Creat✦ism

A PRACTITIONER WHITE PAPER

Human-in-the-✦ Lead

Operating systems for AI-augmented knowledge work — and why the structure has to be yours.

Diego Bonifacino

The New Rhythm

creatism.substack.com

thenewrhythm.com

linkedin.com/in/diego-bonifacino

v1.0 · July 2026

EXECUTIVE SUMMARY

A powerful model with no structure is a *brilliant stranger*.

A quiet shift is underway in how individuals and organizations work with artificial intelligence. The first wave was about access — getting a capable model into everyone's hands. The second wave, now underway, is about *structure*: the realization that a powerful model with no persistent context, no shared memory, and no governing rules is a brilliant stranger who walks into the room every morning having forgotten everything. The people and organizations getting real leverage from AI are the ones building an operating system around it.

This paper makes three claims. **First**, the dominant failure modes of knowledge work — context lost between sessions, systems that collapse under their own maintenance, and outputs no one can trust or trace — are not solved by more model autonomy. They are solved by better-governed augmentation.

Second, the governing principle that makes such augmentation trustworthy, especially in high-stakes professional and public-sector settings, is **human-in-the-lead**: the human sets the intent, holds the judgment, and owns the accountability, while the machine does the tireless work of capture, synthesis, and bookkeeping. This is a stronger and more honest stance than "human-in-the-loop," which too often means a person rubber-stamping machine output they don't understand.

Third — and this is the part most of the field misses — the structure has to be *yours*. A system you adopt wholesale from someone else is a system you don't understand and therefore can't trust, can't maintain, and won't lead. Building your own operating system is what produces ownership. Skip it and you have a system you can operate but never lead.

The paper maps the current landscape, states the human-in-the-lead thesis as a set of design constraints, shows how the same principles scale from a personal workflow to the governance of a large multi-party program, lays out an operating cadence borrowed from agile practice, and confronts the real limitations — security, hallucination, over-engineering, and the discipline of building in the open while honoring confidentiality. It closes with a set of open tools you can use and adapt.

The audience is dual: a decision-maker can read to the end of the limitations section and have everything they need; a builder can continue into the technical appendix and start assembling their own system the same afternoon.

SECTION 1

The Problem: Why Knowledge Systems Collapse

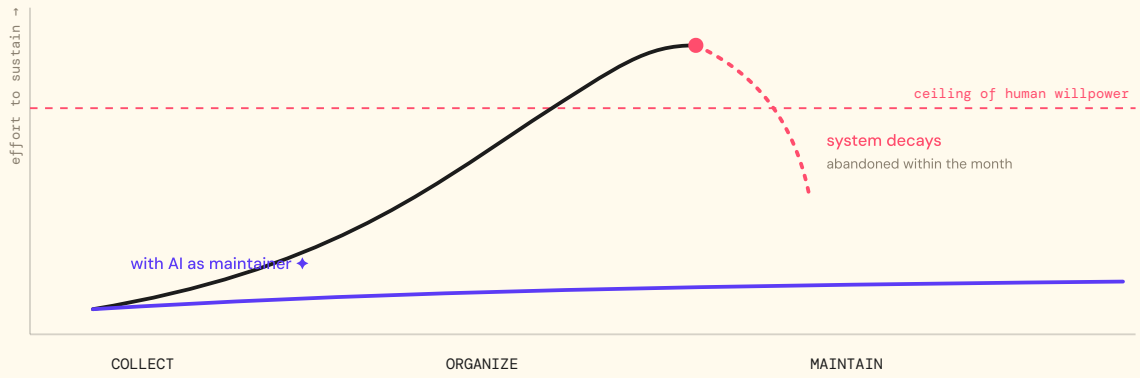
Everyone who has tried to build a "second brain" knows the graveyard. The note-taking app adopted with enthusiasm and abandoned within a month. The folder of five hundred bookmarks no one will ever read again. The beautifully structured vault whose graph view went stale the week after it was set up. The shared drive that became a landfill.

The tooling is rarely the problem. The problem is that building a knowledge system has three stages — collect, organize, maintain — and the difficulty rises sharply across them. Collecting is easy. Organizing is hard. Maintaining is, for most people, effectively impossible at scale, because every new input demands reading, summarizing, linking, cross-referencing, and reconciling against everything already there. Nobody does this consistently. So the systems decay.

FIGURE 1 · THE PROBLEM

Where knowledge systems *collapse*

Collecting is easy. Organizing is hard. Maintaining is, for most people, impossible at scale — so systems decay against the ceiling of human willpower. AI moves that ceiling.



STAGE 1
Collect
Easy. Anyone can capture. The graveyard of bookmarks proves it.

STAGE 2
Organize
Hard. Linking, structuring, deciding what matters takes real work.

STAGE 3
Maintain
Where it dies — or where AI carries the load that willpower can't.

