Model Driven Engineering in Finance

Jack Higgs, December 2022

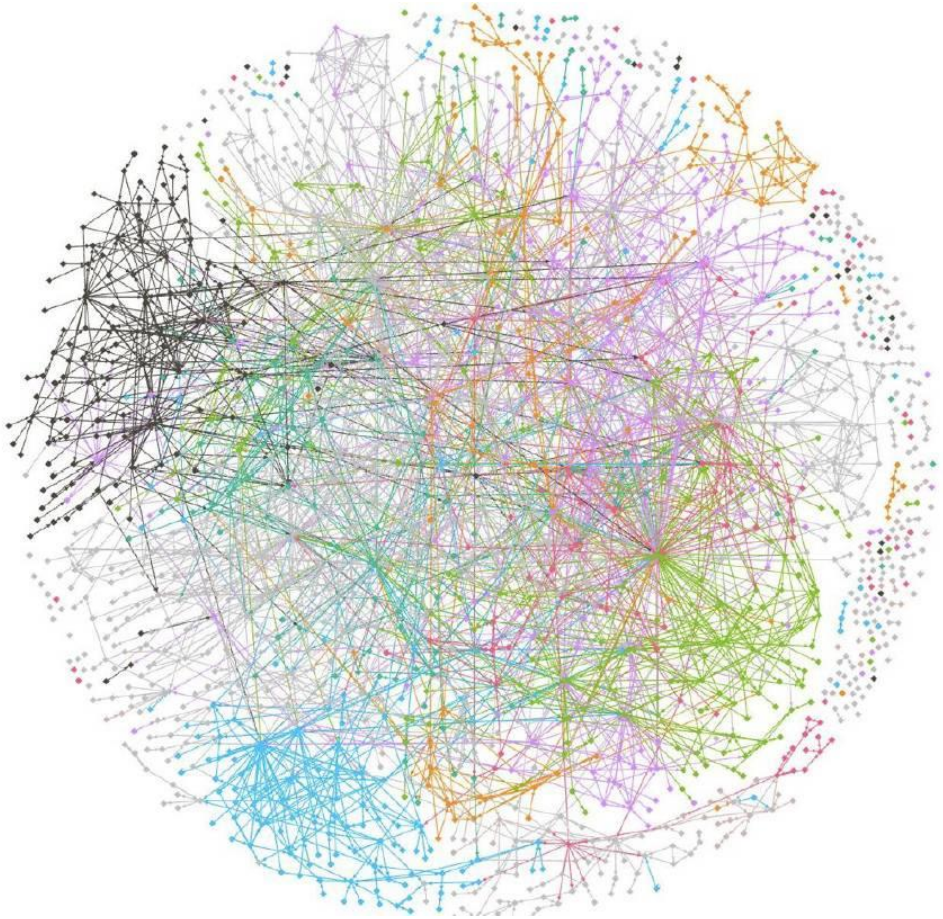
Who am I?

- 5 years at Goldman Sachs
- 7 years at JP Morgan
- Variety of roles across Investment Banks
  - Trading & Sales Tech
  - Finance Tech
  - Regulatory Reporting
  - Core Engineering & Architecture
- 3+ years implementing aspects of Model Driven Engineering using Functional Programming

Agenda

- Setting the scene:
  - Functional/Organisational Complexity
    - Monzo
    - Mortgages at a Retail Bank
  - Regulation
- Applying Model Driven Engineering in Finance
  - Data Lakes
  - Data Mesh
  - Model Driven Engineering and Data Mesh
  - Modelling Languages and Tooling
- Call to actions

