GLORIA GALLO · ENTERPRISE ARCHITECTURE · 2026

The Propagation *Architecture Framework*

The operational reference

This is the operational reference for Propagation Architecture. The structural logic, the operating principles, the design specifications, and the implementation tools, organized in one working document.

"Objects move. Objects carry meaning. Systems evaluate. Signals emerge."

Gloria Gallo

GLORIAGALLO.COM

Contents

1 *The Two Economies*

> *Inside almost every enterprise, a second economy operates alongside the one that creates value. This economy exists to stabilize the movement of that value across disconnected systems.*

Every enterprise runs two economies simultaneously. The first creates value: products, contracts, commitments, revenue. The second compensates for the fragmentation that prevents value from moving coherently. The second economy is not what the company sells. It is what keeps operations moving despite the architecture.

The Value Economy

The Value Economy is where products are created, commitments are formed, and economic outcomes are generated. Its costs are understood and accounted for. It is what the organization was designed to do.

The Compensation Economy

The Compensation Economy consists of re-entries, approvals, coordination, and reporting: the work required to keep operations functioning when architecture is fragmented. It is not visible in financial statements. It appears as operational velocity, or the absence of it.

	Fragmented Architecture	Coherent Architecture
Object at boundary	Loses structural continuity, must be reconstructed	Carries its meaning, propagates without reconstruction
Governance	Applied after the fact, checkpoint after commitment	Embedded in execution, evaluated at the moment of transition
Intelligence	Reconstructed, reports assembled from system exports	Generated, signals emerge from movement in real time
AI deployment	Text generator, output interpreted and re-entered manually	Operational system, structured object evaluated by gates
Cost structure	Value creation plus compensation layer	Value creation only

Four Patterns of Architectural Fragmentation

Fragmentation produces four recurring patterns inside every enterprise.

- **Pattern 1, Data reconstruction.** Information is recreated between platforms. Product attributes defined in engineering are re-entered into commercial systems. Teams spend significant effort recreating information that should travel automatically.

- **Pattern 2, Governance checkpoints.** Governance rules are not embedded within execution systems. Compliance checks occur after contracts are signed. Governance appears as friction rather than infrastructure.
- **Pattern 3, Human relays.** People function as connectors between systems that cannot exchange operational information directly. This is not because the work requires human judgment. It is because the architecture requires human translation.
- **Pattern 4, Reporting reconstruction.** Operational systems do not generate reliable signals during execution. Organizations reconstruct performance after the fact from system exports assembled overnight and reviewed in Monday meetings.

> *The question is not whether the Compensation Economy will appear in a fragmented enterprise. It will. The question is whether the organization understands what it is paying for it.*

2 *The Structural Test*

Before deploying AI into any operational flow, run this test.

THE STRUCTURAL TEST

After the AI produces its output, how many compensation mechanisms are required before it becomes an operational decision?

If the answer is more than zero, the system is Type One. The AI is producing text for interpretation. The compensation chain is intact. The question is not whether oversight is required; it often is. The question is whether the AI output is structured enough that a system can evaluate it before review.

	Type One, AI as Text Generator	Type Two, AI as Operational System
What AI produces	Text: summary, draft, output	Structured object: named fields, confidence scores, flags
What crosses the boundary	A summary, not evaluable by systems	An operational object, evaluable by gates
Next step	Human reads, interprets, re-enters	Gate evaluates, routes, or stops
Audit trail	Decision point with no structured record	Evidence record produced at moment of decision
Architecture	Unchanged, compensation at AI speed	Redesigned, propagation with AI as intake

3 *Why AI Inherits What the Enterprise Built*

Most enterprises are deploying AI into architectures that have not changed. The model arrives. The fragmentation remains. The result is not a more capable organization. It is a more expensive Compensation Economy, now operating at algorithmic speed.

The AI does not fix broken architecture. It accelerates it. The scale and speed that make AI attractive are the same properties that make architectural debt more dangerous than it has ever been.

Architecture determines both operational performance and AI performance. They are not separate problems. The architectural decision precedes the AI decision. An organization that deploys AI into a fragmented architecture has not solved its operational problem. It has amplified it.

> *What is moving, and does the system keep its meaning as it moves? That is an architectural question. It has always been an architectural question. AI makes the answer more consequential.*

4 *The Operational Object*

DEFINITION

The operational object is the unit of movement inside the enterprise. Not the workflow. Not the process. Not the task. The structured entity carrying the meaning of the transaction across every boundary it crosses.

