Published in Slalom Data & AI

Matt Darwin Follow
Oct 15, 2021 · 10 min read



# Data Mesh: is the argument a strawman?

> *noun:* ***strawman***
>
> *1. an intentionally misrepresented proposition that is set up because it is easier to defeat than an opponent's real argument.*

The Data Mesh is a concept of decentralised data architecture and ownership introduced by Zhamak Dehghani of Thoughtworks, and is currently gaining a lot of traction in the data world. On the face of it, it is an incredibly elegant and compelling narrative, one of which I can relate to with several of the organisations I have worked with. I won't delve into all the details of the concept and architecture as these have been covered very well by Zhamak here and here which I would encourage you to read and digest as I will refer to parts during the article; and as with anything, just because I hold some opinions, it does not necessarily make them correct: please do share your thoughts and interpretations in the comments; is this article in itself a strawman?

However, in some respects, the argument presented by Zhamak appears to have an element of the strawman about it; the finger is pointed firmly at centralised architecture, namely centralised data lakes, being the problem, but then delves into issues with *centralised data teams* and the bottleneck that they cause. In this article, I'd like to explore this, and look at where simply implementing decentralised ownership of data without looking at decentralising the architecture would likely resolve the key pain points an organisation is suffering. There are several benefits to this approach, amongst which throw away existing work that

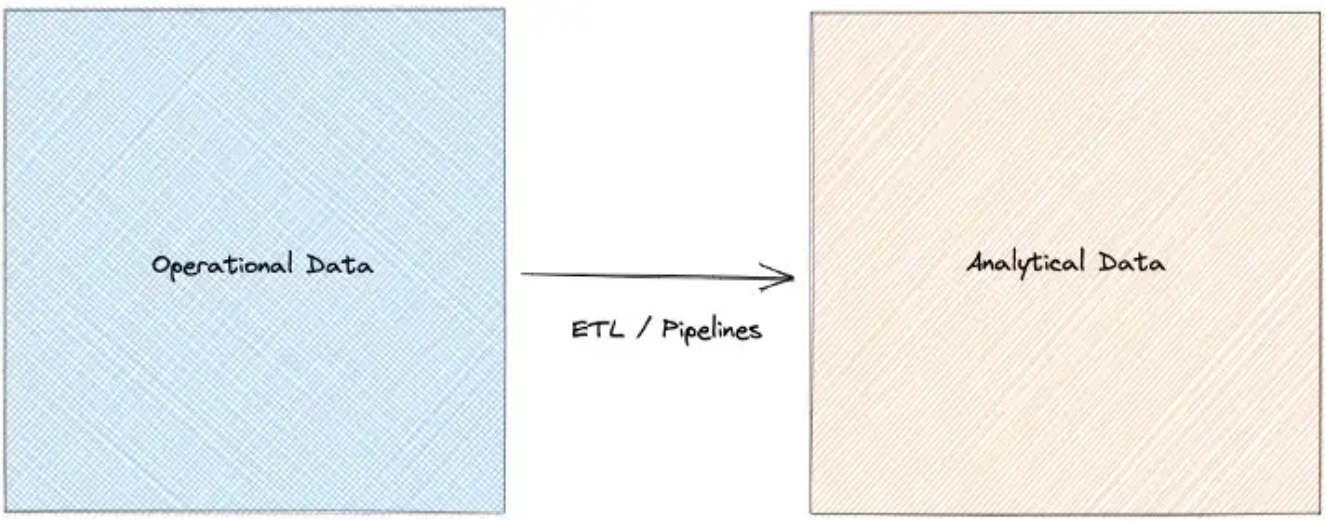
may have been undertaken to implement and embed a centralised data store, also that by making incremental changes we can see and measure how effective those changes are. There is always a risk that too much change simply creates new unforeseen problems that are as bad, if not worse, than the problem we are trying to solve.

