# Data Engine Thinking with AI

*LLM-assisted modeling with deterministic delivery*

Roelant Vos
DDVUG - 2026

# A lifetime spent automating data

## History

Consultant, engineer, trainer, software vendor and corporate decision maker - a broad perspective, defined by seeing the same challenges from every side of the table.

The consistent thread: automation, code generation, reusable patterns, model-driven design.

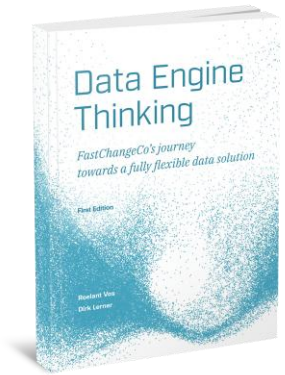


## Nowadays

The approaches and implementation techniques are published as **Data Engine Thinking**, written with Dirk Lerner: an end-to-end methodology for data solutions that are designed to change.

Through **Agnostic Data Labs** they are implemented as free-to-use software: design in the browser, with the metadata and the generated code in your own git repository.

New writing and working examples appear at roelantvos.com and agnosticdatalabs.com.




{ADL}

**This session is about what happens when AI joins the party - where it helps, where it must be kept away, and how to use it for data automation with confidence.**

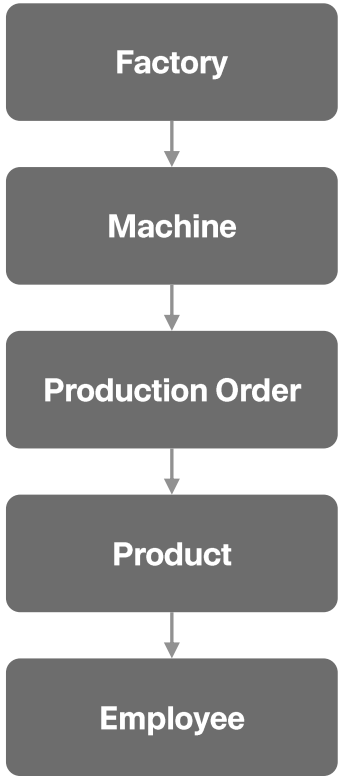
# One prompt...

*“Design a simple 5-object data model for a factory, then propose the data logistics to load it into a data solution, as pseudo-SQL.”*

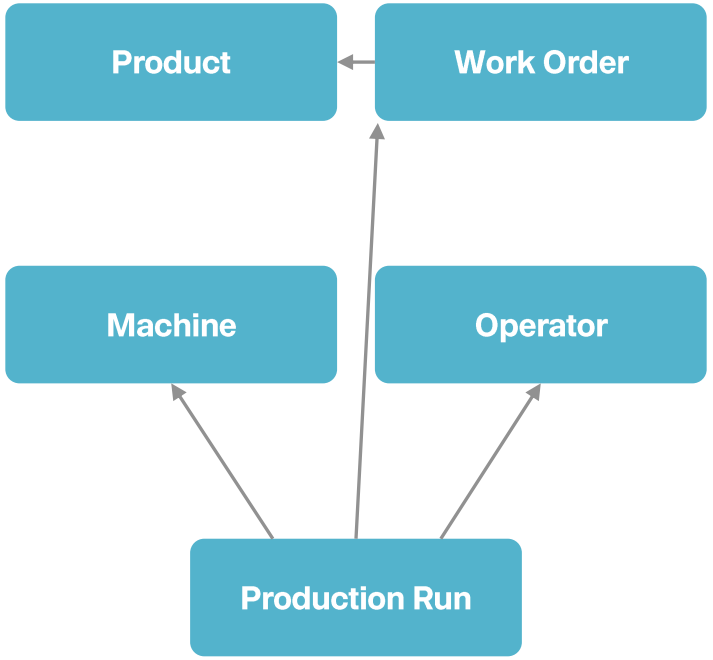
ChatGPT 5.6 Luna · Claude Fable 5 · Qwen 3.7-Plus

