A.10 Expert J

Interviewer
Hi, I'm Tom. I will first give you a short introduction about myself. I'm 24 years old, I'm Dutch. I have a background in industrial engineering, so I came from some kind of business background. That was what I'm doing, that was the thing I was doing in my bachelor. And now I'm doing a master in data science and entrepreneurship, where we try to combine the business aspects with the data science aspects, more technical side of data. So I'm really interested in data engineering and now in data mesh and I think data mesh has a lot of similarities with the business perspective as well. So when we think about data mesh, there are some business aspects as well as technological aspects. So we will deep dive more in the data as a product principle in just a few minutes. So that's some background about myself. By the way, I'm interning now for Bain & Company, which is a strategy consulting firm. And yeah, that's where I'm doing my master thesis right now. So yeah, that's more some background information about myself, if you can introduce yourself as well.

Expert J
Thanks, Tom. Well, *, as you know, a 35 year old in the meantime. A little bit more about the professional context. So the last two years I've been working with *. Before that, I had the opportunity to work two years as the team lead of the data team at the energy company *, also known in Holland, in the Netherlands, of course. Although in Belgium it was a lot smaller, but still also had the chance to work with some Dutch colleagues. So very nice. And before that, I think it's about 11 to 12 years of various IT stuff where I first started in technical support and then got in touch with data. And then started doing BI for I think about eight or nine years. The business intelligence development is basically what you would call data engineering right now. With the difference, of course, that the BI development in the beginning was more built on ETL tooling. Whereas now you have a lot of data engineers who simply write their code and do the thing that the tools used to do before you. But so, yeah, I think almost 15 years experience when working with data in the meantime, loving data. And now the last, what is it, I think five months, I've been asked or assigned as you wish by * to focus on the data mesh transition we're going to do within *. Great. Yeah, very great. And although for me it's a big step because I've always I was, I used to develop, then I was operationally responsible for keeping things running and then assigning the teams, etc. Then when entering *, I focused on the agile way of working for all our data teams. So a lot of operational and a lot of management and now it's kind of a bit of visionary strategy stuff that I need to do. And yes, it's adapting a bit for me, but I love a challenge.

Interviewer
Yeah, because I've read one of your, well, one of * articles on Medium, which was about * and the transition towards a data mesh. I think * was the one who wrote this article. Yes. So I thought your data mesh was already applied, but you're still in some kind of transition or am I misinterpreting things?

Expert J
No, you're not misinterpreting. The thing we as a company sometimes do, or some of my predecessors did, was stating a bit too much that we're already in a data mesh situation. Whereas we simply, well I think two or two and a half years ago, we committed ourselves as a company towards going to a data mesh system. And at this point in time, we're still in a transition. I would even dare to say that we're not there yet. We're taking steps. We have something we call a data product. But when looking at things from a theoretical perspective, I would not say

that we fully check the boxes of a data product as it should be. That's my interpretation of the data product, but in the company we call it a data product. And we're taking steps towards building the mesh with several data products, but we're not there yet.

