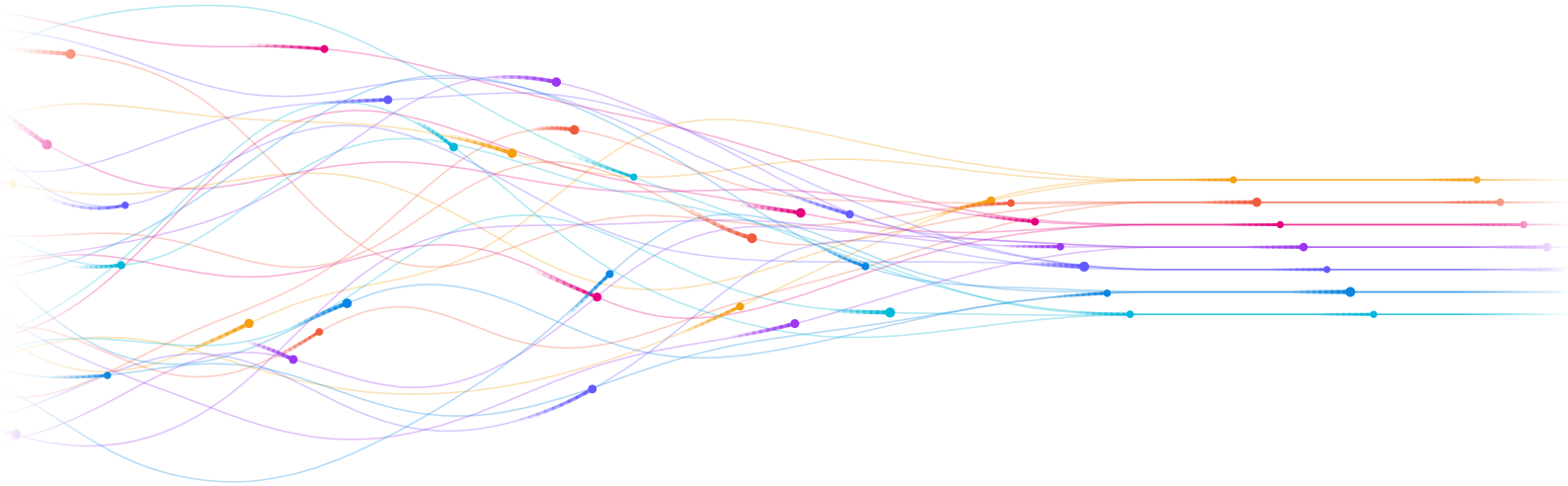


Essay

Integration Islands

Why the quietest problem in your software stack became the loudest limit on your AI.

27 July 2026 42 minute read Aditya Gaur

CONTENTS

I	The Founder on the Ferry Between Islands	2
II	The Ceiling Moved	7
III	How We Built an Archipelago	9
IV	Individual Islands, Defined	13
V	The Cost of Islands	17
VI	What “Data That Just Works” Feels Like	23
VII	Escaping the Archipelago	28
VIII	The Next Decade of Enterprise Infrastructure	32
Sources		35

The print edition of an essay published at adityagaur.xyz/writing/integration-islands. Its figures are interactive there; here they are set as still plates.

I. The Founder on the Ferry Between Islands

Software today looks sophisticated from the outside: a CRM here, an HRMS there, payroll, ERP, ticketing, analytics, and now an AI assistant promising to make sense of all of it. Up close, most of them feel like a collection of systems that each know a different version of the truth, and none of them know the truth in full.

For twenty years, that was a quiet, tolerable problem — a tax you paid in spreadsheets and reconciliation meetings, the kind of thing you delegated to "the data team" and tried not to think about. Then we handed the whole mess to AI and asked it to think for us, and the quiet problem became the loudest one in the building.

This piece is about that shift. We'll name the pattern — **Integration Islands**, made up of **Individual Islands** that each govern their own little world — put a number on the **Integration Tax** they create, and show why the thing we filed under "back-office data hygiene" has quietly become the ceiling on what AI can actually do for your company.

This is Aditya, writing another wondering on the weird things we humans have normalized.

Most modern company stories start with a login screen.

You're a founder with a small team, a few early customers, and a product that barely exists outside your head. No one is talking about "enterprise architecture." You just need things to work by Monday. So you start picking tools, one urgent decision at a time.

- Sales needs somewhere to put leads, so you spin up a **CRM**.
- You're hiring your first operations person, so you add an **HRMS**.
- You need to actually pay people, so you choose a **payroll** tool.
- Support is drowning in emails, so you grab a **ticketing** system.
- Finance wants clean books, so you connect accounting and maybe a lightweight **ERP**.

None of these feel like irreversible decisions. They're quick calls, usually made in a Slack thread.

> **“We just need something that works. We can always change it later.”**

You click through onboarding tours. The tools are slick, opinionated, self-contained. You connect a few “native integrations” while you’re at it, because the settings page makes it look effortless: *Click to connect CRM. Click to connect payroll. Click to sync tickets.*

For a while, it really does feel like magic. Leads show up in the CRM. Offers sync to payroll. Tickets flow into the helpdesk. Invoices appear in accounting. The dashboards look surprisingly grown-up for a company whose company-wide “data strategy” is still a two-page doc. Nothing is perfect, but nothing is obviously broken either. You feel sophisticated enough. You feel in control.

Then the company grows. Headcount doubles. Then doubles again. You add more SKUs, new countries, new billing models. The original structure of “one product, one customer type, one way of doing business” quietly dissolves. The tools that once felt breathable start to get cramped.

Sales has built a maze of custom fields and deal stages to model your real funnel. HR is running performance reviews in spreadsheets because the HRMS wasn’t built for it. Finance is spending more time in CSV exports than in the accounting system itself. Support is juggling three tools because chat, email, and in-app messages never quite landed in one place.

At the same time, your investors ask a simple question:

> **“How many active paying customers do we have, by segment, with full unit economics?”**

On the surface, that sounds like a one-slide answer. In practice, the number lives in five different places. The CRM knows which accounts are “Closed Won,” but some of them churned months ago. Billing knows who is still paying, but it doesn’t really understand segments or sales stages. The data warehouse has product usage, which doesn’t map cleanly to billing accounts. Finance has the revenue and margins, but grouped by legal entity and account code. Support has a list of organizations that feel like customers but aren’t always paying ones.

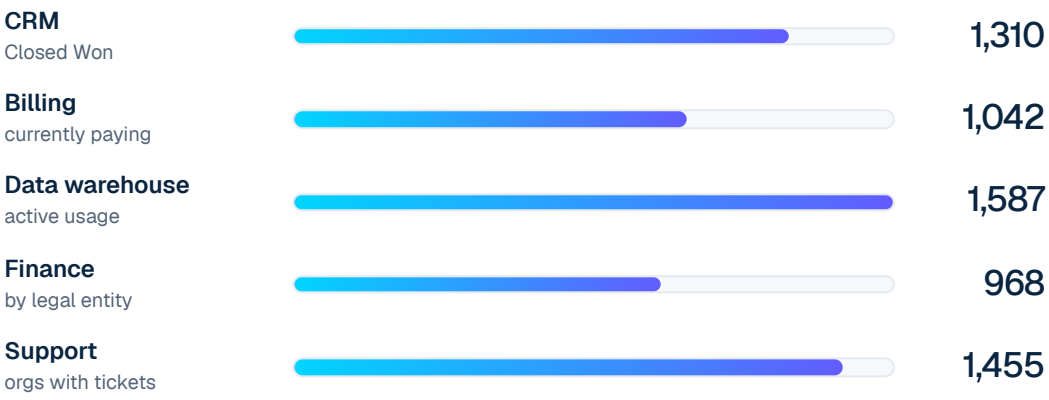
Everyone has a number. No one has *the* number.

