TBSO · OPENSKL · 001

The Agentic Enterprise OS

A field guide for the next operating model for companies and the five hard problems that will decide whether it works.

ÉRIC LARCHEVÊQUE	TBSO · OPENSKL	MAY 2026	V1.0

§00

EXECUTIVE SUMMARY

▲ INTRO

Éric Larchevêque is a French entrepreneur, investor, and engineer, best known as the cofounder of Ledger and TBSO. His perspective on agentic enterprise systems is grounded in three decades of building, scaling, investing in, and operating technology companies.

Most small companies run on human clicks across software. Your team spends hours switching between Gmail, Slack, Notion, HubSpot, and Stripe. Instead of making decisions, serving customers, or building products, they *navigate interfaces* carrying context between tools that can't see each other.

You know AI should help. But after automating a few workflows, or using your favorite chatbot, you hit a wall. You get brittle integrations, cannot find anyone in-house who can maintain them, and you don't have clear expertise on what to do next. The real problem is also that your company's knowledge is scattered across twenty surfaces with no single version of the truth.

> This paper describes the **Agentic Enterprise OS**: a single control surface where you express intent, agents execute across your tools, and the company's memory stays unified and governed. SaaS tools don't disappear; they move into the background.

Pieces of it are already in production at large enterprises (Salesforce, Microsoft, Atos...). But no one has made it real for a 1-50 person company yet, and only a few have addressed the hard problems that determine whether it works or becomes another failed AI pilot.

This paper focuses on five of those problems:

01	**Memory** Your company doesn't have a single source of truth. Different tools own different facts. Agents can't act correctly without knowing *which source wins* for *which field*, and *when*.
02	**Security** A single cockpit is also a universal attack surface. One poisoned email can trigger actions across every connected system unless containment is designed in.
03	**Latency & UX** A chatbot can't run a company. A full cockpit must handle monitoring, execution, and urgent interrupts simultaneously.
04	**Human oversight** Approving every agent action creates fake governance. Real oversight requires tiered autonomy that learns from every human decision.
05	**Migration** The transition itself is the challenge. Running old tools and the new OS may simultaneously create *institutional hallucination*, two competing versions of reality.

We also acknowledge what this paper is *not*. It is not a product spec, not a vendor-neutral analysis, and not a guarantee. TBSO is building toward this vision through its openskl.ai initiative. This paper is our honest assessment of what must be true for it to work.

§01

THE UNIFIED AGENTIC COCKPIT AS CONTROL ROOM

▲ THESIS

From interface labor to supervised execution

Your day probably looks something like this: email for a customer request, CRM to check their history, Slack to ask your team about pricing, back to email to send a quote, then a task created somewhere so someone follows up. Maybe a spreadsheet update. Maybe a mental note that gets lost.

None of this is hard. All of it is time-consuming. The work isn't thinking, it's *navigating*.

The Agentic Enterprise OS replaces this with a **cockpit**. Imagine one screen where you see what matters, tell the system what you want, review what it proposes, and approve what it does. The CRM, billing system, email, and project tool still exist, but transition into background services called by the cockpit.

To understand exactly what this means, let's consider a basic scenario where a contact form is filled requesting a quote for a product in an SMB environment.

MODE 01 · TODAY

1. Interface labor in a typical small business

In most small companies today, that request lands in an inbox or CRM queue. A human then:

- Opens the form submission email or CRM record.
- Checks which product and options were selected.
- Looks up the prospect in the CRM (if there is one) to see if they are an existing customer.
- Switches to email to ask for missing information or send a pricing grid.
- Pings the team in Slack (or calls a colleague) to ask what kind of discount can be offered

- Someone remembers this prospect had a prior discussion with the owner about a specific situation.
- A new email is drafted with an amended proposition reflecting those previous exchanges
- A spreadsheet or CRM field is updated.
- A task is created in a project tool to make sure someone follows up, or a mental note is made and hoped for later.

Nothing in this sequence is conceptually hard. The work is navigating five tools and remembering what to do next. The "process" lives in someone's head and in scattered checklists. The prospect gets the best attention possible and the proposition is drafted with the most up to date information, but it took a lot of time and energy.

MODE 02 · BOLTED-ON AUTOMATION

2. Deterministic AI automation bolted onto SaaS

An agency or internal builder would typically do this:

- The form submission triggers an automation that creates or updates a contact and a "Quote request" deal in the CRM with basic fields filled in.
- The workflow queries the CRM to check whether the prospect already exists and sets a simple "new vs existing" flag.
- Based on hard-coded rules (product type, deal size, geography), it selects a pricing template and discount band.
- An AI agent is called with that template and a few fields to draft an email to the prospect, inserting name, product, and price into a standard pattern.
- The system sends the email via Gmail or the CRM's email integration and posts a notification into a sales Slack channel.
- A follow-up task is created automatically in the CRM or project tool with a due date and owner.
  This removes a lot of clicking but the automation still only sees the form and a handful of CRM fields; it cannot incorporate side conversations,

special promises, or account nuance without being re-wired every time the real process changes.