§1 Why Knowledge Systems Collapse

Creat✦ism

For years my own answer to this was analog. I ran my life out of a bullet journal — Bujo, the method's adherents call it — which borrows more from agile practice than most of its users realize: a running log, rapid capture, a monthly migration that is really backlog refinement by another name. Parts of it I still hold priceless: the act of writing by hand, a notebook I can leaf through at a café when the time away from a screen is the whole point. But a paper notebook cannot summarize itself, cannot cross-reference, cannot tell me that a commitment I made in March contradicts one I made in June. Its

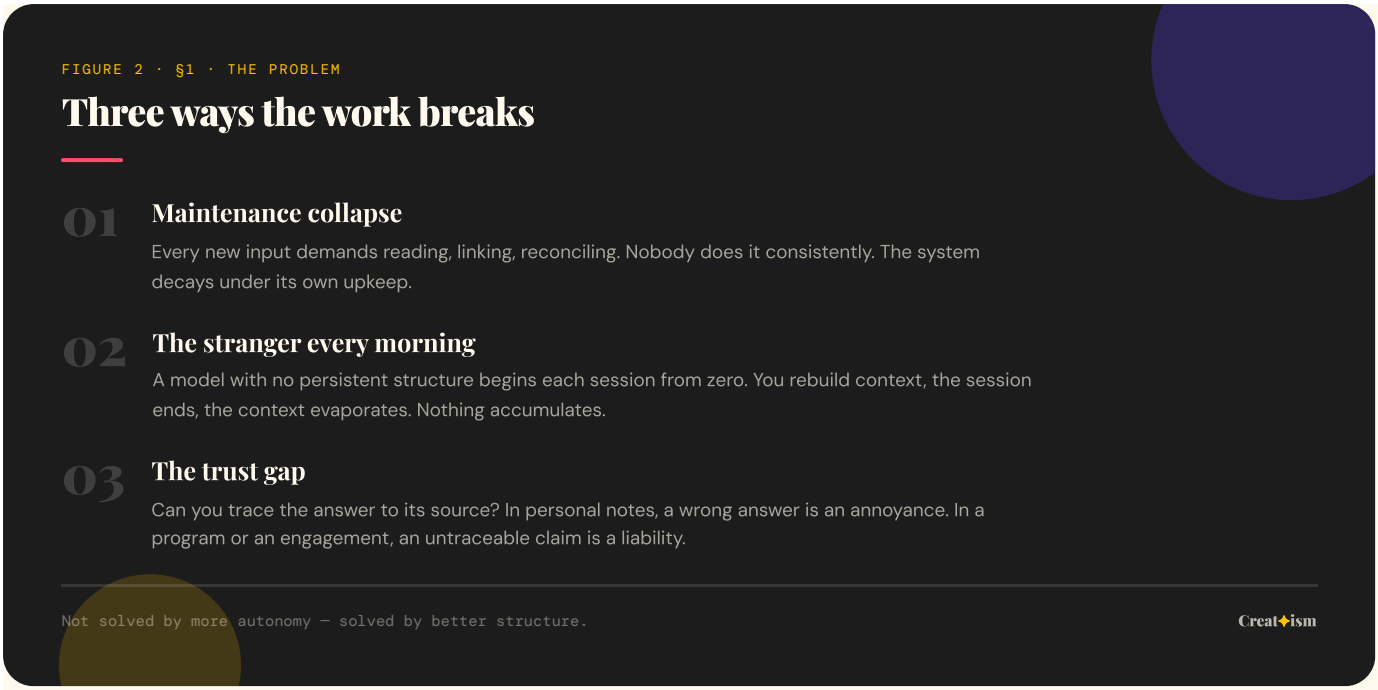
maintenance ceiling is the hand that holds the pen. The question that animates this paper is what happens when you keep the discipline the notebook taught you and hand its ceiling to a machine.

Generative AI changes the calculus, because the very work that kills these systems — the filing, the summarizing, the cross-referencing, the contradiction-checking — is exactly what a language model does tirelessly and at near-zero marginal cost. This is the insight behind the recent wave of "AI second brain" and "personal operating system" experiments. For the first time, the maintenance burden can be carried by something other than human willpower.

But this introduces a second, subtler failure mode, one that organizations feel even more acutely than individuals: **the stranger-every-morning problem.** A capable model with no persistent structure begins each session from zero. You spend twenty minutes rebuilding context — who you are, what the project is, what was decided last week — and then the session ends and the context evaporates. The model is rediscovering your world on every query. Nothing accumulates.

And there is a third failure mode, the one that matters most the moment the stakes rise above personal productivity: **the trust gap.** When an AI system produces an answer, can you trace where it came from? When it makes a claim, can you find the source? When two documents contradict each other, does the system surface the contradiction or silently average it away? In personal note-taking, a wrong answer is an annoyance. In a government program or a client engagement, an untraceable claim is a liability.

