NEW ESSAY · ARAMAI

Coherence is a capability.

Meaning that survives many handoffs.

Not a platform.
Not a committee.
A practice you grow, with the people who already know.

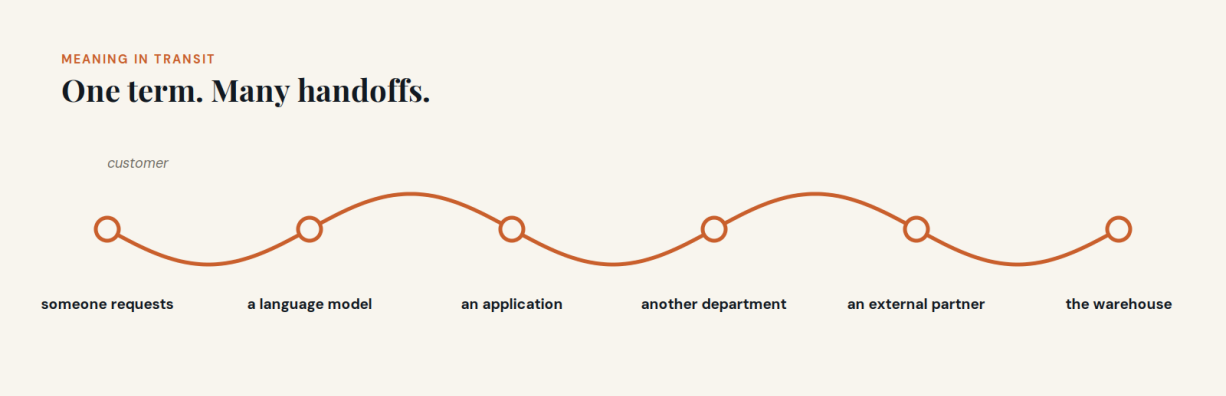
Person to model. Model to model. Department to department.
Partner to partner. Agent to agent. Meaning has to survive every one.

Cruce Saunders
Look up before you make up.

Essay attached
aramai.io

Coherence: meaning survives many handoffs

A request inherits terms the estate holds. Those terms are mapped to meaning across many uses. When the meaning survives every mapping, that is coherence. It comes from humans, AI, content, data, graphs, and systems working well together.



Meaning in transit. One term, many handoffs.

Every outcome is a composite of every input

Nothing an assistant produces is made from nothing. Every answer, draft, module, and plan is a composite of what was fed in, and what was fed in was made by someone, somewhere, earlier. Let's start there, with a request.

Someone asks an assistant for something the company should already be able to produce. Who is the customer on this account. A summary of what the policy says. A first draft of the release notes, the integration, the rollout plan. It might be a manager in a chat box, an analyst asking for a report, an engineer asking for a module, or an agent asking on someone's behalf. It is all the same shape.

What happens next is ordinary now. A prompt fires. A window gets assembled from last week's tickets, a CRM record, a help-center article, a paragraph somebody lifted from a playbook. Tools get called. Something clean comes back, and it is stored for the next turn.

That is the request. It starts when someone asks and ends when they get what they asked for, built from nothing each time and leaving only its product behind. It fails in an afternoon, a bad instruction or a missing file or a loop that won't stop, and somebody on the platform team sees it fail, because a request failing is loud.

A familiar failure pattern, often disguised

Here is one that doesn't look like a failure but is costly, and the costs get worse over time.

The request. An account manager asks for a renewal summary.

What the assistant reads. Three systems, one word, three senses:

- CRM: *customer* is a segment. Mid-market.
- Billing: *customer* is a contract tier. Enterprise, from a legacy bundle.
- Help center: *customer* is an audience tag. Which articles the account gets shown.

What it writes. "Enterprise customer, mid-market segment, renewing in Q4." Fluent. Blended. Nobody in the room can tell which sense carried the pricing.

What happens next.

- The renewal email goes out with enterprise language.
- The summary is stored.
- Next week another agent builds the quarterly deck from that memory.
- "Enterprise" is now a fact about the account that no system ever asserted.

Notice what the failure was not. It was not that billing and the CRM disagree. Those are real meanings, settled by people with good reasons, and neither is wrong. The failure is that at the moment of the request nothing could say which sense was in force for this handoff, so the assistant blended them, and three hops later the blend looked like knowledge.

Coherence is not making *customer* mean one thing everywhere. It is knowing there are several senses, which one this handoff needs, and who is allowed to change each, and carrying that across every join. Looking a term up does not mean fetching the one true definition. It means retrieving the binding that applies at this join, and when two authorities conflict, recording the conflict instead of inventing a third meaning.