Work does not move through enterprises. Objects do. Orders move from commercial formation to operational fulfillment. Shipments move from release to delivery. Contracts move from signature to execution. In every case, what moves is a structured entity carrying the meaning of the transaction.

Record vs. Operational Object

Most organizations have records. They believe these records are their operational objects. They are not.

	A Record	An Operational Object
Purpose	Stores information inside a system	Carries meaning across systems
At a boundary	Loses continuity, must be reconstructed	Propagates, meaning travels with it
Governance	Applied externally after the fact	Embedded as governing fields
History	May be overwritten	Immutable, every version preserved
Evaluability	Requires interpretation	Evaluated directly by gates

THE EXTEND, DO NOT OVERWRITE RULE

The single most important implementation rule for operational objects: downstream domains extend the object, they do not overwrite it. Every modification is a version with actor, timestamp, and reason. There is no silent edit. This is not a discipline. It is a design constraint. Build it into the system or the audit trail does not exist.

5 *The Five Field Categories*

Every operational object carries fields in five categories. Every field in the object exists because at least one gate requires it to evaluate a condition. Fields that no gate uses are documentation, not operational object fields. Remove them.

Identity Fields

Field	Purpose	Mutability
object_id	Unique identifier across all systems	Immutable
object_type	Classification of the object	Immutable

Field	Purpose	Mutability
version_id	Current version of the object	Increments only
origin_system	Where the object was created	Immutable
created_at	Timestamp of object creation	Immutable

Semantic Fields

Domain-specific fields that define what the object represents operationally: customer_name, product, quantity, destination, delivery_date, price. Populated by AI extraction from unstructured input. Extended, never overwritten, and version-controlled.

Governing Fields

Domain-specific fields carrying regulatory, contractual, and policy conditions: export_classification, screening_status, license_reference, authorization_state. Populated only from authoritative enterprise systems, queried at intake, never from AI inference. Extended, never overwritten, and version-controlled.

Diagnostic Fields

- **confidence_scores:** Per-field extraction quality, not document-level.
- **missing_fields:** Required fields the AI could not extract.
- **uncertainty_flags:** Fields extracted but below confidence floor.

State Fields

- **current_state:** Where the object is in its lifecycle.
- **allowed_next_states:** Transitions permitted from current state.
- **state_history:** Complete record of all transitions.
- **gate_results:** Evidence records from each gate execution.

Governing fields are never populated by the AI. The AI extracts semantic fields as lookup keys. Those keys query authoritative enterprise systems to populate governing fields. An AI that populates governing fields from unstructured input is generating compliance data from inference. That is not a compliance record. It is a guess with formatting.

> *Unknown does not equal Empty. Unknown means the AI attempted extraction but information was insufficient. Empty means the field was not extracted at all. These trigger different exception workflows. Conflating them produces the wrong exception, assigned to the wrong person, attempting to resolve the wrong problem.*

6 *The Lifecycle and State Design*

An operational object has a lifecycle. The lifecycle defines the states the object can occupy, the transitions between states, and the conditions that govern each transition. The lifecycle is not a process flow. A process flow describes activities. The lifecycle describes states.

STATE DESIGN RULE

States must be mutually exclusive and collectively exhaustive within the object's operational domain. An object occupies exactly one state at any moment. State names describe where the object is, not what is being done to it. States that describe tasks rather than object conditions are a design error.

Correct: intake_exception, describes the object's condition. **Incorrect:** pending_review, describes an activity.

Reference State Set, Commercial Order Object

State	Meaning	Entry Condition
intake_pending	Object created, extraction in progress	Object creation event
intake_complete	Extraction complete, ready for gate evaluation	AI extraction completes
intake_exception	Intake gate produced Route output	Intake gate: Route
commercial_approved	Commercial boundary gate: Proceed	Commercial gate: Proceed
commercial_exception	Commercial boundary gate: Route	Commercial gate: Route
released_for_fulfillment	All pre-fulfillment gates: Proceed	Final pre-fulfillment gate: Proceed
fulfillment_exception	Fulfillment gate: Route or Stop	Fulfillment gate: Route or Stop
closed	Terminal state, no further transitions	Delivery and financial realization, or cancellation with evidence

7 *The Minimum Propagation Contract*

DEFINITION

The minimum propagation contract is the specification of what the object must carry to traverse every gate in its lifecycle without triggering a missing-information route. It is derived from the gates, not from what seems useful to capture.

