Architecting a Data Platform for Enterprise Use

Strata New York

September 2019

Mark Madsen
Todd Walter
Why are we talking about architecture?
The market shifted from not enough data in the 90’s to too much data: the problem has become managing not just size, but scope and variety.
More complex needs drive more complex technology
Market hype and IT workforce skill gaps lead to FOMO

The pressure on IT to “just buy something” is high. The usual IT response is based on procurement, not innovation or integration. The data infrastructure solution tends to be: buy tools, collect data.
Development accumulates in a disorderly fashion
The end result in the IT landscape is complexity

Do not underestimate the attraction of complicated technology puzzles
A McKinsey survey this year asked executives if their company had achieved a positive ROI with their big data projects: **7% answered “yes”**

Gartner statement in 2018: **only 15% are reported to be successful**

**Gartner Finding:**

only 17% of Hadoop deployments are in production in 2017

Survey Analysis: BI and Analytics Spending Intentions, 2017
Market and IT complexity requires a change in strategy

IT complexity means we should focus on simplifying.

Diverse but narrow analytic needs mean we must focus on identifying business value and use cases.

Many diverse needs across multiple projects means we must shift emphases from technology to the data ecosystem because more components are required.

- Link shared data and infrastructure across multiple opportunities, to avoid one-off solutions.
- Functional silos may be ok, but data silos are not.
The solution to our problems isn’t technology, it’s architecture.
Use
Over time
Use
Over time
By people
Bricks are not buildings

We don’t think this is equivalent to this.

Architecture is not technology. It’s not a product you can buy.
Blueprints are not architecture.
Architecture is not...

Technical wiring diagrams
Product lists
Pretty pictures
Hand-waving abstract statements and rules
A thing you can purchase

...so what is it?
What is this?
Architecture is an abstraction – a pattern that supports a purpose

You need purpose, therefore focus business goals, outcomes, use cases
Architecture is not Static – it is a Process

Rarely does anyone talk about a core problem: preexisting conditions.

You have something new. How does it affect the old?

▪ Replaces the old?
▪ Adds to what you have?
▪ Overlaps with the old, forcing you to make decisions about what parts to keep, change, throw away?

The heart of this problem is the process of architecture: integrating changes to systems over time. The integration is not purely technical, it’s practices of use, operation, deployment.
HISTORY: HOW DID WE GET HERE?
Where we started: the data warehouse and BI

The data warehouse solved a key problem: access to data from multiple OLTP systems to provide a unified view of key information.

It is built on some assumptions: you must model the data before use, the data must be cleaned first, the data must be in tables.

The DW is built for repeatable data use, not for one-time uses of information or unknown-value datasets.
After a while, user self-service was required.

Eventually we reached maturity with the DW, driving an increasing rate of new data requests by departments and individuals. A backlog of smaller data requests built up around it. We got self-service tools that actually worked.
The real problem is time: business analyses are often needed in a week or less. If the data is not in the DW then the analyst has to wait – the industry average for making data available in a DW is 10 weeks. They need quick access to new data rather than reusable, cleaned, common data. This means they need another way – and self-service tools offer an answer other than “wait.”
Data is rarely used in isolation, DW data is often needed

New data usually needs to be linked to existing data.
Users don’t just need access to data, they need a place to work with and store data too.
Ignore this requirement and you have runaway copies, extracts, and files tied to specific tools, with have no visibility into what is happening.
There are limits to what you can do with queries

Some questions are not answerable with queries. Deeper analysis is required to answer the question. The truth is, and always has been, that tables and a database are not the only technology in the analytic ecosystem.
As BI matured, information needs grew more complex. New analytics, new data, higher volumes drove creation of new techniques and new processing engines. New techniques, new engines, means new structuring and positioning of data is required.
More data, more approaches, technologies arrive monthly.

