



POINT OF VIEW

The Orchestration Build Trap

~~Build vs. Buy~~  *is the wrong question*

Buy the foundation. Build what's yours.

Chris Vessey, VP of Innovation & Customer Value

The orchestration build trap

Every large enterprise eventually asks whether it should just build this itself. The honest answer is that the build is becoming easier—but maintaining it is not. Orchestration is not a feature you ship once; it is a layer that has to survive every ERP upgrade, new regulations, every new country, every model release—held together by a team that has to exist for as long as the layer does. That is the bill nobody puts on the business case, and the team nobody has.

Build vs. buy is the wrong question

In 2026 the classic build-vs-buy argument has collapsed, because the premise it rested on is gone.

Building software used to be expensive and slow; with AI assistance, a competent platform team can stand up an intake form, a routing engine and an LLM-backed classifier in a quarter. So the debate as usually staged—can we build it?—now has an easy answer. Yes. You can.

That answer is a trap, because building was never the expensive part. Maintenance is. Every custom workflow starts decaying the day it ships: the ERP moves, the tax authority changes a format, a data provider deprecates an endpoint, a regulator adds a disclosure, a model provider retires a version, procurement re-cuts its category tree, the company acquires a business in a country you've never onboarded a supplier in. None of that is in the build estimate. All of it is in the run cost, forever.

The real question is not build or buy. It is which layer you should own. And the answer for orchestration is unusually clean, because the thing that makes orchestration work is almost entirely not code. It is procurement semantics—a normalized supplier, contract, transaction, and risk data model—plus a



very large body of maintained reference truth: tax identifier rules in 120+ jurisdictions, banking formats and bank reference data, questionnaire libraries, risk domain models, ERP field defaults, connector contracts. A build team can write the workflow engine in weeks. It cannot write the reference truth at all; it can only start paying to maintain it.

Orchestration is the operating system, not the app that runs on it. You can build an app on a weekend. You cannot build—and keep—an operating system as a side project of the procurement function.

So the reframe we put in front of every enterprise weighing this: **buy the foundation, build what's actually yours.** Your category rules, your policy thresholds, your approval matrices, your supplier segmentation, your buying-channel logic, your risk appetite—those are genuinely differentiated, they change constantly, and you should absolutely build them. Build them in a no-code layer that was designed to be changed, on top of a foundation that already speaks procurement. Don't spend eighteen months rebuilding the foundation first.

Six ways the build fails

These are not hypotheticals. They are the pattern we see in enterprises that tried, and the pattern digital and IT leaders themselves name when they're asked honestly what it would take.

01 The scope illusion

What gets scoped is “an intake form, some workflow, and an SAP interface.” What is actually required is a several-hundred-item capability surface per module—most of it invisible until you're live. Section 04 is that surface, module by module. The first release always looks achievable; it is release four, when the exceptions arrive, that breaks the programme.

02 The team doesn't exist—and procurement pays for it twice

The skills a procurement orchestration build needs sit at an intersection almost nobody staffs: procurement domain depth and platform engineering, in the same team, for years. You can hire either; the pair is scarce, expensive and hard to hold. And the second bill is the one nobody forecasts: a build levies a heavy, sustained design tax on the procurement function itself, senior category and process people writing requirements, adjudicating edge cases and testing instead of doing procurement, before a single basic feature works. Section 03 sets out the whole roster, and the queue it has to join.

03 Fit failure, not change failure

Homegrown and generic-platform builds become shelfware, and the post-mortem almost always blames change management. It is usually wrong. Users abandon the tool because it doesn't understand what they're doing—no category semantics, no supplier risk tiers, no notion of a preferred supplier or a buying channel—so day two produces a workaround. Non-adoption destroys the entire value case, because every other benefit is downstream of people actually using the compliant path.

04 The maintenance treadmill

Orchestration sits between systems that all move independently. Regulatory change, ERP upgrades, provider API changes, e-signature and tax-service versioning, and now model churn. On a bought platform this is absorbed under subscription—ORO Labs ships weekly, with scheduled regression suites customers run against their own configuration across five environments. On a build, every one of those events is an unbudgeted sprint, and the run team's capacity goes to standing still.

05 A build learns only from itself

A platform used across many global enterprises accumulates patterns no single company can reach: how duplicate detection actually has to behave at 100,000 suppliers, what a supplier extension needs to re-ask and what it must not, where approval matrices break at reorganization. You get that learning as a product, continuously, without running the experiments. A build gets one company's experience, once, and pays full price for every lesson.

06 Behind the Innovation Curve

The reality of many in-house development teams is that they're usually partly, or even significantly, behind the innovation adoption curve. Especially in larger organizations where new technology requires significant due diligence and review, new innovation adoption can slow. External providers can often adopt new technology faster and make it enterprise-ready, while taking on much of the implementation and operational risk themselves. The accelerating trajectory of AI development has brought this reality into sharp focus.

The team you'd have to build first

Cost arguments are easy to dispute, because every assumption in them is arguable. The skills argument is not, because it resolves to a roster—and a roster is checkable. Before anything else, write down the people a procurement orchestration build actually requires, and ask two questions of every line: do we have this, and can we hold it for five years?

Here is that roster, against the same capability delivered on a bought platform. The point of the right-hand column is not that buying removes the need for skills—it doesn't. It's that buying removes the need for the skills you can't get, and lets you spend the ones you already have.

