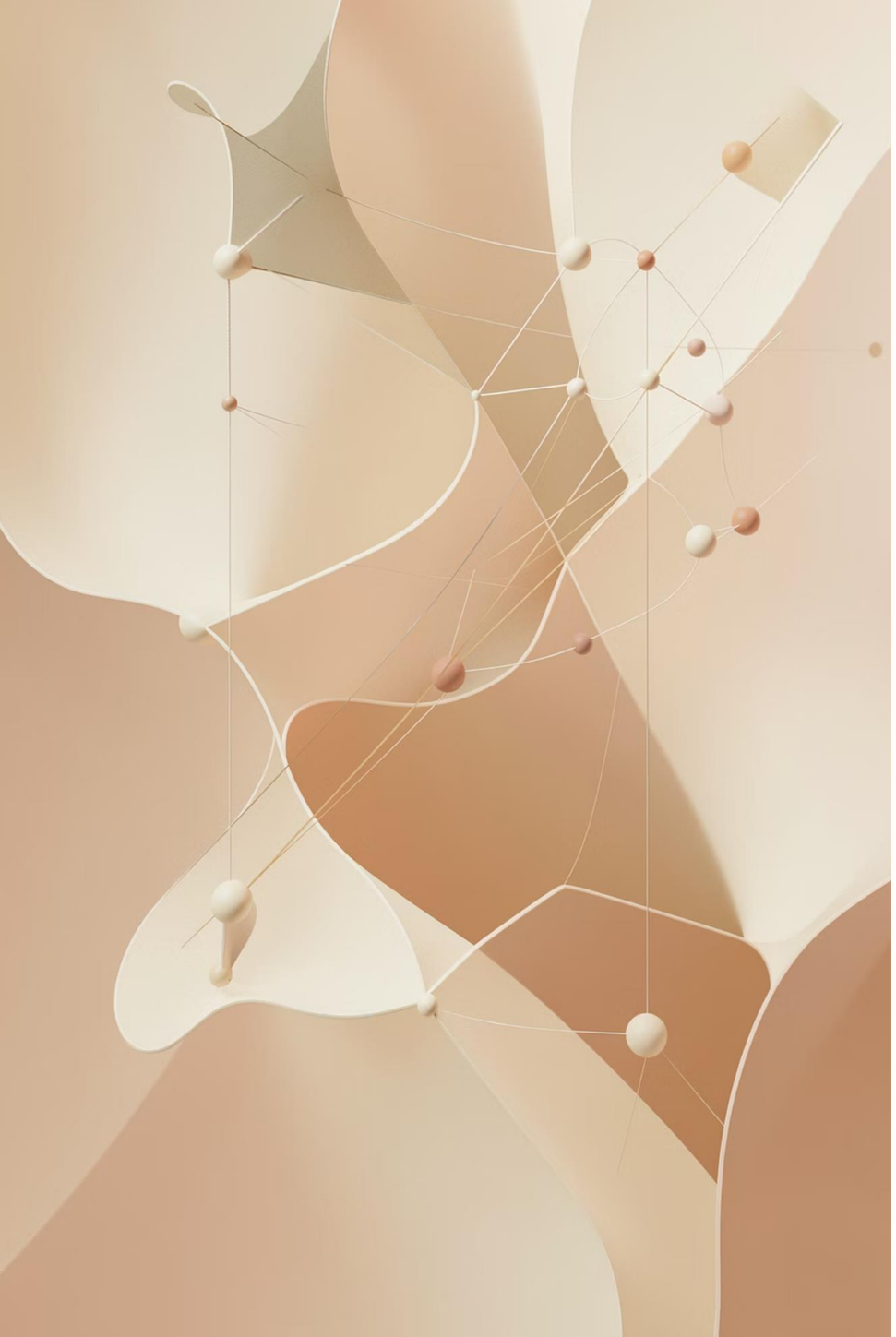
From Opaque Data to Queryable Knowledge

Making manufacturing knowledge queryable, traversable, and operational

Reply



The System Context

This is not a broken system — it is a system that became **difficult to understand at full depth** over time.

