Process mining is dead

AI doesn't need a map of your business before it starts work.

It needs what every new hire needs: a task, a mentor, and permission to ask why.

[01] Discover while building.

[02] Learn through exceptions.

[03] Improve with evidence.

Kognitos

The New Hire Test

Walk into a restaurant and say you want the cashier's job. They look at your resume, agree on pay, and ask when you can start. You say Monday. On Monday they tell you to work with John for a few days, and John shows you how it's done.

Walk into a finance department and say you can do an accountant's job, and the process is much the same. They review your experience, ask a few questions, and pair you with a mentor. Within a week or two you're productive.

Now AI knocks on the enterprise door. We ask what it can do, and it answers, in effect: anything you do, and probably faster. And at that moment, the enterprise reverts to an old reflex. Before technology can start, we must first figure out exactly what people do. We commission process discovery. We mine the logs. We spend months building a map.

> Meanwhile, AI is still standing at the door.

This paper makes a simple argument. Exhaustive process mining and discovery, as the entry ticket to automation, is dead. It was a workaround for machines that could not handle what they did not know. AI that is built to investigate, ask, and learn does not need that workaround. It should be onboarded the way we onboard people: give it the task, show it the standard process, answer its questions, and let it learn on the job.

We are writing this because we believe it is true, and because too many organizations are losing a year to a ritual they no longer need.

Key takeaways

[01] **Process mining solved yesterday's problem.** It existed because automation broke on anything it hadn't been told. That constraint is gone.

[02] **Logs tell you what happened. They never tell you why.** And the why is what an automation needs to make the right call on the next case.

[03] **Onboard AI like a new employee.** A goal, access, a standard process, and a person to answer questions. That is the minimum sufficient assignment.

[04] **The non-negotiable guarantee: it doesn't make things up.** When Kognitos hits a doubt it cannot resolve, it stops and asks. People answer. It learns.

[05] **Tribal knowledge stops walking out the door.** Every answer is recorded in plain English, with its reasoning, for your people and for the next generation of AI.

What we are retiring, and what we are not

Process mining as the front door to every automation project is over. As a precision instrument for a specific question, such as a conformance audit or a targeted bottleneck analysis, it still has a place.

> The difference is who is in charge: the question, not the ritual.

[01]

The old ritual

Map everything before you move

Anyone who has run an enterprise automation program knows the sequence.

1

Instrument

Pull event logs from SAP and Oracle. Put recorders on desktops. Collect weeks of activity data.

2

Model

Out comes a spaghetti diagram: hundreds of variants, loops, and exceptions tangled together.

3

Present

A committee studies the map and asks, with genuine surprise: is this really what we do?

4

Re-engineer

Debate why the mess exists and what it should become. More workshops. More months.

5

Finally, automate

Only now does anyone start building. The map may already be stale.

This ritual has two fundamental problems.

It captures the what and misses the why. A log records that an invoice was held, that two receipts were combined, that an approver overrode a match. It does not record why. Was the override a policy exception, a one-time judgment call, or a mistake? The why is precisely what an automation needs to handle the next case correctly, and it is exactly what the logs cannot give you.

It burns time the business does not have. Every month spent mapping is a month the AI is not working. The map is often stale by the time it is approved, because the process has already moved on.

Why the ritual existed in the first place

The ritual was not foolish. It was a rational response to the technology of its time.

Traditional RPA follows predefined paths. When it meets a variation nobody anticipated, it breaks. Even UiPath documents the fragility of selector-based automation and the recovery mechanisms it has had to build around it.[17] If your automation shatters on the unexpected, you have no choice but to catalogue every possibility upfront.

Early LLM-based automation introduced the opposite failure. Instead of breaking on a missing rule, it could invent one, producing a decision that was plausible and wrong. NIST classifies this confabulation as a risk that requires explicit evaluation and controls.[10] Once again, the defensive response was to specify everything in advance.

> The real problem was never a lack of process maps. It was automation that could not handle doubt. Fix that, and the ritual loses its reason to exist.

[02]

A better model

Hire AI the way you hire people

Think about what a good new employee actually needs on day one. Not a complete map of every variation of the job. They need a clear goal; the authority and access to do the work; the standard process, or someone to watch perform it; a sense of what good output looks like; and a mentor who can answer questions.