CAPABILITY REQUIRED	TO BUILD IT YOURSELF	TO RUN IT ON ORO
Procurement process design	Deep and continuous—every rule, threshold, and exception written from nothing	Deep and continuous, expressed as configuration, by procurement people
Product ownership	Dedicated owner with genuine procurement domain depth, not a generic BA	Deep and continuous, expressed as configuration, by procurement people
Solution & data architecture	Data model, integration topology and security architecture, designed from zero	Inherited
Backend engineering	Workflow engine, object model, APIs, persistence —3–4 FTE minimum	None
Front-end engineering	Two distinct products: an internal application and an external supplier-facing one	None
UX design	Real design capability—adoption is the value case, and friction kills it	None
Integration engineering	SAP IDoc / OData / BAPI / MDG plus every point solution, built and maintained	Deep and continuous, expressed as configuration, by procurement people
Data engineering	Supplier master, category taxonomy, reference-data pipelines and reconciliation	Light: master data ownership and mapping

Jurisdictional tax & banking knowledge	Sourced in-house or contracted, for 120+ countries, and maintained forever	Maintained in the product
AI / agent engineering	Agent design, prompt engineering, evaluation harnesses, model operations	Agents are configured, not coded
AI governance	Invented from scratch: tiering, audit evidence, model policy, access scoping	You set policy inside a framework that already exists
Security engineering	An internet-facing surface for tens of thousands of external organizations, plus permanent pen-test remediation	You govern access and data; the platform surface is the vendor's obligation
QA & test automation	A full regression suite, written and then maintained against your own changes	Configure and schedule the suites the platform provides
DevOps & release engineering	Five environments, change packaging, promotion, rollback	Provided
Agent operations	A permanent new role that did not exist in your org two years ago	Light:monitor, tune, retire
Distinct skill sets to hold	15, indefinitely	4 - and procurement and IT already have all four

This is the roster for the platform itself. It sits on top of the procurement-side design capacity in section 07's cost model—the category, risk, tax, and process experts whose time the build consumes before anything works.

Three scarcities, not one

Scarcity 01

The hybrid you can't hire

Procurement domain depth and platform engineering, in the same team. You can hire either one. The pair is rare enough that most enterprises never assemble it—so every requirement passes through a domain expert, into an engineer, and back, losing fidelity in both directions. That translation tax is precisely why a build demands so much procurement time: the knowledge has to be extracted from your people because it isn't in the tool.

Scarcity 02

The AI skills everyone is bidding

Agent design, evaluation, and model operations are among the most in-demand skills in the enterprise right now—and procurement is not where they typically get allocated. The function's own readiness data says the same thing: 85% of procurement executives are piloting or using AI and 73% are deploying agents, but only 49% clearly understand how agentic AI works and just 39% report formal AI training across their teams.

Scarcity 03

Key-person risk on a payments path

A team this small is two or three resignations away from a capability outage—on a system that governs who gets set up as a supplier and whose bank details reach the ledger. The knowledge lives in people, not in a product, so every departure is an unrecoverable loss and every handover is a re-learning cycle. Five years is a long time to hold a team together.

And then there's the queue



Suppose the skills exist somewhere in the enterprise. They are not idle. A procurement build joins a queue outside the IT and Digital door, and that queue is ordered by proximity to revenue and by regulatory mandate. Customer platforms, revenue systems, and anything with a compliance deadline go first. Procurement is a support function arriving with a cost-avoidance business case—it is structurally near the back, and it stays there.

The failure mode this produces is worse than being refused. Being refused is cheap: you keep your money and buy something. The expensive outcome is being approved and then starved—the slot is granted, the program starts, and the first time a revenue system slips, procurement's engineers are the ones reassigned. The build stalls half-finished. You have spent the money, you have no working system, and you still need the run team you haven't hired.

This is the part a CPO can put to a CIO without it becoming a turf fight, because it isn't a criticism of IT—it is an accurate description of how any well-run IT function prioritizes. The question isn't whether the digital team is capable. It's whether procurement will ever be the thing they are allowed to work on, for the five years this takes.

Buying is how procurement stops queuing. The configuration capacity sits inside the procurement function, so procurement's rate of change is no longer set by someone else's backlog.

“We'll use a systems integrator.”

This is the honest fallback once the skills gap is admitted, and it solves exactly half the problem.

An SI can supply the build-phase roster; that is what they are for. What they cannot supply is the run phase, and orchestration is almost entirely run phase.

What you are left with on day one after go-live is a bespoke system whose design rationale left with the engagement, no product roadmap, no other customer finding the bugs before you do, and every subsequent change priced at a day rate.

BUILD PHASE

An SI can supply the build roster.

Architecture
Engineering
Integration
Configuration
Testing

RUN PHASE

The dependency never leaves.

No roadmap
No pooled learning
Every change at day rate
Permanent supplier dependency

You have converted a skills problem into a permanent dependency on a supplier—with no product behind it and no other customers funding its improvement.

The scope you'd be taking on, module by module

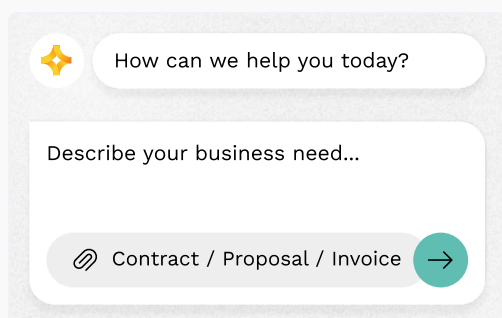
Each block below states what a build team typically assumes the module is, what the capability surface actually contains, and the specific items that never make it into an internal estimate. Capability-area counts are drawn from ORO's own Solution Definition documents—the customer-binding scope statements per module.

