the shared storage. I think shared storage is counter to the data mesh principles and I think it introduced dramatic complexity. So here's the challenge, data mesh is all about you know, having clear boundaries around your data. And if I have a clear boundary

I have an owner who can make the decisions for my data product. OK and if I share it. Have an unclear boundary and I have unclear ownership. Who gets to decide what? And if there's clear delineation, then why do you have shared data? So I don't believe shared storage is not in keeping with the principles, and I think by the way, independent of the principles, it just makes things very, very difficult in practice. If you can't figure out who gets to make the decision about data, you'll be forever in a spin cycle trying to get decisions made. So shared storage. I would never as a practitioner recommend that, independent of the fact that it's against the data mesh principles. It just doesn't work very well. It causes too many problems now. There are instances where I have existing legacies. Legacy may be the wrong word, but my centralized enterprise data warehouse, for example, almost by definition it is shared storage. But here's how we deal with that. And then I'll go to the other items here. But the way we work here, is how we bring an enterprise data warehouse, centralized storage blob if you will, into the data product world. What we do is we say, first off, I'm going to trivialize. But the data warehouse may have all the data for customer account, products etcetera in the organization. But let's take customer as an example, we're going to carve out and say those tables or those set of rows are for retail, retail banking customers in Netherlands, OK. And those, these other ones, these data tables or rows are for commercial or business accounts. OK. So now and then some of them maybe these are for another one. These other tables are for international customers for bank, I'm using the bank obviously example. So what we've done now is we've used some, you can use domain driven design to do some of this. But now what we have is we've carved out our logical domains and what we do is we translate those logical domains into physical domains by virtue of putting boundaries around them, ownership and we start to provide the access mechanisms, some of those patterns there and we associate it with the data product. The set of the smaller granular set of rules as opposed to the broader thing. So even there is an enterprise data warehouse, which is by definition shared storage can be broken into parts. So I'll leave that part there. But let me talk about the others. Now that I have my data product, OK, there's a lot of most folks when they think of a database, they think of SQL, OK. But as we saw earlier in our discussion, a data product can be, conceivably, it could be one file, although I've never seen that unless it's really, really big. Like I've seen a very big climate data set that's like 10s of gigabytes big for precipitation data, for example. For all known history, that's a very big file, and that may be a single file data product. But for the most part, data products are sets of files. They could be databases, they could even be, they could be SQL or NoSQL, they could be, you know, ancient IMS systems or kick systems which have NoSQL plus the data products may actually provide interfaces via APIs, Restful APIs or GraphQL or gRPC or something. That's very the access mechanism is I as Tom I'm going to go and issue a request to my data product and I will get a response. The other pattern is the event streaming one where they say I'm going to register to get a data product and tell me when it changes. Posed to me going to get the data and polling for example. I'm gonna actually sit back, wait and listen. I'm going to subscribe to a topic and that's the event streaming approach. What we're finding is that is actually almost, although there's a huge legacy of SQL and stuff like that. As organizations, especially in banking and insurance are moving towards this real time enterprise they actually want to move towards I want to listen and wait for events in real time as opposed to constantly. Polling or waiting till the end of the day for a batch cycle to run. So what we find is, to make a long story short, your access methods to a data product will be SQL. Yes, APIs absolutely and listening for changes in the data product. So all of those are absolutely part and parcel of every data project. One thing I will say about the SQL access before I talk about some of the catalog and the registry because I think that's crucial. Is. In general, there's no organization that I know of on the operational side or on the analytics side that allows free form queries. There's no such thing as ad hoc queries. OK. It's for probably some very practical reasons. You know, data science, let's say, let's say we give Tom, who's the new data scientist, his first