Everything the request touched was already there before it started, and the outcome is a composite of all of it.

- The account object. The billing plan. The article. The topic in the content system. The policy with a version number on it.
- None of it was created for this request. All of it sits whether or not anyone opens an assistant today.
- That standing material is **the estate**. It fails slowly, over years, and nothing looks broken while it does.
- Underneath the estate is **the ground**: what the terms mean, who says so, and whether that still holds when work crosses a seam.

Most of the answer already exists there, written down in four places, differently, by people who argued about it. The request does not know that. It only knows how to carry what it is handed.

Meaning lives or dies in the joins

So: coherence, as I mean it here, is meaning that survives transit. Between a person and a model, between one model and the next, between an application and the language it puts in front of a customer, between departments that never meet, across a warehouse, across data in flight and data in use. Meaning is realized, and lives or dies, in those joins.

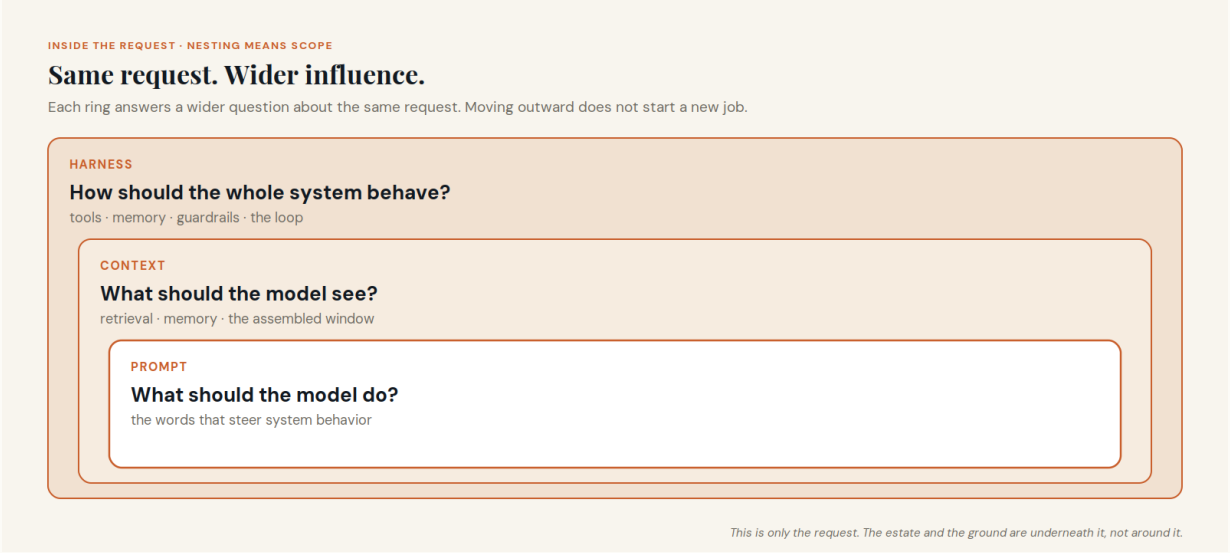
None of this is new to most of us. By virtue of reading this, you too deal in knowledge. All of us in knowledge work recognize the patterns intuitively. The patterns are eternal. In my work in content engineering long before language models: consistent structure and

semantic patterns an organization could share, so a component meant the same thing on the website, in the app, in the various content systems, across translation transforms, personalized variants, and in the PDF the regulator got. Language models multiplied the handoffs and sped them up. The discipline that held meaning steady across channels now has to hold it across models, agents, warehouses, and everything they touch.

And all of it is human work. The question, the records, the article, the definitions, the argument that produced them: people did that, and people will have to keep doing it, as practices. A lot of those practices already exist inside the enterprise. What they mostly lack is composition into something durable, chartered, funded, and given the authority to act, so that they become part of the infrastructure of AI and of everything that happens with knowledge downstream of it, as it reaches customers, partners, supply chains, employees, and now, critically, agents.

Why the request can't fix this from inside

Organizations don't adopt AI. They grow capabilities. Each wave of the last three years arrived as a question, and each question, once an organization learned to ask it well, grew into a practice with a team and a budget line. What should the model do: prompt engineering. What should it see: context engineering, the assembling of retrieval, memory, and the window itself. How should the whole system behave: harness engineering, the tools, memory, guardrails, and loop around the model, which is where agent work became systems work and stopped fitting inside one job description. In a lab, whoever writes the prompt also assembles the context and owns the data. In a bank, those are three departments and two acquisitions. But in every organization using AI to do anything at all, the questions get asked and answered somehow, even if only by default.