Intake	9 Capabilities	Supplier onboarding	17 Capabilities
Supplier Risk Management	12 Capabilities	Bank Fraud	17 Capabilities
Supplier front door	8 Capabilities	Contracts	17 Capabilities
Sourcing	10 Capabilities	PO management	10 Capabilities
Invoicing	7 Capabilities	Agentic workflows	cross-cutting

Intake Management

THE BUILD ASSUMPTION

“A smart form and a routing engine.”



How can we help you today?

Describe your business need...

Contract / Proposal / Invoice →

What it actually is. Conversational intake that interprets **intent**, not keywords; tile-based entry points; forms that restructure themselves on inputs; a configurable workflow engine; routing into at least **seven distinct downstream destinations** - sourcing, contracting, purchase requisition, invoicing, purchasing, supplier onboarding, risk; buying-channel guidance; punchout catalog integration; and assignment logic across procurement, legal, finance and risk. Not to mention other queries, knowledge repositories or data lake options.

NEVER IN THE ESTIMATE

Punchout is a protocol project of its own (cXML round-trip, session authentication, per-supplier certification). The buying-channel matrix is a living policy artifact. Routing has to survive every reorganization, which is why it must be driven by function × entity × category, not by named people.

SKILLS IT DEMANDS:

Conversational-AI and intent design, cXML punchout protocol engineering, and the policy authority to own a buying-channel matrix.

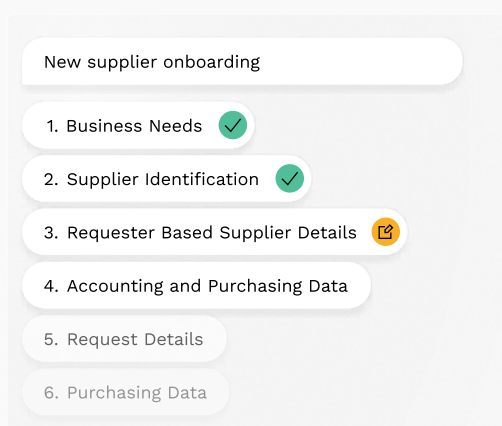
Observed:

35% of demand into a green channel (Bayer) · buying-channel automation above 95% across all categories and regions (Roche)

Supplier Onboarding & Management

THE BUILD ASSUMPTION

“A supplier form and an interface into SAP.”



New supplier onboarding

1. Business Needs ✓
2. Supplier Identification ✓
3. Requester Based Supplier Details ✎
4. Accounting and Purchasing Data
5. Request Details
6. Purchasing Data

What it actually is. The largest surface in the platform.

- Guided intake with complexity-based duration estimates;
- AI category recommendation;
- Preferred-supplier guidance;
- Duplicate detection across names, legal identifiers, tax numbers and addresses;
- Enrichment via external supplier data providers;
- Dynamic questionnaires triggered by category, geography, service type and regulation;
- Document intelligence extracting from tax certificates, incorporation documents and bank proofs;
- Tax validation across 120+ countries;
- Payment validation against IBAN, ABA, IFSC and local standards with a maintained bank reference dataset;
- Workstream routing spanning hundreds to thousands of combinations;
- A supplier-facing submission interface;
- Per-change-type maintenance approvals;
- Supplier extension with delta logic;
- Block/unblock synchronized to ERP; Supplier 360;
- Multi-ERP routing by company entity across SAP ECC, S/4HANA, MDG, Oracle, NetSuite and Sage with accounting and purchasing defaulting.

NEVER IN THE ESTIMATE

Most of this is maintained as reference truth, not code—120 countries of tax rules and document requirements, bank identifier formats, jurisdictional document types. That is the single worst asset class to own: it never finishes, it changes without warning, and being wrong is a compliance event. Delta onboarding —knowing what not to re-ask when extending an existing supplier to a new entity—is the feature every homegrown tool omits and every user demands by month three.

SKILLS IT DEMANDS:

International tax and banking-compliance knowledge, master-data governance, multi-ERP integration engineering (SAP MDG / IDoc / OData), and document-extraction ML.

Hackett world-class risk coverage:

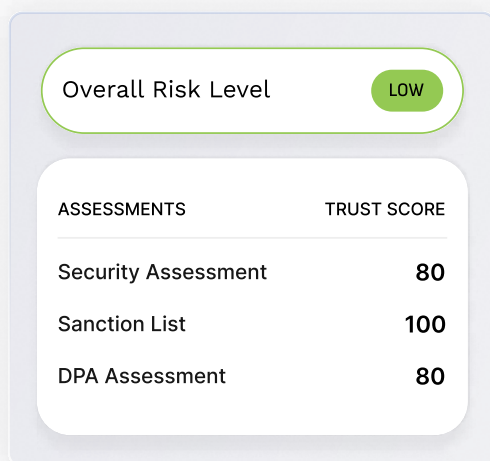
low-value spend 40% → 62%, medium-value 33% → 50%

12 capability areas

Supplier Risk Management

THE BUILD ASSUMPTION

“A questionnaire tool and a score.”



What it actually is. Risk assessment triggered inside intake; questionnaire management with conditional branching and scoring logic; multi-domain assessment across business continuity, cybersecurity, data protection, regulatory compliance, financial viability and concentration risk; a scoring model; centralized tracking; assessment lifecycle management; **automated renewals**; delta assessments on supplier extension; **parent-child risk inheritance**; an aggregated supplier risk profile; and risk visibility inside Supplier 360.