Analysts
Discovery

Anyone
BI

Data scientists
Data science

Financials

tabular

? (Orange)

SQL

tables

? (Orange)

R

Python

Spark

Tensor
Flow

e.g. Emails, images, more events

? (Orange)

Array / matrix

Time series

Graph

Warehouse events

? (Orange)

customers

sales

inventory

products

? (Orange)

More data, more approaches, technologies arrive monthly.
The end result of years of addition: accidental architecture
Today’s environment has (and still needs) different engines.
Some engines require specific data structures and positioning.

- **Analysts**
  - Discovery
  - Engines: ??
  - Data stored to engine needs: tabular
- **Anyone**
  - BI
  - SQL
  - Engines: SQL
  - Data stored to engine needs: tables
- **Data scientists**
  - Data science
  - Engines: R, Python (Spark), Tensor Flow
  - Data stored to engine needs: Array/matrix, Time series, Graph

Some engines require specific data structures and positioning:**

- Financials
- Customers
- Sales
- Products
- Inventory
- Warehouse events
- Clicks
- e.g. Emails, images, more events
Distributed data is the norm, stored in multiple repository types.

Analysts
Discovery

Anyone
BI

Data scientists
Data science

Raw data

- Financials: RDBMS
- Customers: RDBMS
- Sales: RDBMS
- Inventory: RDBMS
- Products: RDBMS
- Warehouse events: RDBMS
- Clicks: HDFS
- E.g. Emails, images, more events

- Tabular: SQL
- Array / matrix: R
- Time series: Spark
- Graph: Tensor Flow

- SQL tables
- R
- Spark
- Tensor Flow

- RDBMS
- HDFS
How much of the data science problem is tools vs engines?

The dirty secret of data science: more than 80% of the enterprise market does it on laptops. Most does not need specialized platforms, just data.
The arrows in architecture diagrams hide the hard work and make things seem simpler than they are. The arrows are where the integration costs and labor are buried.

What’s missing: loosely integrated data stored for provisioning.
Managing this complexity is a growing challenge

The challenge with use of data science shifts to operations: how to deploy and manage a production environment with so many technologies and copies of data?
You need a layer to separate the mess of raw data below from the distributed, context-specific uses of data above. This minimizes the cost of change, provides user control of data via separation of data management from data use.

Data architecture is needed to fill the gap
Renovating the accidental architecture requires a transition path to separate data infrastructure from the uses of data.

For most organizations, this is your starting point. Not the clean slate of the web companies. It's renovation.
"Always design a thing by considering it in its next larger context - a chair in a room, a room in a house, a house in an environment, an environment in a city plan." – Eliel Saarinen
Think of IT as a city
Where does a building fit?
Design is the art of separation, grouping, abstraction, and hiding. The fulcrum of design decisions is change. Separate those things that change for different reasons. Group together those things that change for the same reason. — Uncle Bob Martin
Key focus for the organization: Infrastructure vs Application

Infrastructure enables value, applications deliver value.

Enable applications by pushing the reusable elements down into the platform.

The infrastructure is a hidden combination of technology, process and methods.
Complexity requires a shift: separate the application from the infrastructure – we are focused on the block, not the buildings.

**Buildings above**: flexibility, repurposing, quicker change above, funded separately

**Applications**

**Utilities below**: stability, reuse, slow predictable change below, funded centrally

**Infrastructure**

We built the DW as a single entity that combined the building and the infrastructure in one complex monolith. The same mistake is applied with data lakes.
The bigger picture of what we create

The *ecosystem* is like the city and all of its extended neighborhoods.

The different applications or types of projects are like the neighborhoods with specific types of buildings.

Each individual building (application) has its own unique *blueprint*.

Underlying all the applications is shared data infrastructure, like city services. The shared infrastructure is the foundation of the analytics architecture for a company.
The IT promise: What you see  

The IT reality: What users see
Who and what are you designing for?
How an organization works
Many applications, many activities
How do you get the full picture?