Scale

Large, business-critical manufacturing landscape spanning production, logistics, quality, and maintenance systems

Complexity

ERP, MES, PLM, and SCADA — each carrying years of accumulated business logic and process knowledge

Longevity

Long-running operational systems with process knowledge spread across many contributors and organisational layers over time

THE PROBLEM

The Handover Risk

The risk was not theoretical. A **a supplier transition and internal reorganisation mandated a formal knowledge transfer** — and the knowledge gap had to be addressed head-on before that date arrived.

⚠ Operational understanding was concentrated in a small number of people who knew only the parts they had personally touched.

Documentation Drift

Docs fell out of sync as the system evolved — no longer reliable as a source of truth

Siloed Knowledge

Each team understood their system; no single person held the full picture across production, logistics, and quality

Regulatory Pressure

Mandated provider rotation created a hard deadline — knowledge had to transfer completely

THE PROBLEM

The Operational Gap

When an issue arose, operations engineers faced a critical challenge: a fragmented system landscape that turned root cause analysis into a time-consuming, intuition-driven struggle.

1

Time Drain

Significant delays in identifying the affected system, process, or data entity due to scattered knowledge.

2

Knowledge Void

Lack of deep process understanding forced reliance on guesswork and inefficient fixes.

3

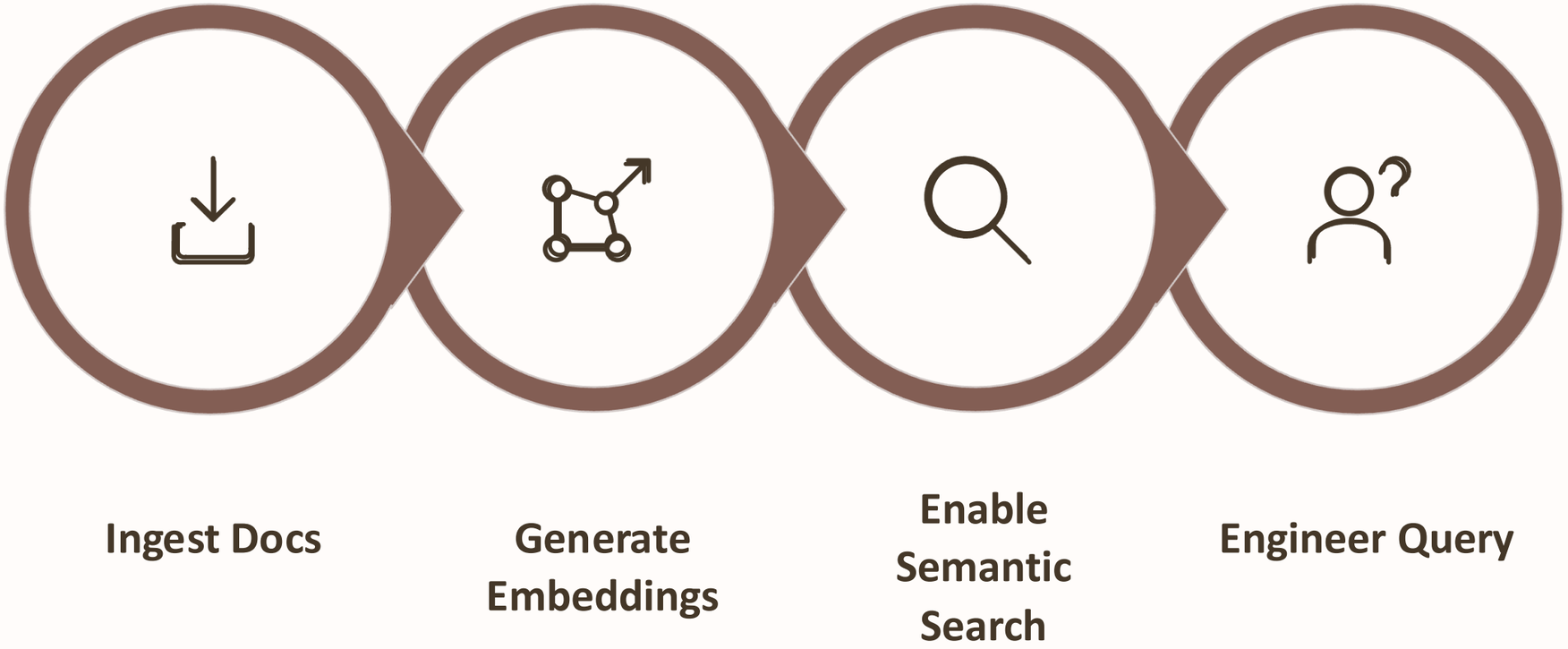
Trial & Error

Engineers resorted to intuition rather than informed diagnostics, increasing risk and downtime.

This bottleneck highlighted an urgent need for an intelligent assistant to bridge the gap between disruption and recovery.

First Approach: Embeddings

The natural first move — ingest the system documentation and process specs, generate embeddings, and enable semantic search. **It improved discovery, but did not solve the full operational problem.**



This approach gave engineers a useful starting point for exploration — but locating a candidate is only the beginning of operational understanding.

Why That Was Not Enough

What Embeddings Gave Us