Interviewer
Okay, wow. This sounds perfect for this interview because we're now going to talk about data products. I will show you some of my frameworks I've developed the last couple of months actually. And everything is based on grey literature and grey literature is actually everything that hasn't been officially published in an academic journal. So you can think about YouTube videos, Medium posts, white papers from Google Cloud, AWS, Azure, etc. So everything that hasn't been published in an academic journal, because there is not a lot of academic literature on data mesh yet. The academics, they are a bit behind the potential. It always takes a couple of years before they are triggered by the idea. So yeah, these decisions over here, this is a very high level framework I've developed. This is the, yeah, it gives you some kind of guidance, considering the other frameworks I've developed. So the first decision I observed was to check for yourself what kind of data product you want to develop. So let me get a laser pointer that's more clear. So this would be the first decision is more conceptual, but I think it has a lot of influence on the other decisions you make in your data mesh journey. Because if you want to start with a source line data product, there will be different needs on the output boards, on the data product anatomy. So what's inside the product. So that's why I defined this as the first data product. And if you want to expose your data product as raw data, or if you want to expose derived data, it all has influence on your other decisions. And if we go one step further, which approach is chosen for the creation of a data product? This is more from the migration perspective or from the greenfield development perspective. So do we want to start from scratch? Or do we prefer to migrate from a legacy architecture? So do we want to do legacy modernization? Or do we want to do cloud acceleration where we actually convert all our resources to the cloud? At the same time, we can make a decision on the outside layer. So what is happening between data products? How does our data product communicates with the self-serve platform? How does it communicate with some kind of government layer and of course, the consumers? Then we have a decision on the data product itself. So now we really deep dive into the data product anatomy. So what kind of architectural decisions do we need to make over there? We can think about the data product interface / contract as well. So what kind of ports do we need on the data product perimeter, let's say. And of course, the last step is to deploy your data products. So there are certain decisions related to deployment. So if we start with the first one, the question over here is what type of data product can be developed? And I identified three core options. So the first one is to expose your data product as raw data. There are some frictions on this perspective, because some practitioners say that you should never expose your data product as raw data to the consumer because yeah, it's dirty, it's not clean. You shouldn't do that. But other practitioners mentioned the fact that it can be really interesting for the consumer, but it depends on the consumer. The second option is to expose your data product as an algorithm. And we can specify this even further in an optimization based decision support system. So for example, a BI tool, a dashboard, or in an AI slash machine learning model. And the third option is to expose your data product as derived data. So some small transformation steps have been applied, etc. And we can think about the some kind of hybrid form as well. So with a hybrid form, I mean, that's your data product has, for example, two output ports, an output port for raw data and an output port for derived data. So you can expose both of these aspects. And there can be a composite data product and a composite data product is some kind of merged version of, yeah, for example, derived data in an algorithm. So it's like merging the best of both worlds. This is actually what I found in my literature research. And I am really curious if some of these aspects, if you resonate with some

of these aspects, if they come, if they are familiar to you.

Expert J
Yeah, the things that are familiar are specifically the output ports, like you mentioned, and what to do with the products itself. But from our perspective, at this point in time, we don't really think about a type or the data type product, as you mentioned here, because from our perspective, you have data product. And for the data product, you need to abide to a set of principles. And those principles should all be the same, no matter what you are delivering. If we if we take for example, the only data product we have right now. It's the subscriptions data product. And it's all about subscriptions and there are some stuff that are that are aligned to that. And if you're a purist, you could say okay that needs to be a new data product but let's skip that discussion. So we have a data product with some data around it. That's all about subscriptions. If you're looking for subscriptions, you go to that data product. And there we have two different output ports, where we have the output port to put it in technical terms it's S3 where we have some files. And on the other hand we have Snowflake, where we have 13 or more tables, which you can neatly query and blah blah blah. So, if I look at this schema, then you would call it some kind of a hybrid product. While, in fact, we don't make that difference. So I wonder what the value is in specifying these different types. But besides that, we ask that no matter what output port you give for the specific data product, the data is the same. Well, yeah, we think about interoperable as a principle between our data products, but specifically for the consumers. Like you say, there are, let's say if you're an analyst and you're thinking about developing some kind of report you're probably going to dive into the Snowflake output port and be able to view the data bits, play with it, query it a bit. While on the other hand, if you're working on a machine learning model, then querying the table won't suffice. You need to have multiple versions of the data ready, etc. So there you would simply go and look at S3 data and start grabbing the files you need to do the logic unit to form a new data product. And that's a bit how we look at it. So the couple of decisions you make here, we don't think about them. Yeah. So I wonder, yeah, question I already asked, what's the specific value for determining which output ports you're going to use and then choosing another data type?

Interviewer
Yeah, this was more some kind of manual. So to just make the user aware of all the possibilities to expose your data, so it's not mandatory to just choose one. You can, for example, choose the hybrid data product as well. It's just to give you some idea on what's out there. So, of course, you can expose your data product as raw data. This would be the source aligned data products.

Expert J
Just to mention, in our company, we have already put a strict no on the raw data. OK, to expose it to the consumer. Never or never will we allow raw data to be exposed in one way or another. We will start with source aligned data products that read the raw data, create a data product, and then we have our consumers tapping onto those data products.