What does a person do when there’s a problem?
One report from each application isn’t a sustainable answer
The goal is to make decisions, not get reports
This is the “decision support” model

KPIs, metrics

Analyze
Decide
Act
We had batch (and stream) models decades ago
e.g. segmentation, NBO queue, churn, fraud

Usually batch, ran from the DW (but probably not on it),
resulting data loaded into the DW for use

Someone oversees and acts on the information
Applying analytics within a process context

- Analyze
- Decide
- Act

E.g. purchasing changes, upsell/cross-sell recommendations

Machines gain agency, humans lose it; “act” is curtailed

This is the *human-in-the-loop* model
When there’s a problem, the fix is a message

The model’s results should be visible via the KPIs and metrics

Act: People call people to see what’s happening
Somebody built the models – the data scientist

More communication is required

The data scientist needs to observe and change system behaviors
Enter the black boxes – the “autonomous” model
Black boxes still need oversight
Black boxes beget gray boxes because of speed
Three model deployment categories

Autonomous

Human in the loop

Decision support (aka BI) is final arbiter of success.
Three model deployment categories

All of these independent / separate architectures are dependent on some level of shared context.

That means shared operational data, managed over time.
How to design, extreme #1: Focus purely on business need and agile your way to a shantytown
How to design, extreme #2: design everything first - the users will surely follow
Persisting data is not the end of the line.

If you stop here you win the battle and lose the war
“Begin with the end in mind”

The starting point *can’t be* with technology. That’s like starting with bricks when designing a house. You may get lucky but...

The goals and specific uses are the place to start

▪ Use dictates need
▪ Need dictates capabilities
▪ Capabilities are solved with technology

*This is how you avoid spending $2M on a Hadoop and spark cluster in order to serve data to analysts whose primary requirements are met with laptops.*
Data use shifted from center to edge, drove greater speed

1. Making decisions at the edge of the organization drives latency of data: cf “active DW”

2. Change is more rapid at the edge, so new data is needed faster than a classic DW architecture can make it available

3. The next “edge” is embedded analytics, which means machine interfaces, machine data, and machine latency: cf “analytic ops” and “embedded ML”

4. Analytics / data science and self-service / discovery share traits: one key element of which is the need for new data and light integration
The analytics process at a high level

1. Collect Data
2. Do we have the right data?
   - Yes → Build it...
   - No → Research
3. Are we finished or do we need more...?
4. Do we have the infrastructure to analyze the data?
   - Yes → Get results
   - No → Run experiments

Diagram: Kate Matsudaira
The nature of analytics problems is researching the unknown rather than accessing the known.

Diagram: Kate Matsudaira
Where do analysts spend their time? *mostly data work*

% of time spent

| 70% | 30% |

Define the business problem

Translate the problem into an analytic context

Select appropriate data

Learn the data

Create a model set

Fix problems with data

Transform data

Build models

Assess models

Deploy models

Assess results

Source: Michael Berry, Data Miners Inc.
The nature of data science and BI differs

• In BI, the required data is known at the start. One can model the required data in advance.
• Build once, deploy many.
• The same schema can be used by many tools that consume data. The purpose of the schema is to permit easy data reuse via query-generating tools.
The nature of data science and BI differs

- In data science, the data is unknown at the start. The process creates a data model. The same schema may not be reusable.
- Build once, deploy once is the norm.
- The equivalent to a report is not a model. That would be the model’s output. The equivalent to a model is more like ETL.

Somewhere between the sources and the extract is a point of reuse. Standard models, but not fully integrated data.
Market Solution? Build a Data Lake
Build silos on a death star. This is an old pattern.
History: This is how BI was done through the 80s

First there were files and reporting programs. Application files feed through a data processing pipeline to generate an output file. The file is used by a report formatter for print/screen. Files are largely single-purpose use.

Every report is a program written by a developer.
History: This is how BI ended the 80s

The inevitable situation was...

Data pipeline
code
History: This is how we started the 90s

Collect data in a database. Queries replaced a LOT of application code because much was just joins. We learned about "dead code"
BI evolved to hiding query generation for end users

With more regular schema models, in particular dimensional models that didn’t contain cyclic join paths, it was possible to automate SQL generation via semantic mapping layers.