# One prompt...

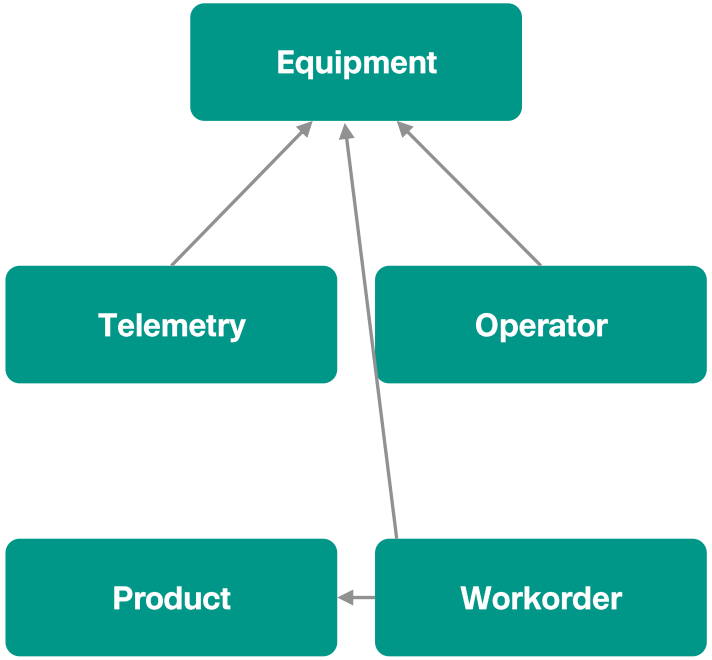
### ChatGPT 5.6 Luna




### Claude Fable 5




### Qwen 3.7-Plus




Data Engine Thinking

# One prompt...

## ChatGPT 5.6 Luna

**Stage**

```
CREATE TABLE STG_FACTORY AS
SELECT * FROM SOURCE_SYSTEM.factory;
```

**Clean**

```
DELETE FROM STG_FACTORY
WHERE factory_id IS NULL;
```

**Merge reference data**

```
MERGE INTO FACTORY target USING STG_FACTORY source
  ON target.factory_id = source.factory_id ...
MERGE INTO MACHINE target USING STG_MACHINE source
  ON target.machine_id = source.machine_id ...
```

**Merge transactions**

```
MERGE INTO PRODUCTION_ORDER target
USING STG_PRODUCTION_ORDER source
  ON target.order_id = source.order_id ...
```

**DQ and serve**

```
ASSERT NO_DUPLICATES(PRODUCTION_ORDER, order_id);
CREATE OR REPLACE VIEW FACTORY_PRODUCTION AS ...;
```

**What stands out?**

- Upserts overwrite in place - yesterday's values are gone
- Deleting in staging, for data quality???
- References before transactions (dependency)

# One prompt...

## Claude Fable 5

**Land**

```
INSERT INTO land.machine
SELECT m.*, CURRENT_TIMESTAMP AS loaded_at,
  'mes_extract' AS record_source
FROM source.machine AS m;
```

**Historize**

```
INSERT INTO psa.machine
SELECT l.*, HASH(machine_type, line, status) AS row_hash
WHERE new OR changed;
```

**Integrate keys**

```
INSERT INTO int.machine_key (machine_number, ...)
SELECT DISTINCT s.machine_number, ...
WHERE NOT EXISTS (...);
```

**Integrate events**

```
INSERT INTO int.production_run
SELECT ... FROM psa.production_run AS r
WHERE NOT EXISTS (...);
```

**Serve**

```
CREATE VIEW serve.machine_current AS
SELECT * FROM psa.machine
QUALIFY ROW_NUMBER() OVER (...) = 1;
```

**What stands out?**

- No DDL
- Insert-only: nothing is ever updated or deleted
- History is maintained. But where does the PSA come from?
- Fun fact: the chat suggests to ‘template this’

# One prompt...

## Qwen 3.7-Plus

**Bronze**

```
COPY INTO raw_equipment
FROM 's3://factory-landing/erp/equipment/*.csv'
FILE_FORMAT = (TYPE = 'CSV' SKIP_HEADER = 1);
```

**Silver**

```
CREATE OR REPLACE TABLE stg_equipment AS
SELECT CAST(equipment_id AS INT), TRIM(name),
  UPPER(status), TO_TIMESTAMP(install_date)
FROM raw_equipment WHERE equipment_id IS NOT NULL;
```