Interviewer
Yeah, I've actually one question, because during your explanation, I noticed that the S3 buckets can be observed as more some kind of source aligned domains. So the source aligned domains contain the S3 buckets, and as soon as the data flows into Snowflake, it's more on the consumer aligned data product side. So I don't know if I'm interpreting this right or do you also do source aligned data products on the Snowflake side as well?

Expert J

Yeah, so it's the same product, in fact. So we have our source aligned data product subscription, and if you dive into the details, we have a couple of operational subscription systems. We tap into all the data over there. That's our input port, several applications, then we do our logic. And then in the end, we expose data products about their subscriptions, but in two formats, in two different output ports. So one is for Snowflake, you have 13 tables. And on S3 you have 13. I'm not sure I'm not going to, so don't, how to say, don't pin yourself on this, but I would suspect that we have 13 file parts in S3, where these files with the same data actually are dumped but in another format. So, then you would have probably JSON files. So, actual data remains the same, but how the data is exposed to the consumer is different.

Interviewer
Yeah, got it. Okay, thank you. I think in terms of time, we can quickly go to the next framework, because I think we discovered all of these aspects already. So let's quickly go to, yeah, this is more about your strategy. So you have defined your data product type, let's say, and now you want to choose between migration or greenfield development. And greenfield development, well, that part is perhaps not really exciting. It's more excited to, I'm more excited to talk about the migration patterns I observed. So, yeah, I actually observed four patterns that are implemented in data mesh quite a lot, actually. So, master data management, of course, there is some kind of, let's say we have a customer, and this customer can be in the media player domain, but it can also be a listener and it also has subscriptions. So we need to have some kind of central authority that keeps track of all the definitions of the customer, well, the customer is related to multiple domains, and we need to have some master data management tool to keep track of, yeah, everything that is happening to this entity. So Strangler Fig is more about decomposing your current monolithic architecture, and at the same time, we're slowly building up our data mesh architecture. Do you have an idea about what Strangler Fig is? No, I've never heard it before, sorry. No worries, I can quickly show you that Strangler Fig pattern. So, this image explains it really well. You have a monolithic architecture, and you want to migrate to a data mesh architecture. Well, the thing you do is extracting a service one at a time, and your monolithic architecture shrinks over time, so it's getting smaller and your data mesh architecture, so your ecosystem of data products is slowly building up, but you still have your monolithic architecture. Yeah, at the same time, so yeah, along your data mesh journey. So that's what Strangler Fig is. Zero Trust architecture is that you apply fine grained access control to all of your data products. So if I have access to one data product, it doesn't mean that I have access to all the others as well. CQRS is about segregating your Read and Write function to make sure that if you read something in your data products, it doesn't mean that something is changing as well. So just about separating these functions. Do you recognize these patterns or do you think that I'm missing out on some patterns as well?

Expert J
No, no, no, whatever, all you described in here is something we see returning as well. But I was wondering, so are those four different approaches to the creation of a data product?

Interviewer
No, you can use all of them actually. You can use all of them to make your data mesh. It's not mutually exclusive, so you can choose multiple options.

Expert J
Yeah, because to put it into practice, we're using the Strangler Fig pattern then to do our migration. So that's for sure. We're taking specific pieces who are very valuable, and we have a high amount of stakeholders. Those are the ones that we're targeting so we can put them in our

data mesh as a data product. But on the other hand, we do a lot of the zero-trust architecture in the meantime, so all of our accesses for those data products are really fine-grained. It's on an individual level. You get added to a group. And that means that you're able to have access to an output port for a data product. If you want access to another output port, you need another access. On the other hand, we also have the CQRS pattern. That's a bit of a difficult one that we're still struggling with. Because for the data product, as I mentioned before, you have several output ports, but basically you would want data products to be bitemporal. And that's something we struggle with for specifically the Snowflake output port. For the S3 stuff, you could say, okay, I render, I do my transformation, I process my data and I put it there, and that's version X. Next day, X plus one, X plus two, pop, pop, pop, pop, pop. But for Snowflake, of course, you want to expose your users to the data. They should be able to query it, but they're not people who work with tables. Mostly another way of versioning or the bitemporality or using newly processed data. They view it more as an operational database where, I mean, the way you operate with the data is more as an operational database. Like a new customer enters. Well, immediately you get a new line in it. Customer has entered, customer leaves, then you put a flag somewhere, customer is deleted. But of course, if you're looking at the data product and the bitemporality of the data product, that should be a practice. You're missing that. So you should actually want different versions as well in Snowflake. Yeah. And that's where I think the CQRS pattern comes in a bit, which at the moment we haven't really found out or answered that question yet.