We developed data pipeline building tools (ETL). Query via business terms made BI usable by non-technical people.

Life got much easier…for a while
Pragmatism and Data

Lessons learned during the ad-hoc SQL era of the BI market:

When the technology is awkward for the users, the users will stop trying to use it. Even “simple” schemas weren’t enough for anyone other than analysts and their Brio...

Led to the evolution of metadata-driven SQL-generating BI tools, ETL tools.

One of the reasons the "Big Band Era" ended.
Today’s model: Lake + data engineers, looks familiar...

The Lake with data pipelines to files or Hive tables is exactly the same pattern as the COBOL batch Dataflow code.

We already know that people don’t scale. Don’t do this.
Evolution: 3 stages of maturing data practice

1. Data lake: store all the raw data. A data lake is not entirely worthless, just mostly worthless
   ▪ This was the big data fad, because data in one place is easier than data from source, and the DW has (largely org / process) problems

2. Feature repository: finalized data for use in models
   ▪ Next stop, because we all want to reinvent data marts
   ▪ There is no difference between “feature repo” and “data mart”

3. Standardized data: not conformed like a DW, standardized
   ▪ Data is the primary element of reuse, not code
   ▪ Final stop because eventually you realize a DW was the right conceptual model all along, just misapplied
What do the experts say?

TIDY DATA: Hadley Wickham makes the case for Tidy data sets, that have specific structure, are easy to work with, that free analysts from mundane data manipulation chores – there’s no need to start from scratch and reinvent new methods for data cleaning.

“If your boss asks you, tell them that I said build a unified data warehouse”

Andrew Ng
Leading AI Researcher

Everyone wants shortcuts. There’s aren’t any shortcuts
How did we get to this state with analytics?

There’s a difference between having no past and actively rejecting it.
We started with silos, and they were useful

**OPERATIONS**
- Inventory
- Returns
- Manufacturing
- Supply Chain

**FINANCE**
- Revenue
- Expenses
- Customers

**CUSTOMER CARE**
- Customer
- Products
- Orders
- Case History

**SALES**
- Orders
- Customers
- Products

**MARKETING**
- Customers
- Orders
- Campaign
- History

How many batteries are in inventory by plant? 54
What is the trend of warranty costs? 32
How many people made a warranty claim last week? 29
How many sales have been made quarter to date? 49
Which customers should get a communication on extended warranties? 66
Given the rise in warranty costs, isolate the problem to be a plant, then to a battery lot. Communicate with affected customers, who have not made a warranty claim on batteries, through Marketing and Customer Service channels to recall cars with affected batteries.
Along came BIG DATA
How many visitors did we have to our hybrid cars microsite yesterday?

What are the temperature readings for batteries by Manufacturer?

What is the sentiment towards line of hybrid vehicles?

Which customers likely expressed anger with customer care?

Which ad creative generated the most clicks?
Curation Required

- High, Well Known Quality
- Well Defined Data Model
- Directional Quality
- Curated JSON, XML, DB
- Extracted Attributes
- Unknown/Low quality
Minimum Viable Curation

OPERATIONS
- Out of Deviation Sensor Readings

FINANCE
- Fraud Events

CUSTOMER EXPERIENCE
- Social Media Influencers

MARKETING
- Online Price Quotes

SALES
- Abandoned Carts

Minimum Viable Data Quality

11234

teradata.
Data Comprehension, Pipelines

OPERATIONS
- Out of Deviation Sensor Readings
- Make/model/year selection
- Vehicle/version normalization
- Sensor data formatting
- Unit Normalization

FINANCE
- Fraud Events
- Social Media Influencers
- Online Price Quotes

CUSTOMER EXPERIENCE
- Abandoned Carts

SALES

MARKETING

11234
User Base and Sharing

- Enterprise Wide Use
- Action list, Report, Dashboard Users
- Share across Enterprise
- 10s to 100s of Users
- 10s Users
- Analysts / engineers
- Share in department
- Share over cube wall
Evolving Consumption

- Enterprise production analytics
- Departmental analytics
- Exploratory analytics
- Data Labs
- Repeatable, auditable results
- Ad-Hoc Query, Self Service
Given the rise in warranty costs, isolate the problem to be a plant and the specific lot. Exclude 2/3rd of the batteries from the lot that are fine.