That is the minimum sufficient assignment. It is enough for a person to start contributing, and it is enough for well-designed AI.

A good new hire does not wait to be productive until they know every edge case. They do what they have been taught. They look things up. And when they hit something they cannot resolve, they ask. Over a few weeks, the questions become rarer and the work becomes routine.

The guarantee that makes it work

There is one condition, and it is not optional. You extend trust to a new employee because you know they will not make things up. When they are unsure, they stop and ask.

AI must offer the same guarantee. It must recognize missing evidence, conflicting rules, and unsupported cases, and it must halt the affected action rather than guess. This cannot be a line in a prompt that says “ask if unsure.” It has to be engineered behavior. Anthropic’s guidance on building effective agents makes the same point: agents should seek human feedback at blockers and run with clear stopping conditions. [3]

Kognitos is built around this guarantee. When it cannot justify the next step, it asks. A person answers. It learns from the answer and continues.

> An automation should be judged like a good new hire: not by how much it was told in advance, but by what it does when it doesn't know.

[03]

Discover while building

Quill: the new hire who reads everything on day one

Kognitos Quill is the English-as-Code builder that turns a high-level goal into an executable process.[13] It is where onboarding happens.

Give Quill a goal and authorized access, and it does what a diligent new hire would do in their first week, only faster. It explores the connected ERP or CRM: the relevant tables, fields, relationships, and sample records. Then it does the equivalent of watching John work: it examines a focused sample of recent transactions and how they were adjudicated, and infers how the job is actually done. From that, it constructs the steps, tests them, and asks a pointed question only when a necessary detail remains unresolved.[14]

Compile once. Execute many times.

Quill's exploration happens once, when the automation is built. Every runtime execution reuses the resulting mappings and steps against current data. Quill does not rediscover your schema or rebuild the process for each transaction.[14]

The underlying techniques are proven. LangChain demonstrates agents that inspect schemas, retrieve sample rows, and refine queries; Anthropic documents on-demand tool discovery.[1, 2] What Kognitos adds is the discipline around them: exploration directed at a specific goal, bounded by authority, and paired with the guarantee to ask rather than assume.

A new hire who helps write the job description

Here the analogy gets more interesting, because Quill does not just follow instructions. It helps author them.

With access to ERPs, databases, knowledge bases, context graphs, and documents, Quill can investigate how work should be done and propose the instructions itself. People set the objective and settle the genuinely ambiguous questions. They no longer have to specify every implementation detail by hand.

On work dominated by searching and cross-referencing information, which describes much of finance and operations, this is where AI can outpace a human new hire. It can compare system structures, recorded outcomes, and written policies in a single pass and present what it found for review.

And like a sharp new employee, it asks why. Why is this validated here and not earlier? Why is this data entered twice? Those questions lead directly to better ways of working. Repeated exceptions might point to a missing validation step upstream. Duplicate entry might point to a direct lookup. Quill can propose a revised sequence with the evidence behind it, and the process owner decides whether to adopt it.

> Process re-engineering starts on day one, grounded in the actual work, instead of after a year of mapping.

[04]

Capture the reasoning

The why is the asset

A schema tells you how data is structured. A log tells you what happened. Neither tells you what a decision meant or why it was right. That knowledge usually lives in one place: people's heads.

This is the quiet cost of tribal knowledge. An experienced accountant knows why partial deliveries are combined, which vendors always need a second look, and when a mismatch is acceptable. When that accountant leaves, the knowledge leaves with them.

Kognitos changes that. In English as Code, every rule carries its reasoning alongside its action.[16] Here is what that looks like:

An English-as-Code rule

```
Reconcile against the quantity received. Add receipts
belonging to the same order line, because deliveries
may arrive in parts and matching should recognize the
full quantity without counting unrelated goods.
```

One rule captures what to do, how to do it, and why. The why makes the rule reviewable by a human, portable to similar situations, and safe to question when circumstances change. If Kognitos observes an outcome but cannot find the reason for it, it asks, because an outcome alone does not justify a rule.

> Institutional memory, written in plain language, readable by your people and usable by the next generation of AI. Knowledge that used to walk out the door now accumulates.