Functional Complexity: Monzo

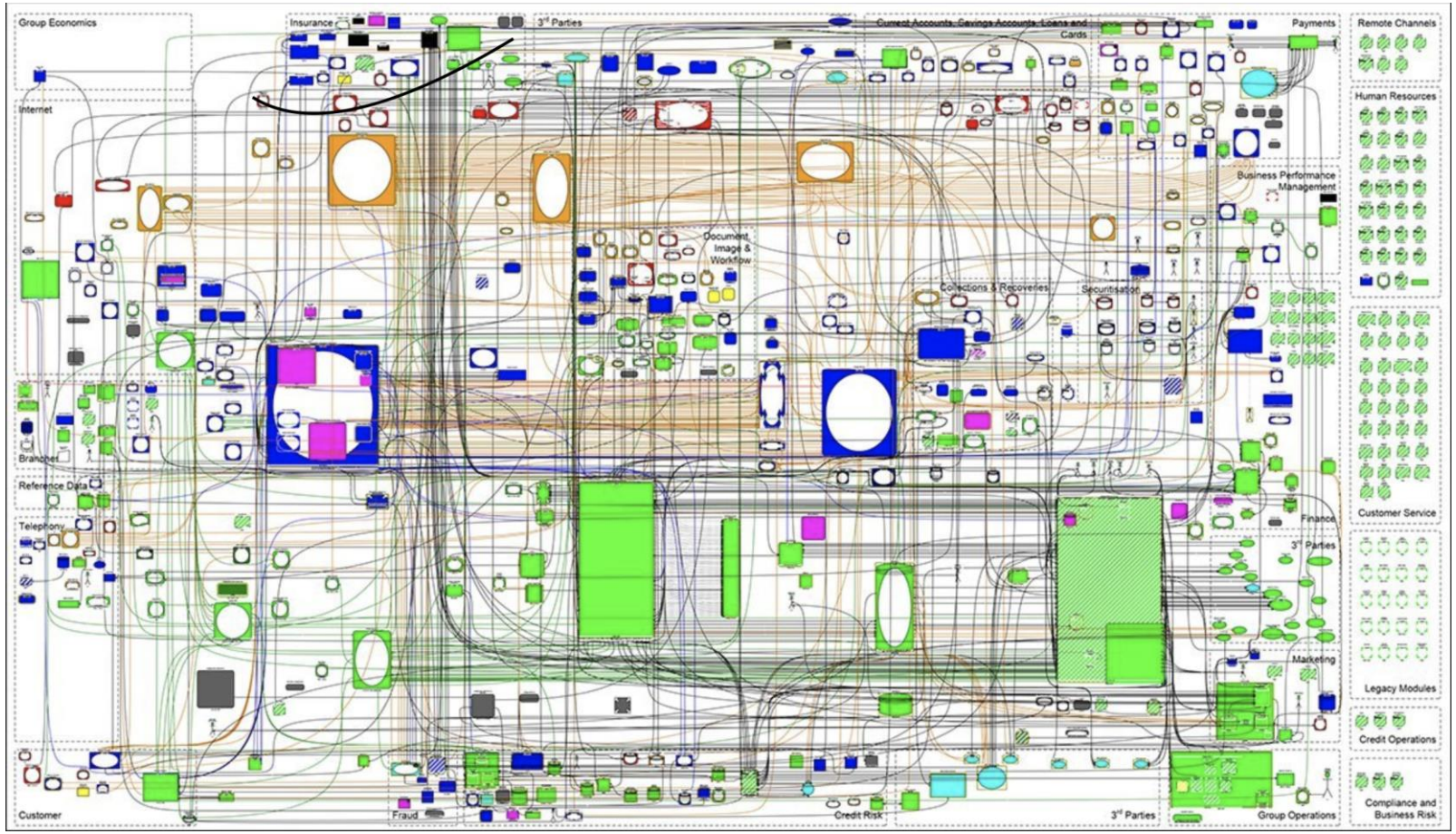


- 1600+ microservices to support 8 products
- Each individual service is simple
- Finance is fiendishly complicated!
- Siloed expertise

"There is a shared core library, which is available in every service… engineers are not rewriting core abstractions like marshalling of data"