Figure 1 Five systems, five answers

Interactive online →

YOUR INVESTOR ASKS

"How many active paying customers do we have, by segment, with full unit economics?"



Gap between answers	Highest vs lowest	Systems that agree
619	+64%	none

WHAT EACH SYSTEM MEANS BY "CUSTOMER"

CRM	An account whose most recent opportunity reached Closed Won. It has no idea whether that account is still paying — some of these churned months ago, and nothing in the sales pipeline records that.
Billing	A subscription in an active, non-delinquent state this cycle. Precise about money and blind to everything else: it doesn't model segments or sales stages, so it can't answer the second half of the question.
Data warehouse	Any organisation with product events in the trailing window. Generous by construction — it counts trials, sandboxes, and free seats, because usage is what it was built to measure.
Finance	A billed counterparty grouped by legal entity and account code. The most defensible number in an audit and the least useful in a board meeting, because one customer can be several entities and several customers can be one.
Support	An organisation that has raised a ticket. It knows about people who behave like customers, including ones who never paid and ones who left but still email.

One question, five systems, five answers. Each number is correct inside the system that produced it.

Illustrative figures, as specified in the draft.

So you do what every company does now. You buy an AI. The pitch is exactly the antidote you were hoping for: connect it to your systems, ask in plain English, get the answer no single dashboard could give you. You wire it into the CRM, the billing tool, the warehouse. You type the question your investor asked.

It answers in seconds. Confident. Well-formatted. A number, a segment breakdown, even a little commentary on the trend.

It is also wrong.

Figure 2 **The confident answer** Interactive online →



What the assistant returned. The disclosure is real, present, and easy to miss.

Not obviously wrong — that would be easy to catch. Quietly wrong. It read "Closed Won" in the CRM as "active," missed the churn that only billing knows about, and leaned on the warehouse's definition of "customer," which never quite matched either. It didn't lie. It did exactly what you'd do if someone handed you three of the five islands and told you to hurry.

The assistant didn't fail because the model was dumb. It failed because you asked it to reason about a company that doesn't exist in any one place — and unlike your RevOps lead, it didn't know enough to hedge. It just answered.

That is the moment the archipelago stops being a reporting nuisance and turns into something sharper. For years, the cost of Individual Islands was *your* time: the working group, the three spreadsheets, the board slide that looks precise but that

nobody fully trusts. Now it's the quality of every decision you hand to a machine that answers faster than you can check.

And that realization gets louder the moment you try to change something. Switch CRM? That means rewriting lead routing, integrations to billing and support, dashboards in BI, contracts in legal workflows. Upgrade the HRMS? Payroll, IT, finance, security, and legal all have to unwind years of ad hoc decisions that "we'll fix later." Implement a "proper" ERP? It feels less like a software project and more like a temporary reorganization of the entire company.

At some point, you notice you have stopped running one coherent system and started running a ferry service. You shuttle between a cluster of systems, each with its own idea of who your employees, customers, invoices, and obligations are. Each is internally consistent. None quite matches the others.

Your company is not one stack. It is a group of individual islands.

You did not plan it that way. Almost no one does. Yet here you are, captaining a small archipelago and wondering when everything quietly became this hard — right as the most important new tool you've ever bought turns out to be only as good as the map you can give it.

II. The Ceiling Moved

For the first time in computing, the scarce resource isn't the machine. It's what you put in front of it.

For most of computing history, the thing you couldn't buy your way past was the machine. A faster chip, more memory, a bigger disk. Software was written to fit inside whatever the hardware could do, and progress meant waiting for the hardware to get better. For the first time, that is no longer where the ceiling is.

The models can reason. Andrej Karpathy describes the new arrangement as a kind of operating system: the model is the CPU, and the context window is the RAM — the small, precious working memory that holds whatever the system is thinking about right now. Intelligence, in this picture, is increasingly abundant and cheap. What stays scarce is what you manage to load into that working memory at the moment it matters.

Strip the marketing from the word "agent" and you get something simple. An agent is a model that runs tools in a loop to achieve a goal: it reads from your systems, decides, acts, and reads again. Every turn of that loop, it has to pull the right context into the window. And here is the part every team building these things discovers the hard way — **more context is not better context.**

Anthropic, whose models sit behind a large share of enterprise assistants, now treats context engineering — curating the limited set of tokens a model actually sees — as the discipline that most determines how good an agent is, noting that quality depends "less on the model itself and more on how its context is structured and managed." Independent testing sharpens the warning. Chroma's context rot study found that all eighteen frontier models it tested degrade as the input grows — well before the context window is anywhere near full; a model advertised at 200,000 tokens starts losing the thread around fifty thousand. Stanford's "lost in the middle" work showed models reliably use information at the start and end of a long context and quietly miss what's buried in between.

Read those findings together and the intuition you started with flips. You cannot fix a fragmented company by hosing all of it into a giant context window and trusting the model to sort it out. Pour in more islands, unreconciled and contradictory, and the assistant gets *less* reliable, not more. Even in narrow technical domains, researchers building autonomous agents keep landing on the same conclusion —

one recent paper on agentic process design is titled, flatly, "Context is all you need."

So the bottleneck moved. Model → context. And "context," for a company, is not a clever prompt. It is a coherent, current, permissioned picture of your customers, employees, contracts, and money — the precise thing an archipelago is structurally unable to hand over.

You can watch this fail at scale

MIT's NANDA initiative reviewed more than 300 enterprise AI deployments and found that roughly 95% of generative-AI pilots deliver no measurable impact on the P&L. The reason they give is not model quality. It is a "learning gap" — tools that, in their words, "don't learn, integrate poorly, or match workflows." The failures cluster not at the model layer but at the seam where the model meets the business.

The analysts see the same wall from the budget side. Gartner expects more than 40% of agentic-AI projects to be cancelled by the end of 2027, citing escalating costs and unclear value — and warns of "agent washing," vendors repainting old automation as something autonomous.

> **The pilots aren't failing because the models can't think. They're failing because we ask them to think about a business whose own systems can't agree on what a customer is.**

It is tempting to read those numbers as "AI is overhyped." That is the wrong lesson, and an expensive one to act on. The model is ready. Your archipelago is not. Individual Islands withhold exactly the context an agent needs to be trustworthy — and unlike a slow quarterly report, a confident agent doesn't wait for you to notice. The rest of this piece is about how the islands formed, what they really cost, and what a company that actually wants working AI builds instead.

III. How We Built an Archipelago

Nobody set out to build islands. They accumulated, one reasonable decision at a time, across five eras.

No founder, no CIO, no architect ever stood at a whiteboard and said: let's split our understanding of the company across forty systems that don't agree with each other. The archipelago is not a decision. It is a *sediment* — the build-up of many small, sensible choices, each correct in its moment, that together produced a landscape no one designed. To see why it's so hard to undo, it helps to watch it form.

Era 1: The Monolith — One Big Island

Era 1: "the system" meant one room, one vendor, one database — and one version of the truth.

Before the browser tabs and the app zoo, there was the monolith. If you walked into a large enterprise in the '90s or early 2000s, "the system" usually meant one thing: the ERP. One vendor. One database. One version of the truth, running on hardware you could point at.

