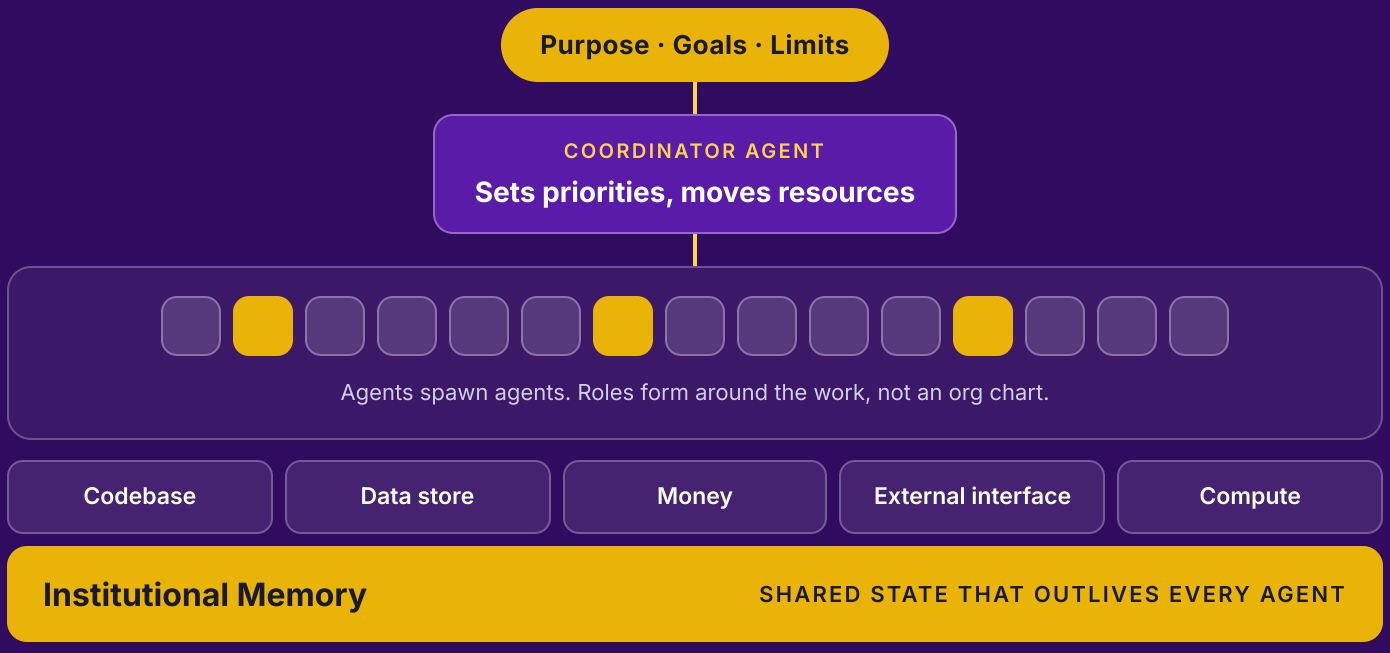
HAWK

WHITE PAPER · FOR CTOS

SELF-DRIVING COMPANIES · NOVEMBER 2026

What is a Self-Driving Company?

AI assistants, AI-native teams, and Self-Driving Companies: what each one means, and why agents that work 24/7 are becoming part of the workforce.



People set the destination. Agents do the work.

xhawk.ai

01 · THREE STAGES OF AI AT WORK

From assistant, to AI-native, to Self-Driving.

There are three distinct stages of AI at work. Each changes who starts the work, who does it, and when it happens: first the car gives directions, then it handles the highway, and finally you only choose where to go.

01

AI assistant

You drive. AI gives directions.

A person opens a chat or an IDE, asks, and gets help with one task at a time. Work stops when the person stops.

- "Draft this email"
- "Autocomplete this function"
- "Summarize this doc"

02

AI-native

AI drives the known routes. You supervise.