Source: https://monzo.com/blog/we-built-network-isolation-for-1-500-services

Functional Complexity: Mortgages at a Retail Bank



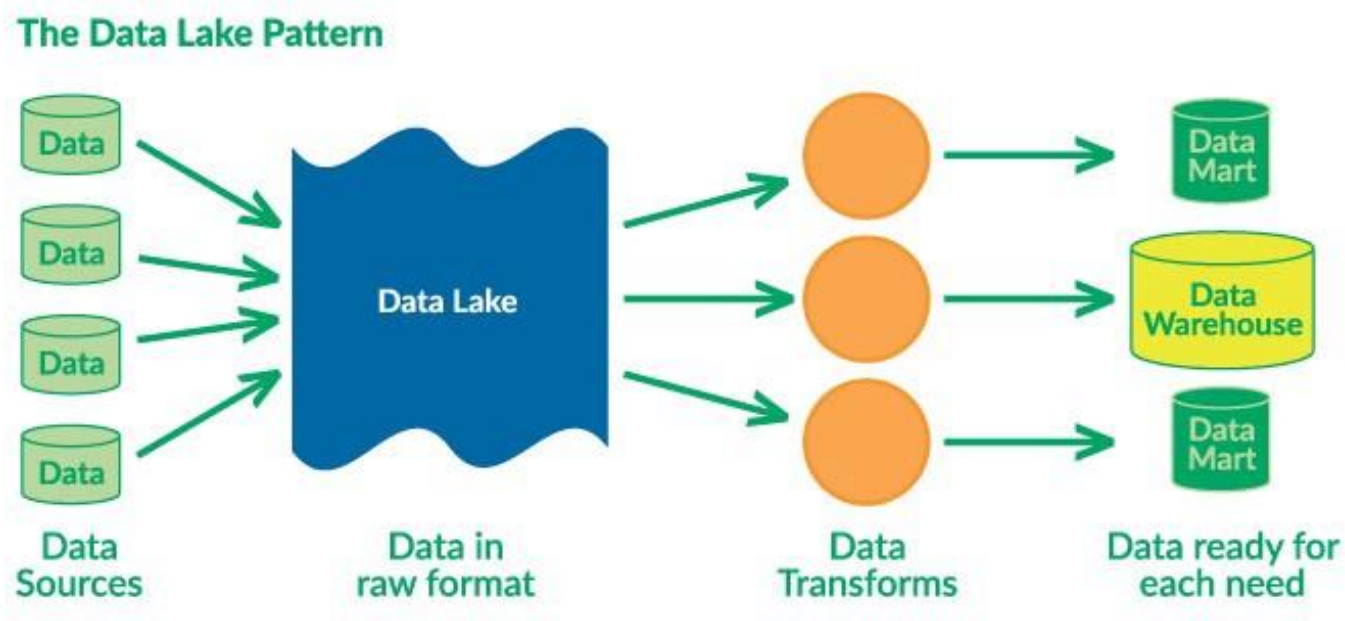
Source: https://www.computerweekly.com/news/4500248392/The-diagram-that-scares-the-next-generation-of-banking-IT-professionals

Regulation

- Data Governance has become something regulators and therefore finance companies are focused on.
  - Basel Committee on Banking Supervision's standard number 239
  - European Union's (EU's) Solvency II Directive
  - ECB Targeted Review of Internal Models (TRIM)
- Response focused on:
  - Data Lineage
  - Data Quality
  - Data Access and Entitlements

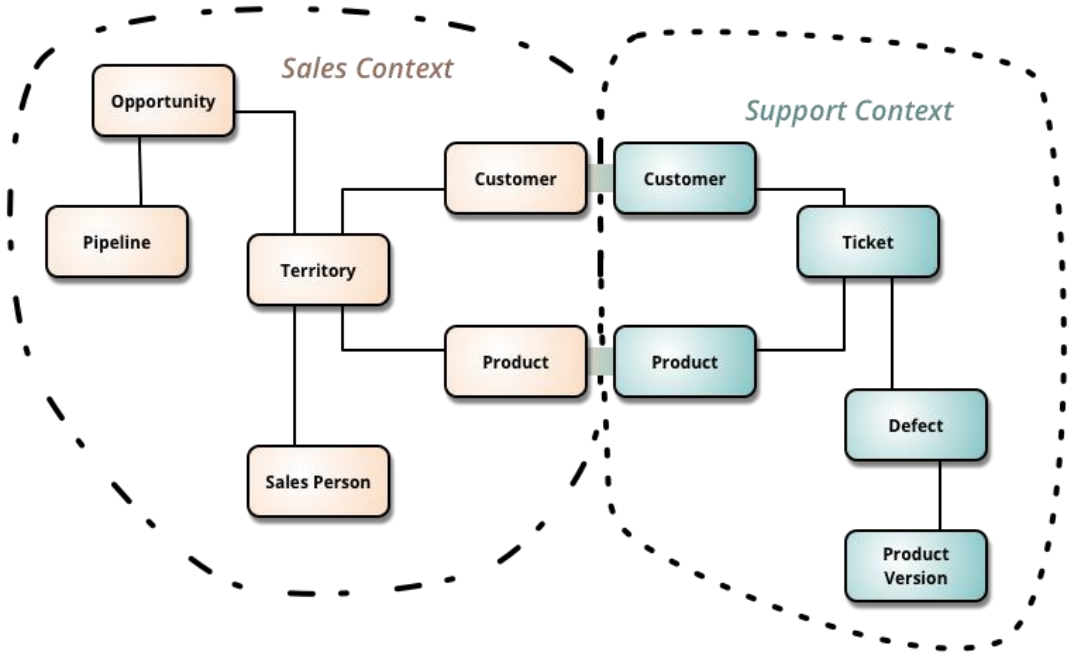
Yet another Data Lake…

- Often a response to data quality gaps, regulatory pressure or frustration on sourcing data.
- Ingest data to a centralized place, map it to common representations, then deliver reports/apps