gig after his internship at Bain and he goes and queries the Enterprise Data Warehouse, or, you know, a fairly large data product and does select star from whatever. OK, and you have a query that, you know, takes a long time, sucks up all the resources, and if the database is, you know, Snowflake on the cloud, you're going to, you know, you're going to have some pretty significant cloud charges, all of which was unintended and all of which as soon as Tom realizes the issue of this crazy query, he wants to stop it and he can't. So never do we actually see, and this is independent by the way of the security implications of doing it, having ad hoc access to sensitive data. Um almost always what we call them federated queries. They're preconfigured pre tested queries that we call them vetted queries and the data product has a set just like APIs a standard set of APIs to access. You have a standard set of queries and those queries have been optimized for speed performance cost etcetera and you know they address the hopefully over time the majority of the use cases so a lot of what I would characterize as new practitioners say that you know data scientists should have you know unbounded access to data stores. I find that very impractical and never happens in the real world now so again the key here about SQL is it's federated access, federated queries but vetted queries is the most important part Now let's talk about the top two, the data product catalog or as I call it now, the data registry, to distinguish from things like Collibra which are really oriented towards governing data at the highest level, the data product catalogs or the data registries I call it. What it says is if I have a data product with an owner and some clear boundaries, what's actually in it? So the catalog or the registry says this is the metadata associated with you know the customer data product, but it does a bunch more. The registry, this is what I call it, a registry. Not just a catalog, but it also provides useful information for the practitioner. The ingestion methods, what are the consumption patterns or consumption API, vetted queries, the topics I can list to. But it also shows what are the observability or operability information. So if I'm a data product owner, the registry also shows you what alerts occurred in the last 24 hours that I should be aware of and perhaps fixed. What are the usage carrier who's been using my data product? I should be able to look at that. So the data product registry provides not just information about what's in the data product. But what's happened to the data product over a period of time and all the other information that's related. So this is the one stop shop for everything about a data product.

Interviewer
Everything. Literally everything related to the metadata, right?

Expert D
Well, the metadata, the usage data, the operator operations data, all of that. Yeah, so it's much, much more than just the metadata. In fact, when we build registries for my clients, it has the slash metadata endpoint, just to, again, dramatically simplify, but it also has a slash usage who's been using my data product, slash logs what's are the patterns? You know what, what events were consumed, what data has actually changed, what the slash alerts. So when something when a data product broke and its processing, what happened was a security violation, I should be able to see all those. So, so one stop shop to see anything about a data product, runtime, security, metadata, etcetera. Now the schema registry is one thing that's in particular. The industry is not standardized the message format between these, so there's no common such form between SQL queries API, Restful, GraphQL or gRPC, nor events. So there's a variety of different I just call it in general schema registries. Kafka calls their registry of schema registry, so I use it in the general term. So any schema, any message pattern that comes in or goes in or out of a data product is part of the schema registry. And again. Is accessible through the data product registry, so I want to be respectful of time, but that's how I would react to this, but I think you've covered it very well.

Interviewer
Great yet to be honest I've all the time in the world. So it depends if you have some more time left.

Expert D
Yeah I can, I can extend a bit. I can extend a bit.

Interviewer
Oh great that would be amazing. Yeah I think we pretty much covered all of these already. All of these patterns over here and these practices. So yeah thank you very much and I will take the feedback into account to specify it even further. If we now look more in depth inside the data products. I was not pretty sure about this one. The immutable chains audit log. There were some articles where I thought you were referring to this as being a pattern outside the data product, but other articles were more like it's happening inside the data product. Do you understand the confusion over here?

Expert D
Yeah. Yeah, I can. Uh, I'll let you, I'll let you explain this and then I'll just quickly explain this.
Interviewer
The change data capture of course, which is related to the event streaming backbone. And yeah, I think this is a really crucial part of the data product. Internal storages, we encountered some kind of shared storage before, but each data product has at least should have an internal storage to be autonomous and maintain the single source of truth. Of course, the data catalog where you can register data sets. And there can be a distinction between these kinds of data sets. So the domain data set is more about when we have HR, for example, the HR domain. All the data products can exchange data sets within the HR domain. That's the domain datasets, but it can also exchange datasets with the sales domain and then it gets called a core data set, so then it can be exchanged with other domains. So that's the distinction over here. Yeah, the data onboarding is more about data ingestion. This is where all the transformations are happening et cetera. All the calculations, everything that happens to the data. And the control plane and the observation plane. The observation plane creates some kind of interface for the consumer to observe the quality metrics regarding the data. And a control plane is more some kind of interface on top of the data product, where the Federated team can make sure that every policy is applied. Make some adjustments if necessary, et cetera. That's how I observed all these options.