I did not arrive at any of this cleanly. I failed my way into it. I had been juggling several approaches at once, not one. I began by building a single standing context for my whole life — every personal and professional project in one place. Then I went deep on a single client and found myself building a rich, client-specific context that the general one knew nothing about. Then a second client, and a third. Inevitably, the problem stopped being capture, or even maintenance; it was that my systems did not know each other. I was the integration layer, holding in my head what each context could not see in the others — and the moment I needed to trace a claim back across them, the seams showed. Watching well-built systems fail to compound because nothing connected them is what convinced me the missing piece was never a better model. It was a better structure.



These three failures — maintenance collapse, context loss, and the trust gap — define the problem this paper addresses. They are not solved by giving the machine more autonomy. They are solved by giving the human better structure.

SECTION 2

The Landscape, Honestly Mapped

I did not invent any of this in a vacuum. A community of builders has been working the same territory, and the honest thing is to map what they have gotten right and where they stop short.

The LLM-as-maintainer pattern. The most influential recent articulation comes from Andrej Karpathy, who described what he calls an "LLM wiki": you collect raw sources into an immutable folder, and the model compiles and continuously maintains a structured, cross-linked knowledge base from them. His framing — *the editor is the interface, the model is the worker, the knowledge base is the artifact* — captures the core inversion. The human stops doing the bookkeeping; the human does the sourcing, the questioning, and the judgment. This pattern is sound, and this paper builds on it.

The layered personal OS. A parallel strand, emerging from practitioners building daily-use systems, organizes the AI's world into layers: a standing context file the model reads at the start of every session (who you are, what tools you have, what to remember), a set of connected tools and integrations, a library of reusable skills for repeated tasks, and scheduled routines that run proactively. This is a genuine architecture, and it maps cleanly onto how an organization actually delegates work.

The agile-ceremony reframe. A third strand, mostly from software practitioners, noticed that the rituals we complain about in project management — planning, backlog refinement, briefing documents — all exist to solve one problem: *context transfer*. And that problem reappears, almost unchanged, when the teammate is an AI. We did not need to invent new disciplines for working with machines. We needed to point decades of process engineering at a new kind of worker. (I will return to this in Section 6, because it is more useful than it first appears.)

Where does the community fall short? I have built on all three; that is what earns the right to say where each one stops. Two places. First, almost all of it is *individual and engineering-flavored* — built by and for solo developers optimizing their own productivity, largely silent on what happens when the system has to serve multiple stakeholders, survive an audit, or carry legal and political weight. Second, it is mostly silent on *accountability*. The community is fluent in autonomy ("the agent does it for you") and underdeveloped on governance ("who is responsible when it does it wrong"). For knowledge work that matters — consulting deliverables, government programs, regulated environments — that silence is the gap this paper fills.

SECTION 3

The Thesis: *Human-in-the-Lead*

The phrase that organizes everything in my practice is **human-in-the-lead**. It is a deliberate sharpening of the more familiar "human-in-the-loop."

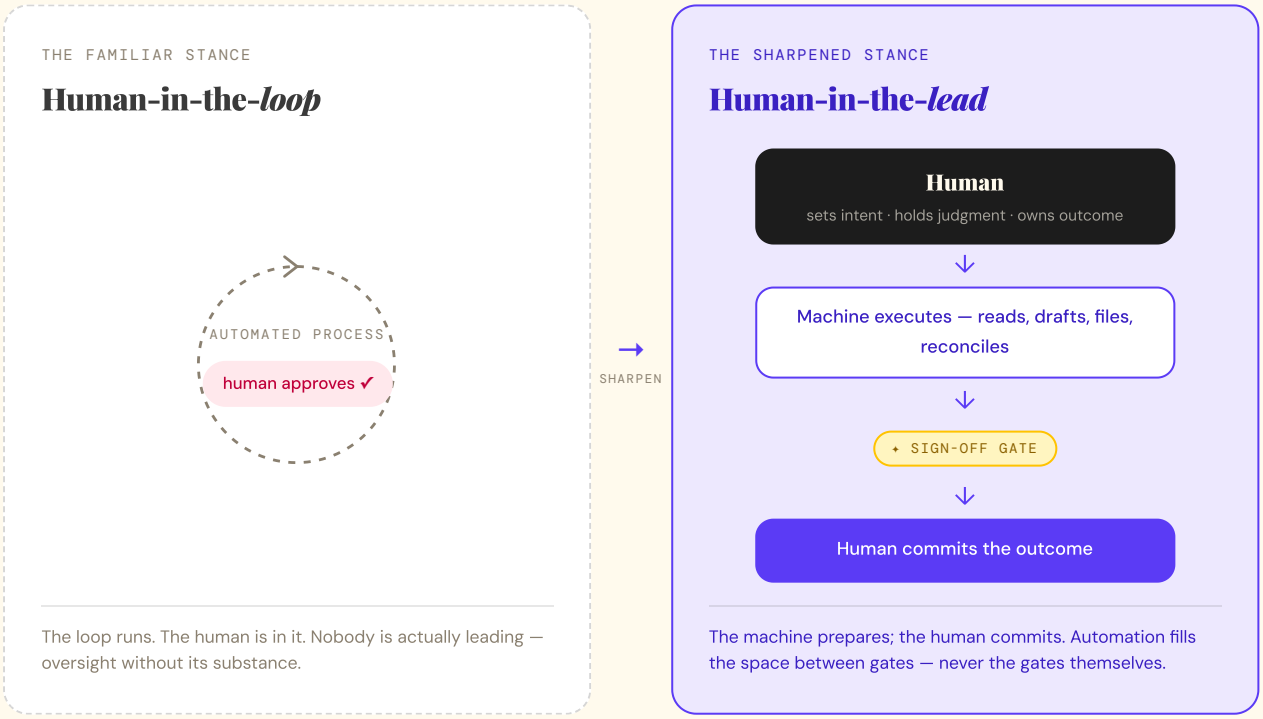
Human-in-the-loop describes a person positioned somewhere in an automated process, available to intervene. In practice it has decayed into something thinner: a human approving outputs they did not shape and often do not understand, a checkbox that creates the appearance of oversight without its substance. The loop runs; the human is in it; nobody is actually leading.