The promise was seductive — put everything in one place: finance, inventory, procurement, HR, payroll, sometimes even CRM. If you could shoehorn your business into the ERP's worldview, you got consistent data structures, end-to-end processes inside one system, and a single place to enforce controls. You weren't ferrying between islands. You were living on one giant, heavily fortified one.

The cost of that comfort was rigidity. Implementations took quarters or years. Customization meant stored procedures and ABAP, not clicking a toggle. If your business model changed faster than your ERP could, the ERP stopped looking like a nervous system and started looking like a concrete cast of how your company *used to* work. So when the cloud era arrived, almost everyone was primed to swing the other way.

Era 2: SaaS — The Island Explosion

Cloud and SaaS arrived with the opposite pitch. No servers, no installs. Try before you buy, pay monthly, cancel any time. Instead of one vendor claiming to do everything, you got a galaxy of specialists: the CRM that only did sales, the HR system obsessed with employee experience, the billing tool that lived and breathed subscription edge cases.

The logic was impeccable. Why suffer a clunky all-in-one when you could have the best tool for each job? So companies optimized locally. Sales picked the CRM they liked. HR picked the HRMS that suited them. Each decision made sense on its own. And each app did its job well.

But something was happening in the background: the number of core systems touching customers, employees, contracts, and money quietly exploded. It is no longer a vague "lots of tools." The 2026 MuleSoft Connectivity Benchmark puts the average organization at **957 applications** — up from 897 the year before — and finds that only **27% of them are connected to one another** (down from 29%). What used to live inside one ERP now lives across a thousand services, and roughly three-quarters of them don't talk. You didn't move from one big island to a few. You moved to a chain of hundreds.

957
applications in the average organization

27%
of them connected to one another

897
applications the year before

In theory, APIs were supposed to bridge the gaps — "integrates with everything" badges, marketplaces full of connectors. In practice, each app still brought its own assumptions about what a customer is, how a contract works, what "active" means. The more best-of-breed you added, the more vocabulary you had to reconcile. The first five apps feel like freedom. It's app number thirty, or seventy, where the pattern starts to feel like gravity — and by then the best-of-breed story has quietly morphed into something else. You haven't just stopped being a monolith. You've become an archipelago.

Era 3: The Modern Data Stack — A Parallel Archipelago

As SaaS spread, so did something else: data that refused to stay in one place. Executives wanted cross-tool reporting. Finance wanted reconciled numbers without juggling fifteen dashboards. The response was the “modern data stack” — pull data out of many systems into a central warehouse, with dedicated tools for ingestion, transformation, and visualization. ETL became ELT. “We’ll fix it in the warehouse” became the informal motto.

For analytics, this genuinely worked. You could finally see, in one place, a combined view of signups, product usage, revenue, and support load. But the warehouse was built for queries, not for running day-to-day operations. It knew your customers and employees in a statistical sense, not as live entities you could update and act on. It sat on the side, separate from the systems that actually sent emails, moved money, or assigned work.

And it brought its own sprawl: an ingestion tool, a warehouse, a transformation layer, a BI layer, maybe a catalog, a lineage system. You had created a *parallel* archipelago. Operational islands on one side — CRM, HRMS, payroll, ERP, ticketing. Analytical islands on the other — ingestion, warehouse, transformation, BI, catalog. The two traded exports and definitions, but they never shared a single, living model of your company. The warehouse was a mirror held up to a fragmented reality. It could describe that reality, sometimes beautifully. It couldn’t unify it. The modern data stack gave you better charts about your islands. It didn’t give you a mainland.

Era 4: Integration-as-a-Category — Connecting Islands With More Islands

As both sides grew more complex, someone had to own the bridges. If you were lucky, an early engineer quietly became “the integration person.” If you were less lucky, integrations accumulated wherever there was a free afternoon and a Zapier account. Eventually the pain got big enough that a whole new category appeared: integration platforms and workflow automation tools — iPaaS.

It was a powerful idea. Pre-built connectors instead of custom scripts. Drag-and-drop flows instead of logic hard-coded inside each tool. A central place to monitor and retry. But it introduced a new twist. The integration layer itself became critical infrastructure. Build dozens of flows in one platform and switching it becomes as daunting as switching any core system. Business logic that used to be trapped inside your CRM was now trapped inside someone else’s workflow engine. For every dependency you removed between two islands, you added a dependency on a new

one. And many integrations still encoded brittle, local assumptions: they knew how to move a field between two specific systems, but they **didn't embody a shared understanding of what that data meant to your business.** You had added a coordination layer. You hadn't added a coherent model.

Era 5: AI — The Stakes, Raised

Then AI arrived, and every major vendor shipped a copilot. We made the core argument in the previous section, so here it's enough to place AI in the lineage: it did not abolish any of this. It moved in.

The same islands, the same fragile bridges, the same Integration Tax — but now feeding a system that *acts* on what it's given, quickly, and sounds confident doing it. The collision is already visible in the same benchmark data. In the 2026 figures, 64% of IT leaders doubt they can meet their near-term AI goals specifically because of data fragmentation, and **half of all AI agents already operate in silos** rather than as part of a connected system — even as organizations plan to grow from roughly 12 to 20 agents each over the next two years. The report's own conclusion is blunt: the effectiveness of AI agents "hinges entirely on achieving seamless, debt-free data integration."

In other words, the agent era didn't escape the archipelago. It sailed in and started building more islands — agent silos on top of data silos. How that happens, and what it costs, is the rest of this piece.

IV. Individual Islands, Defined

The precise definition of an island is also the precise reason an agent stumbles on it.

We've been using "Individual Islands" intuitively. Time to pin it down — because the definition isn't just tidy vocabulary. Each defining trait of an island maps onto a specific way that AI breaks.

What an Individual Island actually is

An Individual Island is not merely "another tool in the stack." It has three defining traits.

It has its own model of the world. Inside a CRM, the core objects are accounts, contacts, opportunities. Inside an HR system, employees, contracts, positions. Inside payroll, pay cycles, pay items, tax codes. Each system decides which entities matter, which fields are required, which relationships exist, and which states are legitimate — locally, with little regard for how other systems think about the same people and events.

For an agent: *the same word resolves to a different object on every island. "Customer" is one shape in the CRM, another in billing, a third in the warehouse. The model isn't confused because it's weak — it's confused because the company genuinely holds five incompatible definitions and quietly asked the machine to pick one.*

It runs its own logic. Workflows, approvals, automations, triggers, status changes, notifications — these all live inside the system's walls. "What happens when a deal moves to Closed Won?" "What happens when someone's employment status changes?" The answers are encoded in rules and flows that only that system fully understands.

For an agent: *an agent that acts has to predict consequences — but the rules governing those consequences are scattered and opaque. Flip a status in the HRMS and an unknown number of downstream flows fire across payroll, access control, and benefits. The agent can take the action in one click. It cannot see the blast radius.*

It enforces its own governance. Users sign in through that system. Permissions are

granted there. Audit logs, if they exist, attach to that product. The system knows who clicked what and when — but only within its own borders.

For an agent: *give an agent broad access and you've handed a fast, tireless actor a key to every island, with no single place that decides what it may touch or records what it did across all of them. Governance that was merely inconvenient for humans becomes a live risk surface for software that acts in milliseconds.*

The key is not whether a system has an API. It is whether the system was designed *from the inside out* or *from the outside in*. Most enterprise products are built inside-out: make the internal world coherent, then expose ports at the edges. Very few are built on the assumption that they are one view onto a larger model that lives elsewhere. That is what makes it an Individual Island — and it is the same property that makes agents — models that direct their own tools and actions — so brittle on top of it: they are asked to be the one thing the architecture never provided, a single coherent view.

