



AI agents and workflow orchestration

Designing supervised systems that can reason across tasks and take bounded actions.

WORKS WITH THE ASSISTANTS TEAMS ALREADY USE



EDITION 02 / EMBER & ROSE

EXECUTIVE WORKING EDITION - 08 OCT 2026

02

AT A GLANCE

Designing supervised systems that can reason across tasks and take bounded actions.

An AI agent is a system in which a language model decides which step to take next, uses tools to act, and loops until a task is done or it hands off. That flexibility is useful for messy, multi-step work and risky when actions are not bounded. This playbook explains when an agent is the right design and when a fixed workflow is better, how to bound permissions, where humans must approve, and how to test behavior across whole runs rather than single answers.

WHO IT IS FOR Operations, product, technology and risk leaders deciding whether and how to let AI systems take multi-step actions inside business processes.

TIME TO USE 60 minutes to read; one design session to apply

What you will have at the end

- A clear rule for choosing between a fixed workflow, an assisted workflow and an agent
- A permission and approval design for one candidate process
- A test plan that evaluates whole runs, including failure paths
- An ownership model for supervising agents in production
- A 90-day plan from supervised pilot to a scale decision

Contents

01	What agents are, and why the word needs care	03
02	Where agents fit and where they do not	04
03	How agent systems work, in plain language	06
04	Operating model, roles and ownership	07
05	Risks, controls and human review	09
06	Measuring quality and value	10
07	Implementation guide: from read-only to bounded action	12
08	Common pitfalls, an illustrative scenario and vendor questions	14
09	A first quarter that produces evidence	15
10	Terms used in this publication	16

EXECUTIVE BRIEF

What agents are, and why the word needs care

01



Agent is used for everything from a chatbot with a tool to a system that runs a process end to end. The useful distinction is who decides the next step.

In a conventional automated workflow, a designer decides the sequence in advance: receive form, check fields, route to queue. In an agent, a language model decides at run time which step to take, which tool to call and when it is finished. The model reads the situation, plans, acts through tools such as search, a database query or an email draft, observes the result and continues. This allows the system to handle variety that a fixed flowchart cannot, such as a case where the needed documents differ each time.

The same flexibility is the main source of risk. A fixed workflow fails in ways you can enumerate. An agent can fail in ways nobody designed, such as looping, choosing the wrong tool, acting on a misread instruction or following text hidden in a document it was asked to read. For that reason the central design question is not how autonomous the system can be, but how much autonomy each step deserves given what could go wrong.

- Orchestration means coordinating steps, tools, data and people, whether or not a model chooses the path
- Most business value comes from workflows with model steps inside them, not from open-ended agents
- Autonomy should be earned in stages: suggest, then draft, then act with approval, then act within limits
- Every action an agent can take should be listed, bounded and reversible where possible

DESIGN RULE

Use the least autonomy that achieves the outcome. If a fixed sequence with one model step solves the problem, build that. Reach for an agent only where the path genuinely varies.



WORKING QUESTION

For your candidate process, is the path truly different each time, or does it only look that way because nobody has written the rules down?

WHERE IT FITS

Where agents fit and where they do not

02

Agents suit variable, multi-step work with checkable results and tolerable error. They are a poor fit for high-stakes actions that cannot be undone.

Think of a spectrum of designs. At one end is a deterministic workflow with no model. In the middle are workflows with model steps, such as classify, extract or draft, where the path is fixed. At the far end are agents that choose their path. Moving right buys flexibility and costs predictability, testability and often speed and cost. Most organizations should spend most of their effort in the middle.

DESIGN	BEST WHEN	MAIN RISK
Fixed workflow, no model	Rules are stable and inputs are structured	Brittle when inputs vary
Workflow with model steps	Path is fixed, but some steps need language understanding	Step errors compound; needs per-step checks
Single agent with tools	Path varies and each case needs research or lookup	Wrong tool use, loops, unbounded cost
Multiple cooperating agents	Distinct roles with clear handoffs justify the added complexity	Hard to test and debug; error propagation
Human-led with agent assist	Stakes are high and judgment is central	Over-trust in suggestions

CANDIDATE TASK	SUITABILITY	REASON
Research a supplier and prepare a briefing	Strong	Read-only, reviewable, path varies
Triage inbound requests and prepare a draft response	Strong with approval	Bounded actions, human sends
Reconcile exceptions across two systems	Moderate	Needs strict permissions and logging
Issue refunds or payments autonomously	Weak until proven	Financial action; needs limits and approval
Change production configuration or customer records unsupervised	Not suitable	Hard to reverse; high blast radius

COMMON MISTAKE

Starting with an agent that can write to systems. Begin with read-only agents that research and recommend. Add write actions one at a time, each with its own approval and test.