**Gold**

```
CREATE OR REPLACE TABLE dim_equipment AS
SELECT * FROM stg_equipment;
```

**Serve**

```
CREATE MATERIALIZED VIEW mv_daily_production_summary AS
SELECT DATE(fwo.start_time), e.name, SUM(fwo.actual_qty), ...
FROM fact_work_order fwo JOIN dim_equipment e ON ...;
```

**What stands out?**

- S3???
- Medallion
- Full rebuild everywhere (no history, not deterministic)

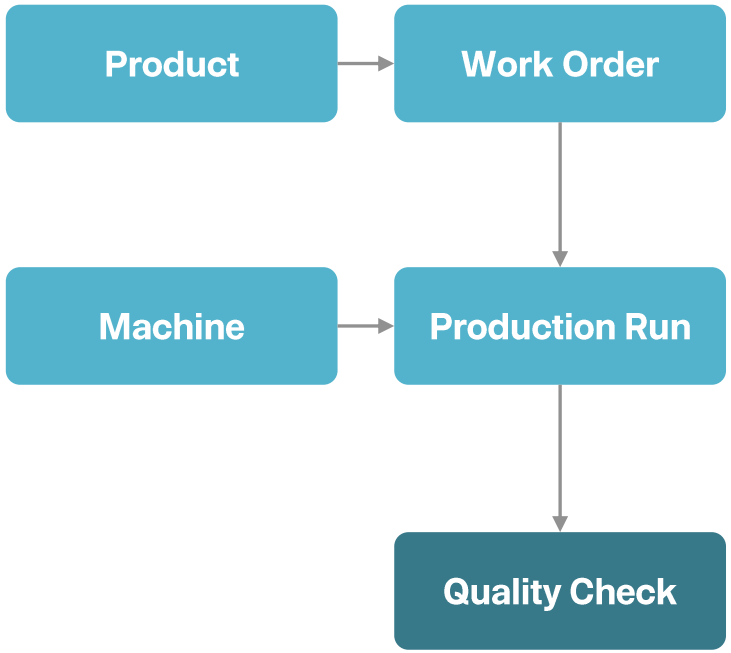
# One prompt...

At this stage, you could argue, it’s time to set some boundaries, add some structure - prompt conventions, house rules, an Obsidian vault providing context (or just ask better questions).

**And we will. But first, a control experiment: same prompt, same provider – but one week later...**

# One week later...

**Claude Fable 5.1 (incognito, no memory)**




*Operator is gone, Quality Check appeared.*

**Some of the things you don't control**

- Sampling - the same prompt is a new draw, every time
- The model - Fable 5 became Fable 5.1 in between, on the provider's schedule, not yours
- Context - incognito, no memory: whatever it 'learned' last time is gone

**And the code?**

- A complete Data Vault - Hubs, Links, Satellites, hashdiff - without any hinting. Haven't seen it since, though...
- A MERGE-based star schema that references columns its own model never drew while still claiming to be idempotent.