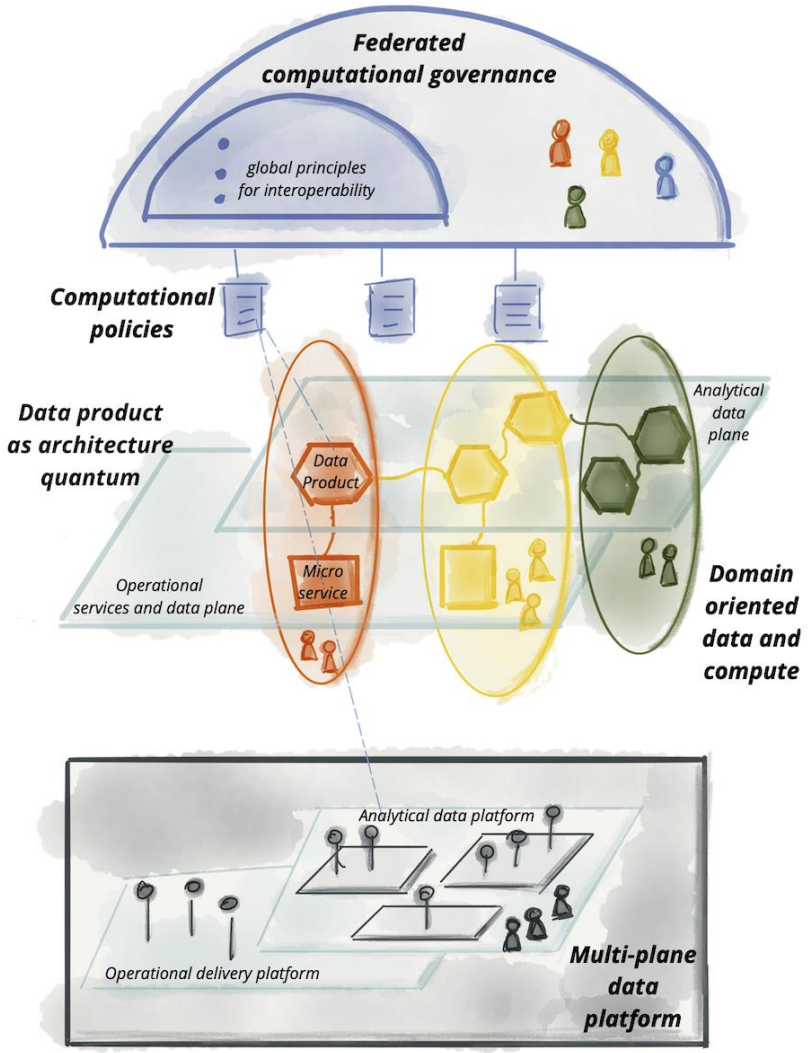
Data Lakes and Bounded Contexts

- Software exists in specific contexts
- Different contexts may have different views of the same domain
- A data concept like “customer” may be ingested into a lake from multiple sources
- We often shoehorn into a unified physical model
  - Loss of design traceability
  - Lack of data lineage
  - Compromised data quality
- Who owns the uniform model?
- Data Lakes do not stop ad-hoc physical transfers of data across siloed organizations.