Interviewer
But have you found some kind of work around so that you can still provide value?

Expert J
Well, at the moment, yeah. The lucky or the lucky, I'm not sure if it's luck, but in practice, we have a lot of people who are working with dashboard loads. So they're doing a lot of dashboarding and those dashboards, well, they just want to show one version. For them, it's still there's one version of the truth. And we want a view on that. So they look at the database Snowflake, this case, and you have all these reports living around in the company. And that's a good thing. On the other hand, you want to detect for machine learning, for instance, the subscriptions. You want to see what happens, coupled with other events in the company. And for that, you really want that versioning and you want to be able to, like I have my customer now, but six months ago, how many customers were there? How was their inflow? Where did those customers come from? And how is it right now? So for the S3 stuff, we're already doing that. For Snowflake, we're not. So it's a bit of a practicality that we're still needing to refine because it's not OK as it is right now. And that's what I mean. Like we have a data product, we call it a data product, but we're still refining. There's still some rough edges that we need to shave off.

Interviewer
Oh, wow, that's really interesting because at *, we're also doing all our stuff in Snowflake. So I haven't really thought about this problem, but this is certainly something I can have a look at.

Expert J
Yeah, but it's logical if you're working with data. For me personally, working with data, it was never like this. OK, you had the slowly changing dimensions in the previous warehouses where you could link, etc. You have some kind of version history, but I think it doesn't suffice for the needs for data in the current day, basically.

Interviewer
Yeah, it's never that black and white. No, exactly. OK, great. Now we can have perhaps a look

at the outside layer, which is a more extensive diagram. So let's just start on top. I identified a schema registry as a very important element of your data mesh. And a schema registry is actually the component in your architecture that translates a change event into a more user friendly event. So if something is changing in my internal database of the data products, a change event is going through the change data capture, while the change data capture notices this change. And he sends the change events to the event streaming backbone, for example, Kafka. And the data products that are subscribed to a certain Kafka topic, they will all get notified once something is changing. And this change also is stored in the central data product catalog. Yeah, changes through Kafka are not that user friendly to read. So the schema registry actually makes sure that this change event is made more user friendly to read. So it's more readable. The central data product catalog is the catalog that keeps track of all your metadata inside the mesh. So it's more like a static tool where all the metadata is stored and you can discover everything about your data mesh. So it's like a one stop shop for everything. So if you want to discover some new data or if you want to search for specific columns, etc, you can just go to this central data product catalog and it will help you in finding it in the mesh. The event streaming backbone, I think we covered this already. It's the Kafka, for example, Kafka. It takes care of event streaming inside your mesh. It keeps track of the change events, etc. Some practitioners were talking about a shared storage. And yeah, there were a lot of conflicting opinions on this one because on the one side, on one side, you want to make your data product as autonomous as possible. And if there is a shared storage, for example, data product A has to wait for data product B when one of the data products is getting versioned. So they are kind of dependent on each other when you use a shared storage. But on the other hand, we really need a shared storage. Otherwise, there will be like 10,000 different storage accounts to manage if you have like 10,000 different data products. So that can be really costly as well. So there can be some trade-offs in this one. API invocation. So how is my data product accessible for the consumer? This can be realized by REST API, GraphQL, gRPC, but we can use SQL access point as well. And on the bottom, we see more like a non-functional requirement. So we have the security controls, a memory cache to keep track of the most complex queries, etc. And a query catalog. And what a query catalog actually is, is some kind of manual for your data analyst where all the possible queries are stored. So that he or she can just look all the queries up and they don't have to define that query by themselves. So this is really the high level overview of the... Yeah, it's more like the infrastructure layer. What's happening around the data products? Yeah. And do you recognize some patterns? It's quite a bit actually.