Pointed engineers to **likely locations** in the documentation — a meaningful improvement over manual search

What Was Still Missing

- No explanation of the **process** behind the behavior
- No visibility into **how systems and processes connected** to one another
- Engineers still had to **reconstruct end-to-end behavior by hand** before any change work could begin

RESULTS

Measured Impact

EFFICIENCY GAIN

30–40%

of triage time saved

Engineers moved from hours of manual investigation to minutes of targeted diagnosis using the agent.

TOKEN USAGE

~3,000

tokens per query

Average consumption per agent-assisted query — lean enough for high-frequency operational use.

Broader Benefits

The impact of a graph-powered manufacturing agent extends far beyond immediate ticket resolution, delivering compounding value across the organization.

Documentation Clarity

Effortlessly identify documentation gaps across production systems, with direct links to the relevant processes and configurations.

Rapid Onboarding

Accelerate onboarding of new operations engineers, enabling faster productivity in complex manufacturing environments.

Enhanced Visibility

Gain deeper insights into under-used or poorly understood parts of the manufacturing system landscape.

Increased Confidence

Give every operations engineer full assurance of comprehensive knowledge transfer across systems and processes.

PART 2

TECHNICAL DEEP DIVE

Architecture • Hybrid Retrieval

The Insight: Knowledge Is a Graph

Flattening knowledge into embeddings preserved semantic meaning — but discarded the **structural relationships needed for operational reasoning**.

Packages → Packages

Modules call other modules, forming deep dependency chains

Procedures → Tables

Every data touchpoint is a traceable, structured relationship

Alerts → Actions

Quality alerts and threshold breaches trigger downstream actions — causal edges in the graph

Jobs → Schedules

Scheduled jobs execute on defined patterns — time-based structural nodes

The structure of operations is **inherently a graph** — treating it as one unlocks full operational understanding.

ARCHITECTURE

What We Built

Graph Nodes

Systems, processes, data entities, equipment, alerts, and schedules — each a first-class graph node