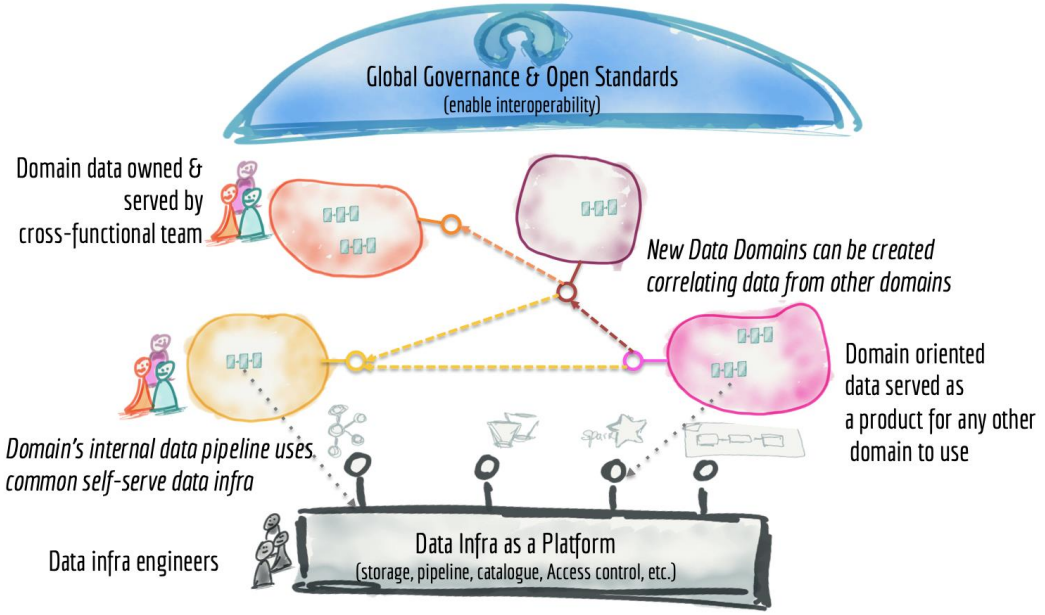
Data Mesh

- Paradigm shift in how organizations share data between systems
- Going from a “push down” to a Data Lake to a “serve and pull” model
- Teams aligned with business domains build data products to serve data in a consistent way
- Shared infrastructure and federated governance



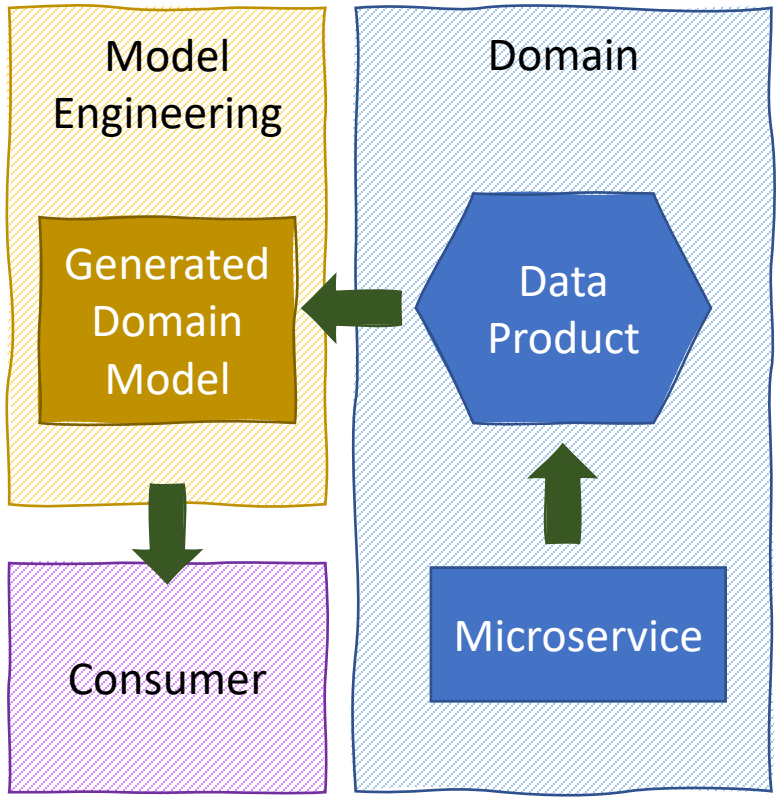
Data Mesh – Issues for Banks

- A wide range of programming languages, databases, middleware. No standardization of physical schemas.
- Compatibility between nodes in the mesh is not addressed in the literature. There is no underlying principle here.
- Culture eats policy. Implementation of a mesh requires a lot of effort (and $$$) and a change in mentality from siloed teams.
- The idea of domain data cross-functional teams is unrealistic. We need easy to use tooling rather than forcing LOB teams to hire Data Engineers.