Every gate in the enforcement layer evaluates specific fields. If the object arrives at a gate missing a field that gate requires, the gate cannot evaluate. It routes. The minimum propagation contract is the complete set of fields that all gates require: nothing more, nothing less.

TARGET FIELD COUNTS

Identity fields: 3 to 5 · Semantic fields: 5 to 12 · Governing fields: 2 to 6

If counts exceed these ranges, remove fields until each remaining field has a gate condition you can name. An object bloated with fields that no system evaluates is as dangerous as an object that carries too little.

The contract is a governance document. It is approved by the operational domain owner, reviewed when gate conditions change, and treated as an architectural change when fields are added or removed.

8 *What a Gate Is*

DEFINITION

A gate is the evaluation mechanism at a boundary. It evaluates specific fields in the operational object and produces one of three outputs: Proceed, Route, or Stop. Gates do not describe what should happen. They determine what is allowed to happen.

The Three Gate Outputs

- **Proceed.** All conditions evaluated by this gate are satisfied. The object transitions to the next state. Evidence is recorded. Movement continues.
- **Route.** One or more conditions cannot be evaluated or do not meet the required threshold. The object enters the exception workflow. A named operator is assigned to resolve the specific condition. Evidence is recorded.
- **Stop.** A defined stop condition is met. The transaction does not proceed. The stop condition was approved by a named authority owner before implementation. Evidence is recorded. The stop cannot be overridden without generating its own evidence.

THE SYSTEM BOUNDARY

A system is complete when every condition it evaluates is derived from the object it carries, and every decision it produces is recorded as evidence without external dependency. If a condition exists that is not represented in the object, the system cannot evaluate it. If a decision requires information outside the object at the moment of evaluation, the system is not operational. It is interpretive.

9 *The Three Gate Types*

Gate Type 1, The Intake Gate

High volume, route-dominant. Certifies that the object is evaluable, not that the transaction is approved. Fires when the AI extraction is complete.

- **Evaluates:** Completeness (all required fields present), coherence (no internal contradictions), confidence floor (all semantic fields above defined threshold).
- **Normal behavior:** High volume of Route outputs is correct. Unstructured input is inherently imperfect. The intake gate absorbs that imperfection before it enters the enforcement layer.
- **Key rule:** An object that fails the intake gate does not enter the flow. It returns to intake with specific, actionable routing information, not a generic needs review status.

Gate Type 2, The Boundary Gate

Domain-specific, primary enforcement. The primary enforcement mechanism of the propagation architecture. Fires at every significant lifecycle transition.

- **Evaluates:** Domain-specific conditions at each boundary. The commercial boundary gate asks whether the enterprise is authorized to make this commitment. The operational boundary gate asks whether the enterprise can fulfill it. The financial gate asks whether payment can be received without regulatory consequence.
- **Key rule:** Every field in the operational object exists because at least one boundary gate requires it. Design the gates first, then derive the object fields from the gate conditions.

Gate Type 3, The Exception Gate

Governs operator review. Ensures the operator review process is bounded, auditable, and conclusive. Prevents exception queues from becoming deferral mechanisms.

- **Evaluates:** Review assignment (named individual, not a queue), review completion (a decision, not a comment), escalation trigger (time limit exceeded), decision validity (consistent with governing constraints).
- **Key rule:** Every exception is a data point. Recurring exceptions identify extraction failures. Escalating exceptions identify ambiguous governing conditions. The exception gate produces intelligence, not just decisions.

10 *Stop Conditions and Authority*

STOP CONDITIONS

Stop conditions define when no decision can authorize movement. They are governance documents, approved by authority owners before implementation. If no authority owner will sign off, the condition becomes a Route.

Three Sources of Stop Conditions

- **Regulatory hard stops.** Not judgment calls. The system does not route them for review, it stops them. A sanctioned entity, a prohibited destination, an expired license. The operator can investigate the stop. The stop cannot be overridden without a process that generates its own evidence.
- **Contractual hard stops.** Delivery to a location not covered by the contract. Pricing outside the authorized tier. Quantity exceeding the license balance. These are not interpretation problems, the contract specifies the condition.
- **Architectural hard stops.** Objects with fields below the minimum confidence threshold. Objects with unresolvable field conflicts. Objects that have been routed for review more than the permitted number of times without resolution. These stops protect the integrity of the system itself.