Your existing knowledge is a head start, not a prerequisite

Most enterprises already hold valuable context in knowledge bases, policy documents, semantic layers such as Snowflake's semantic views, [5] and increasingly, context graphs that link entities to policies, evidence, and past decisions. [15] Quill can draw on whatever is available to locate established mappings and applicable policies faster.

None of it needs to be complete before work begins. Context grows as the automation runs, from validated discoveries and approved answers. The rule is simple: check where knowledge came from and whether it is still current, and never treat one past exception as permission for the next.

Question during the build	How Quill answers it	When it asks a person
Which data matters?	Inspects schemas and samples; verifies existing business mappings.	A definition is ambiguous or inaccessible.
How do records connect?	Follows recorded relationships; checks keys and examples.	A relationship cannot be verified.
Which steps achieve the goal?	Inspects actions and policies; builds and tests a sequence.	Requirements conflict or are undocumented.
Does a past decision apply?	Checks its evidence, policy version, approval, and scope.	Authority is missing or the case is materially different.

Table 1. How Quill resolves questions during the build, and when it asks for help.

[05]

Learn through exceptions

Astral: learning on the job

A new hire's real education happens on the job, through the cases that don't fit the manual. Kognitos works the same way.

When a question comes up during the build, Kognitos asks it precisely, stating what it already checked:

> "These two receipts cover the invoiced quantity, but the policy does not say whether they may be combined. Is aggregation permitted for this order type?"

That is the kind of question a good employee asks: specific, informed, and easy to answer. Where authority is missing, Kognitos asks for approval. It never infers permission.

At runtime, the compiled process handles the standard work. When a case falls outside it, the exception goes to the Kognitos workbench, where Astral resolves it together with a person. Astral asks why the person chose that resolution and records the reasoning. As patterns emerge across exceptions, Astral documents them as rules. People review and approve those rules before they are enabled in auto-mode for future cases.[16] One-off judgments stay one-off; reusable answers become reusable instructions.[9, 13]