Expert J

No, no, no, but I recognize all of them. Oh, great. The thing is, there's some stuff I would like to talk about specifically and that's like the shared storage. I understand what you're saying specifically, like, otherwise you have 10,000s of stuff or locations to manage whatever. Yeah. But there we come with our platform. So we have the self-serve platform in here. I'm not really sure if we look at it the same way, but you have... We are building a data platform which allows or which helps our users building their data products. Yeah. For instance, the shared storage. We will never work with shared storage in such a way because what you say is a big problem. But if you're... I'm not sure how to look at it technically, but suppose you would be working with some on-prem data mesh components where you have a physical disk. You want to share the storage. If you run out of storage while your data product fails, you could impact multiple data products. So that's a firm no for us. On the other hand, from our data platform, we think we should be able to manage everything and we should also allow some autonomy for data products. So what we're going to do there is allow a specific spectrum of storage options for our data products. So Snowflake is one of them. S3 is another which we already use. But if there's a new data product, they will get their own S3 folder. If you're looking for Kafka

as well for the streaming options, whatever, they will all have their separate topics or their separate infrastructure so they do not impact each other. Of course, it's a bit difficult to say because Snowflake, it's one account. It is shared storage 100%. On the other hand, you have, if you look at the technicalities of Snowflake, it can't go down. If it goes down, then there are more problems out there in the world than simply Snowflake going down. So that's a thing I would like to mention. But on the other hand, the product, it's the same. So like the data product catalog, we also have it. We need to register for the discoverability principles, the trustworthiness, schema registry. I think we're also talking about the same there. And the APIs, there it's a bit, as a company, we're still struggling a bit with that. Because if you look at it from a data platform perspective, you actually want every option or every pattern you see here to be solved by using an API. As a data product developer, you should say, okay, I want storage, I want storage like this. I call an API, pop-up platform puts it ready for me, and here's my storage I can use. Same goes for the data catalog. You have your information, your metadata, you just call the API and it gets pumped into the catalog. And for that, we're being a bit more practical at the moment. So we don't build our APIs yet ourselves for the platform, but we help our data product developers by using the APIs for the infrastructure we have chosen. So for data product catalog, we're using data hub. And there we have an API, which we help our data product developers set up so they inject the correct metadata we ask of them so that our data products in the mesh are discoverable and have all the principles we want them to adhere to.

Interviewer
Okay. Interesting. Yeah. So, for example, in terms of policy enforcement mechanisms. I can imagine that you have something like fine grained access control or is each data product accessible for each Snowflake account, for example?

Expert J
No, no, no. We have a big segregation in our Snowflake rule set up for that. So basically it starts with, if you as a data product developer or I'd say a data product owner are responsible for data products, you control the access. So you start with your own team, your own domain has access to your data because you're all working the same domain so it's logical. It makes no sense if you would have a guy working on the operational side of the system who has no access to the data product because it's the same data. Yeah. But on the other hand, if another domain wants access to your data, they need to request access for that. And if so, then they look at okay what team are you, for what reason, and then they give the team access to the data. And that's how our access control works. But for that as well, you would also hope or my dream at some point is where you have the platform, and you're simply someone who's looking at data, okay you're trying to discover sales data or something. Sales data about European quarters about that kind of product category. Okay, I want access to that. Just click a button, data product owner sees okay this guy with this role from that team wants access. I also have a chat with his responsible product owner and I can give him access or deny. And that's the dream of course.

Interviewer
Yeah, yeah, that would be amazing. It would take, yeah, that would take all the technical aspects away for the data product developer. He or she can just do an API call and everything is provided to him or her.

Expert J
But that's also how we look at the data mesh, so we don't only see it as an architectural view we really see it as a new way of working with a new way of ownership where you have the

governance that the governance of all the data is being held by the data platform itself. So you as a developer data consumer are able to work or develop data, and just focus on that. Not all of the other stuff that you have to take care of. It becomes an operational responsibility of the data product owner to make sure his access is in control. And at some point you would want our data platform to say, oh but look at this, you have a data product, but you're giving access to 80 different kind of groups of people. We think this is abnormal, or this is way way up to par or whatever. And that's the kind of system we're looking at. Yeah, we're aiming for.