Figure 3 **What makes an island**

TRAIT	ON THE ISLAND	WHAT BREAKS FOR AN AGENT
Its own model	Each system decides which entities matter and which states are legitimate. **CRM** accounts, opportunities **HR** employees, positions **Payroll** pay cycles, tax codes	"Customer" resolves to a different object on every island, and the agent has to pick one.
Its own logic	Workflows, approvals, and triggers live inside the system's walls, readable only by that system.	It can flip a status in one click. It cannot see which flows fire downstream in payroll, access, and benefits.
Its own governance	Sign-in, permissions, and audit logs attach to that product and stop at its borders.	It holds a key to every island, with no single place deciding what it may touch or recording what it did.

Each defining trait of an island, and the specific way it breaks an agent.

Why “Islands,” not “Silos”

“Silos” is the word everyone reaches for. It is also the wrong word, and the wrong word leads to the wrong fix.

A silo is an accident. It suggests two teams that should talk but don’t. It carries the implication that if people simply collaborated more — if someone just “broke down the silos” — we’d be done.

An island is meant to be self-governing. It has its own rules, institutions, and incentives. Trade with other islands is possible, but it happens on each island’s terms. Product teams build features to keep users inside their surface. Pricing rewards deeper adoption. Integration is, at best, a backlog line — not the organizing principle.

“Silos” also undersells the duplication. In a silo story, there is one copy of the truth in one place and a drifted copy in another. In the island version, *there was never a single shared truth to begin with.* CRM, billing, ERP, support, and the warehouse each grew their own concept of a “customer.” None of them is exactly right; each is tailored to the questions its island cares about.

Language matters because it sets the strategy. If you think you have silos, you try to break them with working groups and “single pane of glass” dashboards. If you realize you have islands, you start asking sharper questions: Where is the mainland supposed to be? Which island gets to decide what a “customer” really is? What happens when two islands disagree? Those are exactly what an AI initiative collides with in week two.

The four island clusters

Once you see islands, you start seeing them in clusters. Most organizations have a few dense groups orbiting the same kinds of entities. Roughly four of them.

Work & Money — HRMS, payroll, ERP, accounting, CRM. The systems that touch employees, customers, contracts, invoices, and pay runs. Where obligations are recorded: to pay, to collect, to deliver, to report.

Operations — ticketing and incident tools, ITSM, project management, time tracking, procurement, logistics. How work actually moves through the organization.

Data & Analytics — ingestion pipelines, warehouses, transformation layers, BI tools, catalogs, experimentation platforms. The cluster meant to reflect the rest of the organization — usually at a delay.

Integration — iPaaS, workflow automation, unified APIs, event buses, internal glue services. This layer owns few entities in principle. It owns the relationships and the movement between the other systems.

Each cluster is an archipelago in its own right, and the integration cluster sits in the middle trying to keep everyone talking. This is why the comforting “we just have a few tools” story is almost always an illusion. A payroll tool that isn’t connected to HRMS is an incomplete island; connect it and you create another set of assumptions and flows. By the time you’ve grown into a normal mid-sized company, you don’t manage “a stack.” You manage several interacting archipelagos — and you’re about to ask an AI to treat them as one.

That is the anatomy of the problem: self-governing systems, each internally coherent and externally misaligned, grouped into clusters that a coordination layer struggles to hold together. None of it was a mistake, exactly. But it adds up to a bill — one that, until recently, you paid quietly.

V. The Cost of Islands

First it was a tax you paid in time. Now it is a ceiling on what AI can safely do.

The human cost — the Integration Tax

Every bridge between two islands costs something. Not once, but over and over. The first cost is obvious: you pay engineers or an integration vendor to wire up APIs, configure connectors, schedule export-import jobs, and glue workflows together with scripts.

The ongoing costs are quieter. A data engineer chasing down why one integration silently stopped two weeks ago. A RevOps lead reconciling pipeline from the CRM with bookings in billing and revenue in the ERP. A payroll manager double-checking that new hires made it from the HRMS to payroll before cut-off. Then there are the costs you only see when something changes — an API version deprecated, a field made mandatory, a new country, an acquisition. Each change punches a hole in one or more bridges, and someone has to find it and patch it — often only after something important breaks.

This is the Integration Tax: the recurring, compounding cost of making Individual Islands behave as if they were one coherent whole. How big? In the 2026 MuleSoft Connectivity Benchmark, technical teams report spending an average of **36% of their time designing, building, and testing custom integrations** — more than a third of your most expensive people, employed as full-time bridge-builders. And the meter is still running on the AI era: IT leaders expect **19% of their IT budgets** to flow into agents and "agentic transformation" over the next twelve months, much of it spent fighting the same fragmentation.

36%
of technical time spent building and testing integrations

19%
of IT budgets going to agents

54%
of organizations with centralized governance

You pay it in several currencies

Money is only the most visible currency. The Integration Tax is also charged against things that don't show up cleanly on a budget line.

You pay in time. Every new edge case, schema change, and new system demands attention. Someone has to design mappings, update flows, adjust queries, rewrite tests.

You pay in attention. Teams keep a mental map of where truth lives and what depends on what. "If we change this field here, what breaks over there?" becomes a constant background question.

You pay in trust. When people meet conflicting numbers, they learn to distrust systems. They hedge. They build private spreadsheets "just to be sure."

You pay in risk. Integrations fail quietly and degrade. A missed sync can turn into missed invoices, incorrect pay, or wrong regulatory reports.

You pay in optionality. The more you build on a set of islands, the harder it becomes to leave any one of them. Integrations, workflows, reports, training, muscle memory — all conspire to make the current choice feel permanent.

When a simple question becomes a multi-week project

If you want to know whether your architecture is working, ask a simple question that cuts across a few systems and watch what happens. Not a moonshot — something boring and essential: Which customers are actually profitable once we include support and collections costs? How many people work here right now, by location and contract type?

Someone asks in a leadership meeting. Heads nod — "we should know that." The room assumes it's a data or ops task. A couple of people pick up action items. They go back to their desks and discover the question touches half the stack. None of those systems were designed together; each has its own identifiers and its own way of marking someone active, churned, or paused. So the team starts stitching: exporting from each island, cleaning column names, reconciling IDs or falling back to fuzzy matches, and arguing about definitions — what counts as "active," which period to use, which costs to include.

Days go by. The first cut emerges in a spreadsheet. By the time it returns to the meeting, it arrives wrapped in caveats: "This is roughly right, but there are still some mismatches between systems."

> **On the surface this looks like a reporting problem. Underneath, it is the cost of islands — two weeks of work to get a "directionally correct" answer that nobody fully trusts.**

Every system in that chain was doing its job. None was built to be part of a global view. Without a shared model, the organization has to assemble one on demand, question by question. That two-week tax is the version of this problem you already know — the human one. The next is the version that is just arriving, and it changes the math entirely: what happens when the thing asking the question, and acting on the answer, is no longer a person who knows to hedge.