Human-in-the-lead inverts the emphasis. The human is not a checkpoint inside the machine's process. The machine is an instrument inside the human's process. The human sets the intent, defines what good looks like, holds the judgment calls, and owns the outcome. The machine carries the load that judgment doesn't require. The reading. The filing. The tireless, thankless part. The division of labor is not "machine works, human approves." It is "human leads, machine executes."

FIGURE 3 · THE THESIS

Human-in-the-loop vs. *Human-in-the-lead*

The same two actors, the emphasis inverted. On the left the human is a checkpoint inside the machine's process. On the right the machine is an instrument inside the human's.



Creat✦ism

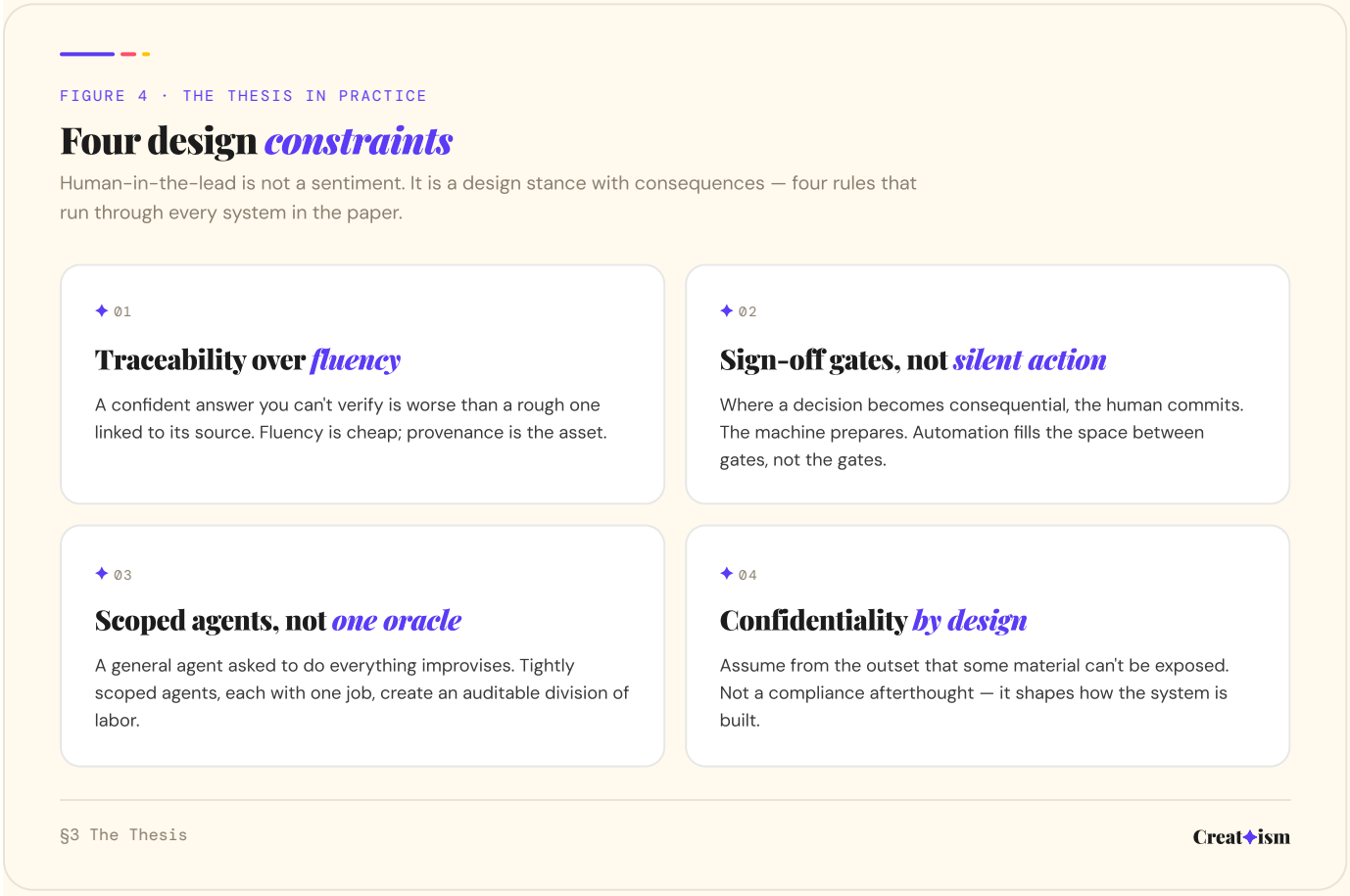
This is not a sentimental preference for keeping people involved. It is a design stance with concrete consequences. It produces four design constraints that run through every system in this paper:

Traceability over fluency. A system that gives a confident, well-written answer you cannot verify is worse than one that gives a rougher answer linked to its source. Every claim should be traceable to the document, the actor, and the moment it came from. Fluency is cheap; provenance is the asset.

Sign-off gates, not silent action. The points in a workflow where a decision becomes consequential — a commitment is made, a document goes out, a position is taken in a negotiation — are explicit human gates. The machine prepares; the human commits. Automation fills the space between gates, not the gates themselves.

Scoped agents, not a single oracle. A general-purpose agent asked to do everything tends to lose context and improvise. A set of tightly scoped agents, each with one job and clear boundaries, creates natural checks and an auditable division of labor. This is the same logic that makes a team of specialists more trustworthy than a single overconfident generalist.

Confidentiality by design. In professional settings the structure must assume from the outset that some material cannot be exposed. Confidentiality shapes how the system gets built. Bolt it on at the end and it never holds — so I treat it as a first-class design problem in Section 7.



We have learned, almost overnight, to brief a machine with a care we have never once given a human colleague. We will spend twenty minutes crafting a precise prompt — supplying context, intent, audience, the definition of success — and then spend thirty seconds briefing a human colleague and wonder why the work comes back wrong. I have been caught on the other side of this too — letting a machine's fluency stand in for thinking I had not actually done, handing over a confident draft as if confidence were the same as judgment. It is the consultant's oldest party trick in a new costume. The machine forced the discipline on us because, unlike a person, it will not fill in the gaps. *The*

quality of the output is mostly a function of the quality of the input, and the input is our responsibility. That sentence is the whole of prompt engineering, and it is also the whole of good leadership. Human-in-the-lead is what happens when you take that discipline seriously in both directions at once: you bring to your machines the clarity of a good brief, and you bring back to your people the clarity you were forced to learn for your machines.

SECTION 4

Build the Structure You Trust *(Creatism)*

There is a reason this paper insists that the operating system be *built*, not adopted. It connects to a body of work I have been developing under the name **Creatism**.

Creatism begins from a simple, almost uncomfortable observation about modern professional life: most of us are professional *consumers*. We consume frameworks, tools, methodologies, other people's systems, an endless feed of content — and we produce almost nothing that is actually ours. The ratio of what we consume to what we create has inverted, and our sense of agency went with it. The corrective is not to consume better. It is to create more than you consume, and to make the things you depend on rather than renting them from whoever marketed them most effectively.

Applied to AI operating systems, Creatism explains *why adopting someone else's system wholesale tends to fail* — and it has nothing to do with how good the system is. A borrowed structure is one you never reasoned your way into. You don't know why the folders are arranged that way, what the rules are protecting against, where the system bends and where it breaks. So when it needs maintenance, you can't maintain it. When it produces something odd, you can't diagnose it. When you should override it, you don't trust your own judgment over its apparent authority. You end up serving the system instead of leading it — which is precisely the failure human-in-the-lead is meant to prevent.

This is also the sharpest critique of the LLM-as-maintainer pattern, and it deserves an honest answer. If the machine does all the filing, summarizing, and cross-referencing, does the human still *understand* their own knowledge — or do they end up with a comprehensive library they have never actually read? The bookkeeping, after all, is where a good deal of understanding forms. The risk is real. Creatism's answer is that the human's creative work moves up a level rather than disappearing: you stop doing the manual filing, but you take on the design of the *structure* that governs the filing — the rules, the schema, the definitions of what matters and what connects to what. You write the constitution; the machine executes it. The understanding lives in the structure you authored, and the structure is something you can read, defend, and change. This is why the constitution file at the heart of these systems is not configuration. It is the artifact where your judgment is encoded, and writing it is the single highest-leverage act in building one of these systems.

There is one more thread from Creatism that belongs here, because the problems that make AI worth deploying are rarely the tidy ones. The challenges that matter most in organizational life — transformation, adoption, cultural change, ambiguity — do not sit still and do not yield to a purely linear, analytical attack. They require *associative* thinking: the lateral, pattern-finding, connection-making mode that surfaces problems before they are well defined. Here is the productive division of labor this whole paper is, in a sense, an argument for: the machine is extraordinary at the analytical, sequential, exhaustive work, and weak at the associative leap. The human is the reverse. Human-in-the-lead is not a constraint we accept reluctantly because the machines aren't good enough yet. It is the correct allocation of two different kinds of intelligence to the kinds of problems each is actually built for.