MODE 03 · AGENTIC OS

3. Agentic Enterprise OS: intent, memory, and supervised execution

In an Agentic Enterprise OS, the same contact form submission becomes the trigger for an intent-driven, supervised sequence rather than a fixed workflow. It looks more like this:

- The form submission is ingested as an event in the cockpit. An orchestrator agent decides which subagent is best suited to manage it.
- The chosen subagent queries company memory across systems: whether this is a new or existing customer, current contract value and payment history, recent support tickets, open opportunities, and any prior conversations in email, Slack, or WhatsApp that mention this account or product.
- Using that context and the company's written pricing and discount policies, all updated to the latest internal decisions, the agent assembles a proposed plan. How to classify the opportunity, what price and discount band fits both policy and history, what risks or exceptions should be flagged, which internal owner should take it, and which follow-up tasks are needed.
- The cockpit shows this plan as a structured card proposing classification, key context pulled from memory, suggested quote, and a drafted message to the prospect.
- The sales owner can adjust the quote or message, see which prior decisions the agent relied on, and approve. The agent then executes the allowed steps across CRM, email, and task systems.
- The company memory is automatically updated to reflect latest actions and changes.

In this approach, the work of handling a quote request stops being a chain of clicks and ad-hoc decisions, or a mostly hardwired deterministic flow, and

> becomes a governed, flexible recurring task, owned by the cockpit and supervised by humans.

Why This Is More Than an Agent Orchestrator

Five things make this different from classical agent orchestrators :

- **Scope** — Orchestrators automate one workflow or one domain. An OS runs sales, support, finance, and founder tasks on the same backbone.
- **Memory** — Orchestrators pull a few fields into context. An OS maintains governed company memory based on which system owns which fact and using temporal proofs.
- **Governance** — In an orchestrator, each agent has its own rules. In an OS, a central safety layer classifies every action and enforces human oversight uniformly.
- **Interface** — Most orchestrators run invisibly. The OS puts the cockpit front and center, where you work.
- **Adoption** — An orchestrator assumes someone knows what to automate. The OS observes your existing work and suggests where to start.

An agent orchestrator is one building block inside the Agentic Enterprise OS, but the OS is what turns a collection of flows into a coherent way to start, grow, and run an agentic company without requiring the founder to become an AI workflow engineer.

Why it must be a control room, not a chatbot

A company can't be run through chat. Enterprise work has three modes that need different UX:

- **Monitoring** — What changed? What needs attention? (Always running in the background)
- **Execution** — Do this. Show me the plan. Then do it. (Staged, deliberate)
- **Interrupts** — Stop. Something's wrong. (Fast, with rollback)

A single chat thread is too slow for monitoring, too ambiguous for execution, and too noisy for interrupts. The cockpit needs to feel more like a Bloomberg terminal than like ChatGPT.

The new role of SaaS System of Records

SaaS tools don't disappear, but they retreat to the background. CRMs, billing systems, wikis, and project trackers become headless infrastructure the cockpit queries and updates.

The company's primary interface becomes the cockpit, and SaaS applications become **infrastructure** the cockpit calls into, rather than destinations where humans spend their days.