WORKING QUESTION

Which step in your process requires judgment about what to do next, and which steps are simply rules you have not documented?

HOW IT WORKS

How agent systems work, in plain language

03

An agent is a loop around a model: look, decide, act, observe. Understanding the loop shows where to place limits.

An agent system has a few parts. A model reads instructions and the current state. A set of tools lets it do things: search a knowledge base, query a record, call an internal service, draft a message. Memory holds what has happened so far in this run and sometimes facts retained between runs. A controller, written by your team, runs the loop, enforces limits and decides when to stop or hand off. The controller is where most reliability comes from, not the model.

01 Receive the goal

The system gets a task, relevant context and a list of what it is allowed to do.

02 Plan the next step

The model proposes an action, such as look up a customer record or retrieve a policy.

03 Act through a tool

The controller checks the action against permissions, then executes it.

04 Observe and update

The result is added to the working state and the model decides whether it is done.

05 Finish or hand off

The system returns a result, asks for approval, or escalates to a person with a summary of what it tried.

Design choices that matter

- Tool design: narrow, well-described tools with clear inputs outperform one powerful tool that can do anything
- Permissions: give the agent the access of a trainee with a specific job, not the access of an administrator
- Limits: cap steps, time, cost and the number of records touched per run

- State and logging: record every plan, tool call and result so a run can be replayed and audited
- Stop conditions: define when the agent must stop, ask or escalate, including when it is uncertain



PROMPT INJECTION

If an agent reads emails, web pages or documents, text inside them can try to give it instructions. Treat all retrieved content as untrusted data, never as commands, and limit what the agent can do after reading it.

WORKING QUESTION

If your agent misread a document and acted on it, what is the worst thing its current permissions would allow?



OPERATING MODEL

Operating model, roles and ownership

04



An agent in production is an operational service with an owner, a runbook and an on-call path, not a one-time project.

Agents behave more like a new team member than a software release. They need a manager, a defined scope, a way to be corrected and a record of their work. Without an owner, small drift accumulates: tools change, documents change, models are updated by the provider, and the agent quietly gets worse. Assign ownership before the first pilot and write down what the owner does each week.



ROLE	RESPONSIBILITY	TYPICAL ARTIFACT
Process owner	Outcome, scope and acceptance of the agent's work	Process definition, quality bar

ROLE	RESPONSIBILITY	TYPICAL ARTIFACT
Agent product owner	Backlog, behavior changes, release decisions	Change log, release notes
Engineer or integrator	Controller, tools, logging, deployment	Architecture, runbook
Risk and security reviewer	Permissions, data, third-party exposure	Approval record, threat notes
Supervisor or reviewer	Approves actions, handles escalations	Review queue, decision log
Evaluation lead	Test suites, regression checks, monitoring	Test report, dashboards

Supervision patterns

- Approve before act: the agent prepares an action and a person confirms it, suited to early stages and consequential steps
- Act then review: the agent acts within limits and a person samples results, suited to low-risk, reversible steps
- Exception only: the agent handles routine cases and escalates anything outside defined bounds
- Always visible: every run is logged and searchable, regardless of pattern

Define the escalation contract clearly. When the agent hands off, what does the person receive? A good handoff includes the original request, what the agent did, what it found, what it could not do and why, and a recommended next step. A bad handoff dumps a transcript. The quality of handoffs determines whether supervisors trust the system or ignore it.

WORKING QUESTION

Who gets paged, and what do they see, when an agent does something unexpected at 2 a.m.?

RISKS AND CONTROLS

Risks, controls and human review

05



Bound the actions, protect the data, log everything and keep a person accountable for any action that matters.

Human review works only when it is designed to be effective. A supervisor who sees fifty approvals an hour will click through them. Keep approval requests rare, specific and well summarized: show what will change, in which record, why the agent proposes it and what evidence it used. Measure the approval queue itself. If almost every request is approved within seconds, either the agent is reliable enough for a lighter control or the reviewers have stopped looking, and you should seed a few known-bad requests to find out which.



Agent risks combine the usual language model concerns with the risks of taking actions. Governance frameworks such as the NIST AI Risk Management Framework and ISO/IEC 42001 are useful vocabularies, and regulations such as the EU AI Act and data protection law may impose duties depending on use. This playbook is not legal advice. Check current requirements for your sector, then express them as concrete limits on what each agent can do.