SECTION 5

From Personal to Organizational

The claim that makes this paper more than another productivity essay is that the *same* principles scale. The system that organizes one person's work and the system that governs a large, multi-party program are not different in kind. They differ in stakes, in the number of stakeholders, and in the cost of error — but the architecture is continuous.

To make this concrete, consider a real deployment, described here at the level of pattern rather than identity: an AI-native operating system built to govern a large-scale public-sector modernization program, delivered by a multi-party consortium under a multilaterally financed initiative. The kind of program where hundreds of contractual requirements move across dozens of documents authored by multiple parties over a span of years, where commitments made in one meeting contradict commitments made in another, and where the difference between a defensible position and an expensive one comes down to whether you can trace exactly what was agreed, by whom, when, and in response to what.

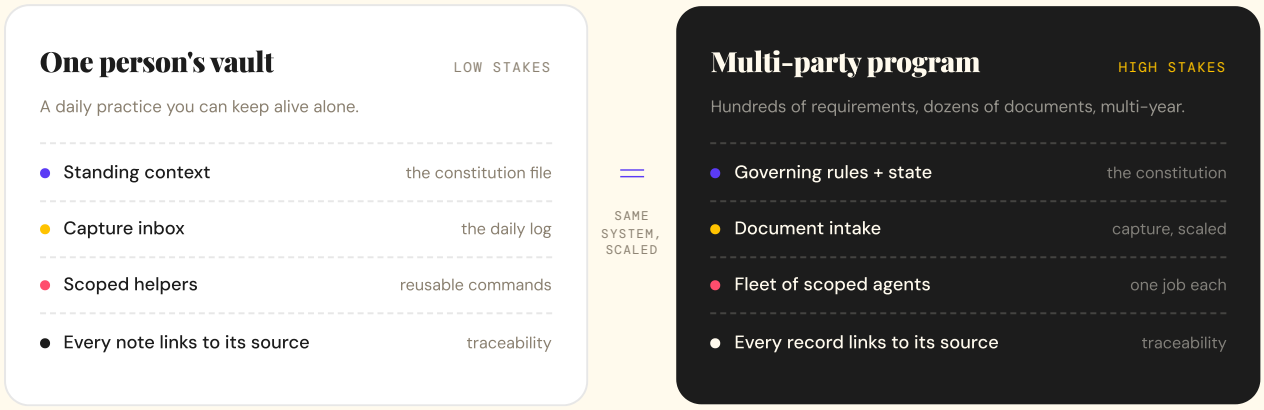
> **The shape, not the fingerprint** *(sanitized scale)*
>
> Hundreds of distinct contractual requirements · dozens of source documents authored by multiple parties · a delivery horizon measured in years · a governance fleet of a handful of single-purpose agents · one relational backbone in which every record links back to its source. No names, sector, program, consortium, or figures — only the order of magnitude, because the structure is the transferable lesson and the specifics are the liability.

The system built to govern it rests on the same layers any personal operating system has, scaled up:

FIGURE 5 · SCALE

Same architecture, *higher stakes*

A solo professional's vault and the governance backbone of a multi-party program are not different in kind. They differ in stakes, stakeholders, and the cost of error — the layers are continuous.



A relational backbone instead of a pile of notes. At the center sits a structured relational database — not a folder of documents, but a graph in which requirements, documents, deliverables, actors, commitments, risks, contradictions, and dates are all distinct, linked records. The design choice that matters most: *every record links to its source document, the actor who authored or signed it, the requirement or milestone it touches, and — where applicable — the contradiction it creates with another source.* That single rule is traceability-over-fluency made architectural. It is what lets the system answer not just "what is the status" but "how do we know, and where is the proof."

A fleet of scoped agents instead of one oracle. Rather than a single assistant asked to do everything, the system runs a small set of tightly scoped agents, each with one job: one reads incoming documents and files them; one extracts contractual requirements and flags when their state has shifted across versions; one captures commitments and action items with their owner and source; one detects what changed when a new document arrives; one watches for risk language and reconciles it against the existing register; one orchestrates the others into a periodic digest. Each is auditable. Each can be reasoned about. None is trusted to act unsupervised at a consequential gate. This is scoped-agents-not-an-oracle made operational.

Human gates at every consequential point. The system prepares negotiation positions, drafts briefings, surfaces contradictions, and assembles the program record. It does not *decide*. The human leads the negotiation, signs the communication, takes the position. The machine's job is to ensure that when the human does decide, they do so with the full, traceable picture in front of them — and that the decision, once made, is recorded against its source. This is human-in-the-lead at the scale where getting it wrong has consequences measured in millions and in institutional trust.