Communicate with affected customers, who have not made a warranty claim, through Marketing and Customer Service channels to recall cars with affected batteries.
To organize and curate data you must understand how it is used. If you don’t understand use then you will not have usable data.
We’re so focused on the light switch that we’re not talking about the light

DATA ARCHITECTURE
The general concept of a separate architecture for BI has been around longer, but this paper by Devlin and Murphy is the first formal data warehouse architecture and definition published.

But 30 years ago we did not expect so many different models of deployment, execution and use.

Analytics for eyeballs and analytics for machines are different
We Need to Support Repeatability \textit{and} Discoverability

Focus is on repeatability
Application cycle time
90% of users \textit{just want answers}

Focus is on discoverability
Analyst cycle time
9% analysts, 1% data scientists
The current needs are beyond lakes and warehouses

We need a new data architecture that is not limiting:

▪ Deals with change more easily and at scale
▪ Does not enforce requirements and models up front
▪ Does not limit the format or structure of data
▪ Assumes the range of data latencies in and out, from streaming to one-time bulk
▪ Allows both reading and writing of data
▪ Makes data linkable, and provides governance
▪ *Does not give up the gains of the last 25 years*
Break down the monolithic architecture – not one building, multiple buildings in a block
Divide the platform architecture to address 3 different goals:

**Collection**
Creation, collection, storage of new data

**Distribution**
Organization and provisioning of data to multiple points of use

**Consumption**
Direct support of data use

Separation of concerns, coordination of process
The goal is to decouple: solve the application and infrastructure problems separately, independently.

Data acquisition should not be directly tied to the needs of consumption. It must operate independently of data use.

Data arrives in many latencies, from real-time to one-time. Acquisition can’t be limited by the management or consumption layers.
The goal is to decouple: solve the application and infrastructure problems separately, independently.

Data management should not be subject to the constraints of a single use.

Data management has historically been blended with both data acquisition and structuring data for client tools. It should be an independent function.
The goal is to decouple: solve the application and infrastructure problems separately, independently.

Data access is already somewhat separate today. Make the separation of different access methods a formal part of the architecture. Don’t force one model.

This separates uses of data from each other, allowing each type of use to structure the data specific to its own requirements.
The full analytic environment subsumes all the functions of a data lake and a data warehouse, and extends them.

Data Acquisition
Collect & Store
Real time
Incremental
Batch
One-time copy

Data Management
Process & Integrate

Data Access
Deliver & Use

Data storage

Platform Services

The platform has to do more than serve queries; it has to be read-write.
The data architecture must align with system components because each of them addresses different data needs.

Data Acquisition
Collect & Store

Real time

Incremental

Batch

One-time copy

Data Management
Process & Integrate

Data Access
Deliver & Use

Separating concerns is part of the mechanism for change isolation.
The design focus is different in each area

**Ingredients**
Goal: available
User needs a recipe in order to make use of the data.

**Pre-mixed**
Goal: discoverable and integrateable
User needs a menu to choose from the data available

**Meals**
Goal: usable
User needs utensils but is given a finished meal
Food supply chain: an analogy for analytic data

Multiple contexts of use, differing quality levels

You need to keep the original because just like baking, you can’t unmake dough once it’s mixed.
Zone 1: Acquisition

- Focus is only on collecting and tracking data
- Direct recording of data from sources
- Each dataset has as much schema as the source provides, could be explicit or implicit or none
- Foundation layer for all subsequent use
Zone 2: Integration - Standardized

- Parsed and cataloged – all structure and form are made explicit so data can be accessed
- Namespace (e.g. keys) managed
- Common elements are standardized
- Datasets are profiled, indexed, available