§02

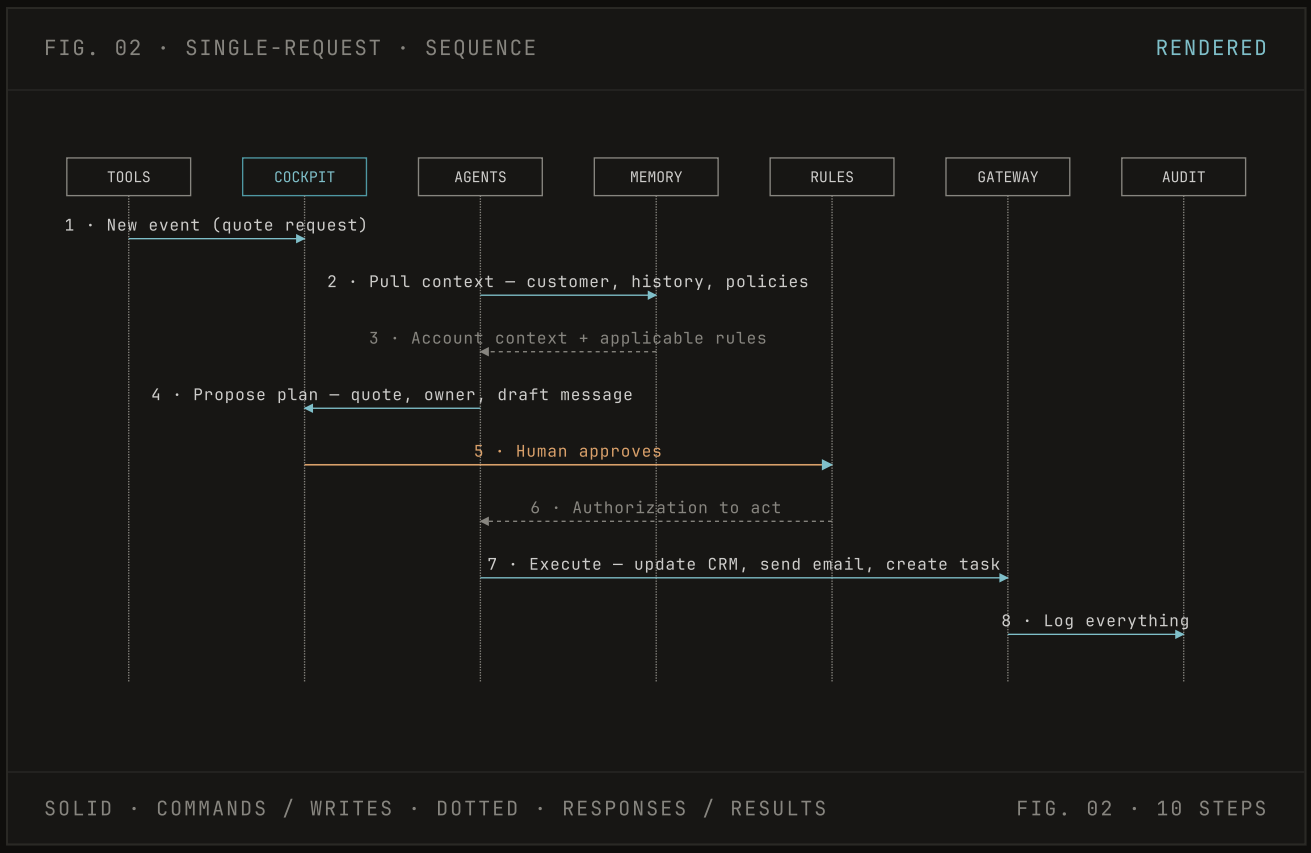
REFERENCE ARCHITECTURE

▲ MODEL

The Agentic Enterprise OS can be understood as seven interacting conceptual layers.

LAYER	CORE QUESTION	MAIN RESPONSIBILITY
Systems of Record	Where does operational data live today?	Existing SaaS, internal tools, databases, documents, and data platforms.
Integration Gateway	How do agents reach those systems safely and uniformly?	MCP servers, APIs, webhooks, connectors, adapters, authentication, and normalization.
Governance & Safety Kernel	What is allowed, by whom, under what conditions?	Policy engine, permissions, classification, delegation tokens, escalation rules, and human-in-the-loop routing.
Company Memory	What does the company know and has decided?	Facts, decisions, roles, authority, context, temporal validity, commitments, and institutional memory.
Observability & Audit	What did the system actually do, and why?	Temporal audit feed, traces, metrics, explanations, approvals, replay, and accountability.
Agent Runtime Layer	Who plans, reasons, and executes?	Planners, orchestrators, domain agents, tool-calling agents, task decomposition, execution, and recovery.
Cockpit	Where do humans see, decide, and intervene?	Control room for plans, approvals, exceptions, monitoring, delegation, and operational supervision.

Here is how a single request moves through the system:



What each layer does

Your existing tools (Systems of Record). Nothing moves, nothing migrates. HubSpot stays HubSpot. Gmail stays Gmail. Stripe stays Stripe. These systems keep doing what they do, which is storing contacts, emails, invoices, tasks. What changes is that you stop opening them directly to get work done. They become background services the cockpit reads from and writes to.

The connection layer (Integration Gateway). This is what lets agents reach your tools safely. It is like a secure switchboard where every agent request goes through it, making sure each agent only touches what it's supposed to touch, with credentials that expire after the task is done. No agent should ever hold a permanent key to your tools.

The rules engine (Governance & Safety Kernel). Before any agent takes an action, this layer asks: is this allowed? It classifies every proposed action from fully autonomous (write a draft, search memory) to requiring your explicit sign-off (send an external email, change pricing, trigger a payment). It also learns over time as actions you approve routinely without editing will gradually require less of your attention and become fully autonomous.

Company memory. This is the layer that makes the Agentic OS different from a chat bot or a workflow. It maps your company (people, projects, products...) and holds what your company actually knows (customers, deals, priorities, decisions, commitments, policies...) Contrary to most approaches, memory is not built as a pile of documents to search through but as a governed record where each important fact has a declared source, an owner, and a freshness date. This is probably the crown jewel of the system.

The audit trail (Observability & Audit). Every action the system takes is logged for future reference (what was done, why, which memory facts were used, who approved it, etc.) This is what lets you ask "what happened with this thing last Tuesday?" and get a precise answer. It's also what gives you a rollback path when something goes wrong, and the compliance evidence if anyone ever asks (useful to deflect blame and organize agent execution to appease auditors).

The agents themselves (Agent Runtime). This is where reasoning and execution happen. There is a planner that figures out what needs doing, specialized agents that handle specific areas like sales or support, and lightweight executors that carry out individual steps. They work in parallel where they can, check memory before acting, and escalate to you when they hit something they shouldn't decide alone.

The cockpit. In theory, the last screen you'll ever use for company work. This is where you see what's happening, tell the system what you want, review what it's proposing, and approve what it does. KPIs, state of your business, priorities, the list of things needing your attention, and the plans agents are about to execute (or are executing).

§03

HARD PROBLEM ONE

▲ PROBLEM-01

PROBLEM-01

MEMORY SOVEREIGNTY

Company memory should be about **truth governance**, not document retrieval.

Why "just connect all your data" fails

Consider the question "What did we promise this client?"