It is worth noting that I am primarily considering this from a Cloud technology perspective. I think that the limitations of on premise Data Lakes are well understood, with a heavy administrative overhead and serious limitations on ease of scale; what I would like to challenge is whether the third generation of cloud platforms, as described in the first article, really are so similar to the on premise data lakes, and whether they do already allow the benefits I see of decentralized ownership to be achieved; are they data platforms rather than single data lakes?

In a future article, I will look at when this might not be enough and the switch to decentralised architecture and even infrastructure would absolutely need to be considered.

## Existing Cloud Architectures

If we consider the 30,000 foot view presented by Zhamak, the architecture is presented with all data sources on the left, consolidated into a big data platform, and then served to consumers on the right.

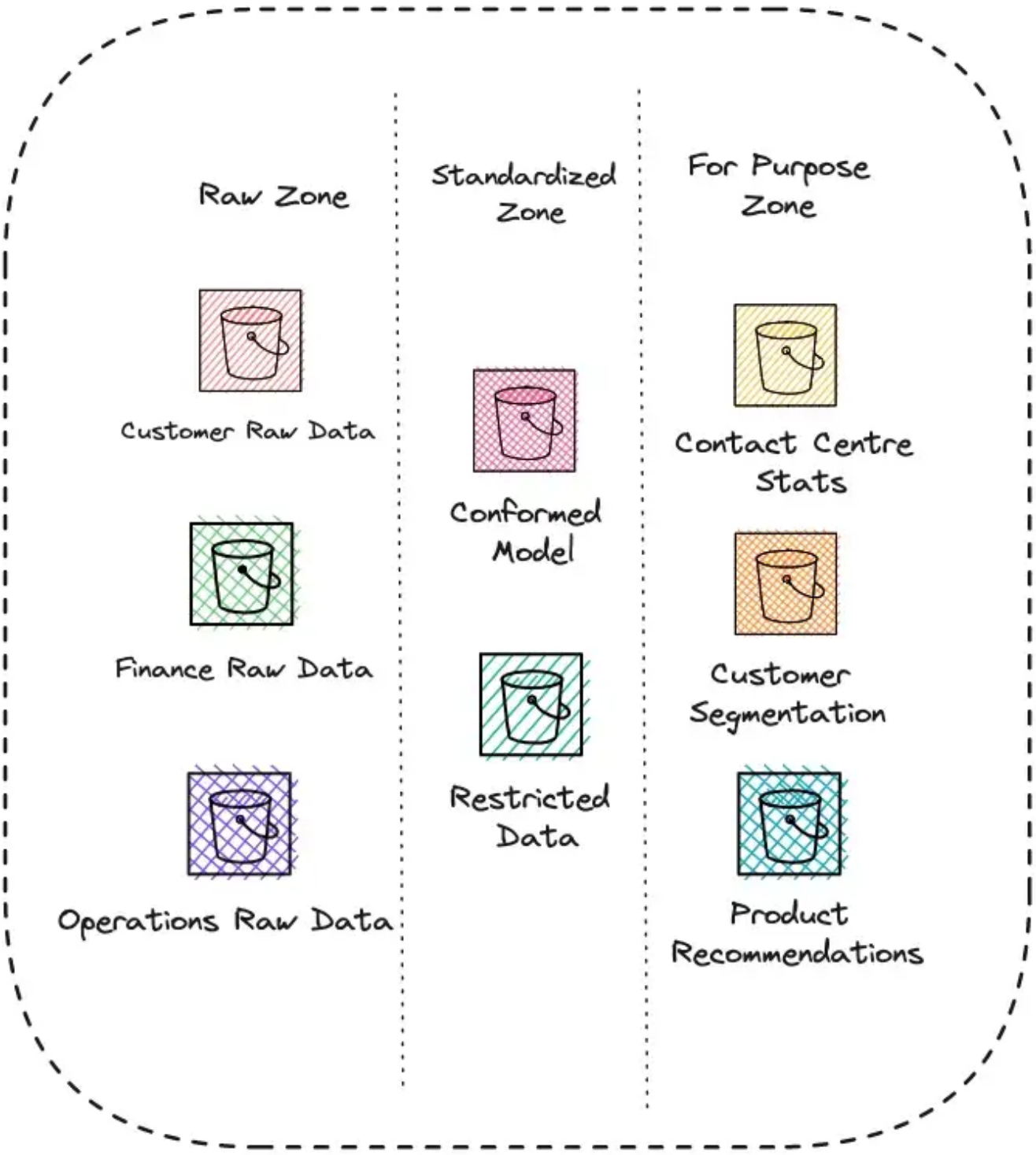



Fragile pipelines linking “operational” and “analytical” data

Firstly, if we consider James Dixon’s coinage of the term Data Lake, it was never envisaged that a Data Lake was a monolith. He even had a rather odd term, Data Water Garden, to describe many Data Lakes interacting, and so there is an