<table>
<thead>
<tr>
<th>Zone 2</th>
<th>Zone 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="images" alt="Images" /></td>
<td><img src="images" alt="Images" /></td>
</tr>
</tbody>
</table>

Transactions  Events  Objects

Copyright Third Nature, Inc.
Zone 2: Integration - Enhanced

- Common shared KPIs, master datasets
- Extracted and derived data available, e.g. NEE output
- Data is linkable, labeled, possibly cleaned

Zone 1

Zone 2

Transactions, Events, Objects
Zone 3: Access

- Data structured to suit the workload
- Integrated data for a specific purpose
- Business rules applied, e.g. filters, controls, etc.
- Designed, explicit structures
- Generally repeatable use
Zone 4: Transient

- Under business / developer control
- The data for one-off projects of unknown value or repeatability, integrated from other layers
- Place for ephemeral analytics output
- The sandbox
Decoupled data architecture

<table>
<thead>
<tr>
<th>Zone 4</th>
<th>Zone 3</th>
<th>Zone 2</th>
<th>Zone 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transient</td>
<td>Managed</td>
<td>Enhanced</td>
<td>Raw</td>
</tr>
</tbody>
</table>

Zone 2

- Standardized
- Enhanced
- Managed
- Raw

Copyright Third Nature, Inc.
The data is in zones of management, *not* isolating layers.

Relax control to enable self-service while avoiding a mess. Do not constrain access to one zone or to a single tool. Focus on visibility of data use, not control of data.
This data architecture resolves rate of change problems

More effort applied to management, slower.

Optimized for specific uses / workloads. Generally the slowest change.

New data of unknown value, simple requests for new data can land here first, with little work by IT.

Not fast vs slow: fast vs right

Not flexibility vs control: flexibility vs repeatability

Agile for structure change vs agile for questions / use
The concept of a zone is not a physical system. It’s data architecture.

The biggest decision is to separate all data collection from the data integration from consumption.

Physical system/technology overlays are separate, depend on the specific use cases and needs of the organization.
Example: data environment, mid-size retailer

- Web Application
- Replication, batch ETL to collect data
- RDBMS
- Persistent store, data warehouse on same DB in different schemas
- Hadoop
- Discovery work usually done in Hadoop, sometimes in database.

Log file fetch & load for clickstream, summaries sent to reporting env
This data architecture uses the 3 zone pattern:

1. **Raw data is stored in two places:** relational and small sets in DB, log collection in Hadoop.
2. **Hadoop** contains raw data in an immutable storage area.
3. **RDBMS** holds standardized or enhanced data and common or usage-specific data.
4. **Web Application** provides access to raw data.
5. **All reporting data and analytics output** is sent to the DW where it can be accessed.
6. **Cleaned, summarized or derived data** is stored in DB.
7. **Hadoop** is associated with data in an immutable storage area.

Copyright Third Nature, Inc.
Data has to be moved, standardized, tracked. There is a lot of data policy and governance to think about.
Given the rise in warranty costs, isolate the problem to be a plant and the specific lot. Exclude 2/3rd of the batteries from the lot that have no problem. Communicate with affected customers, who have not made a warranty claim, through Marketing and Customer Service channels to recall cars with affected batteries.
Data integration is a portfolio in the new environment.

Data engineering
- Data Acquisition
  - Collect & Store
    - Real time
    - Incremental
    - Batch
    - One-time copy

Data curation
- Data Management
  - Process & Integrate

Data preparation
- Data Access
  - Deliver & Use

Data storage
- Data Processing Platform Services
- Data collection
- Data movement
New terminology for our bad-with-words market

**Data engineering** (technical) – the heavy lifting of integrating data, many to many table mappings, cleaning

**Data preparation** (technical/nontechnical) – working with and across datasets to make them usable or consumable by oneself or others

**Data blending** (nontechnical) – “preparation lite”, joining, simple transformation, single output

**Data curation** (nontechnical) – managing data at the dataset level, we don’t do this at all because we always assumed one data model/schema

**Data movement** (technical) – data isn’t just accessed via SQL, it can be in many different forms.
The missing ingredient from most data projects

Specifically, metadata kept separate from the data.
The problem of “too many” forces new ways to organize a data warehouse. As early as 1255: Since the multitude of books, the shortness of time and the slipperiness of memory do not allow all things which are written to be equally retained in the mind, I decided to reduce in one volume in a compendium and in summary order some flowers selected according to my talents from all the authors I was able to read.

