

A JCARDENA.COM FIELD GUIDE

The Mesh

A field guide to event-driven agentic systems — agency after the loop.

The loop is not the architecture of agency; it is local cognition inside one worker. This guide is about what lives *around* it: agents embedded in a continuous field of events, waking on meaningful change, acting in parallel, and composing into chains of reaction a human designs the rules for. Concepts, not products — the ideas that should drive your architecture decisions.

Juan Cardena

Enterprise architect · 25 years building data, software & agentic systems

CONTENTS

01 The Loop Is Local

02 The Event Field

03 The Agent as Projection

04 Choreography vs Orchestration

05 The Minimal Living Mesh

06 The Dark Geometry

07 Designing Event Topology

08 Patterns in Practice

09 Glossary & Sources

CHAPTER ONE

The Loop Is Local

Everyone draws the agent the same way: a loop. Think, act, observe, repeat. It is a good diagram, and I have drawn it many times. But after wiring real systems, I have come to believe the loop is the smallest part of the story — and that mistaking it for the whole is why so many "agentic" projects feel powerful in a demo and inert in production.

Be fair to the loop first, because the easy version of this argument is a strawman. A ReAct-style loop is not a rigid script — within a step the model chooses its tool, its query, whether to stop. That is real, dynamic control flow. Its limitation is not that it is "predefined." Its limitation is **temporal**: it assumes a turn-based world that politely waits for the agent to finish thinking before the next thing happens.

> *The loop is local control flow. The world is global coordination. And the world does not take turns.*

While your agent reasons, prices move, a file lands, a customer cancels, a sensor trips, another agent emits a result. The loop is not the architecture of agency. It is one piece of it — the local cognition inside a single worker. What changes how systems get built is everything *around* that worker.

CHAPTER TWO

The Event Field

The unit the loop is built around is the *turn*. The unit reality is built around is the **event** — a durable fact that something happened, at a time, with consequences. Events do not arrive in order, on request, one at a time. They arrive like weather.

So the question changes. It is no longer "how do I make my agent loop smarter?" It is "**what should wake intelligence up, and what is it allowed to do when it does?**" That is an architecture question, and the architecture already exists. We spent twenty years building systems that treat the world as a stream of events — durable, append-only logs with producers, consumers, replay, and fan-out. We just never pointed agents at it.

CONCEPTS, NOT PRODUCTS

This guide names no single tool on purpose. The thesis is **durable event streams plus autonomous reaction surfaces**. Which log you reach for — a streaming platform, a message log, a database's change feed — is a per-plan decision; the topology underneath is the same. The concepts, not any product, drive the architecture.

CHAPTER THREE

The Agent as Projection

In an event-driven system the durable thing is not the program's current state — it is the append-only log of events. State is a *view* derived by replaying that log. This is **event sourcing**, and it reframes what an agent is.

The shallow upgrade is "the agent is a producer and a consumer." True, but not the profound part. The profound part is that the most valuable agents **maintain a materialized view** over the event stream — a durable, consistent interpretation of a slice of the world, updated continuously as events land.

BE PRECISE (THE REST DEPENDS ON IT)

The agent is **not** the view itself — a view is a deterministic read model. The agent is the side-effecting **decision surface** that reads the log, updates its projection, and emits new events. It stops being a worker that executes and becomes a **membrane** that holds a meaning.

This distinction is not pedantry. Your replay, your causality, your idempotency — all of it depends on separating the deterministic projection from the non-deterministic decision that consumes it.

CHAPTER FOUR

Choreography vs Orchestration

The genuinely radical move here is not technical, it is political: a shift from **orchestration** to **choreography**.

Orchestration has a central conductor — a workflow engine, or the loop itself — that tells each step when to run. Legible, controllable, and prone to becoming a rigid bottleneck. **Choreography** has no conductor: each agent reacts to events it cares about and emits its own, and coordinated behavior *emerges*. It decouples and scales beautifully. It is also far harder to debug.

> *Mature systems need both: choreographed reaction chains for breadth, explicit orchestration for the high-risk, long-running work.*

This is Conway's Law applied to agents: the shape of your event topology becomes the shape of what your system can think. Choosing where to choreograph and where to orchestrate is one of the most consequential decisions you will make — and it is a *design* decision, not a product one.