argument to say that the on premise monoliths we see are actually already flawed by the vision of that architectural pattern. Our collective assumptions about Data Lakes are *wrong*, and it is this that is then addressed, quite correctly, by Zhamak as an issue; as data professionals we have not really been following the theory presented by James, and our reality of a Data Lake is somewhat different.

However, if we then consider how a Data Lake looks in a modern Cloud based setting, this actually, and perhaps surprisingly, fits the perspective of *multiple Data Lakes*; they will usually be composed of several storage buckets, all for different purposes and sources, but likely residing on the same platform, such as AWS S3, Azure ADLS or GCP Cloud Storage. This data is also likely to be segregated into zones — data in its raw format, standardised and cleansed data, and then finally for purpose data which may even be in a Data Warehouse rather than the Data Lake.




A cloud based data lake is typically a number of storage buckets organised into zones

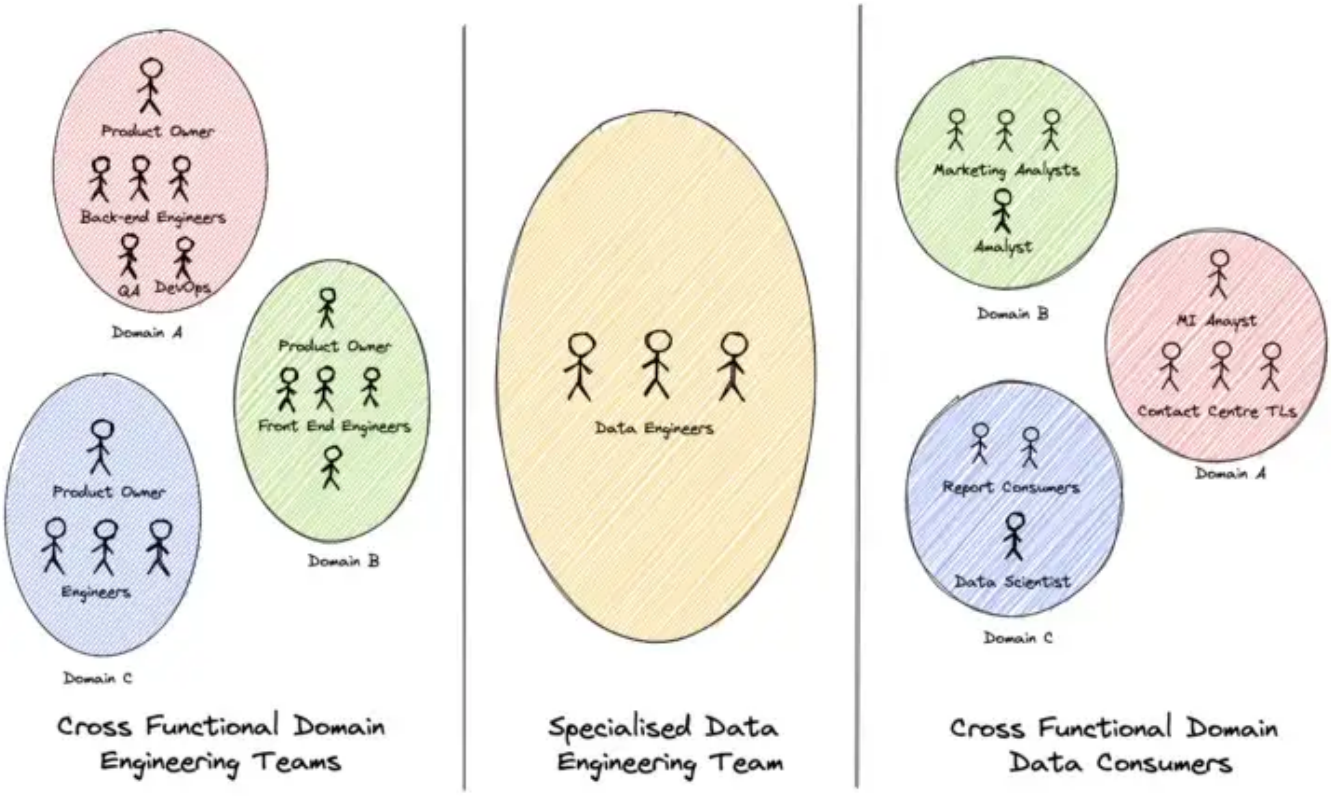
These zones are very much *vertical* zones, as opposed to the *horizontal* domain led zones proposed in the Data Mesh material. It is only if and where they are further subdivided by source and purpose that we truly meet the distributed criteria.