A Stop condition that can be overridden by a phone call is not a Stop condition. It is a routing condition with a more difficult escalation path. Every override must generate evidence: a documented exception, an

authorized approver at the correct level, a recorded business justification, and a flag on the object that the stop was overridden and by whom.

11 *The Evidence Architecture*

EVIDENCE

The machine-readable record of what was evaluated, what was decided, and who or what was responsible. Not a log. Not narrative. Structured, immutable, generated at the moment of decision. Evidence generated at the moment of decision cannot be fabricated after the fact. Evidence reconstructed after the fact cannot be trusted.

Evidence Record, Required Fields

Field	Content	Purpose
gate_id	Identifier of the gate that executed	Links evidence to architecture version
object_id	Identifier of the object evaluated	Links evidence to object history
executed_at	Timestamp of gate execution	Establishes decision timeline
trigger_event	State transition that fired the gate	Establishes why the gate ran
conditions_evaluated	Each condition with field values and result	Proves what was checked and what the values were
output	Proceed / Route / Stop	The decision
operator_review_required	True / False	Routes to exception workflow if true
gate_version	Version of the rule set that produced the decision	Auditable against rule changes

> *An organization with complete evidence chains does not reconstruct its compliance history when an audit arrives. It retrieves it.*

12 *Intelligence Is Not Reporting*

THE DISTINCTION

Reporting describes what happened. Intelligence signals what the pattern of movement means, at the moment it is happening, without anyone producing a report.

	Reporting	Intelligence
When generated	After the fact, weekly, monthly, at close of day	At the moment of execution, as objects move
What it describes	What happened, past state assembled from exports	What is happening, patterns from movement
Actionability	The outcome has already executed before the report arrives	The signal fires at the moment the pattern requires attention
Source	System exports assembled by a reporting function	Gate executions, state transitions, exception patterns
What it replaces	Nothing, it is added on top	The weekly exception report, the escalation call, the reconciliation cycle

Intelligence is not generated by a reporting system. It is generated by execution itself. Every state transition the operational object undergoes produces a signal. Every gate evaluation produces an evidence record. Every routing decision produces a pattern. Intelligence is the structural consequence of building the enforcement layer correctly, not a feature added on top.

13 *Signal Types and Design*

State Signals, Object Position

What they watch: where objects are in their lifecycle, which states they occupy, how long they have been there.

Example: an object has been in intake_exception state for more than four hours without a recorded resolution. Signal fires to operations supervisor.

Recommended first signal: time-in-exception-state signal. Immediately reveals whether the exception workflow is functioning or whether exceptions are accumulating without resolution.

Friction Signals, Volume and Time Anomalies

What they watch: exception rates, routing frequency, confidence score degradation patterns across the object population.

Example: exception rate on customer_entity field exceeds 30% over 48 hours. Signals a systematic extraction failure, not individual exceptions.

Integrity Signals, Data Quality Patterns

What they watch: confidence score trends per field, missing field frequency, uncertainty flag patterns.

Example: average confidence on product field drops below 0.75 across 20 consecutive objects. Signals extraction prompt requires revision, not a data quality issue.

14 *The Enterprise That Sees Itself*

When state, friction, and integrity signals operate together, the enterprise stops depending on a reporting function to tell it what already happened. It sees its own operation as it moves. That capability is not a dashboard. It is the accumulated output of an architecture that generates evidence and signal at every gate, by design, from the first object that moves through it.

15 *Type One vs Type Two*

What the AI Actually Does in a Type Two System

- **1. Reads unstructured input.** Email, PDF purchase order, CRM note, scanned form, portal submission.
- **2. Extracts semantic fields with per-field confidence scores.** Populates named fields with values and confidence scores. Flags missing fields. Records uncertainty markers. Does not invent values.
- **3. Produces the operational object.** The structured JSON object with identity, semantic, and diagnostic fields populated. Governing fields are blank, they will be populated from authoritative systems.
- **4. Hands off to the architecture.** The AI's role ends here. The intake gate evaluates the object. Authoritative systems populate governing fields. The enforcement layer takes over.