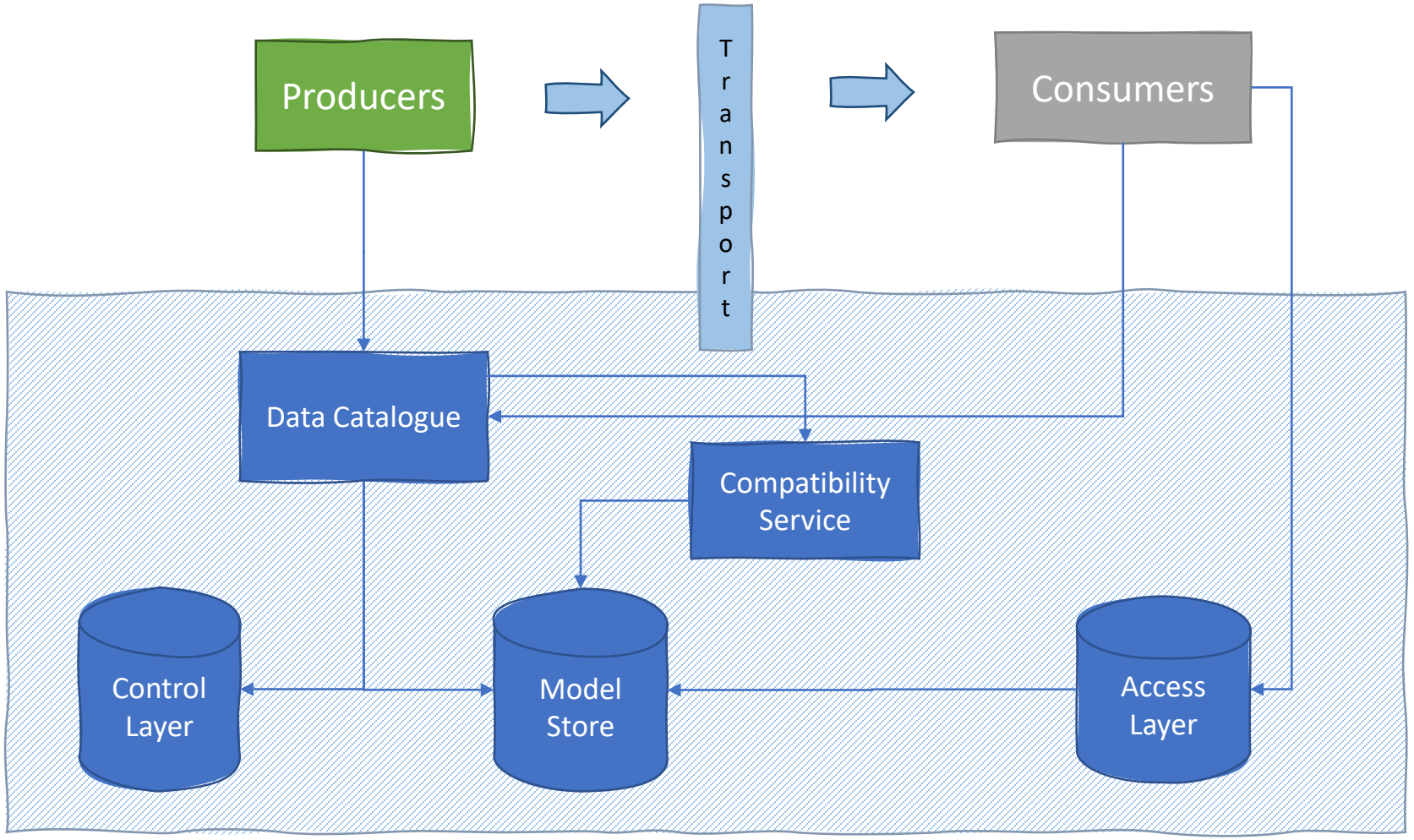


MDE and Data Mesh – Data Products

- Code generate:
  - Logical models and serialization libraries
  - Server / Client Stubs
- Data Products serve instances from code generated libraries
- All data that is published from a Data Product must have a model and be registered in a Data Catalogue
- Access Layer in Shared Infra for enabling row and attribute level entitlements and validations at runtime



MDE and Data Mesh - Infrastructure



Data Infra as a Platform

- Data catalogue to make data products
  - Discoverable
  - Addressable
- Compatibility Service to handle change scenarios
- Control layer stores physical deployment details
- Access Layer for enabling entitlements at runtime

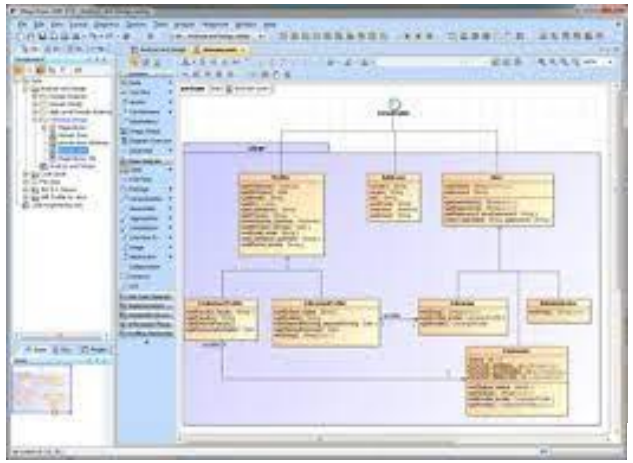
Model Authoring Tooling

awslabs/**smithy**

aws

Smithy is a protocol-agnostic interface definition language and set of tools for generating clients, servers, and documentation for any programming...