So, from a certain point of view and when talking from a Cloud perspective, the proposed monolith is already showing signs of being a distributed system. In fact, if we compare how this looks to the proposed logical architecture of a multi-plane data platform, it almost entirely fits the bill. Assuming you have embraced infrastructure as code (and you should) and have segregated your data into the

appropriate source and layer accounts and buckets, then the Cloud based architecture pretty much ticks the box of the Data Infrastructure Plane of the Data Mesh.

## The presented problem — ownership of data

The presented problem, and one which I firmly see, is the fragility of and lack of domain knowledge within data pipelines, caused by them being owned by a central Data Team. Note that in the presented architecture, data pipelines still are present; however, they exist *within* the domain as opposed to the central space controlled by the Data Team.




The Data Engineering team sandwiched between multiple providers and consumers

This is where I really see elements of a strawman argument — it is presented by Zhamak that *centralized architecture*, i.e. a monolith, is the problem and we need to decentralize that architecture; but primarily to fix symptoms which all relate purely to *ownership*: bottlenecks caused by a single team supporting multiple domains, issues with the quality of data caused by that team not having the domain knowledge of the creation of the data in order to effectively and accurately transform it, and issues where the producers of that data are not incentivised to provide quality data for downstream processing and use.

A sensible counter here, however, could be that the architecture itself simply presents a barrier to decentralised ownership. If we consider the analogy of families living in separate identical houses versus several families living in a single large shared house, the level of ownership that can be achieved is somewhat different. In the former, our ability to shape and mould our dwelling is much higher, with less need for compromise, than if we were sharing the same building; this comes at a greater risk of divergence from the standards we wish to enforce, however. Perhaps the true goal would be closer to an apartment block, where standardised access controls are in place, access and location is the same, and yet each individual unit can be tailored to the needs of that owner. This is where I see a central data platform fully enabling domain level ownership of their data, whilst ensuring that standardisation and governance is not an unwieldy challenge.