A naive RAG (Retrieval-Augmented Generation) system will attempt to synthesize an answer from:

- A signed contract in Drive.
- A Slack thread where a sales engineer promised a feature.
- A CRM note summarizing a call.
- An email thread negotiating a discount.
- A Notion page describing the standard plan.

The real problem isn't model hallucination, it is that the **company itself has unresolved truth**. There is no single source of truth because different fields are legitimately owned by different systems and roles.

Before your agents write anywhere, you need to answer four questions:

1. Who is allowed to assert which facts?
2. From which system or channel?
3. For how long are those assertions considered valid?
4. How are conflicts detected, surfaced, and resolved?

The four memory layers and their failure modes

In an agentic system, there are four distinct layers of memory, each with its own failure mode.

MEMORY LAYER	PURPOSE	TYPICAL FAILURE MODE
Working memory	Current task context and active reasoning state	Context window truncation, causing loss of relevant short-term context
Episodic memory	Past interactions, actions, and conversations	Summaries that lose detail or reflect the model's bias
Semantic memory	Company knowledge, facts, and reference information	Stale or conflicting information with no freshness or authority weighting
Procedural memory	Operational know-how and workflows ("how things are done")	Knowledge remaining implicit and undocumented, living only in people's heads

Agents cannot safely act on behalf of a company if these layers are not designed intentionally.

Decision memory and event sourcing

Companies don't run on data, they run on decisions. But decisions have never been made machine-readable. They are scattered across emails, meetings, and implicit conventions.

What is missing is a context graph that captures decisions and their reasons. One implementation pattern is event sourcing: you record decisions as append-only events and rebuild the current state from them.

This has two major implications:

- Facts cannot be overwritten; they can only be superseded.
- Every answer the cockpit gives can be traced back through a chain of proofs.

The memory constitution

Before deploying agents that write anywhere, a company must write a **memory constitution**.

At minimum, it should define:

- **Authority per field** — For each important fact (contract value, renewal date, payment status, health score, owner, etc.), specify the authoritative system and role.
- **Temporal rules** — For each field, specify freshness requirements, TTLs, and when old facts must be treated as stale.
- **Invalidation responsibilities** — Who is responsible for updating memory when a fact changes?
- **Conflict resolution** — How conflicts are surfaced and who resolves them (who is the memory steward?).

This is not only technical. It forces the company to decide which source is trusted for which fact. If that decision stays implicit, agents will automate the confusion.

§04

HARD PROBLEM TWO

▲ PROBLEM-02

PROBLEM-02

THE COCKPIT AS ATTACK SURFACE

A unified agentic cockpit is, by design, a single authorized pathway to every critical company system. Traditionally, an attacker had to break in to steal credentials, find a zero-day or move laterally across systems.
In an agentic OS, an attacker may only need to compromise the **input** the agent reads.

The trust bridge problem

When an agent can read low-trust inputs and trigger high-trust actions, its permissions become the attack surface. Examples:

- A crafted email with hidden instructions that an agent ingests and uses to update pricing.
- A poisoned Notion page with embedded prompts that steer an agent updating CRM records.
- A compromised MCP tool description that contains malicious meta-instructions executed at discovery time.

The cockpit is a **trust bridge** from low-trust surfaces (email, documents, tickets, external tools) to high-trust actions (payments, critical changes, even code pushed to production). If that bridge is not segmented and monitored, the cockpit becomes the largest lateral movement surface in the organization.

Emerging attack vectors

There are three active vectors already exploited in real MCP ecosystems:

- **Indirect prompt injection** in content the agent is supposed to read (emails, docs, messages).
- **Tool poisoning** in MCP server descriptions read during tool discovery, not logged in conversations.
- **Agent-to-agent lateral movement** via shared memory and context one agent trusts because another agent trusted it.

In 2026, the majority of MCP deployments have no enforced security boundary between agent and tools and only 8.5% of MCP servers use OAuth for authentication [01]; 73% of production AI deployments are vulnerable to prompt injection [02]. OpenAI has formally labeled prompt injection a "frontier, unsolved security problem" [03].

OAuth and the confused deputy

Most current MCP deployments use token passthrough. A user authorizes an agent with a broad OAuth token; the agent passes it to every server. This creates a classic **confused deputy** scenario: the agent acts with the user's full authority in contexts the user never intended. [04]

For instance, one agent may need access to Google Workspace so it can summarize emails, but because it receives the full OAuth token and passes it to every MCP server, a compromised tool can suddenly read confidential Drive documents and send emails on your behalf.

There are, however, emerging best practices:

- The agent is a distinct principal from the user.
- The agent receives a task-scoped delegation token, not the user's full credential.
- Tokens are short-lived and revocable, issued by the safety kernel, not by individual agents.

Universality vs segmentation: the structural paradox

The cockpit's value proposition is universality: one interface, all systems, all context. But at the same time, its security requires a full segmentation, meaning that no agent

can reach everything.

> This creates a paradox. A cockpit that is truly unified at the UI layer must be deeply segmented under the surface. Every connection must have a credential boundary, every action must get an explicit delegation, and every escalation is a structured, auditable handoff.

This presents a real challenge. The architecture must live with this contradiction and own it completely, rather than hand-waving it away with basic solutions like scoped agents.