A data warehouse is less like a library than it is like an encyclopedia.
Today’s market solution: the Data Lake to replace the Data Warehouse

Data hoarding is not a data management strategy
The market solution to the ensuing mess?

Buy a catalog!

Just add more technology to solve your non-technical problem.

You know what’s there - how do you find it? How do you get it?
Who maintains the catalog?

IT is already viewed as a bottleneck.

Many organizations do not have full-time data administrators, and the DW team is already overtaxed.
Let the users manage the catalog!

- Data is already too complex for them to understand their BI tool’s single semantic layer.
- Can end users can solve data usability problems by themselves?
- They can probably access the data, but beyond that it becomes a free-for-all of confused meanings and redundant, conflicting integration.
- Unconstrained tags entered by users are not an answer, just create future work.
Practices need to catch up to technologies

A catalog is a useful, necessary component. It is useless without organizing principles and practices. AKA data curation and data architecture
Data curation is not data modeling

The problem with so many sources, types, formats and latencies of data is that it is now impossible to create in advance one model for all of it.

Data modeling is about the inside of a dataset. Curation is about the entire dataset.

It’s about: creating, labeling, organizing, finding, navigating, retiring.

Data curation, rather than data modeling, is becoming the most important data management practice.
Data and technology architecture needs process

Collection
- Capture metadata (including request)
- Record the structure
- Apply keys
- PII masking, restrict
- Start lineage

Distribution
- Common structures
- RDM and MDM
- Subject models
- Make data findable and linkable
- Data provisioning

Consumption
- Model for target use
- Quality rules
- Track provenance
- Apply SLAs
- As few engines as possible – not fewer

Decide on policies for when to place data in which area
Once again we have to look at the context of what we are building.

"Always design a thing by considering it in its next larger context - a chair in a room, a room in a house, a house in an environment, an environment in a city plan." – Eliel Saarinen
Don’t design for development, design for operation

The design point of many things we see is making it easy to build:

▪ Building models
▪ Building pipelines
▪ Building applications

But value comes only after deployment, and must be sustained
Expanding the perspective beyond the initial bit

- There are upstream parts to the development process: collecting and managing data, both for dev and in prod.
- There are downstream parts, in deployment and then in production operation.
- Data and artifacts are exchanged as part of the workflows.
Learning: could be human methods (manual adjustment) or machine methods (e.g. reinforcement learning), which change the sensing and processing.
Feedback requires lots of data that you must record

Data volumes explode with all the telemetry:

1 execution = raw data in, inputs, the action, each metric used (expected values), execution log, metric data (actuals), deltas, model changes, technical resource information
You need to protect against model execution problems

You have to track the actions / executions and their results, including the OEC, in real time, to protect against failures.

This adds monitors and circuit breakers.
“A production ML system is never all green”

Much of the time, the ML app is a distributed system. Distributed systems are hard. Monolithic architectures are great if you can use them.

This is fine. Everything is fine.
ML is not like code: Monitoring in production

Unlike BI, ML has different metrics for “correct”
The metrics are relative and can change over time

You must monitor performance closely, which is like doing BI on your AI.

“observability”, because a problem may not be the model but the data, or the infrastructure.

- Reduce the time to diagnose, rather than emphasizing the prevention of coding errors
You need telemetry about the entire environment for monitoring, but you also need it for diagnostics.