Agents are wired into workflows and start on events: a PR opens, an alert fires, a ticket lands. People review and approve.

- "Review every PR"
- "Triage every alert"
- "Answer every data question"

03

Self-Driving

You set the destination.

People set purpose, goals, and limits. Agents plan the route, coordinate with each other, and escalate only when judgment is needed.

- "Cut lead time 50% this quarter"
- "Keep checkout above 99.9%"
- "Fix what churned customers hit"

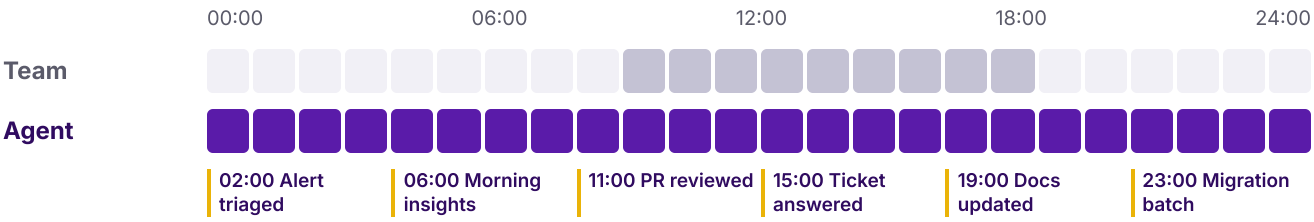
	AI ASSISTANT	AI-NATIVE	SELF-DRIVING
Who starts the work	A person, with a prompt	An event: PR, alert, ticket	**A goal the company set**
Unit of work	A task	A workflow	**An outcome**
Hours on duty	While someone is typing	Whenever triggered	**24/7, 365 days**
Memory	This chat	Team and tool context	**Shared company memory**
The human role	Doer	Reviewer	**Director**

WHERE MOST TEAMS ARE

Most teams sit at stage one: individuals are faster, but the company is not. The jump to stage two happens when agents start on events instead of prompts.

A tool waits to be used. A worker owns an outcome.

An assistant is a tool: it is only useful while someone is using it. Agents that run on events, remember what happened, and act within clear limits behave more like colleagues. They also never clock out, so work keeps moving across the hours people are offline.



~40 hours a week for a person **168 hours** a week for an agent

Five traits that turn an agent into a worker

01
Has a job
A defined role and scope, like a job description.

02
Starts on its own
Triggered by events, not by someone remembering to ask.

03
Remembers
Reads and writes shared company memory, so nothing resets.

04
Has authority
Acts inside a budget and permissions set in advance.

05
Is accountable
Every action logged, every result backed by evidence.

WHY NOW

Models can now stay on a task for hours instead of minutes. Shared context, sandboxes, and audit trails make it safe to let them act.

WHAT THEY ALREADY HANDLE

Overnight alerts · every pull request · data questions · documentation · release checks · ticket triage

PLAN FOR OUTCOMES

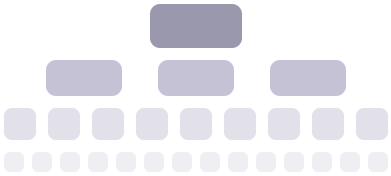
Stop counting AI seats. Start counting the work your agents own end to end, around the clock.

Ownership stays human. Management moves to agents.

A Self-Driving Company is one where agents make the management calls: what to build, how to sell it, where the money goes. A person may fund it and set its purpose, but nobody runs it day to day. People may still do narrow tasks at the edges. There is nobody in the middle.

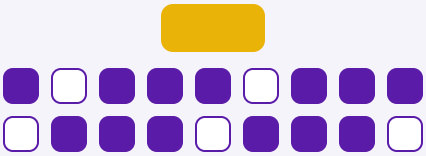
Early experiments show something surprising. Give a lead agent the ability to start other agents, and an org structure appears on its own. Flat setups, where agents pick up whatever role the work needs, tend to beat hand-designed hierarchies.