Inside the request, nesting means scope. Same request, wider influence.

Each of those questions contains the one before it, and the reason to keep widening is leverage. The harness carries whatever it is handed, into tool calls, into memory, into the next agent's input, into the summary written back as fact. Which is the thing the sequence hides. Leverage is symmetric. So a good harness propagates a wrong term exactly as faithfully as a right one, and faster the better it gets. Think of a relay race where the baton

is a word. Every runner is fast, every handoff is clean, nobody drops anything, and if the first runner was handed the wrong baton the team still finishes first, in perfect form, carrying the wrong thing across the line.

The same flow carries a wrong term just as faithfully. Leverage is symmetric.

The industry has not ignored this. Grounding, citations, evaluation suites, and data contracts are all attempts to get at it from inside the request, and each one helps. But they stand on the same floor. A definition pasted into the system prompt does not survive the next agent, the next model swap, or last week's memory write. Retrieval by similarity finds language that sounds like the term; it does not know which contract the term is under at this seam, and it cannot record that it guessed. Evals grade the answer, not the binding the answer was made from. Do, see, behave: the last three years optimized behavior, and they were right to. Not one of those questions decides who is allowed to say what a term means.

We've met the roles we need, and they are already here

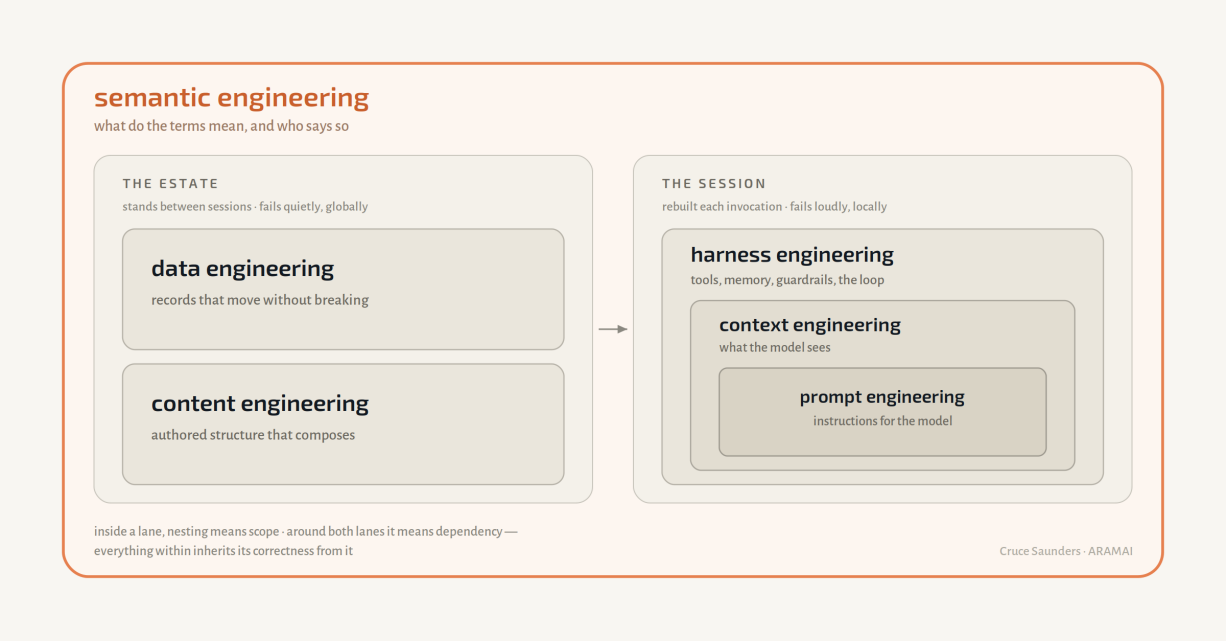
Authority over meaning has a long history. Meaning makers and translators, knowers of all kinds, predate this conversation by many decades. Everything the organization has already written down, modeled, argued over, and settled, kept by people who were doing this work long before anyone opened an assistant. They've gone by many names, but let's look at two of the key practices:

Data engineering asks: what records do we have, and do they survive the trip? Pipelines, contracts, lineage, types. Every ETL job quietly assumes the column called customer_id in the source is the same customer as the one in the target. Mature, well tooled, respected, and present nearly everywhere.