## Product Thinking & Ownership

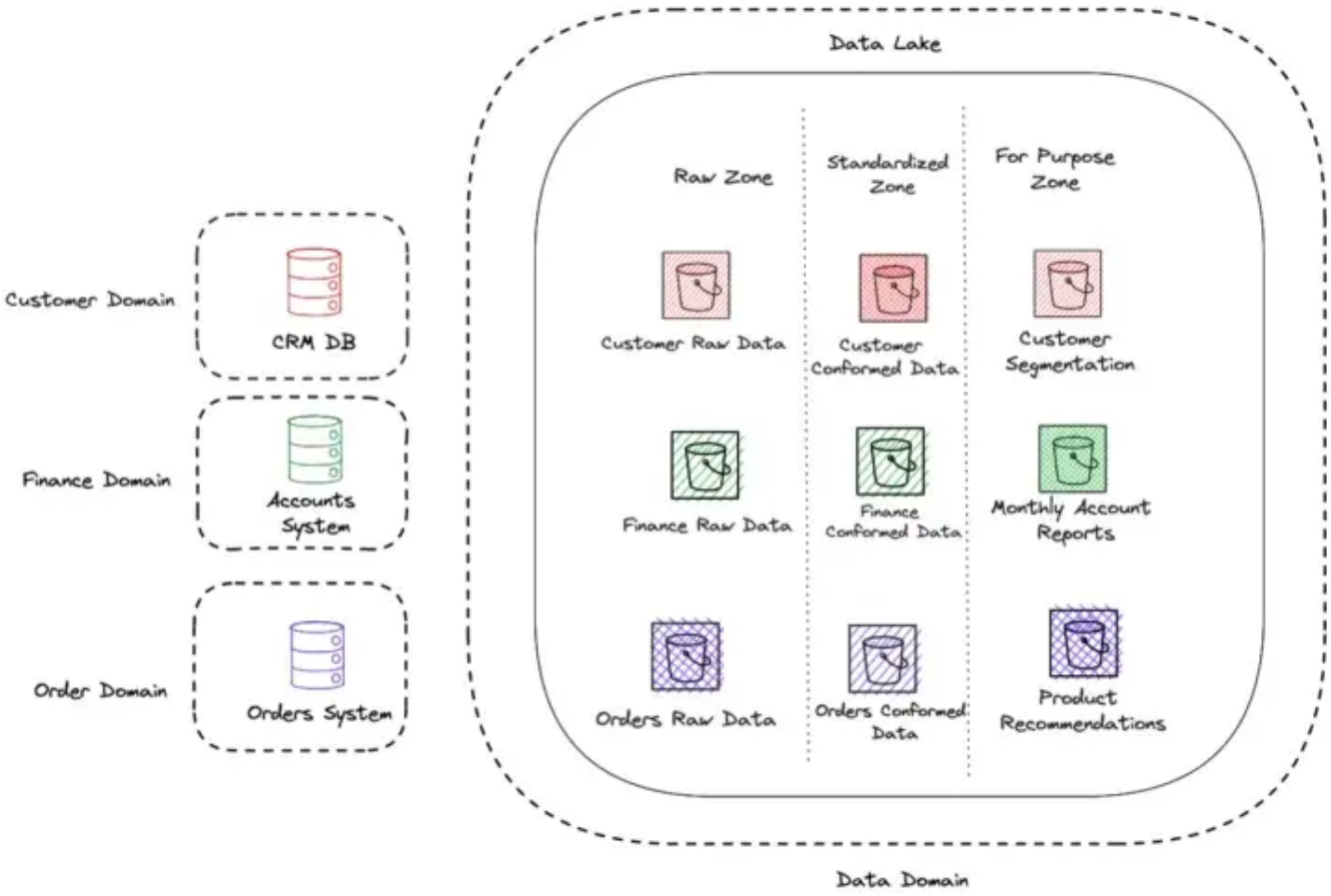
I believe that these problems can be entirely resolved for most organisations without needing to start worrying about the time, cost and effort of rearchitecting the entire data platform along with the challenges and potential pitfalls that may entail; and fortunately, the answer is provided by Zhamak as part of her solution. The proposals to adopt a product thinking mentality for data hits the mark and shifting the ownership of data production to the domain teams who create that data in and of itself feels like a very obvious, yet rarely adopted, solution.

If we take each problem in turn, we can consider how they might be solved *purely by adopting the Data Product thinking*.

## Bottlenecks

Data Team bottlenecks are caused primarily by two issues: multiple providers (the domain teams) and consumers, and the BAU running of the pipelines.