**Only the confidence has been constant ^^**

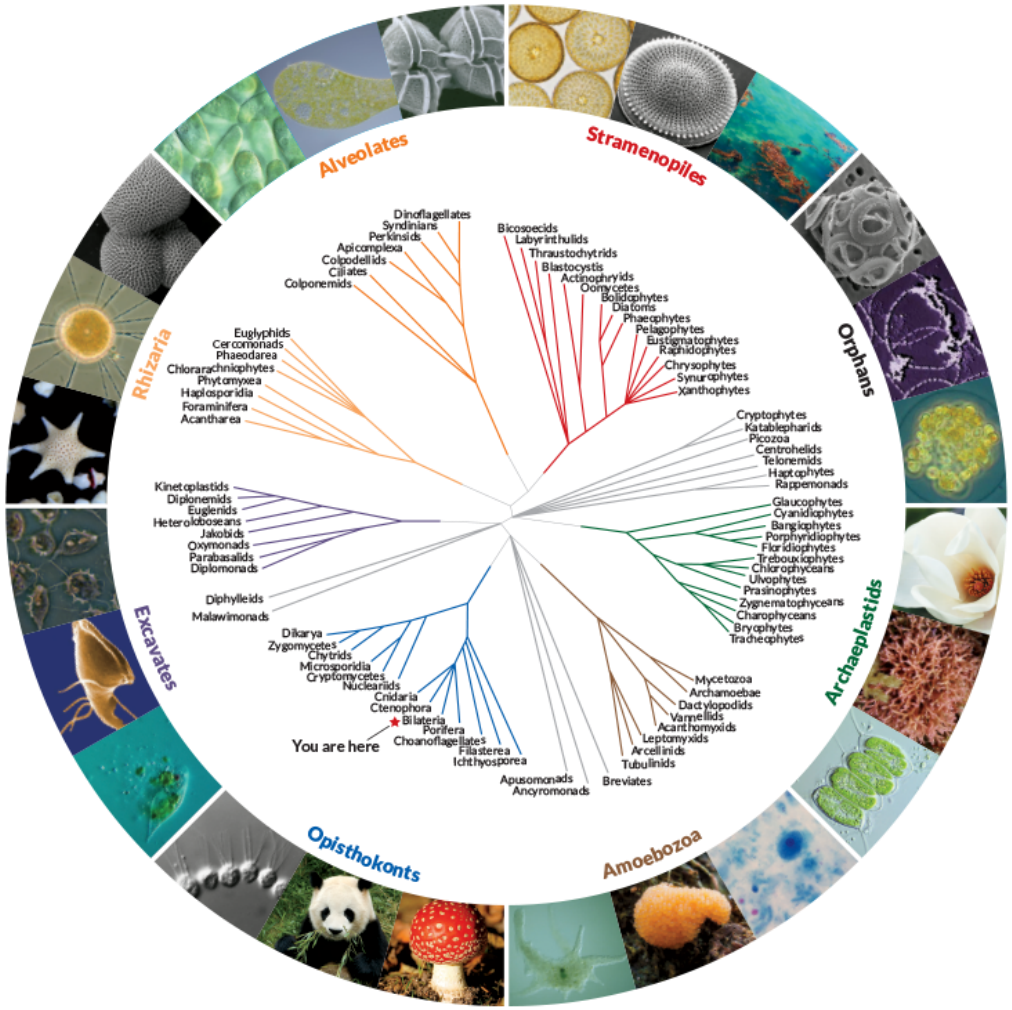
# Today’s session

1. **Data Engine Thinking**
   Capture the design as metadata; generate the implementation from templates
2. **Where AI fits in**
   The deterministic-stochastic border
3. **In practice**
   LLM-assisted modeling in Agnostic Data Labs (with deterministic delivery)

# 1 Data Engine Thinking

Capture the design once as metadata. Generate everything from it.

# The world of data




The tree of life is how the Data Engine Thinking book opens its chapter on the world of data: a landscape richer, older, and more diverse and detailed than any single map of it.

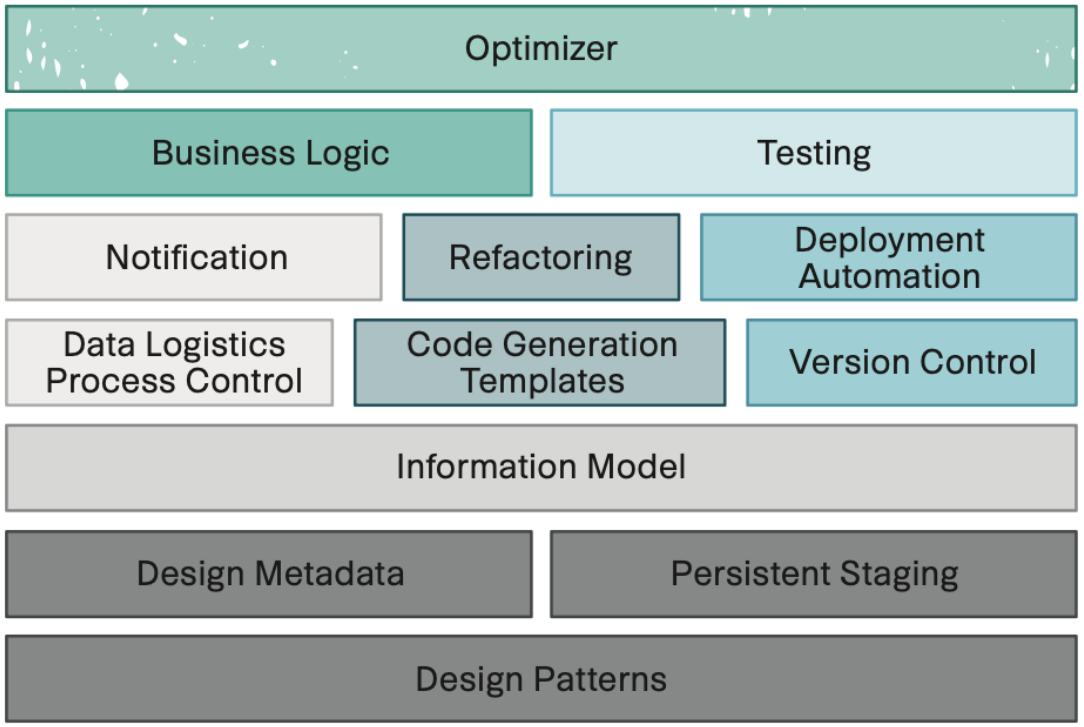
You are somewhere on here.