This means you need to think about observability.
Machine learning is the smallest part of the environment


©2018 Teradata
We need a system of record for analytics
There is an extensive list of requirements to support

<table>
<thead>
<tr>
<th>Primary requirements needed by constituents</th>
<th>S</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data catalog and ability to search it for datasets</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Self-service access to curated data</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Self-service access to uncurated (unknown, new) data</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Temporary storage for working with data</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Data integration, cleaning, transformation, preparation tools and environment</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Persistent storage for source data used by production models</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Persistent storage for training, testing, production data used by models</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Storage and management of models</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Deployment, monitoring, decommissioning models</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Lineage, traceability of changes made for data used by models</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Lineage, traceability for model changes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Managing baseline data / metrics for comparing model performance</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Managing ongoing data / metrics for tracking ongoing model performance</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

S = stakeholder, user, D = data scientist, analyst, E = engineer, developer
ML and AI have a lot of requirements: no shortcuts

THE DATA SCIENCE HIERARCHY OF NEEDS

LEARN/OPTIMIZE

AGGREGATE/LABEL

EXPLORE/TRANSFORM

MOVE/STORE

COLLECT

https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007
We need a discipline of AnalyticOps

We need to enable the full end-to-end lifecycle. No product will do this – it’s a workflow, process, and architecture problem.

©2018 Teradata
Reinforcing relationships resist change, despite radical technology and practice shifts.

- **Organization** defines where the work is done and the roles.
- **Technology** defines what work can be done in a given area.
- **Methodology** defines how work is done and what that work is.

Note how only one third is tech.
What about the technology? Do I need an `<X>`?
When replacing the old with the new (or ignoring the new over the old) you always make tradeoffs, and usually you won’t see them for a long time.

Technologies are not perfect replacements for one another. Often not better, only different.
Blended Architectures Are a Requirement, Not an Option

<table>
<thead>
<tr>
<th>Lake + Hub + Warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Premise + Cloud</td>
</tr>
<tr>
<td>RDBMS + S3 + HDFS</td>
</tr>
<tr>
<td>Commercial + Open Source</td>
</tr>
</tbody>
</table>

You can’t just buy one thing platform from one vendor. We aren’t building a death star. Each of the zones is likely to have products specific to that zone’s usage. The uses differ, the people using them differ, shouldn’t the tools should differ too?
Manage your data (or it will manage you)

Data management is where developers are weakest. Modern engineering practices are where data management is weakest.

You need to bridge these groups and practices in the organization if you want to do meaningful work with data. Remember Conway’s Law when you build.
A good design is not the one that correctly predicts the future, it’s one that makes adapting to the future affordable.

— Venkat Subramaniam
Summary

1. It’s not about storing data, it’s about using data
2. Use drives architecture. Understand the uses, what you are designing for, to drive decisions.
3. Put data at the center, not technology. Don’t let the tech define what you can do or how you do it.
4. The death star is not the answer. The data model is not a flat earth. You are not building a monolith.
5. Know your history. Avoiding wheel reinvention saves time, money, careers.
About the Presenter

Mark Madsen is a Fellow at Teradata in the Technology & Innovation Office. Prior to that he was president of Third Nature, a research and consulting firm focused on analytics, data integration and data management. Mark is an award-winning author, architect and CTO whose work has been featured in numerous industry publications. Over the past ten years Mark received awards for his work from the American Productivity & Quality Center, TDWI, and the Smithsonian Institute. He is an international speaker, chairs several conferences, and is on the O’Reilly Strata program committee. For more information or to contact Mark, follow @markmadsen on Twitter.
Todd Walter
CEO - Archimedata

• Focused on off the grid island living, grandchildren and adventure travel.
• Delivers occasional strategy consulting and speaking engagements. A pragmatic visionary, Walter helps business leaders, analysts and technologists better understand all of the astonishing possibilities of big data and analytics.
• Retired from Teradata after a 31 year career working with organizations of all sizes and levels of experience at the leading edge of adopting big data, data warehouse and analytics technologies. Roles included a decade as CTO of Teradata Labs and a decade as Chief Technologist. He holds more than a dozen Teradata patents and was named Teradata Fellow in recognition of his long record of technical innovation and contribution to the company.