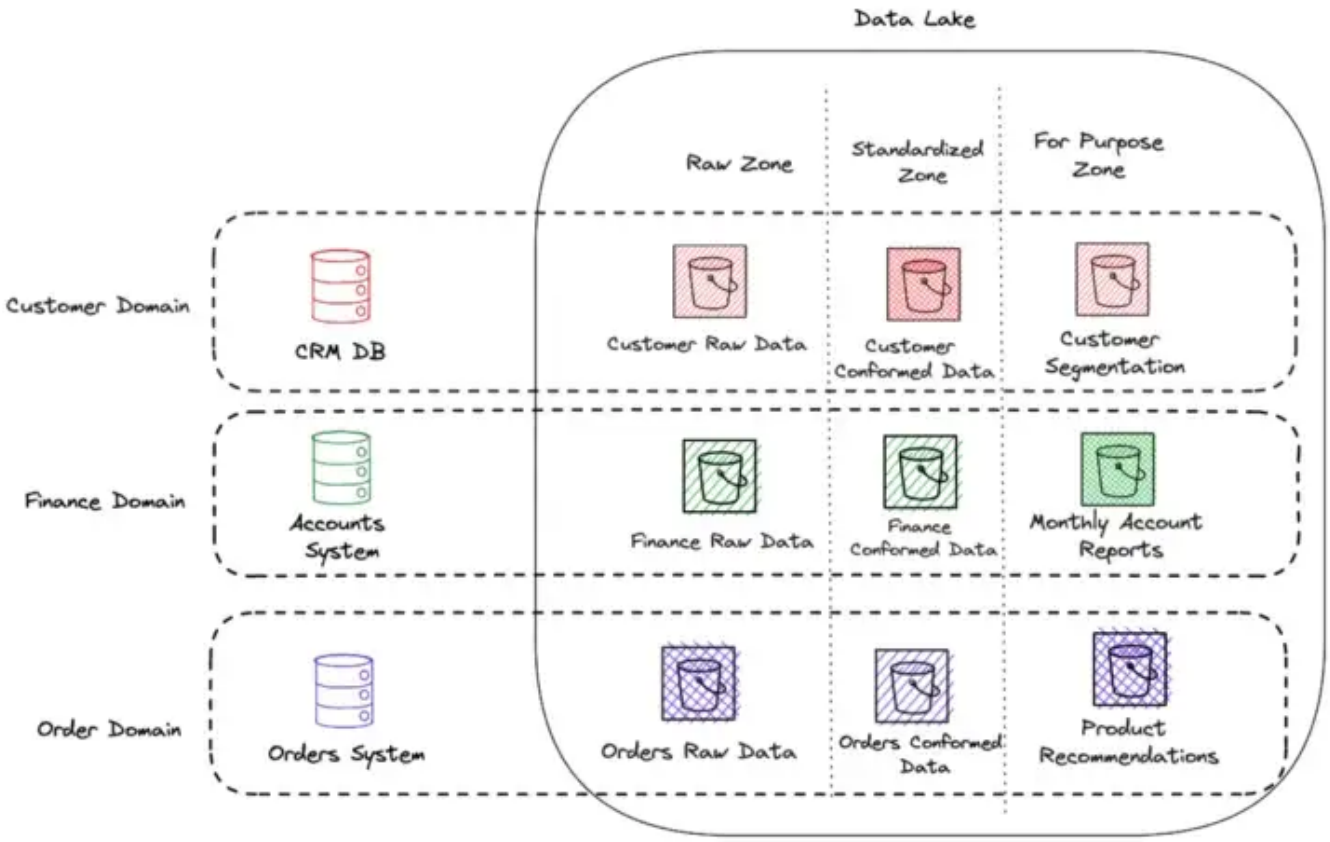
With multiple providers and consumers, prioritisation between those domain units and consumers needs to occur; but this is difficult to achieve when the domain teams are working in an agile manner and pushing their changes to their data producing applications in line with their own drumbeat. If those are breaking changes, they simply need to be resolved *now*, which in turn puts strain on the work the Data Team is undertaking for any of their other providers and consumers.




Product and Data Domain seperation

Likewise, by owning *all* the data pipelines, there is a serious plate spinning exercise required to ensure that everything is up and running on a daily basis, that the pipelines are well massaged to ensure that they are running, and that all downstream dependencies are taken care of.