Expert D
OK, I think uh, I think this is pretty well done and covers most of the points. Let me kind of go through a few of these things. I'm going to start with the observation and the control plane as the kind of macro level things and effectively what the control plane says. This is where I come back to data proc again it has a clear boundary and has an owner, but it also has a standard set of interfaces. One of the set of interest faces that we normally build is the things in the control plane. So again, there's a lot of stuff that I don't mention in my articles, but simplistically control plane says stop and start the data product. Or activate or deactivate, or it may say allow Tom access or revoke Tom's access. So that's the control plane, so we provide. In all of our standard data products, we provide interfaces, we use rest. Rest by the way, almost an open API specification is almost exclusively. But those are the things that allow us to control the characteristics of the data product. So there's a wealth of those,

especially as you start to think about security, observability, operability, etcetera. You have the observation plane. I think that is well worth calling out on its own, mostly because the observation plane covers. It's a pretty wide breadth of capability it provides. So obviously you know. What's the metric statistics? How many requests occurred for my particular data product? All the way through who is, you know, who is executing the various different requests all the way through to. You can include the observable operability domain in this. There's a fine line between the two but I should be able to observe when something went wrong with my data product. I should be able to observe the performance of my data product. So there's a bunch of different endpoints we provide again as part of that standard interface and wrapper around the data product that allow us to observe what's actually happening inside the data product, including usage information. So those are kind of the two on this page. Those are the two core capabilities that stand out loud and clear now. The change data capture, I'll explain that relatively quickly, but this is really an interaction pattern that allows data products to interact in a data mesh. The change data capture says when something happens in one, I'm going to provide a mechanism for other data products or non-data products to listen for those changes and be notified with some happens.

Interviewer
Yeah, it's really like pubsub right?

Expert D
It is very much like pubsub except it's really pub sub for databases. So when something changes. The database notifies somebody else, and it's a crucial capability and it's not trivial to implement. Like raw data events from a change in the table. Are not typically useful, they have to be aggregated. Um, into more useful events to actually make that work. But once you have that ready, I'm done. That's how we that's how we actually facilitate moving as I mentioned earlier from the cloud from the mainframe to the cloud for example, and how we keep the data in the cloud up to date with the mainframe data. We use change data capture and publish them to event streaming platforms. Immutable change audit log is crucial for a bunch of so. So immutable means never changes. So anything historical that's effectively what it means. Umm, if you have a stored record, that means I never delete it. I never update the values in it, but I only append to it so it only grows. Now why is that so crucial? And by the way that doesn't say that I don't have a time window. You know, I may keep data for a year or whatever the business requirements suggest. What is happening, is you guarantee that those things don't change so that you can you can begin to have confidence around your service levels and your governance. So why governance? Well, the immutable change audit log also captures all of the changes that occur in that data. So you know things like address changes for example, things like who has access rights, who's been given access rights, who's been terminated access rights, who violated the security constraints and how do I log that? So we I call that an immutable change audit log to denote the fact that there's important information in there and it is a non-editable interface, but it is used. The primary use for it is around the observation and some of the operability things. So it is a log that is within the data product that captures important events that is there used again for audit purposes, security audits, you know, a variety of other capabilities like

Interviewer
that just makes a lot more sense because at first I thought it was more like a central thing. That keeps track of all the data products, so that's why I just said I'm not sure if this was inside the data product or more like in the infrastructure layer around it.