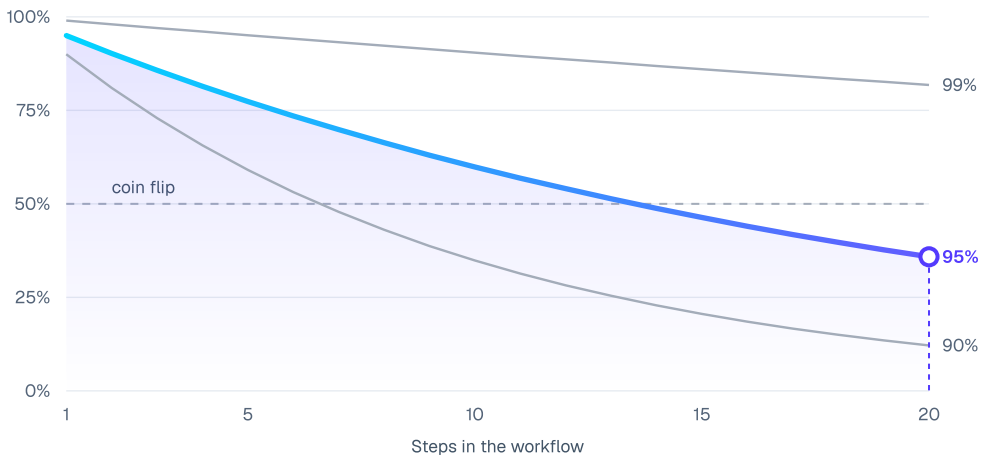
The AI cost — agents act, they don't just read

Everything so far assumed a human in the loop: the analyst who knows which rows to drop, the RevOps lead who knows to hedge. That human judgement has been silently subsidizing the archipelago for years. It absorbs the inconsistencies the systems never resolved. Agentic AI removes that subsidy.

The first wave of enterprise AI mostly *read*: copilots that summarized a deal, drafted a reply, answered a question. The wave arriving now **acts** — it updates the CRM record, triggers the refund, moves the candidate stage, kicks off the payroll adjustment. Reading a wrong number produces a bad slide. Acting on a wrong number produces a bad *event*, in a real system, at machine speed. As Forrester frames it in its 2026 predictions, agents don't merely retrieve information — they interpret, decide, and act. So when the context underneath them is fragmented, the agent doesn't just report a wrong customer count; it emails the wrong segment, applies the wrong discount, or revokes the wrong access — confidently, and faster than anyone is watching.

And the errors compound. An agent completing a real task rarely takes one step; it takes a chain — look up the account, check entitlement, compute the figure, write it back, notify someone. Suppose each step is *95%* reliable, which would be excellent. A twenty-step workflow then succeeds end-to-end only about **36%** of the time ($0.95^{20} \approx 0.36$). Reliability that looks impressive per step collapses across a chain — a point Gary Marcus has made repeatedly about why agent demos dazzle and agent deployments disappoint.

Figure 4 **Reliability, compounded** Interactive online →



36%
succeed end to end

64
failures per 100 runs

0.95^{20}
the whole calculation

RUNS THAT SUCCEED END TO END

Reliability per step	1 step	5 steps	10 steps	20 steps
99%	99%	95%	90%	82%
95%	95%	77%	60%	**36%**
90%	90%	59%	35%	12%
80%	80%	33%	11%	1%

Per-step reliability against end-to-end success, at the article's 95% across twenty steps, with 99% and 90% either side. The collapse is just p to the power of n.

The article's own arithmetic. No model or estimate involved.

This isn't hypothetical fragility. When Sierra built τ-bench, a benchmark of agents doing realistic tasks against tools and policies, it found that leading agents often can't reliably solve the *same* task when the episode is simply re-run — so it introduced a **pass^k** metric just to measure consistency. The wall in front of enterprise agents is not intelligence. It is reliability, and reliability is mostly a function of the context and guardrails you can give them — exactly what an archipelago cannot.

> **It is the same wall as before, one step more dangerous: we point a confident, fast actor at a company whose systems can't agree on what it is acting upon.**

AI needs context — and islands withhold it

On paper, AI should be the antidote to scattered systems. In most companies, what shows up first are app-specific copilots. The CRM gets an assistant that summarizes deals. The support tool gets one that drafts replies. Each is genuinely helpful inside its own island. Each is also, by design, blind to the others. Ask the CRM copilot a question that depends on support history and billing health and it will guess.

The questions that matter most rarely respect product boundaries. "Which customers are at risk?" is not a CRM question — it's about usage, contracts, support history, and collections. To give AI useful work, you need a shared, cross-system representation of people, accounts, events, and obligations. As a16z put it bluntly, data agents are "not a model problem, but a context problem" — and context has to become a real architectural layer, not a prompt. Without that, AI doesn't make the archipelago simpler. It decorates each island with a friendly resident who knows that island intimately and the rest of the world not at all.

Fragmentation becomes compliance risk

Regulators, auditors, and privacy frameworks don't care which tools you use. They care whether you can answer a few basic questions consistently: What personal data do you hold about this person? Where is it stored? Who has access, and on what basis? If your data lives on a handful of systems with clear roles and logs, those questions are manageable. If it lives on dozens of islands, a single subject-access request now means searching across CRM, marketing, support, product, billing, ERP, and sometimes HR — and an auditor asking you to show the chain from contract to invoice to payment to revenue recognition is really asking five islands to agree on identifiers, dates, and amounts.

Agents raise the stakes here too. The same MuleSoft benchmark finds centralized governance in place at only **54% of organizations**, even as agents proliferate — meaning roughly half are deploying software that can read and act across systems without a single place that defines what it may touch or records what it did. Deloitte's enterprise surveys now put risk and governance among the top barriers to scaling generative AI, for exactly this reason. The cost of *proving* compliance rises with every island, and the odds of a blind spot grow with every integration nobody quite owns.

Optionality quietly evaporates

There is one last cost, easy to overlook until it's too late: the slow erosion of your ability to change your mind. Early on, switching tools feels trivial. Fast-forward a few years and that CRM no longer just holds contacts; it holds custom fields that drive pricing and compensation, triggers workflows in other tools, feeds reports other teams depend on. Ask "could we switch?" and the question is no longer "is there a better tool?" It is "can we afford to re-implement our company in another system while still running the company?"

This is how optionality disappears without a dramatic event. You never sign a contract that says "we will never leave this vendor." You simply keep building around them until leaving feels unrealistic. And AI accelerates the trap: if the only AI that works well lives inside one dominant island, that island gains even more gravity. **An island is not a prison on day one. It becomes one slowly, as you terraform it into your only habitable land — and then hand the keys to an AI that can only see that one island.**

VI. What “Data That Just Works” Feels Like

Not one system to rule them all — a few deliberate choices that let the islands stay while a mainland finally exists underneath.

The plumbing isn’t the point

If Individual Islands are the problem, it is tempting to reach for an old solution: “We need one system of record. One place where everything lives.” But we already lived that life. Era 1 — the monolith — was exactly that, and we fled it for good reasons. Going back is not the answer. Neither is the newest shiny version of the same wish: just plug everything into the AI with a universal connector and let the model sort it out.

The industry’s genuine first answer to the agent-context problem is standardized connection. In late 2024, Anthropic introduced the Model Context Protocol (MCP) — a common way for AI systems to reach tools and data, aimed at the “M×N” problem where every model needs a bespoke integration to every system. Google followed with the Agent2Agent (A2A) protocol. This is real progress. After years of bespoke curl scripts, the ferry schedule is finally being standardized.

But here is the crux. A protocol standardizes how an agent *connects* — not what it *finds* when it arrives. MCP can hand an agent a clean pipe into your CRM and a clean pipe into your billing system. It cannot tell the agent that “account” in one and “customer” in the other are the same entity, which system wins when they disagree, or whether the agent is even allowed to write back. The surveys mapping this fast-growing landscape of agent protocols treat exactly these questions — shared semantics and governance — as the open problems.