Understanding a landscape like this is not a project.
**The journey *is* the project.**

# Truly understanding data takes time

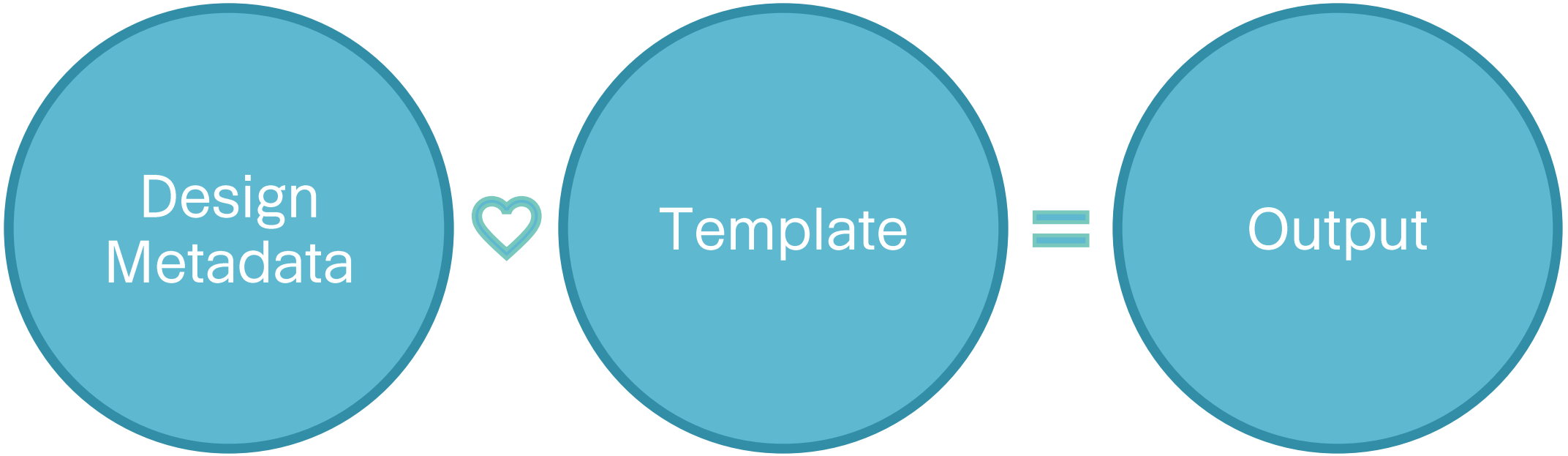
Understanding data is **iterative**: every improvement changes both the design *and* its implementation - the physical model, the mappings, the data logistics.