Expert D

Now what we do the other thing about this though, so for example, a security violation, somebody tried to access data they weren't allowed to. It is crucial that that you know, be addressed obviously within the data product, but there may be a more systemic thing. So the other part of what we do with, it's not part of the immutable change audit log, but the information that's in there also gets sent to other places. So security violations should be sent to the security team also they should be able to come back and look at the audit activities around that security event to see what actually occurred, but the information in the audit log absolutely can go elsewhere. And it almost always goes to your operation system or your security systems. Now I want to talk just briefly. We've talked about the data catalog or as I called the data product registry. So I won't dwell on that. But the last one, the data onboarding, I think is crucial. So every data product has a set of producers and a set of consumers now. Because the data product has a clear boundary and a clear owner, we still need to go and interact with other producers and other consumers. And what we want to do is be able to on, I call it onboarding, but provide the security credentials, the access rights to allow other data products or other users to actually interact with the data product. It's almost always has a lot to do with security and access rights. So, one of the most difficult things that I see in organizations is, is if somebody you know. First off, finding data is very, very difficult. You have to do a lot of different people before you can actually find the right person who knows what this data you're looking for and where it actually exists. But once you actually find out where it exists, trying to get access to it is a very, very difficult exercise. Mostly because every chunk of data, every database, has different practices and policies and processes. So what we found? Is getting access to a data product is one of the most well should be one of the easiest things but unless we think about onboarding is actually one of the most difficult things. Again security should be, should be rigorous but it shouldn't be difficult so that onboarding says if Tom what we actually do is it's another interface but when Tom or another data product wants to have access rights to it we have standard interfaces. So you know again trivialized. Slash access, right? So one data product can query that, provide the credentials. There's probably some manual effort perhaps that needs to take place, but that's largely hidden from Tom if he wants access to it. And then long behold, you know the data product owner has the final say, I would think to say Tom has access to it, and there may be a whole bunch of processes behind the scenes. But what ends up happening is that Tom who wants access to the data product, it's a seamless experience. OK, it's kind of like the analogy I tend to use is like a swan. Swan is beautiful above the water, but when it moves its legs are twirling around like crazy and actually do it's work. That's what the data product onboarding exercise looks like it should be seamless and the developer experience, the data scientists experience should be wonderful. OK, how the data product owner facilitates that going to whatever security systems it needs. That has to happen, but the experience for the consumer should be simple and enjoyable. Same thing that we need to go and get a data source. We ought to have standard contracts, standard service levels so that when we on board a data producer, we have very standard ways of actually interacting with. So while the data onboarding takes is more important on the consumer side, we try and make onboarding of data providers or data consumers. Really, really, really simple. And that's something that I think people do not emphasize enough, but I will be honest with you again. If you have 10s or hundreds of data products running, running, every single one has a different way of interacting and a different, you know, different onboarding pattern. It becomes very confusing very quickly and ultimately the user experience is horrible. This is one of the things that you know with a little bit of foresight and planning, the onboarding experience can allow you to bring people on quickly and grow your data mesh quicker than you normally would. So I'll pause there. I think I've addressed most of these.

Interviewer

The other framework has a lot in common with the observation plane and the control plane and of course the data catalog because this is more about the ports I identified. And I also did to other interviews before you and both they were mentioning the crucial aspect of of course the input port and the output port, which I forgot over here the most obvious one. But yeah, there should be a discovery port and I think that the discovery report is closely related to the data catalog inside the data product where we can get some kind of sneak preview of the data and know what's in there etc. And the control port is connected with the control plane and the observation port with the observation plane. I think we pretty much discussed these already.

Expert D
Like I said if you have the input and output ports, I think you're done.

Interviewer
Yeah, yeah. You think there are five ports as maximum.

Expert D
Well, these are all conceptual ports. So as I mentioned, it depends what you wanna call a port. A port is a logical contract that says I have a channel by which I can access some information. So for example, as I mentioned with observation, what we typically do is we have a slash usage, a slash logs and slash alerts endpoint. So those are all a conceptual. Observation form for it. But in in practice when we implement it there's there's lots of different endpoints so each, again depending on your definition, you either have lots of observation points or you have a logical observation points with many endpoints that so. So as long as you use consistent terminology, you're fine.