Interviewer
Great, I think this is a nice transition towards the data product anatomy as well. So the next framework. Yeah, there are certain aspects I observed so we already talked about the change data capture so this is really about observing changes in your internal storage and notifying all the other participants in the mesh about it. This is the immutable change audit log. This is, for example, if you send a change out there in the mesh. This is where it is stored. So it's like a database, it has an append only mechanism, it's immutable so it doesn't change over time. And if something is changing in my internal database, it goes through the change data capture, it goes through Kafka, and it is finally stored in this one for each data product. So each data product knows what exactly changed, etc. And it is being stored over here. So this is really the change log, it keeps track of all the changes within the mesh. Yeah, internal storages, we briefly talked about this as well. So, yeah, I placed the shared storage in the other framework because if we have a shared storage, it would be outside the data product anatomy, but we free have an internal storage which is also an option, we would have this one inside the data product boundaries. The data catalog, so there can be an enterprise data catalog, which is like an overarching layer that keeps track of everything inside your mesh. And we have an internal, well local data catalog, which is more like a metadata repository that data scientists can observe to check what's going on inside my data product. So it's more specific compared to the enterprise data catalog. There can be an observation plane, and this is more like an interface, which you can check as a data scientist to observe the data quality. So what's the data quality of the data set inside my data products? We have a control plane for the enabling team. So the ones who are taking care of the global policies and they can enforce policies through the control plane over here so they can make sure that all the data products adhere to the global policies and they can adjust it to this port, to this plane. Data onboarding is really about the transformations, the ingestion processes, things like that. It's really like a Spark thing. These are the components I observed in my research. Do you think this is complete or am I missing some components, perhaps some patterns?

Expert J
Well, I was wondering like the data onboarding, you talk about ingestion and you say it's more of a technical thing like a Spark thing. Does that also mean like, yeah, the way you process your data is also handling data onboarding?

Interviewer
Yeah, it's like the internal ETL pipeline. So we don't have ETL pipelines in our mesh, but we can have ETL pipelines inside the data products, more like an internal implementation. That's what the data onboarding is about.

Expert J
Right. Yeah, no, then I think you have pretty much everything I can think of right now at first glance. The only thing I still, that caught my attention is the immutable change audit log. So you talk about the change data capture where you send out change events, but on the other

hand, how does the immutable and the bi-temporal aspects of a data product come into play with this pattern? Or do you say like, for the framework I've developed, I do not take into account the bi-temporality or the immutability of the data sets itself?

Interviewer
Yeah, in fact, it is more about the bi-temporality. So we have different timestamps. And if something is changing, it is stored over here. And how it usually goes, it's not stored anywhere. So there is no data lineage. And if we store it in the change log, we can observe every step that has been applied to the data set. So it's more like keeping track of your changes.

Expert J
And so if you would have two source-oriented data products and you have another aggregate data product, we're taking the two source data products and working with it. You would also expect the aggregate data product to subscribe to the immutable change audit log to discover when data itself changes or what's the specific value? Is it the fail-safe option?

Interviewer
Yeah, I could perhaps show you some graph about it. Because if I'm quick... No, it's not this one. Pattern images. Yeah, so it's really like this. You have your interoperable database. And if a change... Well, if something changes in there, the change data capture notices this and it converts this change into a change event. This change event is first compared to your schema, to your schema registry. So if it complies with your schema registry, it is published in the event streaming backbone. And every data product that is subscribed to this specific Kafka topic, they will get notified. And it actually flows back into the change audit log over here. And if we have multiple data products, what we see here, something changes in this data product and this data product over here will be notified as well about the change. So yeah, this can happen in the aggregate data products, but it depends if the data product owner of the aggregate data product thinks it's necessary. Well, then he can subscribe to this Kafka topic. But if he doesn't think it's necessary for its internal processes, yeah, there is no reason to subscribe to this Kafka topic. So it's really about event streaming. Not everyone in the mesh has to be notified. Only the ones who think it's necessary to provide value.

Expert J
Yeah, okay. So it's a bit specifically for the streaming output ports.