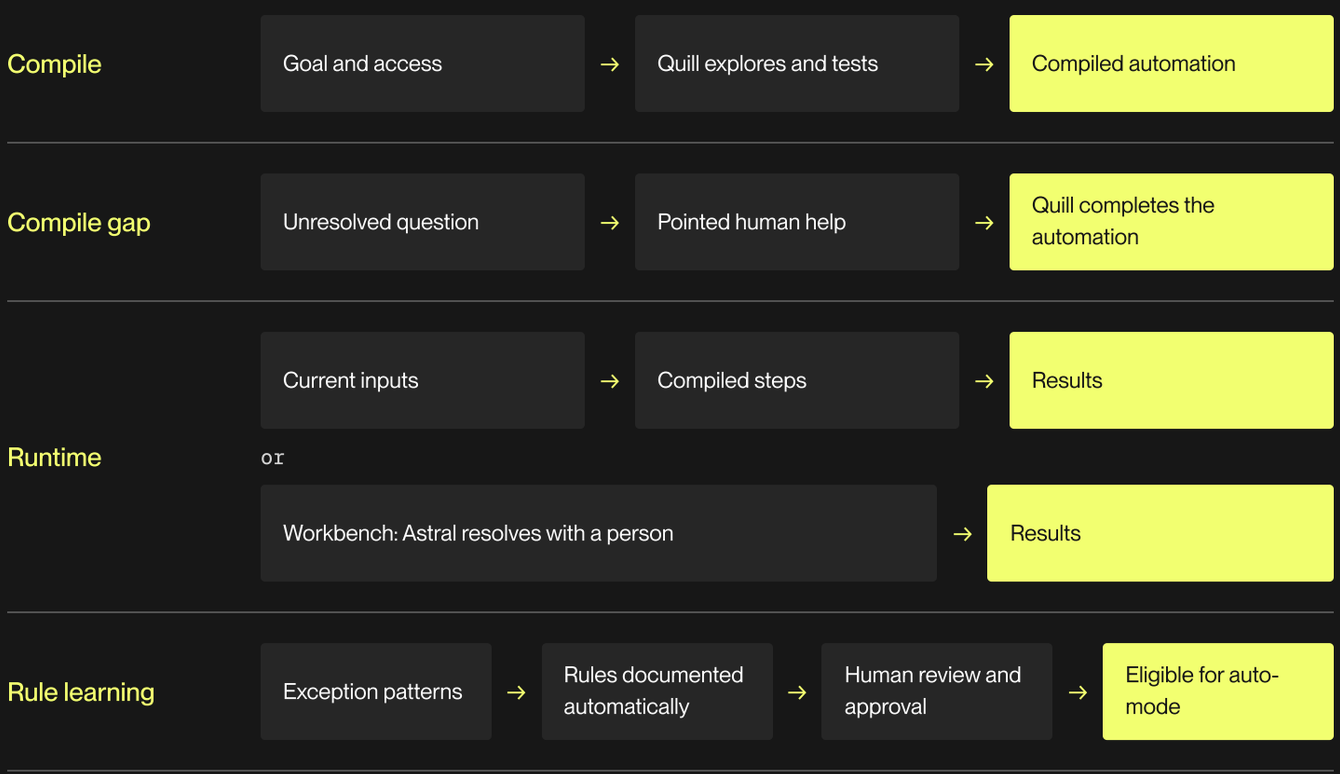

Figure 1. The Kognitos lifecycle: discovery during the build, learning on the job.

> This is process discovery at its highest caliber: grounded in real cases, explained by the people who know, and turned into rules that humans have approved.

Every rule keeps its evidence, scope, approval, and version. Recording a rule does not switch it on; approval does. That matters, because a deterministic system applies a mistaken rule perfectly consistently. The review step is where human judgment stays in charge.

[06]

In practice

Invoice reconciliation, no mining required

Accounts-payable reconciliation: how Kognitos approaches it	A design illustration of a typical task.
The assignment	An accounts-payable team wants invoices reconciled against purchase orders and receipts. They give Kognitos authorized access and their matching policy. They do not provide a table map or a reconstruction of historical workflows. Releasing payments is explicitly outside the assignment.
The build	Quill inspects the schemas and integration definitions, identifies the relevant records, and verifies how they relate. It reviews recent accepted, rejected, and escalated transactions, along with the evidence behind each decision. It checks what it inferred against the written policy, then builds and tests the retrieval, matching, and recommendation steps. [14]
The first question	A sample invoice appears to exceed its receipt. Quill finds a second receipt on the same order line. If the policy allows aggregation, Quill builds that in. If the policy is silent, it asks the one focused question shown above, and moves on once it has the answer.
Going live	The compiled steps run against current records every day. Anything the process cannot resolve goes to the workbench, where Astral and the AP team handle it and capture the reasoning.
Day-one re-engineering	If exceptions keep arising from missing receipt references, Kognitos can propose validating those references earlier in the flow. The process owner decides whether that reduces exceptions without blocking legitimate invoices. No one had to map the rest of the purchasing organization to get there.

[07]

The objection

"But doesn't AI need process intelligence?"

> Yes. We agree completely. The disagreement is about where that intelligence comes from.

Wil van der Aalst, the founding figure of process mining, argues in No AI Without PI that organizational AI must be grounded in process intelligence. [8] He is right about grounding. Where we part ways is the assumption that the grounding must be assembled upfront, through mining, before AI can begin.

Process intelligence can be gathered the way a new employee gathers it: by inspecting the systems, reading what already exists, and asking the people who know. That intelligence is more targeted, because it is tied to the task. It is richer, because it includes the why. And it keeps growing, because every resolved exception adds to it.

Complex, many-to-many business relationships do not change the argument. Standards such as OCEL 2.0 model how an order links to many deliveries and invoices, [7] and those representations are useful when available. They do not require an exhaustive mining exercise before each automation.

Even the IEEE Process Mining Manifesto advises starting from questions rather than extracting everything. [6] We take that advice to its logical conclusion. When a specific question genuinely warrants broad process analysis, commission it. Just don't make it the gate every project has to pass through.

[08]

Prove it

Don't take our word for it

We are confident in this approach, and we think the right way to settle the question is on your own work, with your own data, under conditions you control.