59 Contributors · 70 Issues · 1k Stars · 133 Forks



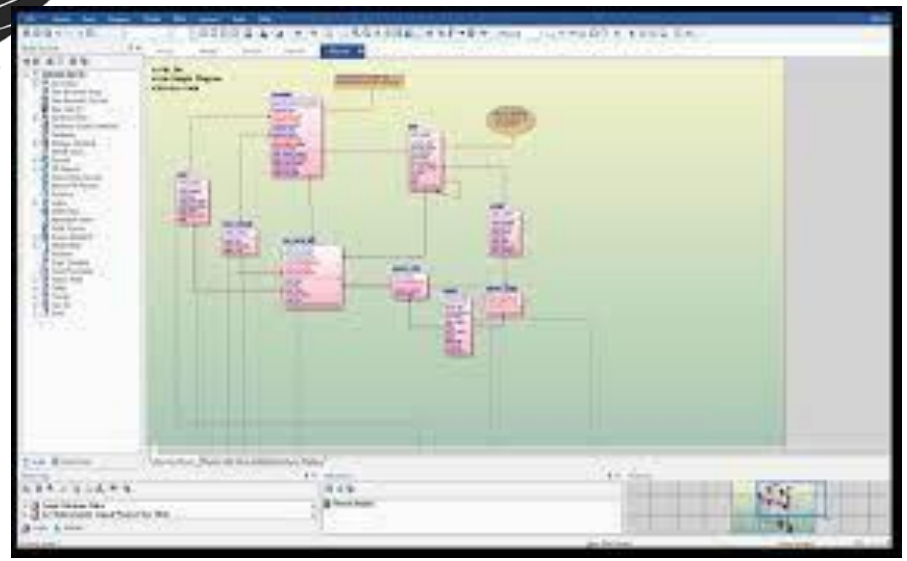
MPS

emf

ECLIPSE MODELING FRAMEWORK

rest.li

GraphQL

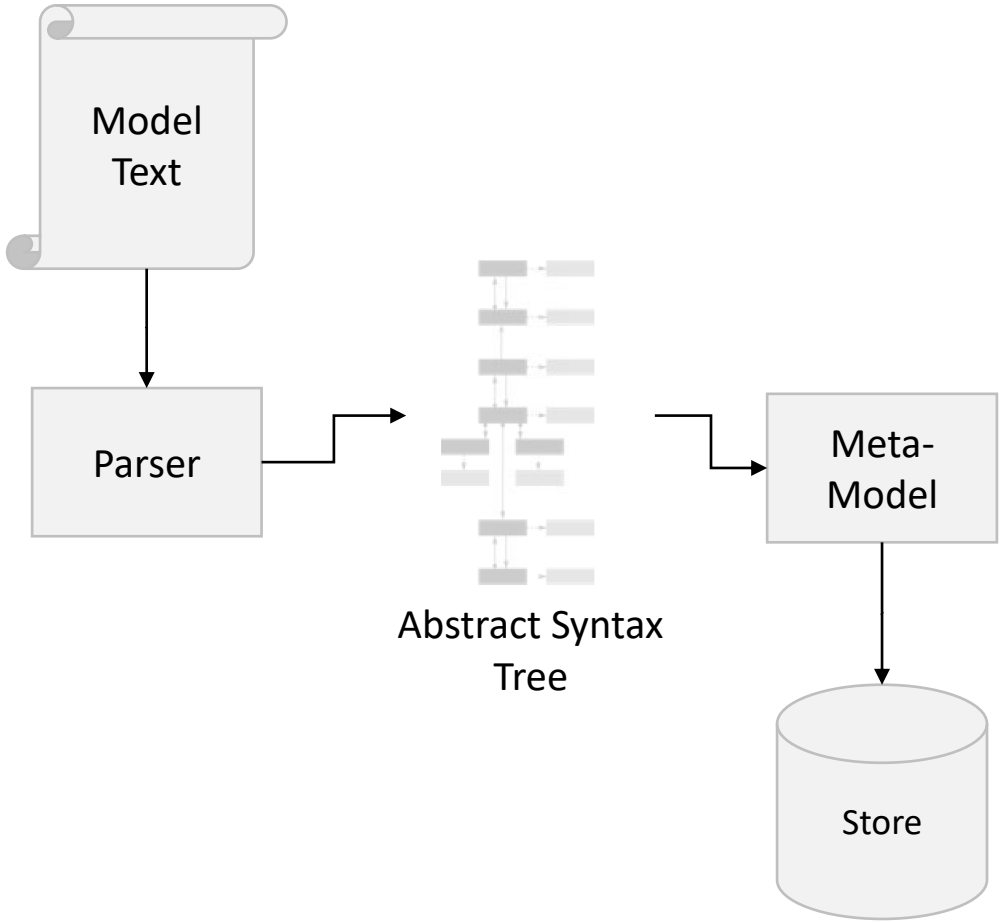


EGEND

ProtoForce

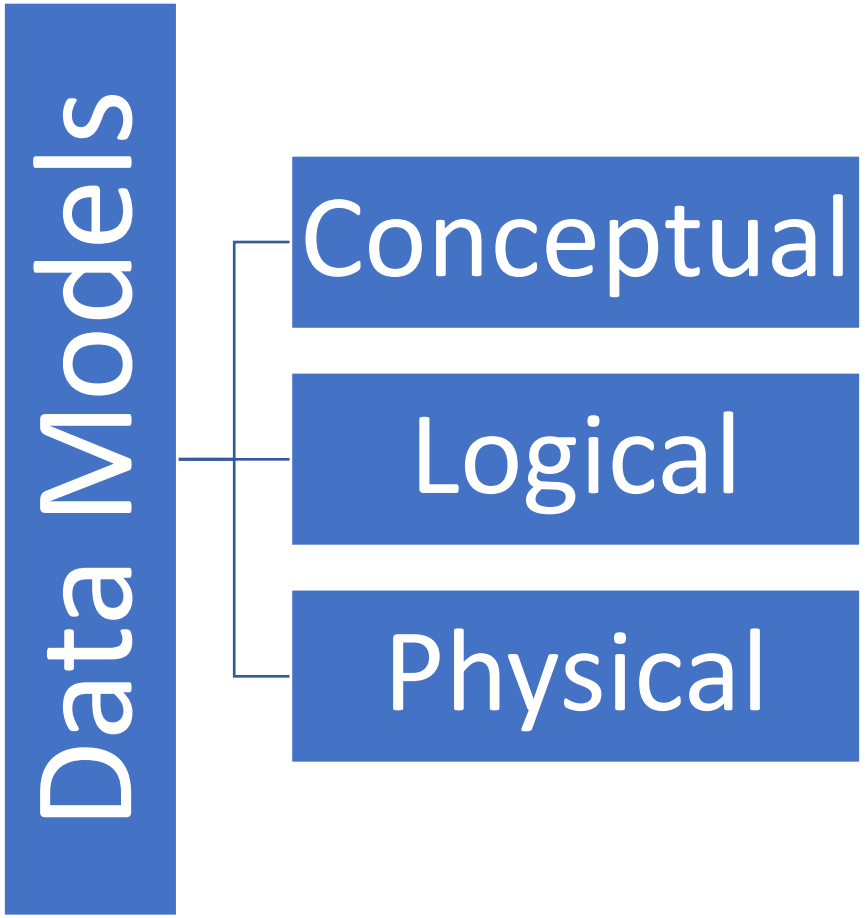
Model Authoring – Data Modelling Languages

```
namespace example.weather

service Weather {
    version: "2006-03-01",
    resources: [City],
    operations: [GetCurrentTime]
}

resource City {
    identifiers: { cityId: CityId },
    read: GetCity,
    list: ListCities,
    resources: [Forecast],
}
```



Obstacles applying MDE

- Developer buy in.
  - Education: A lot of Devs don’t know what a Data Model is or why they should care.
  - Seen as a tax.
- Management buy in.
  - It’s relatively expensive to do Model Driven Engineering.
  - Can’t hire expertise.
  - Tangible return on investment?
- Lack of standardization and open-source solutions.
- Many half-baked data catalogues by data/cloud vendors.



Call to Actions (or tough love)…

- IDEs
  - Build tooling around VSCode/Visual Studio/Intellij ecosystems.
  - Eclipse Modelling Framework gives you a foundation to build interesting things, but the Eclipse ecosystem is being used less and less in industry.
- Research into how MDE fits in with modern data architectures and cloud ecosystems. What could Amazon do with Smithy?
- Languages and authoring tools for capturing model transformations, validations and data entitlement policy.
- Books. ***Please write books aimed at developers***. I can't point my people to anything other than papers or books written in the 90s/00s.
- Standardization and Open-Source.