The continuity is the point. A solo professional's vault — with its standing context file, its capture inbox, its scheduled review, and its rule that every note links back to where it came from — *is the same system* as the program governance backbone, just without the stakes that force the discipline

to be explicit. Building the personal version well is how you earn the right, and the understanding, to build the organizational one. So before you scope an enterprise deployment, look at how you run your own week: a system you can't keep alive for one person will not survive contact with forty.

SECTION 6

The Operating Cadence

A structure without a rhythm goes stale. The community's agile-ceremony reframe — that planning rituals exist to solve context transfer — turns out to be the most practical way to keep an AI-augmented system alive, whether it serves one person or a program.

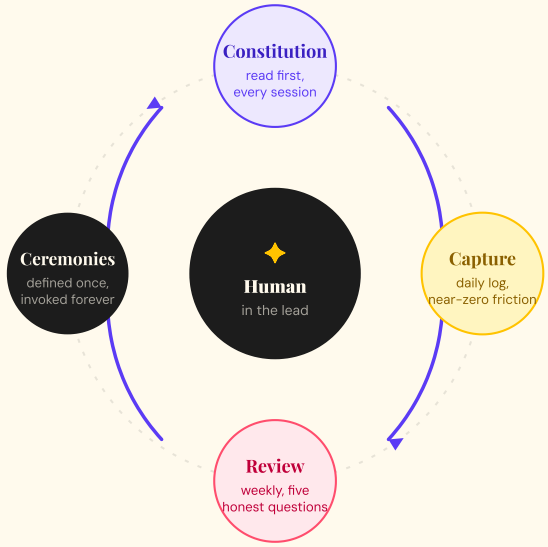
The reframe is this. Every agile ceremony maps onto a context-transfer function that an AI teammate needs just as much as a human one. A briefing document is a standing context file you hand a fresh session. Backlog refinement is breaking a large goal into pieces small enough that one worker — human or machine — can hold the whole thing at once. A planning session is the moment you decide, with intent, what the next stretch of work is actually for. A review is where estimated meets actual and you learn. A retrospective is where the system improves itself.

For knowledge work rather than software delivery, I run a lightweight version of this cadence — and I run it on myself, daily, which is the only reason I trust it enough to describe. It is the same Bujo discipline from Section 1, with the maintenance ceiling handed to the machine.

FIGURE 6 · THE HOW

The operating *cadence*

A structure without a rhythm goes stale. Four ceremonies — each encoded as a command the system runs — keep an AI-augmented system alive, with the human leading at the center.



> *"A ritual you have to remember is one you will eventually skip; a ritual that is a command is one the system can hold for you."*

- **Constitution** — the human-authored file that ends the stranger-every-morning problem.
- **Capture** — the human writes in; the machine reads and produces *separately*.
- **Review** — the machine assembles; the human decides, defers, commits.
- **Ceremonies** — the ritual stops needing willpower and becomes something the system runs.

§6 The Operating Cadence

Creat✦ism

A standing constitution the system reads before every session. In my own setup this is a pair of files — one that defines the operating architecture and its hard rules, one that holds nothing but current state: this week's small number of commitments, what's being carried, what's active. Every session

opens by reading them, so the model is never the stranger-every-morning. This is the artifact Creatism says you must write yourself; mine has been rewritten more times than anything else in the system.

A capture discipline that lowers the friction of getting things into the system to near zero. In practice this is one file per day — a single daily log that any working session appends to, including by voice when I'm away from the keyboard. The rule that protects this layer, learned the hard way: the human writes into the system; the machine reads and produces *separately*, so the system's picture of my thinking always reflects my thinking, not its own prior output fed back to itself. Without that day's log, the rest of the cadence has nothing to operate on — which is exactly why the system flags a day that opened without one.

A periodic review — a scheduled, semi-autonomous weekly retrospective. The machine reads the week's logs, the current state file, and the open client and content threads, then answers a fixed, short set of questions: which commitments shipped and which carried, whether the operator's anchor held, which carried items have been deferred too many times, what pattern is showing up. It assembles; it does not conclude. I then lead the review — decide, defer with intent, commit the next week's small handful of priorities — and the machine writes the refreshed state back. The format used to be a sprawling essay and it killed three consecutive reviews; cutting it to five honest questions is what made it survive.

Ceremonies encoded as reusable commands. The open-the-day ritual, the log-a-session pass when I switch context, the close-the-day reflection, the weekly review — each is defined once and then invoked by name forever. The ceremony stops being something I summon willpower for and becomes something the system runs. This is the whole trick: a ritual you have to remember is a ritual you will eventually skip; a ritual that is a command is one the system can hold for you.

Two honest cautions about borrowing agile wholesale. First, most agile tooling is built for shipping code — pull requests, test suites, merges — and that machinery mostly does not transfer to consulting or knowledge deliverables; what transfers is the *philosophy* of context transfer, not the plumbing. Second, estimation gets strange when the worker is a machine that does in minutes what used to take a day; the practitioners who keep estimation repurpose it to size their own *review and decision* load, not the machine's effort. And there is a standing risk, which Creatism names directly: a cadence can become productivity theater, an elaborate ritual that substitutes for the work. The test is simple and unforgiving — if you are spending more time tending the system than the system saves you, the system has become the work, and it is time to simplify.