NEVER IN THE ESTIMATE

The re-assessment engine. A build ships the assessment and forgets the renewal, so coverage silently decays to near-zero within two cycles and the audit finding arrives in year three. Parent-child inheritance and delta assessment are the two things that make risk workable at scale, and both are structurally hard once you already have 100,000 supplier records.

SKILLS IT DEMANDS:

Third-party-risk expertise across six-plus risk disciplines, questionnaire and scoring design, and the data modelling for hierarchy inheritance.

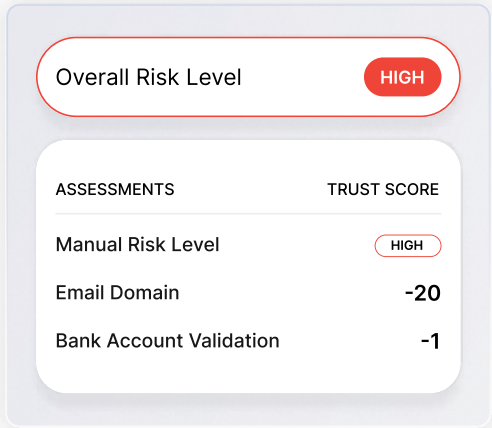
Hackett world-class risk coverage:

low-value spend 40% → 62%, medium-value 33% → 50%

Bank Fraud & Payment Verification

THE BUILD ASSUMPTION

“Validate the IBAN checksum.”



What it actually is. Global bank account validation; US verification via Early Warning; international verification via Nium; supplier trust scoring via CDQ; domain risk intelligence via SEON; bank proof document extraction; global format validation; a risk scoring model; automated risk-based workflows; **out-of-band callback verification**; and invocation from both onboarding and maintenance flows.

NEVER IN THE ESTIMATE

You cannot build this at all. It is not software—it is a portfolio of commercial data relationships with contracts, compliance review, per-check pricing, and volume terms. A build ends up procuring each one individually, at low volume, with no aggregated pricing, and still has to write the risk-tiering and callback orchestration on top.

SKILLS IT DEMANDS:

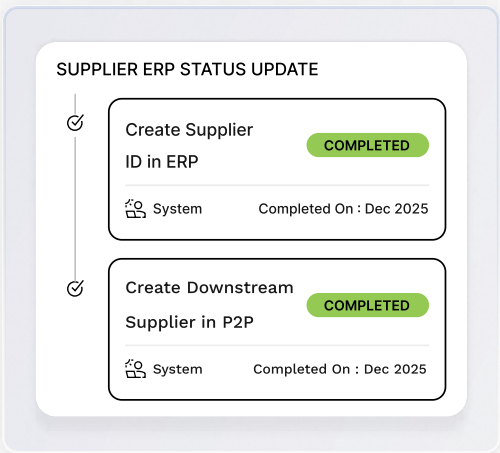
Financial-crime domain knowledge, commercial vendor management across four data providers, and payment-security engineering.

Observed:
\$10M+ prevented losses · 4× more suppliers screened per month · under 30 seconds to verify a vendor’s email and bank details

Supplier Front Door

THE BUILD ASSUMPTION

“A supplier portal—some screens.”



What it actually is. An externally-facing identity and security product. Portal interface; PO, invoice and onboarding status visibility; two-way collaboration; **fraud controls including device fingerprinting, rate limiting, and domain validation**; notification framework; full audit logging; supplier-initiated workflows for price, catalogue, scope, contact, banking, name/ address and certificate changes; logged-out invoice status lookup; rejection and hold reasons; invoice listing and download; PO retrieval; support ticketing and a help centre; announcements and regional contacts; attachment restriction and retention policy; phased enablement; access tagging; **automated vendor-user association, multi-vendor association, unassociated-user handling and access revocation**; and usage analytics.

NEVER IN THE ESTIMATE

This is the module that is universally under-scoped. Authenticating and authorizing tens of thousands of external organizations—whose staff turn over constantly—is an identity problem, and exposing it to the internet creates a permanent penetration-test and security-review obligation your team now owns. “Some screens for suppliers” is how a build acquires its largest attack surface.

SKILLS IT DEMANDS:

External identity and access management, application security, and consumer-grade UX for an audience that does not have to use you.

Observed:

98% reduction in supplier callbacks · designed for 100% of suppliers, not only the digitally mature

9 capability areas

Contracts Management

THE BUILD ASSUMPTION

“A document repository with metadata.”

What it actually is. Intake-to-contracting; configurable contract types and metadata mapping; approved template library; **contract hierarchy** (main, sub-agreement, renewal lineage); approval and review routing;

contextual messaging; external document exchange with counterparties; **AI playbook gap analysis on third-party redlines**; dynamic signatory identification against a signature matrix; counterparty signatory validation; native DocuSign and Adobe Sign execution plus a wet-signature path; post-signature storage and tuned search; **renewal intelligence with notice-period tracking**, configurable and auto-pilot renewal workflows, tiered expiry escalation; committed-spend tracking; RBAC with sensitive-contract marking; bulk legacy ingestion with AI metadata extraction; orchestration to a third-party CLM where one exists; and cycle-time, pipeline, expiry and spend reporting.