CHAPTER FIVE

The Minimal Living Mesh

Here is the smallest real version, scoped honestly, because the lazy critique writes itself: "this is just event-driven ETL with an LLM bolted on."

I run my own news system. Scheduled agents pull, on my cadence, from **specific producer sources I chose** because they align with how I want to see the world — not whatever a generic search coughs up. Each pull becomes events. A central system holds them. Consumers — me, and downstream agents — subscribe, and one agent's output becomes another's input.

What it is, honestly, is **a seed** — the minimal living mesh, the first organism, not the proof of the whole thesis. A production mesh must survive harder questions: replay, idempotency, failure handling, observability, downstream contracts, cost control. Mine gestures at all of them; it has not had to survive all of them at scale. The difference between a demo of event-driven agents and a system you can trust is precisely those unglamorous guarantees.

CHAPTER SIX

The Dark Geometry

Every architecture has failure modes that look exactly like success. Event-driven agentic systems fail *distributed* and they fail *quietly*.

1 · Reaction storms — the new infinite loop

One event wakes many agents, which emit more events, which wake more agents. Past a density threshold, stable propagation phase-transitions into exponential cascade. The system looks alive while it is amplifying noise — and you will not see it in testing.

2 · Causality rot

Logs tell you what happened, not why. With non-deterministic agents and multi-path events, "why did this occur?" becomes statistically answerable but not *forensically*. Dashboards are observability theater; you need true causal tracing.

3 · Idempotency collapse

Mind the vocabulary: *exactly-once* is a broker guarantee. What breaks is *idempotency* — handling the same event twice no longer yields the same side-effects, because the model or the world moved. Move from "same message, same handler" to **"same intent, bounded side-effects."**

4 · Ownership black holes

A triggers B triggers C that destroys value. Who is accountable? The **event-topology designer** becomes the new point of moral failure. You need chain-level governance: audit, rollback, approval boundaries, kill switches.

5 · The always-on tax

A 24/7 mesh discovers every cost leak and bills you. Continuous operation is a *capability*, not a virtue. Without budgets, quiet hours, backpressure, and

sampling, "parallelism" becomes uncontrolled spend.

6 · Immutable hallucinations

The one to fear most. If an agent hallucinates, it appends a *stochastic lie to an immutable log*, and downstream agents read it as ground truth — because the log *is* the truth. The falsehood becomes permanent causality; the bedrock is poisoned. And as agents drift in how they interpret what events *mean*, the mesh slides into incoherence with no error thrown.

> **THE RULE**
>
> The future is not "more events, more agents." It is **better event design** — schemas, filters, priorities, contracts, dedup. An agent that reacts to everything is not intelligent. It is loud.

CHAPTER SEVEN

Designing Event Topology

Notice what every failure mode had in common: it is a *design* problem, not a model problem. That is the real shift in the human's job. You stop writing procedures and start designing the conditions under which the system should notice, decide, and respond.

THE TOPOLOGY DESIGNER'S CHECKLIST

- **Signals.** What event is worth waking intelligence for? Most are not. Filter ruthlessly.
- **Contracts.** Version your event schemas; an event's *meaning* must be stable or explicitly migrated.
- **Legality.** Which reactions are allowed autonomously, and which require a human turn?
- **Damping.** Rate limits, backpressure, circuit breakers — so a storm cannot start.
- **Provenance.** Capture lineage, context, model version, costs, suppressed alternatives — or you cannot answer "why."
- **Idempotency.** Define "same intent" and bound the side-effects per intent.
- **Ownership.** Name who owns each chain, where the audit is, and where the kill switch lives.
- **Budget.** Set the spend ceiling and what happens at its edge.

This is a craft closer to designing a nervous system — or a city's traffic rules — than to writing a script. The human is not replaced. They are relocated: from operating

the machine to shaping the possibility space of its reactions, and holding the switch that stops it.

CHAPTER EIGHT

Patterns in Practice: Anatomy of an Order

The dark geometry told you what to *avoid*. This chapter shows what to *build* — three patterns that recur in every healthy mesh, then a single event traced end to end so you can see them work together.