Interviewer
Yeah, that's actually interesting that you mentioned this because that was feedback I received a couple of times from other practitioners as well that my research was a bit too focused on event streaming, on real time data instead of batch processing as well. So now I'm kind of reconstructuring my frameworks. And for example, I implement a blob storage as well as some kind of output mechanism or a table, a query table, big query table, for example. So, yeah, in my new graphs, I'm making everything in Python so I can change everything really easy instead of, yeah, because otherwise I have to make thousands of draw diagrams after every interview. So, yeah, I'm kind of changing some small aspects. I can show you. Which one was it? Yeah, I don't see it right now. It's an interesting aspect.

Expert J
And I think it's also, from what you tell right now, I see the value. The other thing is, I wonder how...

Interviewer
Yeah, so it wouldn't be really valuable for, yeah, data mesh where batch processing is like the main method.

Expert J
Indeed, but that's a bit how you approach your data mesh, of course. So we at *, we chose to handle our data mesh as a set of data products who have different kind of output ports. So it could be that, for instance, if you have a top 10 articles most read that you provide that in some kind of streaming option, even though it's a bit weird. And on the other hand, you have batch based processing that happens. No, it's not even, it's a good example. So the top 10 most read articles, it's real time. It's calculated real time. And every time you create or you look at the API, you have that moment in time. On the other hand, you have a batch based for perhaps specific reports or whatever that calculates it every 24 hours or every six hours. You get a top 10. Yeah, the change data capture method you talk about is interesting. But for specific workloads, you do not want to subscribe to a log to see the changes. You really want to have the different versionings of set table. So if you would have a table with 10 records in it for the top 10 most articles every six hours, then you would want to be able to see, okay, what has been the movement in time of those 10 articles. And you would simply want to query and say, okay, I want the last 30 states or the last 30 times you processed the data in batch. And I want those to be taken into my data product to do some kind of stuff with it. And if you're then working with an immutable log or change audit log, that would be too much for the batch based stuff, of course.

Interviewer
That's a really good remark indeed.

Expert J
Yeah. On the other hand, I'm talking, I have the background of really working with the data. So I'm a bit practical in this. On the other hand, the simple immutable change audit log, it can be anything. So it's a theoretical focus point. But working out will be a bit different, I guess. Yeah. The practical implementation.

Interviewer
Yeah, the practical implementation. Okay. I think we can quickly continue because we only have eight minutes left. I don't know if you have a few minutes extra. Yeah. Okay, great. Okay. This is more like the interface graph. Actually, I forgot the two most important ports over here, the input port and the output port, of course. So there are actually five different kinds of ports that I observed during my research. So we have the input and output ports. There should be an observation port, a control port and a discovery port. And I read an article from *, he is an author of a book from O'Reilly. And he mentioned the fact that we should split the observation port and control port. So that's why I did this over here. And in the book of Zhamak, she actually defined these two as just one unified port. So, yeah, I'm really curious about your perspective on this. Do you think we can split the observation port and control port or do you not recognize this in your own implementation of Data Mesh?

Expert J
At the moment, we're not really that far that we can talk about all the different mesh ports I would talk about because import output is specifically to the data product itself. On the other hand, the three ports you're talking about here, they come more, they touch more in terms of the mesh. But it all depends a bit like observation and control port. What do you specifically

mean with it?

Interviewer
Yeah, they are really connected to the observation plan and control plan we mentioned over here. So the observation port is like I as a data scientist want to know what kind of data quality is present in specific data products. And the first thing I view is the observation port. So I can connect to this port and check the data quality metrics, etc. The control ports is related to the control plane. So it's like a control interface. As an enabling team, I want to check if all the policies are enforced in this data product. And I want to be able to adjust some policies or policy out automation mechanism, etc. So this is more the entry for the enabling team. The discovery port, this one is connected to the data catalog. So the local data catalog just to check what kind of metadata is present in this data product. So it's more in terms of discovery, checking what is feasible in this data product.