> **Connection is not coherence. Transport is not truth. A protocol moves data between islands faster; it does not hand anyone the shared map.**

Worse, plumbing alone can backfire. Wire an agent into more systems via MCP, with no coherent model underneath, and you have simply let it reach more contradictory islands more quickly — which makes it *less* reliable, not more. The plumbing is necessary. It is not the point. The point is what sits behind the pipes.

Three roles most companies have smeared together

To get to a stack that genuinely "just works," it helps to separate three jobs that most organizations have quietly merged.

A system of record is where the truth for a specific domain lives. The HR system might be the record for employment status; the ERP for invoices and revenue recognition. None has to be the *only* place that data appears. It is the place everyone agrees to reconcile to when systems disagree.

A system of engagement is where people do the work — a CRM where reps live, a ticketing platform, an HR portal. The term comes from Geoffrey Moore's 2011 AIIM paper, which contrasted these knowledge-worker–facing systems with the transactional systems of record beneath them. They are optimized for workflow and UX, not for being the canonical home of data.

The connective tissue is the layer almost nobody names explicitly. Its entire job is to keep the other two honest: reconcile schemas, link keys, resolve conflicts, enforce policy. It doesn't want to be the place people click around. It wants to be the infrastructure that makes everything else easier to change.

Most organizations today have systems of record and engagement mixed together, and connective tissue smeared across integration tools, data pipelines, and tribal knowledge. The goal is not to pick one favorite system and force everything into it. The goal is to assign these three roles **deliberately** — and to stop asking any one system to be all three at once.

Treat interfaces as replaceable

Once you accept that systems of engagement and record are different jobs, the tools people click around in should be **replaceable** — in the way a browser or an email client is. You might stay with one for years, but if something better comes along, switching should feel like pulling out one cable and plugging in another, not demolishing the house. In practice, that means treating CRMs, helpdesks, and HR portals as *interfaces to a deeper model*, not as the model itself: the CRM reads the customer record from a defined source of truth, its automations are triggers into workflows you own elsewhere, and its reporting is views over a shared understanding of entities and metrics. Replaceability is not a feature vendors advertise. It is something you enforce, by where you let data and logic live.

Make systems of record narrow and explicit

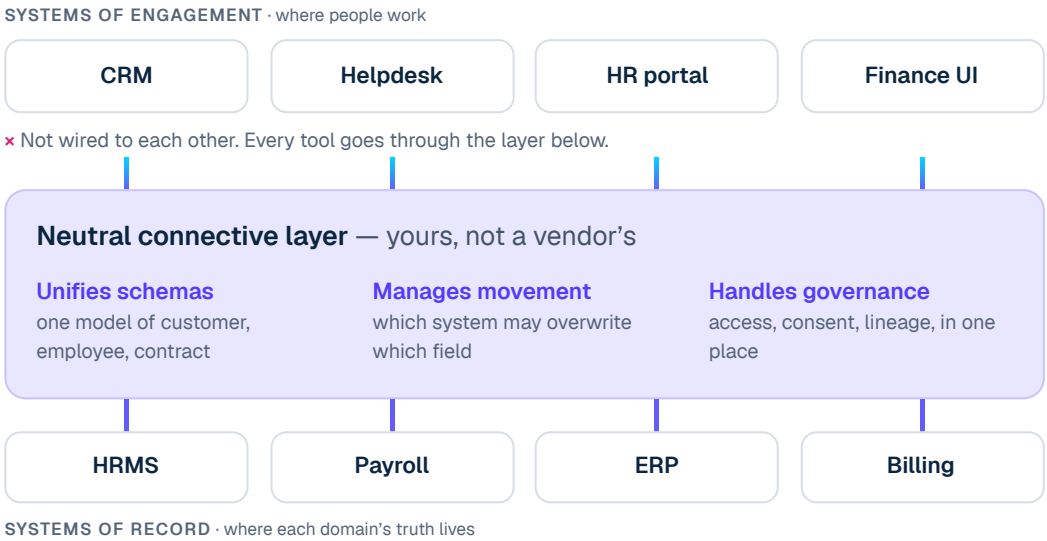
"System of record" can't be a slide in a sales deck. It has to be a sentence you can finish for each domain: "For employment status, our system of record is ___." "For invoices, our system of record is ___." That sentence names an **accountability**: when there's a disagreement, this is where we look. It also forces scope to be narrow — an HR platform may be the record for employment status but *not* for compensation history. Once you write these choices down, integration becomes less arbitrary, conflicts go from political to procedural ("this system vs. the declared record"), AI projects stop starting from a blank page, and migrations stop being pure chaos because you know which downstream flows move with each system.

Build a neutral connective layer you control

Even with clear systems of record and replaceable interfaces, you still need a place where the organization as a whole comes into focus: a neutral integration and data layer whose loyalty is to the company, not any one vendor. It knows how "customer" maps across CRM, billing, ERP, support, and the warehouse; which system may overwrite which field; which events count as hires, churns, or delinquency. It does three things. It **unifies schemas** — keeping a clear model of core entities and how each island maps onto them (a semantic layer, but living in the operational world). It **manages movement** — enforcing your decisions about systems of record, timing, and conflict resolution, behaving like a switching yard rather than a pile of point-to-point flows. And it **handles governance** — access, consent, lineage, retention — in one place you can reason about.

This is where the most credible recent thinking has landed. a16z argues that for AI, context must become "a real architectural layer in its own right… first-class infrastructure". Research on agent reliability points the same way: grounding an agent in an explicit shared schema measurably reduces overconfident, wrong actions. Vendors are racing to occupy this layer — Glean builds a knowledge graph as the "context engine" for enterprise agents, and Salesforce now pitches "Data 360" as a unified context layer feeding every agent. That is useful validation that the connective tissue is the real battleground — but note the irony: buying your "neutral" layer from the vendor of your busiest island is precisely the lock-in this piece warns against. The layer doesn't have to be one product. It does have to feel like *one logical thing* that belongs to you.

Figure 5 The connective layer



SIX CATEGORIES, ONE COMPANY

CRM Salesforce · HubSpot · Zoho CRM	SUPPORT Zendesk · Freshdesk · Intercom	PAYROLL & HR Workday · Gusto · Rippling
BILLING Stripe · Chargebee	ERP SAP · NetSuite	WAREHOUSE Snowflake · Databricks

Examples of each category, not any particular company's stack. Every one of these is a system of record for something, and every one of them holds a different opinion about who your customers are.

The shape the argument points at. Interfaces on top, one neutral layer you own, records underneath.

Put AI on top — not locked inside

With systems of engagement, systems of record, and connective tissue in place, the right home for AI becomes almost obvious. It belongs **on top of the integrated layer, not buried inside each island.** An assistant built this way does not care which CRM you use tomorrow. It cares that there is a coherent definition of "customer," and that it can see the relevant events: deals, usage, invoices, tickets, payments. If you keep AI locked inside individual islands, you get local optimization and leave most of the leverage on the table. If you put it on top of an integrated layer, you can ask questions that cross systems, let it take actions knowing permissions are enforced centrally, and swap interfaces without rebuilding your understanding of the company each time.

The irony is that AI — which many treat as a reason to double down on a single platform — is the strongest argument for neutral connective tissue. The more important AI becomes to how you operate, the less you can afford for your understanding of the world to be trapped inside any one vendor's walls. That is what "data that just works" actually feels like: you type the question once, you trust the answer, and you can let an agent act on it without holding your breath.