Simply by moving this ownership to the Domain team, these problems can be alleviated. Changes to the data producer can now have an item added to the Definition of Done that includes ensuring that all pipelines are also functional before that work is completed. Prioritisation of new pipelines are included in the same prioritisation of new product features. Synergy between the product and the data product can be achieved.




Extending the Product Domains into the Data space

## Intimate Domain Knowledge

Domain teams garner intimate domain knowledge through constantly working on their product, and yet we expect a central Data Team to achieve the same for all domains. It seems rather obvious that if a data engineer worked entirely within that same domain, they would therefore be able to build that intimate knowledge, and as a result build better data pipelines. Or even better still, have tooling provided so that engineers within the Domain Team are enabled to build their own data pipelines.

By doing so, the potential for Chinese Whispers is removed, requirements are given directly to the Domain Team by the consumers, instead of through a chain of feedback to a Data Team who need to confer with the Domain Team, interpret the answer and build a solution, which is less likely to meet the original request's needs. Furthermore, by conversing directly with the producer of the data, collaboration, and suggestion as to expanded or altered solutions can be presented. How often do we hear the phrase "give them what they need, not what they ask"? The Domain Team are best placed to be able to make those suggestions.

## Data Quality, SLAs and Contracts

Data Quality is best addressed at source. In order to do this from a data platform perspective, having the Domain Team responsible for the analytical use cases as well as their operational, application based use cases, encourages them to focus on this for all aspects of their data. Once they fully own the data, the five traits of data quality (accuracy, completeness, reliability, relevance and timeliness) are their responsibility within *both* the operational and analytical planes. Coupled with their domain knowledge, they now have the power to ensure that data logic is consistent and accurate within both planes, and the importance of data quality within both planes is now equal.

In order to measure their success, SLAs can be introduced. When is the data is expected to be delivered? This could be real time or a batch process on a given schedule, but should be expressed within the SLA of the data product. Reliability also falls into the SLA space, denoting what happens should something fail, escalation points and so on.