**Data Engine Thinking treats this as the normal condition, and designs for it.**

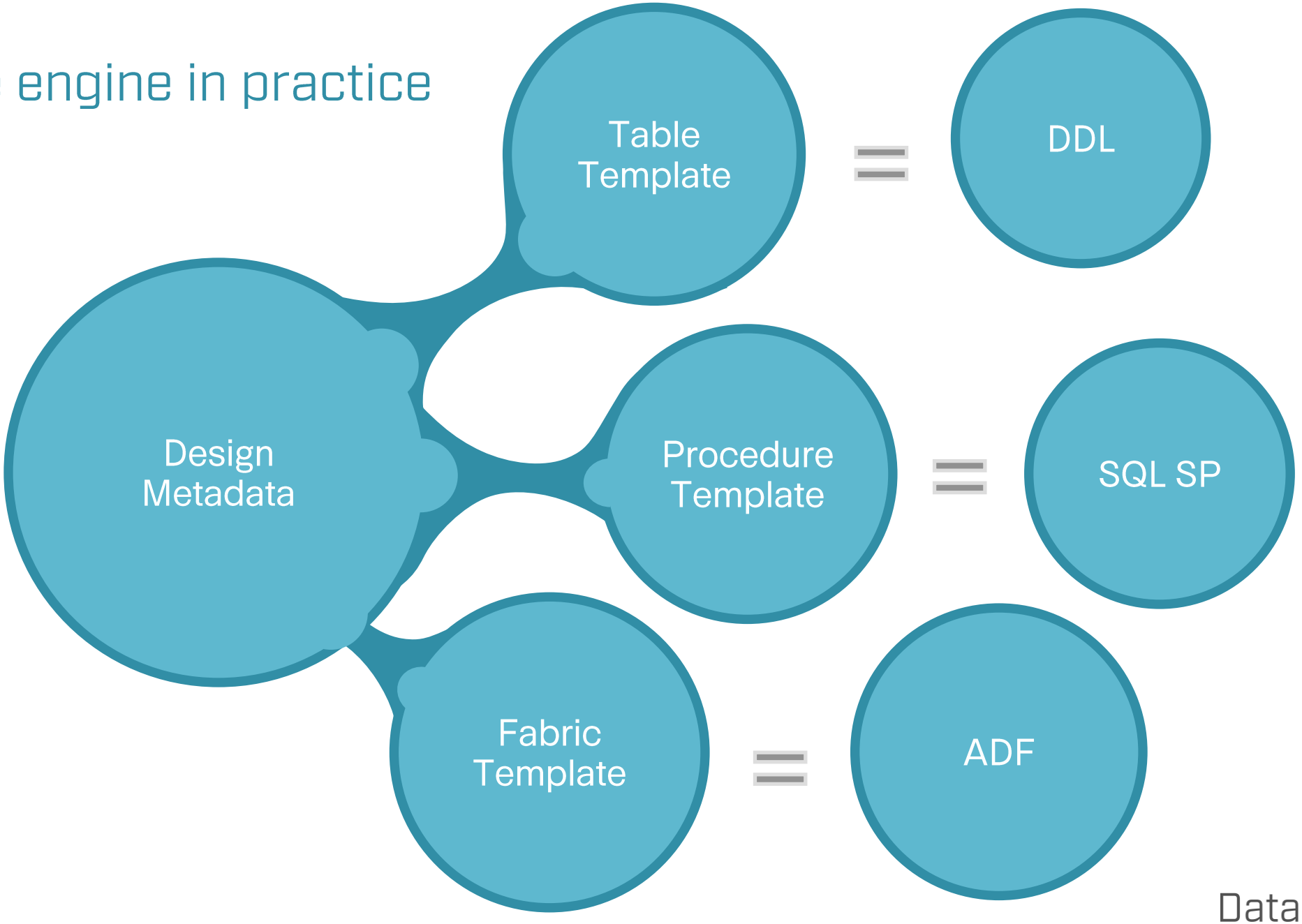



The engine, as the book describes it: patterns and metadata at the base, generation and delivery above.

# The engine in practice



# The engine in practice



# Deterministic templates: same design in, same code out

Quick demo – *one* of the
ways this can work

# 2 Where does AI fit?

Generating code is easy these days - so how far should it go?