SECTION 7

Honest Limitations

A practitioner document that only sells the upside is marketing. The systems described here have real and sometimes serious limitations, and confronting them directly is what separates a trustworthy account from a pitch.

Security is the headline risk, and it is not theoretical. The moment you give an agent broad access to your data and your tools, you create an attack surface. Prompt injection — malicious instructions hidden in content the agent reads — can turn a helpful assistant into a vector for data exfiltration or unwanted action. The always-on, broadly-connected personal agents that have captured the most attention this past year are precisely the ones security researchers warn about most sharply. The mitigations are not exotic but they are non-negotiable: isolate sensitive material, grant the narrowest access that works, start agents read-only, keep consequential action behind human gates, and never wire an agent simultaneously to your most sensitive data and the open internet without deliberate safeguards. Human-in-the-lead is, among other things, a security posture.

Hallucination does not go away; it gets managed. A model will state a plausible falsehood with the same confidence it states a fact. The structural answer is the traceability rule: a system that links every claim to its source lets you verify rather than trust, and a system designed to say "I don't have enough to answer that" is more valuable than one that always answers. But the residual risk is real, and it is why the human leads.

Over-engineering is the most common failure, and the easiest to hide. The seductive trap is building an elaborate structure before you have the habit that fills it — five folders deep on day one, never written in by week three. The discipline is to start smaller than feels satisfying and let the structure earn its complexity. I know this failure from the inside: I have built the elegant five-folder scaffold on a Sunday and watched it sit empty by the third week. The machine should carry maintenance burden; the human should resist adding structure faster than use justifies it. If you have ever done the same, the fix is not more ambition — it is less.

Building in the open while honoring confidentiality is a discipline, not a disclaimer. This deserves more than a line, because it is where my own practice has had to be most careful, and because it is a methodology others can use. The principle is to separate the *transferable pattern* from the *identifying instance*. The architecture, the rules, the agent design, the schema shape — these are shareable and are the substance of this paper and the tools that accompany it. The names, the numbers, the component taxonomy, the specific document set — these stay closed, because in a specialized domain even a structural detail like a module naming convention can act as a fingerprint to someone who knows the territory. The working test, before anything is published: *would this survive a client's legal team reading it?* If a sentence is only interesting because of the specific named entity behind it, it is too specific to publish. Drafting with explicit placeholders, rather than hoping to catch identifiers later, makes the discipline deliberate and reviewable. This is not a constraint on the work; done well, it is a demonstration of exactly the judgment a client is paying for.

SECTION 8 · TECHNICAL APPENDIX

The Tools

For builders. A decision-maker can stop at the previous section with the full argument intact.

The principles in this paper are instantiated in a set of open tools, generalized from real deployments and carrying none of their specifics. They are offered in the spirit of Creatism: not a system to adopt wholesale, but a set of well-made parts to reason your way through and make your own. The recommended path is to read them, understand why each piece is shaped the way it is, and rebuild them to fit your own work — because a structure you reconstruct is one you can lead.

The Constitution Template. The standing context file (and its companions for scoped sub-domains) that turns a general model into a disciplined operator of *your* system. It encodes identity, structure, the session-open and session-close protocol, voice and standards, and — most importantly — the hard rules the system must never break. This is the artifact Section 4 argues you must author yourself; the template is a scaffold to author against, not a finished answer.

The Cadence Toolkit. The ceremonies of Section 6, encoded as reusable commands — planning, review, retrospective, inbox-processing — together with a markdown task-card schema and a reference pattern for a backlog the agent can actually call rather than one it forgets exists. The hard-won lesson built into it: if you want the system to use something reliably, make it something the system can invoke directly, not a passive file it is supposed to remember.

The Reference Vault Structure. The folder skeleton — the immutable raw-source layer, the machine-maintained working layer, the periodic-notes and capture conventions, the traceability rules — that the other two tools operate inside. A starting structure, deliberately minimal, designed to earn complexity through use rather than impose it on day one.

These tools are being released progressively, one working part at a time, through Creatism — announced there as each becomes available.

CLOSING

The promise of AI in knowledge work is not that it will think for us. It is that it will carry the work that judgment doesn't require, and in doing so give judgment more room. That promise is only kept when a human stays in the lead — setting the intent, holding the trust, owning the outcome — and when the structure that makes it all work is one the human built, understands, and can defend.

Create more than you consume.
Build the structure you trust.
Lead the machine; don't follow it.

— Diego Bonifacino · The New Rhythm

✦ CONNECT

Creatism	creatism.substack.com
The New Rhythm	thenewrhythm.com
LinkedIn	linkedin.com/in/diego-bonifacino

diego@thenewrhythm.com · San José, Costa Rica