VII. Escaping the Archipelago

You can't make good decisions about architecture, tooling, or AI if "how bad is the archipelago?" is answered with a shrug.

Step 1 — Score your stack with the ARCHIPELAGO Index

There is no magic migration that turns an archipelago into a mainland overnight. What you can do is change the direction of travel — stop accumulating islands by default and start bending the system toward cohesion. Think of this as a set of moves you can make in parallel, not a linear checklist. But the first step is the same for everyone: get an honest picture of where you stand. Score yourself across seven concrete dimensions.

For each dimension, score yourself from **1 to 5** — 1 means "in good shape," 5 means "this is costing us time, trust, and options." Add them up for a total between **7 and 35.** Low teens is a mostly connected mainland; high teens to low twenties, a managed archipelago; mid-to-high twenties, a growing one; near the thirties, islands everywhere.

> **It is not a scientific instrument. It is a mirror — and you will learn more from the disagreements than from the average. If engineering scores workflow entanglement a 4 and operations scores it a 2, something important is hiding in that gap.**

So don't score it alone, and don't obsess over the exact total. Sit in a room with people who understand the stack from different angles — engineering, data, ops, finance, HR, security — and go dimension by dimension. When you're done, you'll have a shared vocabulary for "how bad is it?" and a shortlist of where change will have the highest leverage. That shortlist is what the next four moves act on.

Figure 6 **The archipelago index** Interactive online →

Circle one number per row: 1 is in good shape, 5 is on fire. Then add them up.

1. Surface area
How many core systems you run, and how much they overlap.
1 · A handful, little overlap 5 · Dozens, many holding the same entities
1 2 3 4 5

2. Integration approach
How your systems are actually wired together.
1 · One observable layer you control 5 · Brittle point-to-point scripts nobody owns
1 2 3 4 5

3. Systems of record
How clearly each domain's source of truth is defined.
1 · One explicit, agreed record per domain 5 · No one can say what's authoritative
1 2 3 4 5

4. Workflow entanglement
Where your critical business logic actually lives.
1 · In a neutral layer you own 5 · Trapped in vendor-specific automations
1 2 3 4 5

5. Exit friction
How hard it would be to leave a core tool.
1 · A project, not a re-org 5 · Leaving any core tool feels impossible
1 2 3 4 5

6. AI context coherence
How coherent a view your AI and agents can get.
1 · Agents read a shared, defined model 5 · Each assistant sees one island, guesses the rest
1 2 3 4 5

7. Governance & semantics
How centralized your access, lineage, and definitions are.
1 · Centralized and clear 5 · Scattered and informal
1 2 3 4 5

Total _____ / 35 7 · mainland 35 · islands everywhere

7–14	**Mostly Connected Mainland**. Your systems behave like one place. The job is to keep it that way as you grow.
15–21	**Managed Archipelago**. Islands exist, but you're holding them together deliberately. Watch the seams.
22–28	**Growing Archipelago**. Fragmentation is outpacing your bridges. AI projects will start hitting this wall.
29–35	**Islands Everywhere**. Your understanding of the company is scattered. Agents can only guess at the rest.

Score all seven. The total matters less than the dimensions where you and your colleagues disagree.

The ARCHIPELAGO Index is an original self-assessment framework for this piece, not a benchmarked external metric.

Move 1 — Buy and design for exit, not just onboarding

The easiest time to change your destiny is before you sign the contract. Most buying processes focus entirely on getting *in*: features, pricing, implementation. If you want to avoid being trapped on any one island, add a second lens: **"If future-us wanted to leave this system in eighteen months, how painful would that be?"** Instead of "does it have an API?" you ask: Can we export *all* our data, including custom fields and history, in structured formats? Are the schemas documented well enough for an outsider to understand? Are there pricing cliffs on data access that will bite later? Instead of "does it have AI?" you ask whether the system can participate in an AI architecture *you* define, or whether its AI requires keeping everything inside its walls. You'll still pay Integration Tax — just on things that matter, instead of on vendor friction.

Move 2 — Decouple critical workflows from vendor automations

If one thing quietly turns islands into prisons, it's workflow. Over time, crucial processes get encoded across dozens of app-specific automations — a CRM rule that fires a webhook, a support macro that pings a bot, an HRMS workflow that flips a status. Together these are how your company actually works, and one of the biggest sources of migration pain. The goal is not to eliminate vendor automations but to be intentional about *where the important logic lives.* Pick your top end-to-end workflows — lead-to-cash, hire-to-pay, ticket-to-resolution — and map where each step lives. If changing one system would force you to rebuild almost the whole workflow, you've found logic that's too tightly coupled to a specific island. Lift the core of those workflows into a neutral orchestration layer you control, and leave the edge tools to handle local concerns.

Move 3 — Clarify domains and systems of record

Turning narrow, explicit systems of record into practice starts with a domain map. Gather the right people and draw it in three columns: the domain, the system of record, and the main consumers.

- Employment status → **HRMS** → payroll, IT, finance
- Pay events → **payroll** → HR, finance, reporting
- Invoices → **ERP** → CRM, collections, analytics
- Customer legal entities → **ERP or billing** → CRM, support, compliance

The goal is to replace shrugs with sentences — to reach a point where, for each domain, someone can say “when we disagree, this is the system we reconcile to.” Then you use the map three ways: update integrations so flows respect the declared record; update reporting so dashboards anchor on it or are explicit about divergence; and update conversations so conflicts start from “this system vs. the declared record” instead of “my tool vs. your tool.” It’s unglamorous, and it’s a prerequisite for giving AI a coherent view of the company.

Move 4 — Use AI as the forcing function

Executives rarely sign off on “let’s refactor our integrations because it’s the right thing to do.” They are far more willing to fund “let’s build an AI assistant that helps our teams make better decisions.” The trick is to link the two honestly. Start with one cross-system question where AI would be genuinely valuable — “for any customer, show me a live view of health combining product usage, contract terms, support history, and payment behaviour” — and work backwards. Very quickly, you discover that the hard part of the AI project is not the model. It is the connective tissue. When someone asks why you’re investing in integration architecture and governance, the answer isn’t “because the stack is a mess.” It’s “because without this, the AI we want cannot see the world clearly enough to be useful or safe.”

The evidence backs the sequencing. BCG finds that only about 5% of companies are capturing substantial value from AI while 60% are not — and the “future-built” leaders are distinguished not by fancier models but by having put the underlying capabilities in place first. McKinsey’s State of AI reaches a strikingly specific version of the same conclusion: **redesigning workflows is the single biggest differentiator** between AI high performers and everyone else. And MIT’s GenAI Divide found the rare pilots that succeed are the tightly scoped, deeply integrated ones. Across three independent studies, the winners are the ones who treated AI as a reason to fix the foundation — not as a coat of paint over the archipelago.

VIII. The Next Decade of Enterprise Infrastructure

The last fifteen years were about apps. The next ten are about whether your company has a coherent model of itself to put AI on top of.

From app-centric to data-centric stacks

For the last fifteen years, most of the energy in enterprise software went into applications. We built better CRMs, HR systems, billing tools, analytics. Each wave added more islands and, sometimes, better ferries. The next decade is about something else. As AI becomes less of a novelty and more of an expectation, and as regulation tightens around data, the limiting factor shifts. It is no longer "do we have an app for that?" It is **"do we have a trustworthy, flexible representation of our business that we can safely put to work?"** That is a data-centric question. A data-centric stack doesn't mean everything in one database. It means the architecture is organized around shared models, flows, and policies — rather than around whichever vendor's app has the most screen time. Systems of engagement are chosen on fit. Systems of record are deliberately assigned per domain. Connective tissue is core infrastructure. AI and analytics sit on top of that layer, not buried inside individual products.