The AI does not populate governing fields, make gate decisions, set state, or determine what stops. The AI extracts. The architecture governs. That is the correct division of responsibility, and it is not a limitation of the AI. It is the design.

The Extraction Prompt as System Component

The extraction prompt is part of the system, not a configuration setting. It instructs the AI to produce the operational object. It must reference every semantic field by name, specify the format, instruct per-field confidence scoring, and specify what to do when a field cannot be extracted. Version-control it from the moment it is first used.

16 *The AI's Role in the Architecture*

The AI is the intake, not the system. It sits at the boundary where unstructured input becomes a structured object, and its job ends there. The architecture, the gates, the evidence, the states, exists independently of which model performs the extraction. If the AI is upgraded, replaced, or fails outright, the object still has its identity, the gates still evaluate, and the evidence chain is unbroken. That independence is the point. An architecture that depends on a specific model to function is not an architecture. It is a wrapper around a model.

17 *Compliance as Infrastructure*

THE PRINCIPLE

Governance becomes effective when regulatory conditions are embedded directly into operational systems, enforced at the moment of transaction, auditable without reconstruction.

Most compliance programs depend on human intervention. Organizations have policies. They have trained people. They still fail. The reason is structural: compliance is a department applied after the fact, not infrastructure embedded in execution.

Control DNA

The governing fields of the operational object are the Control DNA of the enterprise. Export classification, screening status, license reference, destination restriction, authorization state. These fields travel with every transaction from the moment of intake. Every gate that touches a governed transaction evaluates them. Compliance is not checked. It is carried.

> *The organizations that perform best financially are those that design compliance into their architecture, not bolt it on afterward.*

18 *The Minimalist Ring*

THE FRAMEWORK

The Minimalist Ring shows how organizations change when the architecture is corrected. Five stages. One cycle. Perpetual. With every cycle the organization becomes more capable of doing what algorithms cannot: seeing the entire value chain and acting on it with judgment, intention, and consequence.

Stage	Focus
1. Perception	See the system actually running, not the one believed designed
2. Connection	Build the nervous system so objects carry meaning across boundaries
3. Coherence	Align every process, metric, and role to the same strategic intent
4. Automation	Let technology carry repetitive execution and consistent enforcement
5. Self-Adaptive Intelligence	The ecosystem evolves in real time, self-correcting and self-optimizing

19 *The Five Stages*

1. Perception

The organization learns to see the system it is actually running, not the one it believes it designed. Data visibility, signal detection, understanding dependencies. The compensation economy becomes visible as a structural condition, not an operational problem.

Signal: The organization can name the compensation mechanisms it runs and estimate their cost.

2. Connection

The enterprise builds its nervous system, connecting people, systems, and data streams so that operational objects carry their meaning across every boundary they cross. Integration moves data. Connection moves meaning. The distinction is architectural.

Signal: At least one operational object crosses at least one boundary without reconstruction.

3. Coherence

Every process, metric, and role is aligned with the same strategic intent. The noise is stripped away. Governance becomes explicit. What remains is signal. The organization stops optimizing the Compensation Economy and starts removing it.

Signal: Stop conditions are defined, approved, and enforced automatically at at least one boundary.

4. Automation

Technology carries what it was born to carry: repetitive execution, consistent enforcement, real-time routing. Humans focus on design, judgment, and governance. Automation is the consequence of coherence, not the substitute for it. AI deployed here into a coherent architecture produces operational systems, not text generators.

Signal: The first Type Two AI system is operational. Evidence is produced at the moment of decision.

5. Self-Adaptive Intelligence

The ecosystem evolves in real time. Feedback loops make it self-correcting and self-optimizing. The organization functions as a living organism: intelligent, responsive, and balanced. The enterprise sees itself without reconstruction.

Signal: The enterprise generates intelligence from movement without a reporting function producing it.

> *You did not build a business. You built an accumulation. Layers of solutions stacked on top of each other over decades, each one designed to fix the last one.*

20 *The 60-Minute System Spec*

Set sixty minutes on a timer. Work through the seven steps in order against a real flow. The discipline is structural: a spec that requires more than sixty minutes has either chosen too broad a scope or has not done the flow-reading work that should precede it.