NEVER IN THE ESTIMATE

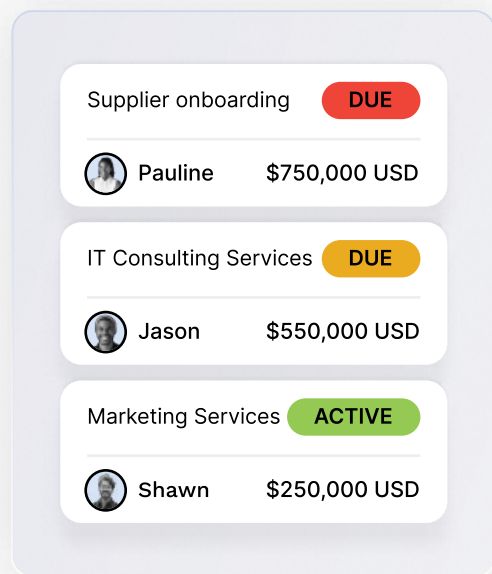
Notice-period tracking. Passive and evergreen renewals are where the money actually leaks, and catching them requires per-agreement notice windows, multi-tier escalation and an owner resolution model. Also the **signature matrix**—determining who is authorized to sign what, for which entity, at what value—which is authority policy, not a field.

SKILLS IT DEMANDS:

Legal-operations expertise, clause and playbook modelling, e-signature integration, and document AI.

Hackett world-class:

contract turnaround and sourcing cycle-time compression; indirect savings per FTE €500k → €1.5m



Sourcing

THE BUILD ASSUMPTION

“An RFQ form and a bid comparison table.”

Procurement Triage

 Pauline Powers

\$350,000 USD

 SOURCING INTAKE

DOC NAME

Brand Marketing Servi..

DESCRIPTION

Request for brand ma..

What it actually is. Sourcing intake; conversational RFx generation; self-service RFQ and n-bid flows; AI-assisted RFP management; conversational scenario analysis with supplier recommendation; autonomous negotiation; supplier collaboration and bid management including **sealed bids**; sourcing orchestration and workflow; integration to an external sourcing engine where one is the system of record; and reporting.

NEVER IN THE ESTIMATE

Sealed-bid integrity and award audit trail. Bid confidentiality is a control, not a feature—it needs enforced timing, access separation and an evidential award record that survives challenge. Getting it subtly wrong is a governance incident, not a bug.

SKILLS IT DEMANDS:

Category and sourcing expertise, bid-security engineering, and scenario/optimization modeling.

Hackett world-class:
sourcing-eligible spend routed through tools, and indirect spend influence 78% → 91%

PO Management

THE BUILD ASSUMPTION

“Push the requisition into the ERP.”

 PO 1687 | Staying Brands

PO DETAILS

Staying Brands

SERVICES

Department

MARKETING

Total

\$12,000 USD

What it actually is. Intake-to-PO creation; PO issuance; supplier order acknowledgement; **change order management**; goods and services receiving; **receipt reversals**; PO accumulators; full lifecycle management; order closure; and orchestration back into the wider flow—all under SOX-minded review and approval governance.

NEVER IN THE ESTIMATE

The amendment path. Every build ships PO creation and then discovers that real life is change orders, partial receipts, reversals and accumulators – and that “amend, don’t re-buy” has to be enforced or spend visibility collapses. Plus the SOX evidence trail, which auditors will test.

SKILLS IT DEMANDS:

ERP transaction knowledge, SOX control design, and finance-process expertise.

Hackett baseline:
\$12–13.6 to process a PO · ad-hoc PR-to-PO cycle 18 → 10 days

Invoicing

THE BUILD ASSUMPTION

“Run OCR over the invoice.”

Invoice

IT Sys. Discovery & Infra Assessment	\$30,000 USD
IT Str. Development & Execution	\$25,000 USD
Perf. Measurement & Optimization	\$20,000 USD
Total	\$75,000 USD

Verification is finished
we've come across 1 exceptions.

INVOICE DATE
Invoice date is within the acceptable invoice duration range as per the PO.

INVOICE AMOUNT
Exceeds the PO available amount (USD 50000.00) by 50%.
The invoice currency matches the PO currency.

What it actually is. Guided invoice intake with **supplier identification from an unstructured document**; intelligent PO matching and invoice classification; AI extraction; unified header and line-item management; automatic payment-term and accounting derivation; configurable approval workflows; supplier financial data management and ERP posting – with **native two-way matching and discrepancy highlighting**, and exception-only human routing.

NEVER IN THE ESTIMATE

Supplier identification. Extraction is the easy half; deciding which of 100,000 vendor records an invoice belongs to, then matching to the right PO at line level within tolerance, is the half that determines whether anything is actually touchless.

SKILLS IT DEMANDS:

Accounts-payable domain expertise, document-extraction ML, and matching-tolerance logic design.

Hackett world-class:

manual processing 73% → 21% · invoice cycle time 6 → 3 days

Cross-cutting

Agentic Workflows

THE BUILD ASSUMPTION

“Wire an LLM into our workflow tool.”

Configure Agent Instructions

LLM PROMPT

TASK:
Analyze the items listed in the “Please describe the business need” section of the Profees_Business_Need form through a procurement lens. After extracting the good th...

TEMPERATURE
0.5

MODEL NAME
o1

PROFEES_BUSINESS_NEED (PROFEES BUSINESS NEED)

PROJECT TITLE

PROJECT DETAILS

Click Test Run to review outcomes. **TEST RUN**

What it actually is. A shelf of production agents drawn into workflows by configuration – PR/PO review, AP fraud, invoice review, vendor operations review, sourcing automation, price updates – across front door, source-to-contract, supplier management and procure-to-pay; plus the ability to build your own. Underneath: **explicit automation tiering** (assist → recommend → auto-with-confirm → autonomous), exception escalation, a measure-and-improve loop, SOX-compliant governance of agent activity, model portability, and auditability of what an agent did and why.