§05

HARD PROBLEM THREE

▲ PROBLEM-03

PROBLEM-03

LATENCY AND CONTROL ROOM DESIGN

Latency in agentic systems is about **how time feels** to the user.

Three latencies

There are three kinds of latencies:

- **Technical latency** — processing time for LLM inference, tool calls, network trips.
- **Perceptual latency** — how fast the system feels, shaped by visible progress and responsiveness.
- **Cognitive latency** — how long the human must hold uncertainty about whether the system will do the right thing.

Technical optimizations (model speed, faster I/O) help, but they do not solve the cognitive problem. A frozen spinner during a critical sequence feels catastrophic, even if the result arrives in three seconds.

Conversational vs command: a false choice

Pure chat interfaces fail because enterprise work is not inherently conversational. Complex, parameterized instructions with constraints do not map cleanly to back-and-forth dialogue.

Pure command interfaces fail because enterprise intent is ambiguous and context-laden. Forcing users to learn a DSL (domain-specific language) to express "fix the deal with customer Y" pushes cognitive burden back onto humans.

The correct pattern is **intent-first, plan-driven**:

1. The user expresses intent in natural language.
2. The system interprets it, asks clarifying questions if needed and generates a structured execution plan with parameters, constraints, and decision points.
3. The user reviews, edits, and confirms.
4. The agent executes under that plan, with stage gates where needed.

Speculative execution and parallelism

Speculative execution is one of the most effective techniques for reducing perceived latency in agentic systems. While the user is reviewing a plan (often 5-15 seconds), the system pre-executes high-confidence dry-run sub-steps whose results are very likely to be needed regardless of minor plan edits.

When the user clicks *Confirm* some results are already available. If the plan is modified in ways that invalidate speculative work, that work is discarded. This is a direct borrow from CPU speculative execution applied to agent workflows.

Additionally, tool calls that do not depend on each other's outputs should never be serialized. Parallelizing them can halve perceived latency with no conceptual complexity.

§06

HARD PROBLEM FOUR

▲ PROBLEM-04

PROBLEM-04

HUMAN-IN-THE-LOOP THAT SCALES

HITL is often invoked as guardrails to keep AI agents under strict control and supervision. But without careful design, it becomes either a bottleneck or a fiction.

Approval fatigue and fake governance

Approval fatigue is a governance catastrophe. Here is what happens when humans receive too many approval prompts:

- They approve faster than they read.
- They treat approvals as a mechanical step.
- The audit log records approvals that were never substantively given.

This outcome is worse than full autonomy! The company gains neither efficiency nor real oversight. To address this, there are three common models for human oversight:

- **HITL (Human-in-the-Loop)** — the agent proposes actions, but execution is blocked until a human explicitly approves.
- **HOTL (Human-on-the-Loop)** — the agent executes autonomously within predefined boundaries, while humans supervise, monitor, and retain intervention authority.
- **HIC (Human-in-Command)** — humans define objectives, constraints, and governance, while the system operates continuously and autonomously unless escalation or override is required.

Four-tier action classification

The solution: classify actions by reversibility and how much damage if it goes wrong:

TIER	CHARACTERISTICS	APPROVAL POSTURE	EXAMPLES
Tier 1 Autonomous	Reversible, narrow scope	Execute and log	Draft internal emails, summarize documents, search internal memory
Tier 2 Notify	Reversible, moderate scope; real systems touched	Execute, surface for review	Update non-critical CRM fields, create tasks, schedule internal meetings
Tier 3 Approve	Partially reversible or sensitive	Stage for human approval	Send external emails, modify pricing on an account, create a new vendor
Tier 4 Governance	Irreversible or high blast radius	Multi-party sign-off, challenge-and-response	Payment execution, contract signature, access changes, data deletion

T3 and T4 actions must stay a small share of total actions. If they are not, the agent's scope is too broad and will bury the user under approval volume.

Challenge-and-response approvals

For T3 and T4 actions, simple Approve/Deny buttons encourage rubber-stamping. A better pattern is a structured checklist the approver must complete:

1. I confirm the intent of this action is what I authorized.
2. I confirm the data sources used are appropriate.
3. I confirm this action is within the agent's declared scope.
4. I understand which systems will be affected and the blast radius.
5. I know how to reverse this action if needed.

This takes 30-60 seconds, but that is appropriate for high-stakes actions.

HITL as learning signal

HITL systems fail when human oversight becomes repetitive friction rather than a learning mechanism.

The correct architecture treats every HITL decision as a training signal:

- Approvals without modification → candidates for T1/T2 promotion.
- Approvals with edits → reveal miscalibrations in the agent's reasoning or prompts.
- Denials → indicate scope definitions that are too broad or misaligned.
- Approved actions that later produce bad outcomes → signals that policy or risk models need revision.

This creates a **HITL flywheel**: every human review helps recalibrate the system, progressively reducing the number of actions that require user intervention.

§07

HARD PROBLEM FIVE

▲ PROBLEM-05

PROBLEM-05

MIGRATION IS THE PRODUCT

The gap between present-day SaaS-centric work and a working Agentic Enterprise OS is real and will create migration challenges.