Content engineering asks a three-part question the others don't. What did we actually say, what shape does it live in, and how does it move? Model, metadata, schema, taxonomy: the authored structure. A DITA map (the assembly plan for a set of structured topics), a component in a CCMS, a topic reused by reference in six places: that isn't data engineering, and treating it as data engineering is how a governed document set gets flattened into a bag of text. Content is more than data. Not opposed to it, additive. A structured component carries what a record carries, plus audience, sequence, variant conditions, the reason a sentence sits where it sits. And unlike a record it never stops moving. Assembled, translated, personalized, delivered, delivered again somewhere else. Treating content as

data alone isn't wrong. It's thin.

These two are peers. Neither contains the other. Both are already living in a lot of large organizations, and they usually report to people who don't attend each other's meetings.



Semantic engineering encloses two lanes: the estate, where data and content engineering stand as peers, and the request, where harness holds context holds prompt.

The ground is underneath, not further out

If the request is well engineered, the data is well engineered, and the content is well engineered, what is still missing? Something is, or the renewal email would not have gone out wrong. Inside a request there are three questions, and each is bigger than the one before it. Moving from the first to the third widens your view of the same piece of work. Nothing new is added underneath. That is one kind of nesting, and it means scope.

The relationship between the request and the estate is a different kind of nesting. The request does not contain the estate and the estate does not contain the request. The request sits on top of it. Everything the assistant produces is made out of records and content that already existed, and every one of those is made out of terms that somebody, somewhere, already defined. The definitions are not a bigger ring around the outside of the picture. They are the floor. Everything above inherits whatever is settled, or unsettled, below.

The request sits on the estate. The estate sits on the ground. Nothing above is more settled than the floor it stands on.

That is why a very good harness over an unsettled *customer* does not give you a slightly worse answer. It gives you a confident, fast, well-formatted wrong answer, every time. So the missing thing is **semantic engineering**. Not another practice added to the list, and not a fourth ring drawn further out. It is the ground the other practices stand on, and it asks a plain question: what do the terms mean, and who says so? When it is present, the other practices stop working around each other and start working through each other: the request retrieves what the estate settled, the estate holds what the ground authorized, and the three together produce something none of them could alone, an answer that is fluent, grounded, and still means what it meant when it left.

The definitions already exist

Most of the answer is written down. The CRM has a definition of an account. The ERP has an order object with fields somebody fought over for a quarter. The CMS has a content type with a required audience field, and the digital experience platform that personalizes the site has a segment that was supposed to match it. A hospital has HL7 and FHIR, the standards its clinical records travel in; its help center has a *patient* audience tag that means "someone reading this who is not a clinician," which is not the FHIR Patient resource, and an agent that can't tell those apart will write a very confident discharge summary. The data contracts already say what a customer is, in four places, differently. Decades of settled argument, written into data models, content models, and standards. Two things are missing, small to say and hard to do: those definitions are not reachable at the moment the request needs them, and they have never been mapped to each other.

Reconciling them means connecting the models, not merging them. The dbt project (the transformation code that builds the warehouse tables) and the warehouse schema have a model of the business. So do the CRM, the CMS, the DXP, the ERP, and many other systems. So do the stores of meaning beside them: controlled vocabularies, taxonomies, glossaries, translation memories that hold what every term is called in every language. Each is a place a definition lives, and each is expert in something the others cannot hold. The warehouse cannot hold what the CMS knows about audience and sequence. The taxonomy cannot hold what dbt knows about lineage. The translation system knows a term in twelve

languages and nothing about which system is allowed to change it. Every attempt to collapse them into one master model has produced a very expensive fifth place where the definition lives, slightly differently again.

Which is also the answer to "we have this." A metrics layer holds measures. A catalog holds lineage. A master data system holds identity. A CCMS holds components. A knowledge graph holds whatever someone loaded into it. Each is real, and none of them holds the mappings between the rooms: which sense applies at which seam, and who may change it. That is the unoccupied job. It is the job we built CoreModels to do, hold each model as it is and keep the mappings between them alive, and it is a job that has to be done whether or not anyone buys anything.