Graph Relationships

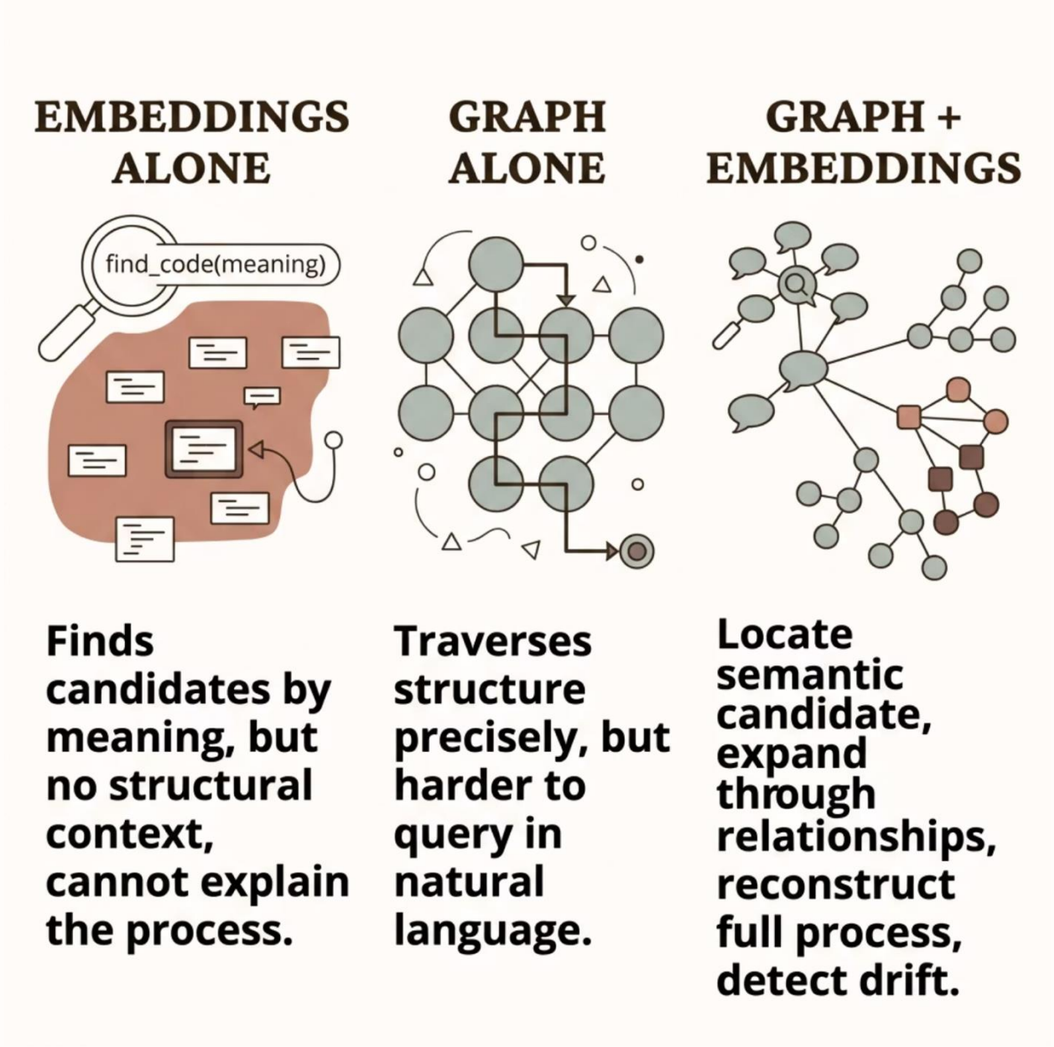
Calls, dependencies, data touchpoints, and ownership — all traversable edges

Hybrid Signals

Embeddings stored on the graph — enabling both semantic and structural retrieval in a single query

Why the Hybrid Approach Matters

This is the core thesis: **semantic retrieval and structural traversal together** create something genuinely more useful for operators than either technique alone.



The combined model lets engineers **describe a production issue in plain language**, then follow the structural path from that entry point through every connected system, process, and alert — automatically.

Drift between what the operational systems do today and what it was originally documented to do becomes **visible instead of hidden**.

Operations Scenario

A realistic operations scenario — an engineer describes an unexpected production issue, the agent identifies the relevant system, reconstructs the current process, and surfaces any drift from documented behaviour.