Empirical data is discouraging

- IDC: 54% of AI proofs-of-concept never reach production. [05]
- MIT NANDA: GenAI pilot failure around 95%. [06]
- PwC CEO survey: 56% of CEOs see no financial impact from AI investments despite adoption. [07]
- Gartner: over 40% of agentic AI projects are predicted to be cancelled by 2027. [08]

However, small companies have a structural advantage. The founder decides, the team aligns in days not months, and the tool stack is 5-15 SaaS products, not 200.

The failure pattern doesn't have to apply if you adopt progressively and avoid one specific trap.

The one trap to avoid: contested reality

In a naive transition, both old interfaces (Notion, Salesforce, Gmail, Stripe) and the cockpit are live and can write to operational truth. This creates a contested reality where two systems are both authoritative in practice but neither formally primary.

A forecast built on partially migrated data can be both precise and wrong. A support response based on cockpit memory can contradict what someone set manually in the underlying tool five minutes earlier.

> The result is not hallucination in the classic LLM sense. It is **institutional hallucination**, multiple inconsistent truths written to systems by multiple humans and agents.

The fix is straightforward in principle: company memory must stay as close to real time as possible, and every important field must have a declared source of authority. If someone updates the CRM, the cockpit should know quickly; if Slack, email, and the CRM disagree, the system should know which source is allowed to decide.

The adoption pattern: connect, observe, act, expand

Migration to Agentic OS for a small team follows a natural rhythm:

01

Connect and ingest (days 1-2)

Plug in your existing tools (CRM, email, Slack, documents, billing...). The OS ingests your data and starts building the company memory. You change nothing about how you work yet.

02

Observe and ask (weeks 1-3).

Use the cockpit in read-only mode. Ask questions, surface conflicts, see what the system understands about your business. This is where you discover gaps, stale data, and contradictions. No execution risk, and opportunity to help the system improve its understanding of your company.

03

Pick one painful loop (week 3-4).

Ask the OS to select a domain that clearly hurts: inbound sales triage, customer follow-ups, invoice chasing, weekly reporting. Let the cockpit propose plans and execute low-risk actions autonomously, while you approve anything sensitive.

04

Trust and expand (month 2+).

As confidence builds, widen the scope. Let the OS add other domains, let the system handle more routine actions on its own, and gradually stop doing that work directly in your SaaS tools.

05

Tools become infrastructure.

At some point, you realize you haven't opened your CRM directly in two weeks. It still runs, it still holds data, but you and your team naturally do the work through the cockpit, not by opening the underlying tool.

The people side

In a large enterprise, migration is politically charged. It threatens middle-management roles, coordination empires, and information gatekeepers. In a small company, the friction is lighter but still real:

- The person who "owns the spreadsheet" may feel exposed when that knowledge becomes shared memory.
- Team members comfortable with their current routines may resist changing habits.
- If the owner doesn't visibly adopt the cockpit first, no one else will.

The fix is leading by example, starting with a domain that clearly saves everyone time, and making early wins visible fast. Once one loop works noticeably better through the cockpit, adoption tends to pull itself forward.

§08

LIMITATIONS AND OPEN QUESTIONS

▲ HONEST

The Agentic Enterprise OS is still destination, not something you can buy off the shelf today. There are still a number of limitations and open questions that need to be addressed.

What about the costs?

For a 10-person company running this daily, you may expect LLM inference costs of $200—$2,000/month depending on usage volume and model choice, excluding Agentic OS vendor fees and the SaaS subscriptions that, at least initially, are still required underneath the system.

At first glance, this can feel like an additional operational burden, especially for small businesses already saturated with software costs. The economic value is in compressing administrative overhead, reducing context switching, and allowing owners and operators to focus on higher-value work such as sales, execution, customer relationships, decision-making, and growth.

For many small business owners already overwhelmed by operational complexity and unsure how to transition into the AI era, this may therefore become a powerful technology investment with a positive ROI.

Can it work for everyone?

This model also doesn't work well for companies whose work is inherently non-procedural (pure creative agencies, early-stage R&D). It works best where there are repeating operational loops that currently live in human heads and scattered tools.

Privacy, GDPR, and employee trust

A cockpit that ingests email, Slack, documents, CRM records, billing data, and calendar events can quickly feel like surveillance if deployed carelessly. For European SMBs, this is not only a cultural concern but a legal one: GDPR imposes clear obligations around purpose limitation, data minimization, lawful basis, retention, and the right to erasure, none of which map cleanly onto an always-on agentic memory layer.

Before deployment, a company needs explicit answers to a small set of questions: what is ingested, what is excluded by default (private DMs, personal email, HR conversations), who can query what, how long memory is retained, how deletion requests propagate across the memory graph, and whether any data leaves the EU through model providers. Data processing agreements with LLM vendors, sub-processor chains, and cross-border transfer mechanisms all need to be worked out before the cockpit touches production data.

Reliability and rollback must be designed in

Agentic execution will fail. Agents will misread context, act on stale data, pick the wrong tool, trigger the right action at the wrong time, or chain several small mistakes into one large one.

A serious Agentic OS therefore needs rollback patterns from day one. Every executed action should be logged with enough context to be explained, undone, or compensated. Irreversible actions (payments, deletions, external communications, contract changes) need explicit escalation paths, dry-run modes, and manual override.