The same work, named

The question	Practice	What it settles	Where it already lives
What should the model do?	Prompt	Words are an engineering surface.	Instruction libraries, playbooks
What should it see?	Context	The surround is designed.	Retrieval, memory, knowledge bases
How should the system behave?	Harness	Leverage compounds both ways.	Tools, guardrails, agent platforms
What records do we have?	Data	Standing material must survive the trip.	Warehouses, contracts, CRM / ERP
What did we say, in what shape?	Content	Structure carries intent worth keeping.	DITA / CCMS, taxonomy, variants
What do the terms mean, and who says so?	Semantic	Every term has an authority.	CRM, ERP, FHIR, DITA, contracts
How does meaning survive the handoffs?	Coherence	Mappings between the rooms, maintained.	Schemas and taxonomies at the seams

Growing the capability

A lot of large organizations already hold most of the practices in that table. What they don't have is composition. The practices live as silos with their own vocabularies and their own meetings, and a term changes meaning every time it crosses a seam. Agents make this worse before they make it better: every department can now run its own self-improving loop tuned to its own metric, and a silo that optimizes itself faster does not become more coherent with its neighbors. It becomes more confidently different. The correction isn't a merger. It's a grammar between disciplines, shared reference structures that let each practice stay expert in its lane while meaning survives the handoffs. Maintained by people, consulted by machines.

Start with the people, since the whole request began with one. The sentence the assistant returned was assembled from things people wrote, structured, argued over, and decided. The topic in the CCMS has an author. The account object has a modeler. The taxonomy has a curator who lost a fight about a term in 2019 and was right. None of them are downstream of the AI. They are upstream of it and beside it, and the quality of what a model can carry is bounded by the quality of what they encoded. Which settles where humans sit, and it isn't "in the loop." Dropping a person into a loop to approve what a machine already did sets them up to rubber-stamp, or to fail. The workable division is older and simpler: humans set direction, encode meaning, and own the judgment and the exceptions; machines do the volume and the assembly. Let knowers know, and machines scale out the knowing.

A weeklong taste test for shared meaning

Here is what a weeklong taster for the capability looks like. Pick *customer*. Write the four definitions side by side, the CRM's, billing's, the help center's, the warehouse's, each in the words its own system uses. Name who may change each one, by role and by act: the person who arbitrates when three systems each claim the customer record, the person who writes the shape a million automated exchanges will conform to. Publish the mapping somewhere a machine can read it, in the schema, not in prose about the schema. Point the harness at it. Then watch the next twenty retrievals. Some will resolve cleanly. Some will hit a seam where two authorities disagree, and the right behavior there is not to pick or to blend but to record the contest and hand it to the owner. Two of those twenty will be surprises: a distinction discovered in a bug fix that deserves to outlive the bug, or a mapping that has to be republished because billing renamed a tier on Tuesday. That is the job. Done, for the first stretch, is a small set of contested terms reconciled and bound, not "coherence stood up." A leaking seam shows itself as conflicting answers, translation mismatches, or a retrieval log that records a guess.

Coherence needs a practice

Two things our example week does not include. It does not include a big-bang ontology program or a merged master model being built on a whiteboard, and it most certainly does not include a platform purchase that counts as the capability. The platforms follow the practice. It also does not need a new department, nor a committee.

Centers of excellence usually fail because they are nobody's job, a side activity asking busy people to volunteer time. The ones that work are funded and chartered, and they centralize the pattern-making, the shared standards and workflows, while leaving implementation where it already lives. Coherence needs a practice of that shape, crossing the rooms that already exist: a shared reference structure both can point at, and a named owner for each seam where meaning is known to leak. Not an owner of the CRM and an owner of the help center. An owner of *customer*, wherever it travels. Really, an owner of the mappings. A small structural change with a large effect, because it turns a definition everyone assumed somebody else owned into one somebody actually does.

The change moves at the pace of the estate, not the pace of a request. The definitions drifted over years; they reconcile over quarters, and anyone who has led a content migration or a data governance program knows the shape. Start with what you have,

because no one starts with nothing. Reconcile the settled definitions first. Name the authorities. Tend the intersections. Let the harness look things up instead of guessing, and let the retrieval logs tell you which terms are still contested. Each reconciliation changes what the next reading can see, and the practice matures by degrees rather than by decree. A reader took me to Hermann Hesse's *The Glass Bead Game*, whose scholars build a perfect symbolic language for all knowledge and slowly find that the game is not the same as living what it describes. True that, and it sharpens the point: meaning gets discovered in implementation at least as often as it gets declared in advance. Authority is not always priority. It's all improvisational intelligence, at the edges, accrued and made solid. Wherever a distinction gets discovered, in a schema review, a bug fix, an argument about a settlement date, the question is whether it has somewhere to live afterward, so the next system can look it up instead of re-deriving it. Mining produces candidates. Governance gives them status. Orchestration gives them reach.