01

Describe the Behavior

The operations engineer inputs a one-line description of an unexpected production issue

02

Agent Identifies & Reconstructs

The agent locates the likely module and walks through the process as it runs today

03

Drift Detection

The agent highlights where current system behaviour has diverged from documented expectation

Real Use #1

Jira ticket details

Ein Kunde hat folgende Anfrage an uns weitergeleitet: Hi Michael, I was made aware of a bug in easyPara. I specifically tested this with part number 515132. If you add, for example, the 51000 - Body Center unit to the functional units, the following error message appears. Could you possibly take a look at this and analyze whether it might be a bug? If you have any further questions, please feel free to ask and thank you in advance for your efforts. Best regards, Max PreparedStatementCallback; SQL []; ORA-02291: Integrity constraint (PFPS.SFG_SRS_FK) violated - parent key not found; nested exception is java.sql.SQLIntegrityConstraintViolationException: ORA-02291: Integrity constraint (PFPS.SFG_SRS_FK) violated - parent key not found StackInfo: easyParaVersion = [4.38.0 (Build date: 11.06.24 14:59)] , appConfig = [p] , classDesc = [ClassDescription: de.porsche.easypara.app.model.StationFunktionseinheitZuordnung] , sql = [INSERT INTO PFPS.STATIONSFESZUORDNUNGEN (SFG_ID,REIHENFOLGE,BESCHREIBUNGID,FESEINGABETYP,SSM_SSM_ID,FET_FET_ID,BLD_BLD_ID) VALUES (?,?,?,?,?,?,?)] , modelObject = [StationFunktionseinheitZuordnung@45503@Verbindung (45503) zwischen Station '515132 - Finish I/Takt 13 Abfahrt re' und Bild null und Funktioneinheit '51000: Karosserie mitte' und Station '515132 - Finish I/Takt 13 Departure right' and Station '515132 - Finish I/Takt 13 Departure right' (C/O /C)] , ... DB-Commit]]

Real Use #2

Jira ticket details

Hi Michael, I was made aware of a bug in easyPara.
I specifically tested this with part number 515132. If you add, for example, the 51000 - Body Center unit to the functional units, the following error message appears. Could you possibly take a look at this and analyze whether it might be a bug?
If you have any further questions, please feel free to ask and thank you in advance for your efforts.
Best regards,
Max
PreparedStatementCallback; SQL []; ORA-02291: Integrity constraint (PFPS.SFG_SRS_FK) violated - parent key not found; nested exception is java.sql.SQLIntegrityConstraintViolationException: ORA-02291: Integrity constraint (PFPS.SFG_SRS_FK) violated - parent key not found
StackInfo: easyParaVersion = [4.38.0 (Build date: 11.06.24 14:59)] ,
appConfig = [p] ,
classDesc = [ClassDescription: de.porsche.easypara.app.model.StationFunktionseinheitZuordnung] ,
sql = [INSERT INTO PFPS.STATIONSFESZUORDNUNGEN
(SFG_ID,REIHENFOLGE,BESCHREIBUNGID,FESEINGABETYP,SSM_SSM_ID,FET_FET_ID,BLD_BLD_ID) VALUES (?,?,?,?,?,?,?)] ,
modelObject = [StationFunktionseinheitZuordnung@45503@Verbindung (45503) zwischen Station '515132 - Finish I/Takt 13 Abfahrt re' und Bild null und Funktioneinheit '51000: Karosserie mitte' und Station '515132 - Finish I/Takt 13 Departure right' and Station '515132 - Finish I/Takt 13 Departure right' (C/O
/C
)] ,
...
DB-Commit]]

What Changed

Before

- Ticket triage felt like **archaeology**
- Engineers spent hours locating the right system or process
- Process understanding had to be **manually reconstructed** before any change work could begin
- Drift between actual behaviour and documented expectation was invisible

After

- Describe the behavior — the agent **locates the likely area quickly**
- The agent walks through the process as it runs today
- Engineers compare current behavior against **documented expectation**
- Drift becomes visible — representative triage moved **from hours to minutes**

Operational knowledge that once lived in a few engineers' heads is now **queryable, traversable, and transferable** — ready for any handover.

The Takeaway

- Operational knowledge trapped in people's heads is heads is a manufacturing delivery risk.
- A graph is a better place to hold that knowledge than disconnected documents.
- When combined with embeddings and an agent layer, it becomes queryable, traversable, and operationally useful.

Questions