BEFORE YOU BEGIN, THREE REQUIREMENTS

- **A flow you can trace end to end.** The actual flow as it operates today, with real boundaries, real reconstruction points, and real exceptions. Not the intended flow.
- **Access to the governing data.** Identify which systems hold each governing field and confirm you can query them at intake before you specify the object.
- **The authority conversation.** Know who holds the authority to define that a condition stops movement before you design gates with Stop outputs.

Step	What You Produce	Time
1, Name the flow	Flow name, initiating event, terminal event, boundary count (2 to 5)	5 min
2, Name the object	Object name (one noun), object type, created from, closes when	5 min
3, Specify the fields	All fields in five categories, each traced to a gate condition	15 min
4, Design one gate	Trigger, evaluation set, decision logic, stop conditions with authority, evidence record structure	20 min
5, Define one signal	Signal name, category, generated when, threshold, routes to	5 min
6, Write the extraction prompt	Per-field instructions, confidence scoring rules, missing field handling	5 min
7, Name what stops	Specific compensation mechanism eliminated, condition under which it becomes unnecessary	5 min

> *Do not start with transformation. Start with one object moving correctly. Everything the architecture becomes follows from that.*

The Next 72 Hours

Within 72 hours of completing the spec, run the extraction prompt against five real inputs from the flow you chose. Evaluate each output against the field specification. Count missing fields. Check confidence scores. Identify systematic failures. Those failures are your first refinement cycle: the design working, not the design failing.

21 *The Propagation Checklist*

Before declaring the first system operational, verify each of the following. Do not proceed to the second gate until every item is confirmed.

- The object specification has been reviewed by at least one person who works in the flow and confirmed that the fields match what the flow actually requires.
- Every semantic field in the specification has been extracted correctly from at least three real inputs.
- Every governing field has a confirmed authoritative source and a confirmed method for querying it at intake.
- The extraction prompt produces per-field confidence scores on every semantic field.
- The intake gate has been tested against at least twenty representative inputs and produces Proceed outputs on at least half.
- Every Stop condition has a named authority owner who has been informed that the condition will stop movement automatically.
- The evidence record structure is confirmed, every gate execution produces a record with the required fields.
- At least one Route output has been tested end to end: the exception workflow received the object, a reviewer recorded a decision, and the gate re-evaluated correctly.
- The signal definition has been tested, the condition was artificially triggered and confirmed to route to the correct actor.
- The compensation mechanism named in Step 7 has been confirmed as eliminated, the specific meeting, re-entry step, or call no longer occurs for objects that clear the intake gate.

22 *Complete Field Reference*

The Core Sequence

- **1. Objects move.** The operational object is the unit of movement. Not the workflow, not the process, the structured entity carrying the meaning of the transaction.
- **2. Objects carry meaning.** The minimum propagation contract specifies what the object must carry to traverse every gate without triggering a missing-information route.
- **3. Systems evaluate.** Gates evaluate the object at every significant boundary. Every condition references a field. Every field is in the object. Every evaluation produces evidence.
- **4. Signals emerge.** Intelligence is generated by execution itself, from gate outputs, state transitions, and exception patterns. Not from a reporting function.

Critical Rules, Quick Reference

Rule	What It Means	Why It Matters
Extend, don't overwrite	Every object modification is a versioned extension, never a silent overwrite	The object's history is its audit trail. Overwrites destroy it.

Rule	What It Means	Why It Matters
Governing fields never from AI	Governing fields come from authoritative systems queried at intake	AI-populated governing fields are inference, not compliance
Per-field confidence scores	Every semantic field carries its own confidence score	Generic thresholds produce generic exceptions requiring interpretation
Unknown does not equal Empty	Different conditions trigger different exception workflows	Unknown = attempted, insufficient result. Empty = not attempted.
Stop conditions need authority	Every stop condition requires a named authority owner before implementation	A stop with no named authority will be overridden the first time it fires
Evidence at moment of decision	Evidence records are generated when gates execute, not reconstructed afterward	Reconstructed evidence cannot be trusted or audited
One system, not a program	Start with one object, one gate, one boundary	Programs that start broad produce lessons-learned reports, not systems
Intelligence is not reporting	Intelligence is generated by movement. Reporting describes past state.	Reporting arrives after outcomes execute. Intelligence arrives as they form.
AI is the intake, not the system	The architecture exists independently of the AI model	If the AI fails, the system must still have its object, its gates, its evidence