RISK	EXAMPLE	CONTROL
Excess permissions	Agent can edit records it only needs to read	Least privilege; separate read and write tools
Runaway loops	Repeated calls that consume budget or hammer a system	Step, time and cost caps; rate limits
Prompt injection	Hidden instruction in an inbound document	Treat content as data; restrict actions after reading untrusted input
Wrong action	Misidentifies a customer and updates the wrong record	Confirm identifiers; approval for writes; undo path
Compounding errors	Early mistake carried through later steps	Checkpoints and validation between steps
Opaque behavior	No one can explain why it acted	Full run logs and replay



Control checklist before any write action



- The action is on a documented list of permitted actions
- Credentials are scoped to this agent and can be revoked quickly
- Identifiers and amounts are validated by code, not only by the model
- A person approves actions above a defined threshold
- The action can be reversed, or the risk of irreversibility is accepted explicitly
- Every attempt, success and failure is logged with a run identifier
- A kill switch exists and has been tested

EXAMPLE (ILLUSTRATIVE)

An agent drafts supplier payment holds. It may read invoices and the vendor list, and propose a hold. Only a finance clerk can apply it. Amounts above a set limit require a second approver. The agent never holds credentials for payment systems.

WORKING QUESTION

Which actions in your process are irreversible, and what approval protects each one?



MEASURING VALUE

Measuring quality and value

06

Judge agents by outcomes across whole runs, including failures, cost and the effort left for people.

A single good answer tells you little about an agent. You need to know how often it completes the task correctly, how it fails, what it costs per run and how much supervisor time it requires. Build a test suite of realistic scenarios, including easy cases, hard cases, missing information, conflicting data and adversarial inputs. Run each scenario several times, because the same

input can lead to different paths.

METRIC	DEFINITION	WHY IT MATTERS
Task completion rate	Share of runs reaching a correct final outcome	Core effectiveness
Correct escalation rate	Share of out-of-scope cases handled off properly	Safety of boundaries
Wrong-action rate	Share of runs with an incorrect action attempted or taken	Risk exposure
Steps and cost per run	Tool calls, model usage and time	Unit economics
Supervisor touch time	Minutes of human effort per case	Real savings
Cycle time	Elapsed time from request to resolution	Customer and operational impact

Evaluation approach

- 01 Record the baseline**
Measure how the process runs today: cycle time, touch time, error and rework rates.
- 02 Build scenarios**
Assemble thirty or more real cases with known correct outcomes, and add deliberate edge cases.
- 03 Test in a sandbox**
Run with mocked or read-only tools first, score outcomes and inspect the traces.
- 04 Shadow mode**
Run alongside people without acting, compare decisions, and review disagreements.
- 05 Limited live release**
Enable actions for a small share of cases with approval, and monitor daily.

ILLUSTRATIVE FIGURES ONLY

If a case takes a clerk 20 minutes and the agent prepares it so that review takes 6, the saving is 14 minutes per case. If one in ten cases needs 30 minutes of rework, include that. Use your own measured numbers.

Watch for regressions. Model updates from a provider, new document formats and changes to connected systems can all alter behavior. Re-run the full scenario suite before every change and on a schedule, and keep a record of results so you can show the trend over time.

 WORKING QUESTION

What is your acceptable wrong-action rate, and who agreed to it?

IMPLEMENTATION GUIDE

Implementation guide: from read-only to bounded action

07



Build trust in stages. Each stage adds capability only after the previous one has met its test.



01 Map the process

Document the real steps, decisions, systems, exceptions and who does what today. Mark which steps vary.

02 Choose the design

Use fixed steps where possible, with model steps for language tasks. Add an agent only for variable parts.



03 Define tools and permissions

List each tool, its inputs and its limits. Start read-only. Document any write action and its approval.

04 Build the controller and logging

Implement step caps, validation, stop conditions and a complete run log before adding features.

05 Test, shadow, then pilot

Use the scenario suite, shadow mode and a limited live release with approvals.



06 Operate and improve

Review logs weekly, update scenarios from real failures and widen scope only when metrics hold.