Intelligence is interdisciplinary and starts with knowers

None of this is learned inside one discipline, which is the quiet difficulty. The content engineer has to understand what the data modeler means by a contract. The data engineer has to understand why a component is not a row. The prompt author has to know that a term has an authority and where it lives. Thus, we find ourselves in a macro professional shift as much as an organizational one, and it is the shift the Kinetic Information Association, a professional body for content, data, and semantics practitioners, was formed to help people make. Everyone working towards the interrelated set of practices an enterprise needs, rather than living in more isolated silos. Knowing is social. The people who structure knowledge are not being written out of the story. Quite the opposite. They are becoming its authors again. The people who know things. The meaning a model carries is meaning *knowers* gave shape to, and the intelligence that grows in an organization, human and machine together, grows from what knowers understood well enough to write down.

Intelligence happens between organizations

One more interaction horizon, and it is the hardest one. No estate is alone. A supplier's system hands off a purchase order whose *unit* is a case where yours is a pallet. A partner's agent reads your catalog with its own sense of *available*. Between estates nobody holds the authority, so meaning cannot simply be looked up; it has to be negotiated at the boundary, at the moment of contact, and recorded so the next exchange does not start from nothing. That is a different essay, The Stranger Problem. But start inside. Within teams. Coherence between estates is built on coherence within them.

The short form

THE REQUEST — prompt, context, harness. What should the model do, see, and how should the system behave.

THE ESTATE — data engineering and content engineering. Peers. What records survive the trip; what we said, in what shape, and how it moves.

THE GROUND — semantic engineering. What the terms mean, and who says so. Every term has an authority, a place, not a person's memory.

COHERENCE — the mappings between the rooms, maintained. Humans encode meaning and own judgment; machines do the volume and the assembly. Reconcile what's settled. Own the seams. Watch for slow drift. Look up before you make up.

We've spent three years making the request smarter. The estate is where the meaning was the whole time. The ground is where it gets settled. And coherence, meaning surviving its own handoffs, is a capability. Not a value statement, not a committee, and not a platform you can buy.

Capability is the thing that compounds. Yet *coherence as an emergent outcome* compounds unusually well, because meaning is shared infrastructure: one team's investment in it lowers the cost of every other team's data and AI work. It may be a lag of a few years before the return is visible but it offers a decade or more of advantage after.

All of these roles, together with systems that can carry settled meaning. And all are the ground of the neurosymbolic future we are collectively walking into (or, waking into), whether or not anyone uses the word: fluent models over grounded structure, each supplying what the other cannot.

An organization whose terms hold their meaning across every system, every handoff, and every agent stops re-deriving its answers and starts knowing them. The organizations growing coherence capabilities now are the ones that will be leading.

ABOUT THE AUTHOR

Cruce Saunders is the founder of ARAMAI and the architect of CoreModels. Before ARAMAI he founded and led [A], a content intelligence consultancy that designed structured content and semantic systems for some of the largest enterprise publishers in the world, including Mayo Clinic, PayPal, Cisco, Adobe, and Microsoft. He is the author of *Content Engineering for a Multi-Channel World*, a founder of the Kinetic Information Association, and a longtime speaker on content intelligence, semantic standards, and the coherence of content, data, AI, and people.

A NOTE ON HOW THIS WAS WRITTEN

This essay was written with AI as part of the process. Claude models helped draft, restructure, and refine the text and rendered the figures, working from three decades of the author's practice, his outlines, prior writing, reference graph, and reader dialogue on earlier drafts. It went through many editing cycles, each a round of the author's notes and the model's revisions, and the whole of it was grounded in a large knowledge graph the author maintains: a queryable record of his concepts, decisions, definitions, sources, and prior work that the model consulted before drafting, so that what it wrote was looked up rather than made up. The author directed the work, made the judgments, and takes full responsibility for every word. That is less a disclaimer than a description of the thing the essay is about.

ABOUT ARAMAI

ARAMAI builds the infrastructure that lets AI systems retrieve meaning from authoritative structure instead of inferring it from statistical proximity. The founding principle is four words long: look up before you make up. CoreModels is the enterprise platform: it takes the models an organization already has, from data warehouses and dbt projects to CMS content types, vocabularies, and standards, holds them as one connected graph, and maintains the mappings between them so a term can be looked up, at runtime, by the systems and agents that need it. The Schematica schema library is the reference collection alongside it. aramai.io