Liability remains unresolved

Most agentic deployments still sit on top of SaaS-era contracts written for passive software, not autonomous execution. If an agent misprices a product, sends a misleading customer message, or performs an action that causes damage, liability will often remain with the deploying company, even if the mistake originated in the agent's reasoning.

This is a major open question for the Agentic Enterprise OS. Companies will need clear internal accountability for agent outcomes, and vendors will need contracts that recognize agents as operational actors, not just software features. Until that legal model matures, high-impact actions should stay gated, auditable, and tied to explicit human approval.

State of the Market

We are not the only ones seeing this shift. Major players are already moving in this direction, each from a different angle:

- **Salesforce** is pushing the "Agentic Enterprise" through Agentforce 360, with Data 360 as context and Slack as a collaborative agentic work surface. [09]
- **Microsoft** is expanding Copilot Studio into multi-agent orchestration across Microsoft 365, Fabric, SDKs, and Agent-to-Agent protocols. [10]
- **ServiceNow** is building AI Control Tower capabilities to govern and monitor agent workforces. [11]
- **Palantir** has long framed AIP, Foundry, and Apollo as an operating system for AI-driven operations. [12]
- **SAP** and **Workday** are turning core systems of record into agent platforms and agent governance layers. [13]:
- **UiPath** is extending automation into an agentic control plane through Maestro. [14]
- **Atos**, in Europe, is framing agentic AI as sovereign production infrastructure through Sovereign Agentic Studios. [15]

These moves confirm the direction that agentic AI is becoming an operating model. But most approaches remain enterprise-led, ecosystem-bound, consulting-heavy, or tied to a specific system of record. The missing layer is still the founder-friendly version: one cockpit for small and medium businesses, working across the tools they already use, with memory, governance, and execution designed together.

§09

CONCLUSION

▲ END

For a small business owner, the real question isn't "should I use AI?" You already know you should. The question is *how to do it?*.

You face four choices:

01

Help one person at a time

You can give everyone access to general AI coworkers like ChatGPT, Claude, or Gemini. This is the easiest start: people draft emails, summarize documents, brainstorm, analyze spreadsheets, and move faster individually. But the range of possibilities is so wide that many business owners do not know where to begin, try a few disconnected use cases, and stop.

02

Make each tool smarter

You can keep buying SaaS tools with vendor agents. Your CRM gets smarter, your helpdesk gets smarter, your accounting tool gets smarter. But each agent remains trapped inside its own product.

03

Automate and connect existing tools

You can hire an agency to build integrations and specific AI workflows. This can remove painful manual steps like lead qualification, follow-ups, invoice chasing, CRM updates or reporting. But it often automates the existing mess instead of replacing it. The tools remain fragmented, ownership remains unclear, and company memory stays scattered across emails, docs, Slack, spreadsheets, and SaaS fields. The workflow may be useful, but the AI still does not really know the company; it only sees the narrow corridor the integration gives it.

04

Change the work surface itself

You can move toward an Agentic OS: one cockpit where the company is seen, understood, and operated.

We believe option 4 is where this is all heading. Not because it's easy (this paper has tried to be honest about how hard it is) but because the alternative is spending the next decade as the human glue between fifteen SaaS tabs.

[01] According to NimbleBrains analysis of the full MCP registry, published March 2026: "The State of MCP Security in 2026" https://nimblebrain.ai/mcp/mcp-security/state-of-mcp-security/ ↩

[02] According to Obsidian Security's analysis of production AI deployments, prompt injection appears in over 73% of enterprise AI deployments assessed. Obsidian Security, "Prompt Injection Attacks: The Most Common AI Exploit in 2025" https://www.obsidiansecurity.com/blog/prompt-injection ↩

[03] OpenAI CISO Dane Stuckey stated, in October 2025 at the launch of the ChatGPT Atlas browser: "prompt injection remains a frontier, unsolved security problem, and our adversaries will spend significant time and resources to find ways to make ChatGPT agent fall for these attacks." Source: Simon Willison, "Dane Stuckey (OpenAI CISO) on prompt injection risks" https://simonwillison.net/2025/Oct/22/openai-ciso-on-atlas/ ↩

[04] The confused deputy problem was originally described by Norm Hardy in "The Confused Deputy (or why capabilities might have been invented)" Operating Systems Review, ACM, October 1988. https://people.cs.vt.edu/~kafura/cs6204/Readings/ConfusedDeputy.pdf; Token passthrough is explicitly forbidden by the official MCP specification, which states: "If the MCP server not only accepts tokens with incorrect audiences but also forwards these unmodified tokens to downstream services, it can potentially cause the 'confused deputy' problem." Official MCP Authorization Specification, June 2025: https://modelcontextprotocol.io/specification/2025-06-18/basic/authorization; For implementation-level analysis of the vulnerability in the wild, see: Obsidian Security, "When MCP Meets OAuth: Common Pitfalls Leading to One-Click Account Takeover", February 2026. https://www.obsidiansecurity.com/blog/when-mcp-meets-oauth-common-pitfalls-leading-to-one-click-account-takeover ↩