NEVER IN THE ESTIMATE

Governance. Building an agent is a week. Governing a fleet of them in a regulated enterprise—who authorized this action, on what evidence, under which model version, with what human checkpoint, and can you prove it eighteen months later to an auditor—is the actual product. See section O6.

SKILLS IT DEMANDS:

ML engineering, prompt and evaluation engineering, model operations—plus an AI governance discipline most enterprises are only now standing up.

Observed:

~70% of PR-review effort eliminated (Novartis) · price changes in one day instead of ten (Bayer) · an agentic implementation whose headcount savings funded the programme (Pfizer)

Counts are capability areas as scoped in ORO's Solution Definition documents, not feature counts—each area contains many behaviors. Benchmark figures are attributed: Hackett Digital World Class where stated, otherwise observed ORO customer results. Neither is a guaranteed outcome for any specific organization.

The floor nobody budgets

Before a single procurement capability exists, an enterprise-grade orchestration platform needs all of the following. On a bought platform they arrive on day one and are somebody else's problem to keep running. In a build they are the first six months, and then a permanent obligation.

Five environments, four connected

Play, Dev, QA, Pre-Prod and Prod—with selective change packaging so two change programs can run in parallel without one overwriting the other. Most builds ship with two environments and discover the problem during their first parallel release.

Automated regression against a moving platform

Schedulable regression suites a customer runs against their own configuration after each weekly platform release. Without this, every upgrade is a manual test cycle you pay for out of change capacity.

Integration Fabric

An embedded iPaaS with 1,000+ pre-built connectors, REST/SOAP/flat-file, real-time and batch—SAP, Ariba, Oracle, Coupa, OneTrust, D&B, and the rest. A build starts from an empty integration platform and a license negotiation.

No-code administration RBAC & Identity

Workflow and form design, full audit trails, versioning, environment management—configurable by the business, not deployed by engineering. This is the difference between a change taking days and taking a release train.

Role Based Access Control (RBAC)

Role-based access control managed in platform, or governed by external data sources. Set up at any level within the application, this ensures process and policy compliance at every level of the organization.

Collaboration

Threaded comments, file sharing, @mentions, real-time updates, Teams and Slack, spanning internal and external participants—with retention and attachment policy controls, because suppliers are on the other end.

Localization and mobile

SSO via Okta, Entra or Google; locale normalization and recipient-language notification fallback; mobile web for approvals. Multi-country means multi-language, and multi-language means notification templates per locale with a fallback chain.

In a build, IT permanently owns procurement's change queue. That isn't a cost line—it's an operating model decision, and it's usually made by accident.

Why AI made the build harder, not easier

The intuition runs the other way, so it's worth being precise. AI has genuinely collapsed the cost of the first twenty percent of an agentic capability: a demo that classifies a request, drafts an RFP section or extracts fields from a tax certificate is now a fortnight's work. That's real, and it is exactly what makes an internal build look feasible in a steering committee.

What AI has done to the remaining eighty percent is raise it. A production agent in a regulated enterprise needs: an explicit automation tier and a defensible reason it sits there; a human checkpoint where judgment belongs; exception escalation that a person actually receives; an evidential record of what the agent did, on what input, under which model version; access scoping so the agent can't reach data its human couldn't; model portability so a provider's deprecation isn't an outage; and a cost model that survives volume. Then it needs to be re-validated every time the underlying model changes—which is now several times a year, on someone else's schedule.

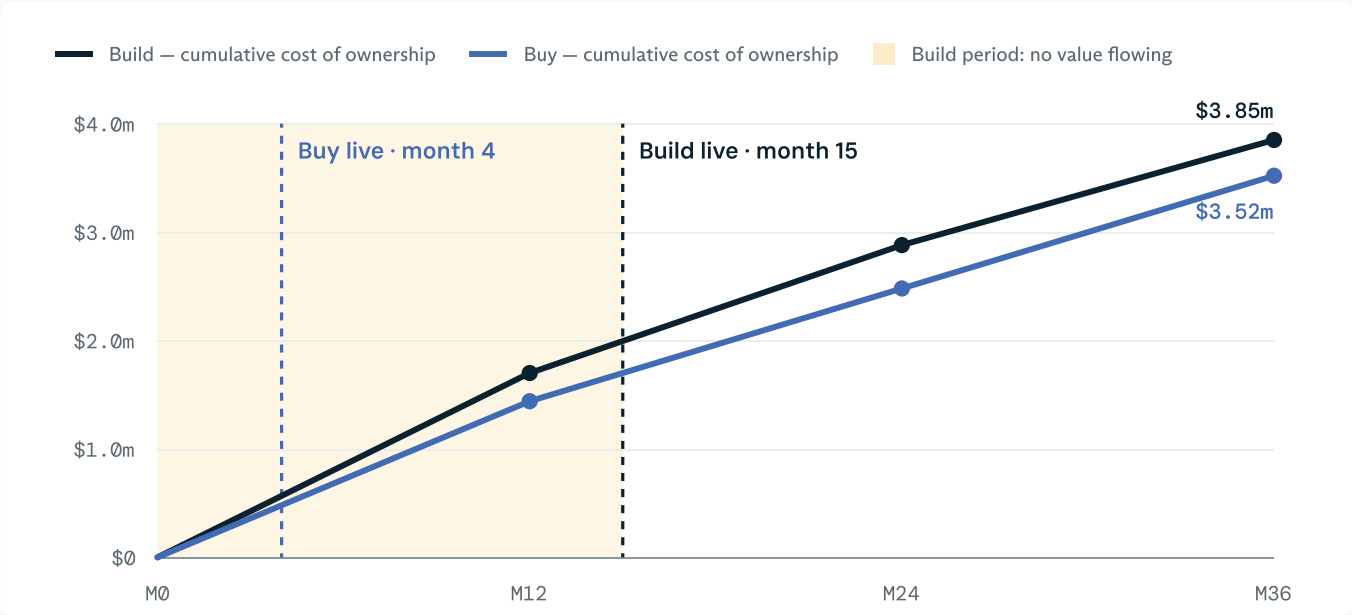
This is why the market's own framing has shifted from “can we build agents” to “can we govern them”. ORO's position is that in 2026 agents execute, orchestration governs, and people lead—and that the governing layer is the hard, durable engineering. That layer is certified and maintained as a product: ISO 42001 alignment, SOX-minded control of agent activity, human-in-the-loop by design, bring-your-own-LLM so model choice stays the customer's, and a predictable consumption model rather than raw token exposure.