Interviewer
Yeah, yeah, sure thing, OK, I can specify this even further. Let's just go to the deploy decision. I think we pretty much covered all the interface and the data product layer decisions. Yeah, I only encountered two options, but I believe there are many more so I encountered Docker a lot. So containerizing your data product have a single container design. So for each data product there can be a container. And at the same time there's some kind of container orchestrator which is in most cases the Kubernetes engine. Um, yeah, and there can be some specifications here, so we can use the Kubernetes as a function as a service and the container as an infrastructure as code, but I think. There can be some improvements over here, so I'm really curious about your opinion on this.

Expert D
Yeah, sure, what I've typically seen. So you have a number of them. First off, everything starts with a docker container or a container in a different, slightly different format. Docker is a proprietary word, but anyway, the the whole idea of containers, absolutely, absolutely. That's the basic building block. Once you have a container, Kubernetes I see, as is very common. I also see, you know, it's not Kubernetes butVMs? So Docker containers get spun up in VMs OK, they see interfaces in some cases depending on which cloud vendor you're on. So you're interface is your function as a service to Lambda functions on AWS for example. Their access mechanism so the bottom line is you have a data product OK and it's just as a Docker container. Data. The one piece that's missing here is the data. The data and the data product are all in a zero trust, zero trust runtime environment. So what that means is you typically don't package. You rarely, if ever, package your data in a Docker container. Doesn't make sense. What you do is you have a docker container, some applications that mounts volumes for example. But ultimately what ends up happening is the data needs to be in a secure location,

a secure runtime environment in which the application, the Docker, the Docker container that contains all those endpoints you observe, and all the rest of the stuff that then is the sole mechanism to access the data in the runtime environment and all of that gets with the exception of the raw data gets deployed in typically a Kubernetes or a VM type environment now. The access channels, you know, the slash observe, there's a whole bunch of different ways you can expose those capabilities in Kubernetes. What you have is you know, ingestion ports and stuff like that. Ingress, sorry ingress ports. In a VM you have different ways of exposing it. And in AWS for example, but I think Azure has a similar concept and so does Google, but you may use AWS Lambda Services as the mechanism to actually access. So that becomes part of the broader data mesh fabric that we actually see out there. So that's how I would characterize it. You're probably missing, like I said, the VM and the whole idea of all this gets packaged into a zero trust runtime environment, yeah, but the parts you have on the right hand side of the screen infrastructure as code. Absolutely. You do not want to be having manual steps in any deployment process at all. And that's you know, that might have been OK in 2005, but not today. Yeah, sure thing. Templated data pipelines, absolutely. There's lots of different old style Jenkins. But there's standardization. Yeah absolutely. So the whole idea here is if I've standard of beauty of this is if I standardize my data product. And it goes into a container et cetera. And it has standard interfaces, OK, they all look and smell and feel the same. Then I can actually inject them into a templated pipeline in a very graceful way. So and I obviously as you saw there leads into a CI/CD process. So I think this is spot on with a few you know gaps.

Interviewer
Yeah some refinements. OK. Well, that leads us to uh, yeah, actually the last question and that's about the inter-decision. So I covered 6 decisions in total. But do you think there should be more decisions you should take into account while implementing data as a product or do you think we covered most of them. And well, what do you think about the relations over here, the chronological order, etcetera?