Three patterns you will reach for constantly

THE AGGREGATOR

An agent that consumes a stream of fine-grained events and maintains one coarse-grained **materialized view** — an order's running total from line-item events, a customer's current risk score from many signals. It is the canonical "agent as projection." Its output is state others can read; it rarely takes irreversible action itself.

THE GATEKEEPER

An agent whose only job is to enforce one rule and emit a verdict event — `stock_reserved` / `out_of_stock`, `fraud_cleared` / `fraud_held`, `human_approved` / `human_rejected`. It is where your "legality" and "human-approval" topology decisions become concrete. Small, single-purpose, and the easiest place to put a kill switch.

THE SAGA

For a multi-step process across several agents that must be able to **undo** itself. Each step emits a success event that triggers the next; any failure emits a compensating event that rolls back the earlier steps (release the reserved stock, refund the charge). This is how you get transactional integrity in a world with no central transaction.

A worked example: one `order_placed` event

Watch a single order move through the mesh. No agent calls another directly; each only reacts to events and emits its own.

1. **The signal.** A storefront emits `order_placed` (a versioned contract: id, items, customer, total) to the durable log. Nothing else has happened yet.
2. **The aggregator wakes.** An *Order View* agent projects `order_placed` (and later events) into a running materialized view of this order's state. It owns the answer to "what is true about this order right now?"
3. **Gatekeepers react in parallel.** An *Inventory* gatekeeper emits `stock_reserved` or `out_of_stock`; a *Fraud* gatekeeper emits `fraud_cleared` or `fraud_held`. Each is one rule, one verdict, independently.
4. **The legality boundary.** If the total exceeds a threshold from your topology checklist, a *Human-Approval* gatekeeper holds the chain and waits for a person — emitting `human_approved` only when a human acts. This is your kill switch made routine.
5. **The saga composes the verdicts.** An *Order Saga* watches for all-clear (`stock_reserved` + `fraud_cleared` + `human_approved`) and emits `order_confirmed`. If any gatekeeper emits a failure, it instead emits compensating events — `release_stock`, `void_authorization` — and the system unwinds cleanly.
6. **Damping & ownership.** Every emitted event carries provenance (which agent, which model version, what it saw). The saga owns the chain; its compensations are the rollback; the human-approval gatekeeper is the boundary; a rate limit on retries is the damping that stops a storm.

> *Notice what this is not: a workflow engine calling steps in order. It is a set of independent agents each reacting to events — choreography — with a saga supplying just enough orchestration where the stakes demand it.*

That is the whole field guide in one flow: a signal becomes events; aggregators hold state; gatekeepers enforce rules and emit verdicts; a saga composes them and

can undo them; provenance, ownership, and damping keep it honest. Start with one event and two agents. The mesh grows from there.

APPENDIX

Glossary & Sources

Event

A durable, immutable fact that something happened, at a time, with consequences.

Event sourcing

State is derived by replaying an append-only log of events, not stored directly.

Materialized view / projection

A deterministic read model built from the event log. An agent *maintains* one; it is not one.

Choreography

Decentralized coordination: agents react to events and emit their own, with no central conductor.

Reaction storm

A runaway cascade where events trigger agents that emit more events — the distributed infinite loop.

Immutable hallucination

A hallucinated event appended to the log and treated downstream as permanent ground truth.

Event topology

The designed structure of signals, reactions, contracts, and boundaries — the human's new craft.

Sources

- Martin Fowler — *Event Sourcing* (martinfowler.com/eaaDev/EventSourcing.html)
- David Luckham — the foundational work on Complex Event Processing
- Jay Kreps — the log as a unifying abstraction; Martin Kleppmann — *Designing Data-Intensive Applications* / "Turning the Database Inside Out"

- The Reactive Manifesto; choreography vs orchestration in distributed systems

> **Part of a bigger picture**
>
> This guide is the architecture underneath something I'm building: **Sphere LLM** — a proof-of-concept set of tools and agents that live on exactly these event-driven concepts. A sphere of interconnected agents reacting to a continuous world of events, with a human designing the topology and holding the kill switch. It's early, and there is much more to come — but the loop is already becoming the mesh.