SINCE 1994

Workflow *or* agent?

A DECISION FRAMEWORK BEFORE YOU ARCHITECT ANYTHING · REX BLACK, INC.

Most "agents" are *workflows* that overshot.

Autonomy is expensive in three measurable ways.

Debug cost compounds with run length.
Observability cost is a prerequisite, not an upgrade.
Drift cost of a system that picks its own next step is

DEFINITIONS

Two shapes. *Both* valuable.

WORKFLOW

Deterministic pipeline.
LLM steps are **local**: extract, classify, summarize.
Control flow lives in **code.**

AGENT

LLM picks the **next step.**
Has tools. Invokes them in an order it chooses.
Control flow lives in the **model.**

Can the steps be *enumerated in advance?*

If the solution path is knowable at design time, **the workflow wins.**

Agents earn their keep when the path is genuinely variable. Most enterprise tasks are not.

Test. Write the solution path on a whiteboard. If you can, it is a workflow.

What is the *blast radius* of a wrong next step?

Agents fail *distinctively:* **they do the wrong thing efficiently.**

A workflow dead-ends with an exception.
An agent dead-ends, tries another tool, writes to something it should not have touched.[3]

Low blast radius. Autonomy is cheap.
High blast radius. Autonomy is a contingent liability priced in incidents.

How *observable* is the decision trace?

WORKFLOW

Observable **by construction.** Steps are in code. Logs are enough.

AGENT

Requires a separate **observability layer:** trace capture, tool-call replay, decision-point annotation.

Build it before the agent.

Unit economics at *full volume.*

At 100/day, the cost difference is noise.

At **100,000 requests/day,** agents cost 2 to 4x more per decision and the surface is harder to bound.

Workflow first is not ideology. It is an economic argument, with the chart to prove it.[7]

The earned-autonomy *principle.*

Build the workflow. Accept 85 to 95 percent coverage.
Instrument the failures. Capture inputs, intermediates, low-confidence paths.
Look at the distribution. Clustered? Solve in the workflow. Long-tail? Evidence for an agent layer.
Add a bounded agent, not a replacement. Narrow allowlist. Explicit territory.

WORKED EXAMPLES

Three *real* decisions.

DOC PROCESSING · 50K/MONTH

Workflow. Fields enumerable. Blast radius bounded. If a doc type fails, *narrow* agent layer for that type.

SUPPORT TRIAGE · 8 QUEUES

Workflow. LLM as classifier. Sequence is fixed: classify, score, route, log. No agent needed.

IT OPS REMEDIATION, 15 SYSTEM TYPES

Agent, cautiously. Tool allowlist, step caps, mandatory human approval on writes, long-running eval on traces.

Failure modes by path.

WORKFLOW

LLM step **swallows** the error.
Pipeline **ossifies** past 92%.
Prompt file becomes **unversioned product.**

AGENT

Observability deferred → rewrite in 6 months.
Unbounded retries → linear cost in failure rate.
Policy drift → system prompt edits that nobody audited.

This week.

Re-scope one in-flight "agent" project as a workflow. Walk through the four questions.
Audit observability on any production agent. If "why did it do X" is unanswerable, the layer is insufficient.
Name one workflow candidate for earned autonomy. Look at the failure distribution.

REX BLACK, INC.

Workflow *first.*
Agent earned.

REXBLACK.COM/RESOURCES/WRITING/WORKFLOW-OR-AGENT