This isn't only our read; it's becoming the consensus among people who fund and study this shift. a16z argues that as raw model capability commoditizes, the durable moat moves from the model to proprietary, well-organized data and context. PwC frames agentic AI as a structural rewiring of how work happens, with trillions of dollars of value in motion — value that flows to the companies whose data can actually feed it.

Incentives: vendors, buyers, regulators

Vendors have strong reasons to behave like islands. The more of your data, workflows, and AI they keep inside their product, the harder you are to leave. Gartner's warning about "agent washing" is a symptom of exactly this instinct: rebrand, capture, retain. But there is an opening for a different kind of vendor — one that leans into portability, clean interfaces, and being a good citizen in a data-centric stack. Such products extract less lock-in rent, but they become easier to adopt and recommend, and as buyers grow more sophisticated, that reputation compounds.

Buyers have more influence than they think. Every time a company asks hard questions about exports, APIs, workflow portability, and AI context — every time a contract is won or lost on interoperability rather than features — it nudges the market and signals that "becoming our prison" is not a winning strategy.

Regulators are slowly raising the bar on what it means to be in control of your data: mapping, minimisation, access logs, explainability. You cannot meaningfully comply on top of a stack where no one knows where the truth lives. In the short term, regulation feels like another tax; in the long term, it may be one of the forces that quietly pushes organizations toward more coherent architectures.

Rewriting the founder story we began with

Go back to the founder from the opening, standing in front of a login screen. In the old story, they pick tools one by one, wire it all together just enough to get by, and by the time the company really cares, the archipelago is already there — optionality gone, AI and analytics working uphill against their own infrastructure, the assistant confidently returning the wrong customer count to the board.

The alternative story does not require becoming an enterprise architect on day one. It asks the founder to treat a few ideas as non-negotiable much earlier:

- We treat tools as replaceable interfaces, not as our only reality.
- We are explicit about which systems are systems of record for what.
- We invest in connective tissue we control, even if it's simple at first.
- We design workflows that don't assume any one vendor will be with us forever.

In that story, picking a CRM is not picking the canonical home for all customer truth. It's picking the current interface into a model that lives elsewhere. Migrations still hurt. Integrations still need work. But the growth of the company does not automatically turn yesterday's choices into tomorrow's prison.

Choosing the future

We have normalised a strange thing.

We have normalised the idea that as a company grows more digital, its understanding of itself should become more *fragmented* — that every new capability must arrive as a new island, with its own language, memory, and idea of what is true. We have normalised paying the Integration Tax as if it were rent or electricity, instead of seeing it as a consequence of how we chose to assemble the stack.

It does not have to stay that way. The answer is not going back to one monolith. It is not pretending API diagrams are a coherent model. It is not slapping AI onto every island and hoping it bridges the gaps. The answer is to name the pattern and change the defaults: to see your tools as individual islands, to measure how islanded you really are, and to insist on clear systems of record, replaceable interfaces, neutral connective tissue that belongs to you, and AI that sits on top of all of it rather than trapped inside any one vendor's world.

If you are a founder, an executive, a builder, or an operator, you do not control everything about the ecosystem you live in. But you do control your own geography. You can keep adding islands and ferries and hoping it somehow adds up. Or you can start building a mainland, piece by piece, until one day the question — *"How many active paying customers do we have, by segment, with full unit economics?"* — really does feel like something your company can answer in one place, with one voice, without summoning a small army. And without an AI confidently handing you the wrong number.

> That feeling is what "data that just works" actually means.

It is still rare. The next decade will be defined by how many organizations — and how many founders, starting today — decide to make it normal.

> The islands don't have to disappear. They just have to stop being the whole map.

Set in Geist. Typeset with pdfcn on Takumi from the same source as adityagaur.xyz/writing/integration-islands, so the two editions cannot disagree.

Sources

1 Andrej Karpathy, *Software Is Changing (Again) / "Software 3.0"*. latent.space/p/s3

2 Simon Willison, *"An LLM agent runs tools in a loop to achieve a goal"* (2025). simonwillison.net

3 Anthropic, *Effective context engineering for AI agents*. anthropic.com/engineering

4 Chroma Research, *Context Rot* (2025). research.trychroma.com/context-rot

5 Liu et al., *Lost in the Middle* (TACL 2024). arXiv:2307.03172

6 Schäfer et al. (BASF), *Context is all you need* (2026). arXiv:2603.12813

7 MIT NANDA, *The GenAI Divide: State of AI in Business 2025* — ~95% of pilots no P&L impact; "learning gap." nanda.media.mit.edu *(primary source)*

8 Gartner, *Over 40% of Agentic AI Projects Will Be Canceled by End of 2027* (2025). gartner.com *(primary source)*

9 MuleSoft (Salesforce), *2026 Connectivity Benchmark Report* — avg 957 apps per org (up from 897); 27% connected (down from 29%); 64% of IT leaders doubt near-term AI goals; ~half of AI agents in silos; agents projected 12 → 20 per org. mulesoft.com *(primary source)*

10 Anthropic, *Building Effective Agents* — an agent is a system where the model directs its own tools and actions. anthropic.com/research

11 a16z (Jennifer Li, Jason Cui), *Your Data Agents Need Context* — enterprise context is "disparate and messy"; agents fail cross-system questions without a coherent layer. a16z.com

12 Forrester, *Predictions 2026: AI Agents…* — agents interpret, decide, and act; without explicit context they guess and misinterpret metrics. forrester.com

13 Gary Marcus, *Marcus on AI* — compounding error across multi-step agent workflows ($0.95^{20} \approx 36\%$). garymarcus.substack.com

14 Sierra, *τ-bench / τ²-bench* — realistic agent benchmark; inconsistency on repeated tasks; the pass^k metric. sierra.ai

15 Deloitte, *State of Generative AI in the Enterprise* — risk and governance among top barriers to scaling. deloitte.com

16 Anthropic, *Introducing the Model Context Protocol* (Nov 2024). anthropic.com/news

17 Google, *Announcing the Agent2Agent (A2A) Protocol* (Apr 2025). developers.googleblog.com

18 Ehtesham et al., *A Survey of Agent Interoperability Protocols (MCP, ACP, A2A, ANP)* (2025). arXiv:2505.02279

19 Geoffrey Moore, *Systems of Engagement and the Future of Enterprise IT* (AIIM, 2011). Systems-of-Engagement.pdf

20 Semantic grounding for agent reliability (OpenReview) — explicit shared schema reduces wrong actions. openreview.net

21 Glean, *Knowledge graph as the context engine for enterprise AI. (vendor source)* glean.com

22 Salesforce, *Welcome to the Agentic Enterprise (Agentforce 360 / Data 360)* (Oct 2025) — market signal, not neutral research. *(vendor source)* salesforce.com

23 BCG, *The Widening AI Value Gap / Build for the Future 2025* — ~5% capture substantial value, ~60% none; leaders built capabilities first. bcg.com *(primary source)*

24 McKinsey, *The State of AI 2025* — redesigning workflows is the key differentiator for AI high performers. mckinsey.com *(primary source)*

25 a16z, *Big Ideas 2026* — the durable moat shifts to proprietary, well-organized data and context. a16z.com

26 PwC, *Agentic AI Reinvention / Value in Motion* — agentic AI as structural rewiring of work. pwc.com