# Same job, two code generators: a template and an LLM

Both use metadata to produce code, and on a good day the output looks identical.
The difference lies in the reliability **every** day.

| A Large Language Model | A code generation template |
| --- | --- |
| **Stochastic: sampled, not computed** | **Deterministic: computed, not sampled** |
| Ask the same question twice, get *might* get two answers | Same design in, same code out |
| Sometimes an answer is confidently wrong | Testable, diffable, reviewable |
| Cannot be reliably unit-tested | Wrong the same way everywhere, so a fix applies everywhere |
| Output needs a reviewer before it can be trusted | Runs unattended (‘at three in the morning’) |
| **A brilliant assistant, but an unaccountable executor** | **A predictable executor** |

# Two ways of working with data (but only one tolerates approximation)

**Code is cheap now**

AI generates good-looking code ‘in a jiffy’, making it tempting to let it do everything we covered so far. Whether that is wise depends on what kind of automation is required.

| Interpretation | Integration |
| --- | --- |
| Analysis, exploration, drafting, classification, summarization. Approximate answers can be useful: a good suggestion saves time where a ballpark figure might be sufficient.<br><br>This is where LLMs really work - and where a review step is already a natural part of the workflow. | Moving and transforming data: loading, historizing, combining, restructuring. Here the requirement is exactness - identical inputs must yield identical outputs every time.<br><br>A data logistics process that is sometimes wrong can fail in ways that cannot be reproduced afterwards.<br><br>Auditors and regulators do not accept 'the model usually gets it right’ (neither should you). |

**Integration requires determinism. Interpretation benefits from assistance.**

# Nine principles every data logistics process must support

From the book: the non-negotiables of data logistics.




The three in green are where a stochastic component violates the principles

**Where stochastic components break stuff**

Idempotent means a process can run again without changing the outcome; deterministic means identical inputs always produce identical outputs; auditable means any result can be traced and reproduced after the fact.

A language model inside data logistics violates all three at once: the second run may differ from the first, the same input may yield different output, and yesterday's result cannot be regenerated to prove what happened. No amount of prompt engineering provides guarantees - only removing the stochastic step does.

**The conclusion is not 'no AI' for your data solution, but that nothing stochastic should touch the actual data. AI works on the design; templates do the delivery.**

# Review hell: one template, or a few thousand drafts?

Imagine the language model generates all the data logistics code - typically, a few thousand processes.
Review effort scales with the number of things that can be **independently** wrong.

| Loading code from a language model | Loading code from a template |
| --- | --- |
| **Thousands of drafts, each one its own sample** | **One template per pattern, plus structured metadata** |
| Fix a bug in one draft, and it is fixed only there | Fix the template, regenerate: fixed everywhere |
| Regeneration re-rolls everything: the diff is mostly noise | A design change shows up in the diff means exactly that |
| Conventions drift, across many objects | Health rules check the metadata, tests check the template |
| After a certain number of plausible merges, the reviewer's attention fades | Review scales with patterns, not with objects |
| **An AI reviewer is a second sampler, not a guarantee** | **Reviewed once, run thousands of times** |

**So, can AI write all the loading code?**

Yes - once, as the template. Let it draft the generator, not the outcomes.

# Where AI shines: design metadata

These are metadata draft-and-review tasks. None of them touch the actual data.
A person reviews every proposal before committing the change, making it part of the design.

| What the AI can do | Example | What makes this OK? |
|---|---|---|
| **1 Drafts definitions & docs** | A description for every table and column of a new source. | A wrong draft only costs a correction |
| **2 Suggests classifications & keys** | 'This column looks like a business key' | Suggestions land in the design |
| **3 Proposes mappings** | A candidate mapping from a source into the integration layer. | The mapping is metadata |
| **4 Drafts template code** | A starting point for a new integration pattern. | Templates are tested before use |
| **5 Answers questions about the model** | 'Which marts depend on this Satellite?' - in natural language. | Reading the design can't corrupt it |
| **6 Spots inconsistencies** | Two objects with the same key; inconsistent naming convention. | Findings feed the health checks |
| **7 Proposes design changes** | Explain, find, map, propose, change using natural language. | Every change is approved as a diff |

**Use AI to manage your metadata. Use deterministic templates to move and transform your data.**

# 3 In practice

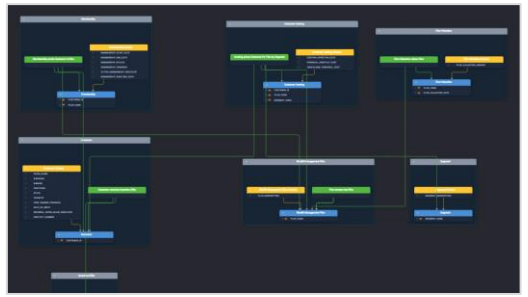
LLM-assisted modeling in Agnostic Data Labs - while delivery stays deterministic