An internal build inherits the opposite shape. It gets to a compelling pilot quickly, and then meets the audit, security and finance functions all at once—typically at the point where the pilot is meant to become the process of record. That is where agentic build programs stall, and the stall is expensive because the pilot has already set an expectation.

The money - and the line that decides it

Here is the uncomfortable part of an honest comparison: over three years, the modelled costs are closer than either side usually admits. A build is not obviously cheaper, and it is not obviously more expensive. If the decision were made on three-year cost of ownership alone, it would be close to a coin toss.

It is not a coin toss, because two things sit outside that comparison and both point the same way: when the value starts, and where the ongoing run-rate is heading.



Illustrative only—a modelled comparison for one global enterprise, not a quote or customer case. Costs are allocated to the periods in which they are incurred or consumed; the chart does not represent invoice timing, cash payments or a particular accounting treatment. Payment terms and the capitalisation of eligible implementation or development costs will vary by contract, accounting policy and jurisdiction. Shared third-party costs are excluded because both scenarios incur them.

LINE	YEAR 1	YEAR 2	YEAR 3	BASIS
BUILD				
Build team	\$1.28m	\$0.32m	-	8 FTE for 15 months, blended \$160k fully loaded
Permanent run team	-	\$0.54m	\$0.72m	4.5 FTE: BA, engineers, integration, agent ops
Platform, infra, model spend, tooling	\$0.15m	\$0.25m	\$0.25m	Rising with volume
Procurement design capacity consumed	\$0.27m	\$0.07m	-	1.5 FTE of senior procurement time, 15 months
Cumulative	\$1.70m	\$2.88m	\$3.85m	Run-rate still rising with scope
BUY				
External One-time implementation	\$0.50m	-	-	Indicative range \$450-550k <validate per deal>
Subscription	\$0.70m	\$0.80m	\$0.80m	Indicative ranges <validate per deal>
Internal product owner + configurator	\$0.24m	\$0.24m	\$0.24m	1.5 FTE, business-owned, no-code
Cumulative	\$1.44m	\$2.48m	\$3.52m	Contracted and predictable

11 months

of benefit the build never captures—and never gets back. A bought platform reaching production in month 4 against a build reaching it in month 15 is not a scheduling difference; it is a permanent, non-recoverable shift of the entire benefit curve to the right.

Put a number on it. On roughly \$100m of addressable indirect spend, a modest 1.5% realized leverage benefit is about \$1.5m a year—so eleven months of delay costs around \$1.4m of value that is simply never earned. That single line is larger than the entire three-year modelled cost difference between the two scenarios, and it is the line that never appears on an internal build's business case. (For an independent anchor on the order of magnitude: SAP's own intake material cites Gartner-sourced estimates of up to \$2.5m of annual savings per \$100m of unmanaged spend - their figure, not ours.)

Three further things sit outside the table, all of them on the build's side of the ledger:

The run-rate direction

The buy line is contracted and flattening. The build line is rising, because every new country, module, regulation, and model release adds permanent run cost. Year four is where the two curves separate for good.

Third-party terms

Both scenarios pay for supplier data, bank verification, e-signature and tax services. Only one negotiates them as a single enterprise buyer at aggregated volume; the other buys nine contracts at low volume and manages nine renewals.

Security and audit obligation

Penetration testing, external-facing identity, SOX evidence, agent auditability, and annual re-certification. In a build these are net-new permanent obligations of the internal team, and they are not optional.

The scarcity premium on every FTE line

The blended rate assumes you can hire these people at market. The hybrid and AI roles in section 03 are not clear at market—they carry a premium, a recruitment lag and a retention cost, and every month the roster is short is a month the schedule slips.

The queue, priced

The model assumes uninterrupted delivery. A support function's program rarely gets that. Each quarter procurement's engineers are pulled to a revenue-facing system, the run-rate keeps burning while the benefit date moves further right.

What you should build—and where

This is not an argument against building. It is an argument about which layer deserves your engineers and which layer deserves your procurement experts. The dividing line is durable:

Buy what is the same at every enterprise, build what is only true at yours.