The public research shows why a real-world test matters. Spider 2.0 shows how demanding enterprise data tasks become once real metadata and documentation are involved.[4] τ-bench tests whether agents follow domain policies consistently across repeated trials.[11] METR shows that performance on software tasks does not automatically transfer to every kind of business work.[12] Benchmarks are useful, but your processes are the only benchmark that counts.

The side-by-side test

Take one real assignment. Give a discovery-first approach and Kognitos the same access to the same systems, policies, and documentation. Measure the build separately from runtime. Include the hard cases: ambiguous rules, split records, missing authority. Start in shadow mode, and agree on acceptance thresholds before expanding what the automation is allowed to do.

Measure	What it shows
Time and total effort	Preparation, discovery, review, and corrections, through to accepted operation.
Verified outcomes	Correct, authorized results across eligible cases, with sample size and exclusions stated.
Independent discovery	Details resolved correctly without human help, compared with a human onboarding baseline.
Handling doubt	Correct pauses, missed ambiguities, unnecessary questions, and the human effort involved.
Learning and redesign	Accuracy of approved rules over time, and the tested effect of proposed process changes.

Table 2. The scorecard for a side-by-side evaluation.

We hold ourselves to a strict standard here. Fewer questions only count if outcomes stay correct. Unresolved cases and failures count as much as completed transactions. And a short, clean demo proves nothing about dependable operation.

That is exactly why we recommend measuring properly.

Conclusion

Stop mapping. Start onboarding.

The playbook is the same one you already use for people. Give AI enough authority and context to start. Let Quill investigate the systems and draft the process. Let Astral and your team handle the exceptions, and capture the why every time. Review rules before they run on their own, and approve redesigns before they change the process.

Process intelligence doesn't have to come before useful work. It builds up as the work gets done, and it stays with your organization.

AI is already at your door. Stop asking it to wait while you draw a map. Hand it the task, introduce it to John, and let it ask its questions.

> Process mining had its era. The organizations that move fastest with AI will treat it less like a machine that needs every instruction on day one, and more like a capable new colleague.

Evidence

References

Reviewed September 27, 2026. Product details draw on Kognitos architecture notes [14] and [16]. The invoice scenario is a design illustration; section 08 describes how we recommend measuring results on your own processes.

[1] LangChain. Build a SQL agent. Official documentation; schema inspection, table selection, query execution, and error recovery.

[2] Anthropic (2025). Introducing advanced tool use on the Claude Developer Platform. November 24.

[3] Anthropic (2024). Building effective agents. December 19.

[4] Lei, F., et al. (2024). Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows.

[5] Snowflake. Cortex Analyst. Official documentation; semantic views connect business concepts to physical data.

[6] IEEE Task Force on Process Mining (2011). Process Mining Manifesto. Question-led extraction, p. 7.

[7] Berti, A., et al. (2023). OCEL 2.0 Specification. Version 2.0, October 16.

[8] van der Aalst, W. M. P. (2025). No AI Without PI. Keynote preprint.

[9] Kognitos (2026). How Conversational Exception Handling Transforms Process Automation. May 5.

[10] Autio, C., et al., NIST (2024). Generative Artificial Intelligence Profile. NIST AI 600-1.

[11] Yao, S., et al. (2024). τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains.

[12] METR (2026). Task-Completion Time Horizons of Frontier AI Models.

[13] Kognitos. Platform architecture, and Binny Gill, Quill as the English-as-Code builder. Accessed September 27, 2026.

[14] Kognitos Quill architecture note, September 2026. Quill explores ERP or CRM structures, sample records, and recent transaction adjudications at compile time to construct steps for a high-level goal; that exploration is not repeated on each runtime execution.

[15] Htet, E., Neo4j (2026). What is a context graph? August 4.

[16] Kognitos English as Code, Astral, and workbench architecture note, September 2026. English as Code rules preserve rationale alongside actions and steps. Astral resolves exceptions with human help and documents emerging rules. User review and approval precede auto-mode.

[17] Seju, R., UiPath (2025). How UiPath Healing Agent solves UI automation's biggest challenges. July 22.

Kognitos

AI is at your door. Put it to work.

Bring one real process, a goal, and authorized access. We'll run the side-by-side test from this paper against a discovery-first approach, in shadow mode, on your terms.

kognitos.com