Expert D
Um. Yeah, there's, there's one. So, so one I'll ask it in terms of a question and I'll try and answer it. But one of the biggest problems my clients have is, is they've read the book, they maybe even read some articles and they say I love this data product and data mesh idea. How do I find my first data products? Now we have talked about domain driven design and I'm a very strong proponent of absolutely the right answer, but there's a lot of ways to short circuit that and go a lot quicker, which I don't think is in here. But when I when I look at you, how do I find my domain? There's a lot of different capabilities that you should be able to have. So for example, in financial services there is a fair amount of industry models by NBIAN, which gives you a list of all the relevant domains, even a hierarchy of domains within a bank. So if you're in financial services you should be looking at something like buy in. If you're in insurance you probably want to look at something like a cord, and in different industries there's probably different standard domain models. You should start with those before you even do custom DDD. DDD is a good exercise, but it you know the content it can take a very long time if you don't have some a starter kit or some starting point. So when I look at how do I find out which data products to go with? DDD absolutely, but I would start with a business architecture or an industry model that lays out the domains in an organization. The next, the last part I would mention is DDD lead you to domains that cross organizational boundaries. So I'll give a trivial example if I would do customer, OK and a naive implementation of a data progress, I'm going to put all customers in this one data product and I think that's a very poor match for a bunch of different reasons. DDD would not lead you to that conclusion. Let's say you did. The problem is customer spans multiple organization units and those organization

may be units, may be you know, I'm a bank in Netherlands and I have, I'm a big bank, maybe I'm ING or something like that. And I have customers globally in in 58 different countries. Every country has a different regulatory framework for customer, which means you're collecting different data, so customers is really not a customer. Every customer in each region is a little bit different, and each organization treats customers a little differently. But if I had a customer spanning all those different organizational units, I'm never going to get a clear boundary, but I'm never going to get a clear owner. So the concept has been, you know established in tech for a long time. It's called Conway's law and it is one of the things that I find most junior to even some senior practitioners forget to consider the organizational constructs, the socio part of the sociotechnical stuff. As you look at your data domains. 9 times out of 10 it may be architecturally impure to select the data product that resides in an organization. But nine out of 10 times it is the one that will get you a clear boundary, a clear owner and clear decision rights and that local autonomy that's required to allow data mesh and data products to grow. So the key here is select your domains wisely and DDD is one tool in your tool belt to use, but is not the only one and it should never be the only one. Rather the quickest way to figure out what data products you want is to start with a business architecture that most organizations typically have. They define the domains in there, an industry model if you haven't got that and look at the organizational model so that you don't, you know, so you take advantage of Conway's law as opposed to ignore it. So that's kind of one of the things we didn't talk about a lot here, but that is one of the things you may want to inject into your analysis.

Interviewer
Really good. Yeah, this is a really good point. And do you think this framework can help people with their implementation. Is it understandable for everyone?

Expert D
Yeah, I think so. Absolutely. So simple answer yes. In its raw form it needs, yeah, you need like, you need context, you need some words, you need to, yeah. Getting started guide like the stuff you have here, Zhamak Dehghani, the data mesh visionary, took 400 pages to write and she didn't, she didn't cover all these things. So what I would say is as with anything it requires a little bit of context. But based on how our discussion went, I think you're covering many of the right decision points.

Interviewer
Yeah, Ohh, thank you very much because this is just one of the first frameworks in a series. And yeah, the upcoming two months I will be deep diving into this self-serve platform a bit more. So yeah, we hope this is a good start and I'm also really looking forward to implement it myself well as a prototype because it takes a lot of time to implement it in an entire organization, but absolutely, really practical stuff.

Expert D
Perfect. And Tom, obviously if you want to have some follow-ups, feel free and then if you want, I'll be happy to volunteer for discussion on around the self-serve platform. But I don't know who you have on your view list. If you're interested, I'd be happy to help.

Interviewer
Definitely would be definitely interested. Yeah. OK. Thank you. *, I want to take any more time off your busy schedule. Thank you so much. And I will send you the transcript to make sure everything is OK. If. If you. Yeah, if that, if you want the transcript.

Expert D
Yeah, absolutely. Perfect. And thank you very much for choosing to select me. I'm flattered.

Interviewer
Yeah, sure. I'm a really big fan of your articles.

Expert D
OK, perfect. Well, thank you very much. Thank you very much. And I'm glad that I was able to help and I look forward to potentially, you know, seeing the results of this work sometime in the future.

Interviewer
Sure we can. We can make that work. Alright, thank you very much for this. Take care.

Expert D
Bye, bye.