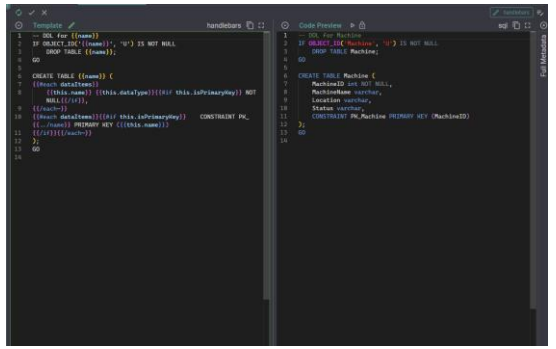
# Agnostic Data Labs

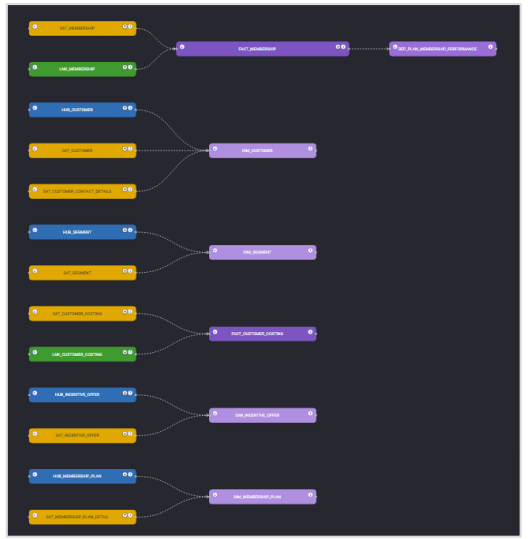
{ADL}

Agnostic Data Labs (ADL) is a working implementation of Data Engine Thinking: design in the browser, manage metadata and templates as plain files in your own git repository, generate using an open-source engine.







## Where the AI sits in ADL

- An AI assistant works on the metadata in natural language
- It calls the model you configure (you bring your own API key)
- You can tweak the agent/prompt if you want
- Every proposal is previewed, and applied only on approval
- Because the metadata is versioned in Git, every accepted change is a diff

Data Engine Thinking

# LLM-assisted modeling, deterministic delivery

Demo

# What just happened?

**AI supports the model generation**
The result is not the model until someone accepts it.

**Change appear as a diff**
Design changes are in Git, until push forward.

**The accepted design is metadata**
Every proposal is reviewed and approved by a human.

**Delivery stays deterministic**
The same templates generated the code from the accepted design.
Run it again tomorrow and the output is identical: same design in, same code out.

# What to take home

### Encode the complete design in metadata

Version-controlled, serialized in your own repository. The design *is* the IP, the durable asset. Everything else can be generated from it.

### Document your patterns, encode them as templates

Deterministic templates turn metadata into code, documentation, and deployment artifacts.

### Let AI draft (but not deliver)

Definitions, classifications, mappings, model proposals: assistance with review, on the design only.

Nothing stochastic touches data logistics: delivery must remain a deterministic function of the accepted and tested design.

### Merge many drafts into one design

Different calls give different answers – this can help generate ideas. Agreement builds confidence; differences open up conversations on business interpretations.

### Draw the line

Use AI to manage your metadata.

Use deterministic templates to move and transform your data.

# Where to go from here

1

## Read the book

Data Engine Thinking, with Dirk Lerner - the complete method · dataenginethinking.com · dtng.link/order

2

## Do the training

Hands-on, three days: from patterns and metadata to a running, generated solution · training.dataenginethinking.com

3

## Use the software

Free in public preview: open a sample, generate something · app.agnosticdatalabs.com

4

## Work together

Contact me at roelant@agnosticdatalabs.com

# Thank you

**Roelant Vos** | roelant@agnosticdatalabs.com

The approach, in article format: roelantvos.com/blog/data-engine-thinking-with-ai

**Data Engine Thinking, with Dirk Lerner** | dataenginethinking.com

**Agnostic Data Labs, with Stefan Johansson** | app.agnosticdatalabs.com - free in public preview

{ADL}