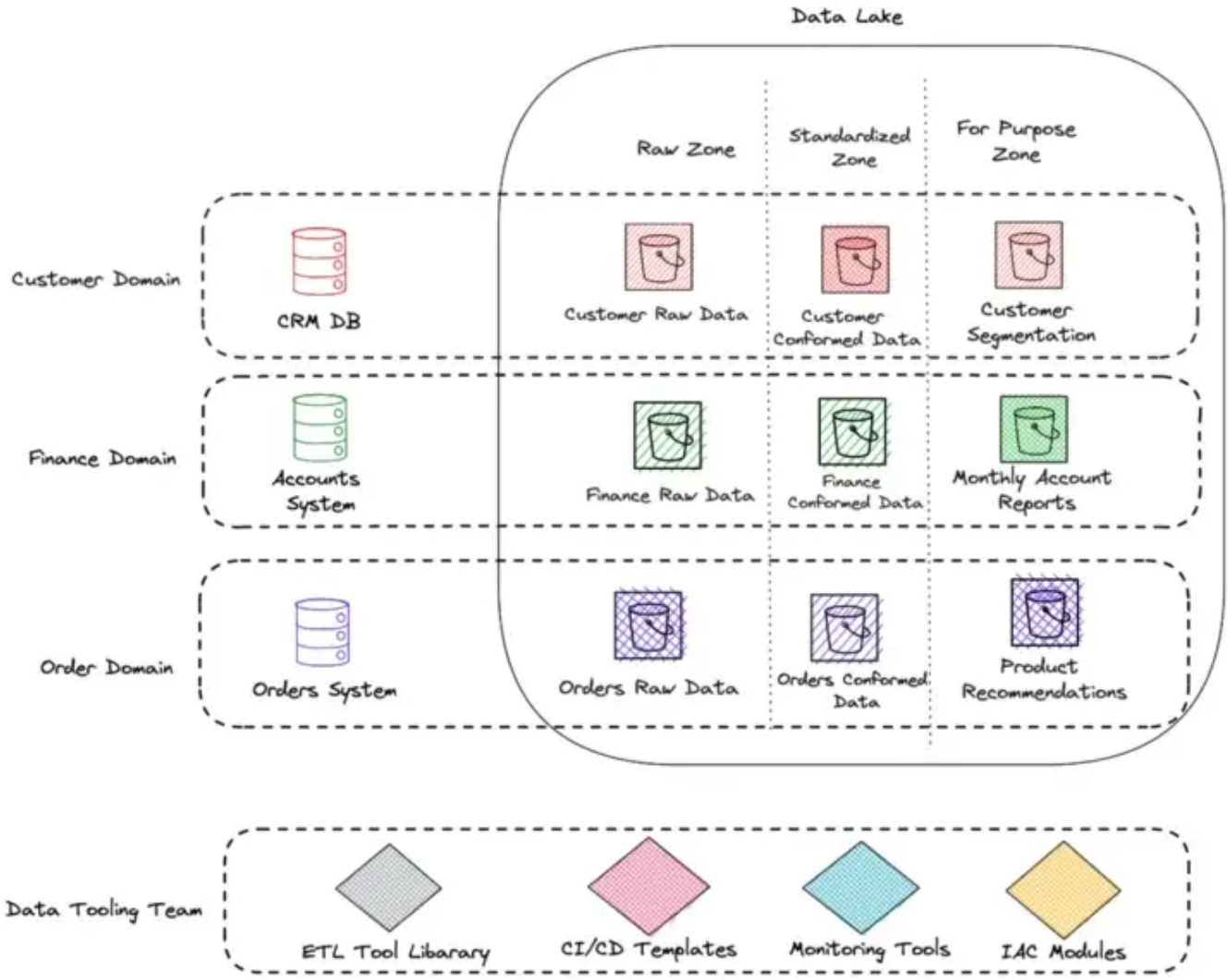
Contracts enforce other elements of data quality. A common, self-describing schema can be created via a Data Virtualisation tool aligned to each version of the data product, and the team can publish lifecycle policies for how long these will be kept in production following future changes. This contract ensures that any downstream consumers know what to expect from the data and when they will need to make a change; gone is the circumstance where an overnight change to a data schema is left un-communicated and so reporting or analytics pipelines break.

What is key to note here is that all of the above can be implemented with existing architecture. There is nothing prohibiting domain ownership of a portion of a database schema, nor from putting contracts and SLAs in place for how and when data will be delivered and supported. They can just be incorporated as part of the data pipeline that the team builds, and communicated out. Ultimately this means that the Domain Team can no longer throw their data issues over the fence to a beleaguered central Data Team; they are now bound to deliver quality data for their consumers.

## Change

The change required then is one purely of organisational change. Similar to the DevOps revolution, the Data Team needs to be freed from its shackles of building and maintaining every domain team's pipelines, into an enabling team that provides standardised infrastructure as code, libraries of functional framework code, CI/CD

templates, monitoring and alerting for the data engineers now within the Domain teams.




Adding the Data Tooling Team to enable the Domain teams in the data space

Alongside this, the production and ownership of the required data pipelines moves to the Domain teams, focusing purely on the data aspect of the pipeline, rather than any of the infrastructure components. The data still adheres to the three zones of a standard data lake, and still sits on a "central" platform; but is owned be different teams. If cross domain data is required, this is simple to deliver from the Standardised layer, ensuring that a single funnel of truth is maintained.

These shifts are by no means small changes, which I believe makes it even more compelling to try prior to rearchitecting a well understood pattern such as the data lake. Operational change is a contentious issue. But it is also possible to begin this change in an agile manner, selecting a single set of pipelines to transition to their respective Domain Team first. By doing so, the required tooling and templates can be built whilst other operations continue as normal. Gradually, over time, more and

more teams will operate in this manner until the central data team no longer exists, and is no longer a bottleneck.

Data Mesh Data Architecture Cloud Data Data Data Analytics