Expert J
Yeah, I understand. Well, I think that how you describe it, the control port and the observation port itself will become the same thing with us. Because the data quality and looking at all of the, how do you say, the implementations of the trustworthiness, like the SLOs, SLAs, whatever, refresh rate of the data. That's all part of the observation game. On the other hand, I think we do need some kind of control port where we are able as a company to push a specific change regardless of, well, regardless, depends, of course. But for us, for instance, you have customers who gave their consent on our website to share their details. And it's a classic example. I think it's also mentioned in the book, by the way, from Zhamak, that you say, OK, I want to be forgotten. I have the right to be forgotten. Just click the button. And there comes in the central aspect of the control port, I believe, where you want the data platform to be able to control your data product in such a way that the data handles the request of the platform. So it's, I think, a shift from specific topics you mentioned in the control port, go to the observation port. But the port itself probably, or we would like to still have such a port where we can push specific changes. But it's there. I don't think it's pooling, like observation port we want to pool. And the discovery port, of course. It's something we have as well. Although I wonder, because in the other schema, you also had the internal data catalog.

Interviewer
Yeah, the local data catalog.

Expert J
Yeah. And the enterprise data catalog. Yeah. What's the difference for you specifically then?

Interviewer
Yeah, the local data catalog is more like a deep dive into the data products. And it makes the data products autonomous. So it does not depend on the enterprise data catalog, but it has its own catalog that it can provide to a consumer and to show the consumer what's inside the data product. And the central data product catalog is like a higher level of metadata in your mesh. So it's defining more like central aspects of all the data products. So it's more like a high level perspective. But I have had a discussion with other practitioners and they mentioned the fact that we should actually split the data catalog and some kind of meta store. Because there is dynamic metadata, for example, and dynamic metadata. This is metadata that changes over time, changes real quick, actually.

Expert J

Sorry to interrupt, but could you put into practice what type of metadata would that be?

Interviewer
Yeah, well, for example, the data lineage. So if something is, yeah, if the data set is being transformed, several steps are being applied. These steps bring some changes to the data set and these changes, they are relatively quick and someone needs to keep track of them. So this is more like dynamic metadata and schema versioning, of course, because schema versioning can... Yeah, I heard from practitioners that that's something you don't store in your data product catalog and perhaps more in a metadata lake or things like that. So there can be a distinction between them.

Expert J
Yeah, I understand what you're saying. And I think it's good that people like you think about it, because in theory, it would be a really, really great addition to the way we work. But in practice, I haven't seen it yet. The internal data catalog or the local data catalog. And also in our plans at the moment, we don't take that into account. So we let the people work how they want. Perhaps some teams might benefit and will do it. But like our enterprise data catalog contains a lot of data. So you have the definitions of all the specific fields that are available for the end user. You have the data types, of course, you have the goal, the meaning of the data product, the table, etc. And that's the only stuff we keep in. And we also have, you know, this discovery report. So that's part of our discovery aspect of our data mesh implementation.

Interviewer
Okay, great. Yeah, I think we can quickly go on to the last one. This is the deployment of a data product. And I was actually too focused on, yeah, I shouldn't use proprietary names here, by the way. So Docker is more like containerization using containerization. And Kubernetes, of course, a container orchestrator. Yeah, orchestration. And we can actually use the function as a service as well as a second option. Yeah, if we reconstruct this framework, it would be like containerization, a function as a service. And someone was, sorry, I think this room is booked. Someone is standing in front of the door. Okay. Sorry, I can go over this room. Oh, no. Okay. Yeah. Oh, yes. Okay, sorry, I need to leave the room, but it was amazing to talk to you, Michio. Yeah, I will perhaps just send you this. This is the last framework and perhaps you can send me some feedback over email and it will be amazing.

Expert J
Yeah, no worries. See you. Bye bye.

Well I can be fairly brief about that. For now, we have chosen to leave the deployment aspect entirely to the data product development team. Again, the desire is that one day a service can be addressed via the central data platform so that the entire product will place itself neatly into components. And so all based on metadata and addressing such a service/api. But we are not that far yet. From what I see in your diagram, I wouldn't peg so much on a "Decision". I would rather go for 1 addressing that triggers the deployment, but based on all kinds of metadata passed along to that service, set out different possibilities. But that's without your voice-over and only with my interpretation of the drawing so take my words with some caution.