Readiness for a supervised pilot

- Process map and baseline agreed with the process owner
- Permitted actions listed, with thresholds for approval
- Scenario suite with a written pass mark
- Run logging and replay working
- Supervisors trained, with a clear escalation path
- Kill switch and rollback tested
- Data and security review completed



Staged autonomy

STAGE	AGENT MAY	PERSON DOES
1. Recommend	Research and propose	Decide and act
2. Draft	Prepare the action in full	Review and execute
3. Act with approval	Execute after confirmation	Approve each action
4. Act within limits	Execute small, reversible actions	Sample and handle exceptions



Resist skipping stages because early results look good. Moving from stage two to stage three changes the failure consequences, not just the workload. Require evidence from the previous stage, such as a stable wrong-action rate over several weeks, before each promotion, and make demotion equally easy if metrics slip.

WORKING QUESTION

What evidence would you need from stage two to feel comfortable authorizing stage three?



PITFALLS AND VENDORS

Common pitfalls, an illustrative scenario and vendor questions

08

Most agent failures come from unclear scope, excess permission and missing supervision rather than model weakness.

- Building an open-ended agent for a process that a fixed workflow could handle
- Granting broad credentials for convenience during a pilot, then leaving them in place
- Testing only happy paths and single answers
- Ignoring cost per run until the invoice arrives
- Treating logs as optional until the first incident
- Splitting work across many cooperating agents before one agent works reliably

ILLUSTRATIVE SCENARIO (HYPOTHETICAL)

A logistics firm builds an agent to resolve delivery exceptions. In testing it looks excellent. In production it misreads a carrier email that contains instructions in a signature block and re-books a shipment. Because it had write access to booking, the error reaches a customer. The firm restores a read-only design, adds approval for bookings and a check that rejects instructions found in inbound content.

VENDOR QUESTION	WHAT TO LOOK FOR
Can we see and export complete run traces?	Step-level logs of prompts, tool calls and outputs
How are permissions scoped	Fine-grained, revocable credentials, not shared admin keys per agent?
What limits can we set on steps, spend and actions?	Configurable caps and approval gates
How do model updates reach us?	Notice, version pinning or a way to re-test before change

VENDOR QUESTION	WHAT TO LOOK FOR
Where does data go and how long is it kept?	Documented regions, retention controls, contract terms
How do we leave?	Exportable definitions and logs; no hidden lock-in



Build, buy or combine

- Process is common to many firms, so a packaged product may fit
- Process is distinctive, so custom orchestration may be worth the cost
- Your team can operate and test the system after delivery
- Integration with identity, logging and existing systems is proven
- Total cost is modeled including supervision time

WORKING QUESTION

If your vendor changed the underlying model next month, how would you know whether your agent still behaves acceptably?



30 / 60 / 90

A first quarter that produces evidence

09



Progress from a read-only, supervised pilot to bounded action only as the evidence supports it.

First 30 days: ground it

- Map one candidate process, with baseline cycle time and touch time
- Decide fixed workflow, model steps or agent for each part
- List tools and permissions, read-only first
- Name the process owner, product owner and risk reviewer



Days 31 to 60: test it

- Build the controller with step caps, validation and full run logging
- Create a scenario suite of thirty or more cases with edge and adversarial inputs
- Run in sandbox and shadow mode, review every disagreement
- Train supervisors and test the kill switch

Days 61 to 90: decide

- Launch a limited supervised pilot with approval for actions
- Review metrics weekly: completion, wrong action, cost, touch time
- Promote or demote autonomy stage on evidence
- Decide whether to scale, change or stop

GLOSSARY

Terms used in this publication

10

TERM	MEANING
AI agent	A system where a model chooses steps and uses tools to pursue a goal.
Orchestration	Coordinating steps, tools, data and people into a managed process.
Tool	A defined function an agent can call, such as search or a record lookup.
Controller	The code that runs the agent loop and enforces limits.
Least privilege	Giving a system only the access it needs for its task.
Prompt injection	Hidden instructions in content that try to redirect an AI system.
Escalation	Handing a case to a person when the system should not proceed.
Shadow mode	Running the system alongside people without acting, to compare decisions.
Run trace	A step-by-step record of what an agent planned, called and received.
Kill switch	A tested mechanism to stop an agent quickly.

TERM	MEANING
Regression test	A repeat test confirming a change did not worsen known behavior.
Approval gate	A point where a person must confirm before an action proceeds.



A note on use

This publication is a public orientation resource, not legal, technical, financial or professional advice. Adapt it to your people, information, workflows and obligations, and check current tool terms, regulations and risks before acting.

TAKE THE NEXT STEP

Take the free Halden assessment at halden.company/assessments, or talk to us at halden.company/contact about turning this into a plan for your organization.