[05] Lenovo CIO Playbook 2026 (4th annual edition), a study commissioned by Lenovo and conducted by IDC, published January 2026, surveying 3,000+ IT and business decision-makers globally. Found that only 46% of AI POCs have progressed to production-scale deployment. Lenovo Newsroom, January 2026. https://news.lenovo.com/pressroom/press-releases/research-reveals-ai-is-paying-off-cios-arent-ready/ ↩

[06] MIT Project NANDA (MIT Media Lab) published *The GenAI Divide: State of AI in Business 2025* in July 2025 (press coverage began August 2025), authored by Aditya Challapally, Chris Pease, Ramesh Raskar, and Pradyumna Chari. Methodology: systematic review of 300+ publicly disclosed AI initiatives, structured interviews with representatives from 52 organizations, and survey responses from 153 senior leaders across four major industry conferences. The

report's precise finding: 95% of enterprise AI solutions delivered no measurable P&L impact. Only 5% of custom enterprise AI tools reached production. Reported by Fortune, August 2025: https://fortune.com/2025/08/18/mit-report-95-percent-generative-ai-pilots-at-companies-failing-cfo/ ↩

[07] PwC's 29th Global CEO Survey, "Leading Through Uncertainty in the Age of AI" published January 19, 2026 at Davos. Conducted September 30 to November 10, 2025, with 4,454 CEOs across 95 countries and territories. Exact finding: "more than half (56%) say they've realized neither revenue nor cost benefits" from AI; "only one in eight (12%) report both" positive impacts. Full report: https://www.pwc.com/gx/en/ceo-survey/2026/pwc-ceo-survey-2026.pdf ↩

[08] Gartner press release, June 25, 2025. "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027." Anushree Verma, Senior Director Analyst, Gartner. https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027 ↩

[09] Salesforce launched Agentforce 360 in late 2025, positioning Slack as the primary front-end interface for its agentic AI ecosystem. Agentforce in Slack grounds agent responses in both Slack conversational data and Salesforce CRM data, enabling agents to take actions such as creating and updating channels, lists, and canvases directly in the flow of work. Sources: Salesforce EU, "Work with AI Agents in Slack using Agentforce" https://www.salesforce.com/eu/slack/agentforce/ ↩

[10] Microsoft Copilot Studio has expanded into multi-agent orchestration, with capabilities reaching general availability in April 2026. The platform enables agents built in Copilot Studio, Azure AI Foundry, Microsoft Fabric, and the Microsoft 365 Agents SDK to collaborate across workflows using open Agent-to-Agent (A2A) protocols. Source: Microsoft Copilot Blog, "What's new in Copilot Studio: Updates to multi-agent systems" https://www.microsoft.com/en-us/microsoft-copilot/blog/copilot-studio/new-and-improved-multi-agent-orchestration-connected-experiences ↩

[11] ServiceNow launched AI Control Tower at Knowledge 2025 (May 2025) as a centralized command center to govern, manage, secure, and realize value from any AI agent, model, and workflow. In May 2026, ServiceNow further expanded AI Control Tower through a deeper integration with Microsoft Agent 365. Sources: ServiceNow Newsroom, "ServiceNow launches AI Control Tower at Knowledge 2025"

[12] Palantir explicitly describes its combined AIP + Foundry + Apollo architecture as an "Enterprise Operating System." Foundry serves as the data operations platform, AIP as the generative AI platform, and Apollo as the continuous delivery platform. Source: Palantir, "Integrated platforms: AIP, Foundry, and Apollo" https://palantir.com/docs/foundry/architecture-center/platforms/ ↩

https://www.servicenow.com/uk/company/media/press-room/ai-control-tower-knowledge-25.html; ServiceNow Newsroom, "ServiceNow expands AI agent governance through deeper integration with Microsoft" https://newsroom.servicenow.com/press-releases/details/2026/ServiceNow-expands-AI-agent-governance-through-deeper-integration-wi ↩

[13] SAP has grown Joule to over 40 AI agents and 2,400+ skills embedded across S/4HANA, Ariba, SuccessFactors, and IBP, and launched an AI Agent Hub within SAP LeanIX for central agent governance. Workday announced its Agent System of Record in February 2025, providing a unified governance layer for managing both Workday and third-party AI agents. Sources: SAP News Center, "SAP Business AI: Release Highlights Q1 2026" https://news.sap.com/2026/04/sap-business-ai-release-highlights-q1-2026/; Workday Newsroom, "Workday Unveils New Agent System of Record" https://newsroom.workday.com/2025-02-11-The-Next-Generation-of-Workforce-Management-is-Here-Workday-Unveils-New-Agent-System-of ↩

[14] UiPath Maestro™ is described officially as "a cloud-native orchestration platform that unifies automation, AI agents, and human interactions into streamlined, end-to-end business processes" using BPMN (Business Process Model and Notation) and DMN (Decision Model and Notation) for visual workflow and rules modeling. https://www.uipath.com/platform/agentic-automation/agentic-orchestration ↩

[15] Atos Group launched Sovereign Agentic Studios on March 12, 2026, as a new operating model to help organizations move from agentic AI pilots to production under full sovereign control. Source: Atos Group Press Release, "Atos Group Launches Sovereign Agentic Studios" https://www.atosgroup.com/en/press/atos-group-launches-sovereign-agentic-studios-bring-ai-safely-production-across-organizations ↩