Buy the foundation

- A procurement-native semantic data model, with suppliers, contracts, transactions and risks represented as first-class objects—not rows in a generic table
- Maintained jurisdictional truth, including tax identifier rules across 120+ countries, banking formats, bank reference data and document requirements
- Commercial data relationships spanning supplier enrichment, bank verification, fraud intelligence, sanctions and adverse media
- An external-facing supplier front door, together with the identity, fraud and security obligations that come with operating it
- An agent-governance layer covering automation tiering, human checkpoints, auditability, model portability and SOX-minded controls
- The integration substrate and environment and release machinery beneath everything
- The domain expertise, support model and implementation experience needed to put these capabilities to work
- Continuous investment that keeps the foundation secure, compliant and relevant—without your team having to maintain every layer themselves

Build what's yours

- Your category taxonomy and the routing it drives
- our policy thresholds, approval matrices and delegation model—expressed as function × entity × category so they survive reorganization
- Your buying-channel matrix and what belongs in each channel
- Your risk appetite: which domains apply to whom, at what tier, on what renewal cycle
- Your questionnaires, templates, playbooks and the clause positions you'll defend
- Your own agents, for the work that is specific to your business—dropped into workflows by configuration
- The integrations to systems only you run

The second list is where competitive advantage actually lives, and it is the list that changes most often. That is the case for putting it in a no-code layer owned by the business and promoted through connected environments in days—rather than in application code, behind a release train, in IT’s queue.

It is also the answer to the fear that buying means losing control. It doesn’t. It relocates control to the people who should have it: **the business owns intent and outcomes; IT governs the guardrails, the integrations, and the data.** IT stops being the custodian of procurement’s forms and starts being the architect of how the enterprise’s systems fit together.

The same division, applied to your people

The dividing line above is also a staffing plan, and it is the reason this argument lands differently with a CIO than a pure cost case does. Buying does not tell an IT function it isn’t good enough. It tells it to stop being an in-house procurement software company and go back to the work it is genuinely best at.

Skills you stop needing

- Backend, front-end and UX engineering for two separate applications
- Solution and data architecture from a blank page
- Jurisdictional tax and banking expertise for 120+ countries
- Agent engineering, evaluation harnesses and model operations
- An externally-facing security and identity practice
- QA automation and release engineering for five environments
- The hybrid procurement-engineer you were never going to hire

Skills you keep - and finally get to use

- **Procurement process design**, expressed directly as configuration instead of translated into a requirements document and back
- **Product ownership** of your own operating model
- **Integration engineering** - the thing IT actually wants to own, against pre-built connectors rather than from scratch
- **Data and access governance**: master data ownership, identity, policy

Two consequences follow, and both matter more than they sound. First, because your people configure rather than code, **the learning curve is days rather than quarters** and a change is reversible—which means process design can be iterative instead of a one-shot specification written before anyone has used anything. Second, **the knowledge stays**: it lives in a configured platform your team can read, not in the heads of engineers or in a departed integrator’s design documents.

Where we stop

A build-vs-buy paper that claims the platform does everything is arguing against itself. Orchestration works precisely because each specialist system keeps its craft. These are the bounds, stated plainly—they are the shape of the offer, not a weakness in it, and they are the clearest evidence that “buy the foundation, build what’s yours” is a real principle rather than a slogan.

The ERP stays the system of record	ORO governs the supplier record’s journey and prepares it; the ERP or MDG platform remains master of the vendor master, and runs three-way matching, PO execution and payment.
Two-way matching is ours, three-way is the ERP’s	Extraction, PO match recommendation and discrepancy highlighting run natively in ORO. Don’t under-claim it and don’t over-claim it.
Redlining stays in the CLM	Where a specialist CLM is in place for legal drafting, ORO owns intake, routing, negotiation orchestration, repository and renewal—and orchestrates to Icertis, Sirion, Ironclad, Agiloft, Conga, Ariba or Coupa.
Complex strategic sourcing may keep a point solution	Democratized sourcing, RFX, n-bids, scenario analysis and negotiation run in ORO; deep optimization-led sourcing legitimately belongs in a specialist engine ORO orchestrates to.
SaaS only	ORO is a hosted, multi-tenant cloud service. There is no on-premise deployment option.
Configuration, not bespoke code	The platform is configured through its no-code surfaces. Genuinely bespoke logic is a conversation about timing and cost before purchase, not an assumption after it.
Roadmap is labeled as roadmap	Forward capability is stated as forward capability. Anything not generally available is named as such.

ORO Labs - point of view, September 2026. Assembled from ORO’s Solution Definition documents (module scope), the Value Framework framework and benchmark library (Hackett Digital World Class and observed customer results), the reusable case-study set, ORO’s standing position on orchestration versus suites and versus generic workflow platforms, and the sponsored Pure Procurement build-vs-buy white paper. All pricing is indicative and is not a quote or a commitment; all benchmarks are attributed and are not guaranteed outcomes.



**What people
think**



**What it
actually takes**



About ORO Labs

ORO Labs is an agentic orchestration company on a mission to make procurement work for everyone. Founded by former SAP Ariba product leaders looking to do things differently, ORO was created to humanize procurement with effortless user experiences that shorten cycle times, mitigate risk, and accelerate value creation. The no-code platform is purpose-built for enterprise procurement complexity and supported by an extensive network of implementation and technology partners. ORO helps Fortune 500 and fast growing global leaders reinvent processes, surpass performance benchmarks, achieve unmatched agility, and scale procurement's strategic capacity for value creation — from Idea to Pay. The ORO platform is trusted by the world's largest brands, including The Coca-Cola Company, Pfizer, Danone, Roche, BASF, Liberty Global, Kyndryl, and Booking.com.