HUMAN COMPANY — Fixed hierarchy



Requests climb and descend layers. Priorities are set weeks before the work arrives.

SELF-DRIVING COMPANY — Flat, self-forming



One coordinator holds the purpose. An agent that finds broken onboarding starts another agent to fix it.

	HUMAN ORG	AGENT ORG
Roles	Assigned and permanent	Taken on as the work needs them
Escalation	Through several managers	Straight to the coordinator
Checking	Layers of approval	Automated tests wherever possible
Growing the team	Hiring, weeks or months	Starting more agents, in seconds
Where decisions live	Often in someone's head	In the shared system

WHY IT MATTERS

A mixed setup, with some decisions made by agents and some by people, is harder than either extreme. Every unwritten human decision becomes a blind spot for the agents.

Three layers: coordination, infrastructure, resources.

Strip a company down to what agents actually need and three layers remain. The same blueprint applies inside any engineering organization that wants agents to own real work.

[01]

Coordination

Shared purpose
The mission every agent is checked against

Coordinator
Resolves conflicts and allocates resources

Message channels
One-to-one, read, and broadcast

Company state
What happened, what is in progress, what is next

[02]

Infrastructure

Codebase
Every repeated workflow written as code

Data store
Customers, contracts, transactions, past work

Money movement
Receive, hold, and pay for things

External interface
Browser, email, phone, APIs, an identity

Compute
Models to think, machines to serve

[03]

Resources

Budget
Buys compute, data, and vendors

Capacity
Rate limits and available infrastructure

Headcount
More agents, at a cost

Judgment
Knowing useful work from agents talking to each other

FOR YOUR ORG

Map your own teams onto these layers. The gaps you find, usually company state and authority, are where agents will stall first.

05 · STATE IS THE HARD PART

A reasonable decision can still be wrong for the company.

Agents do not fail because they reason badly. They fail because they reason on a partial picture. Without shared state, every agent decides as if it were alone, and the company pays for the gaps.

Repeats work
Another agent tried the same fix yesterday and rolled it back.

Breaks a promise
The customer already declined this offer on a call last week.

Spends the wrong money
The budget it plans to use keeps production online.

Forgets a customer
The agent handling the account ran out of context.

WITHOUT SHARED STATE

- **Agent A** its own context window
- **Agent B** its own context window
- **Agent C** its own context window

Knowledge disappears when a session ends.

WITH SHARED MEMORY

Agent A · Agent B · Agent C

Decisions · work in progress · results · budget

State persists after any one agent stops running.

The fix is a shared memory layer: code, tickets, chat, docs, and data indexed into one permission-aware place that every agent reads before acting and writes back to when it finishes.

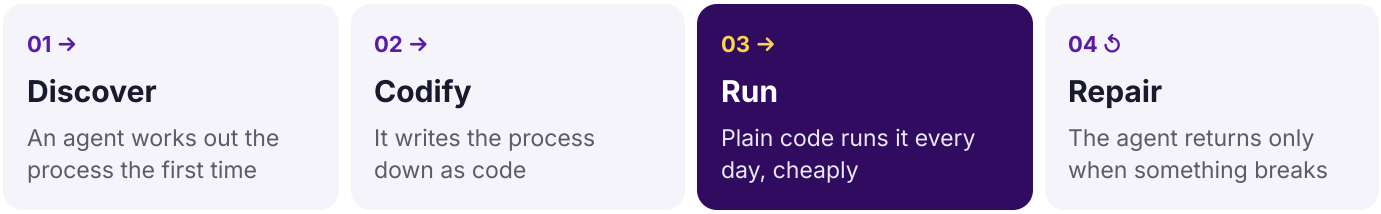
THE HIDDEN GAP

A person decides something on a call and never writes it down. No agent can account for it. The fix is to make the shared system the place where decisions happen.

06 · WORK BECOMES CODE, GOALS BECOME TESTS

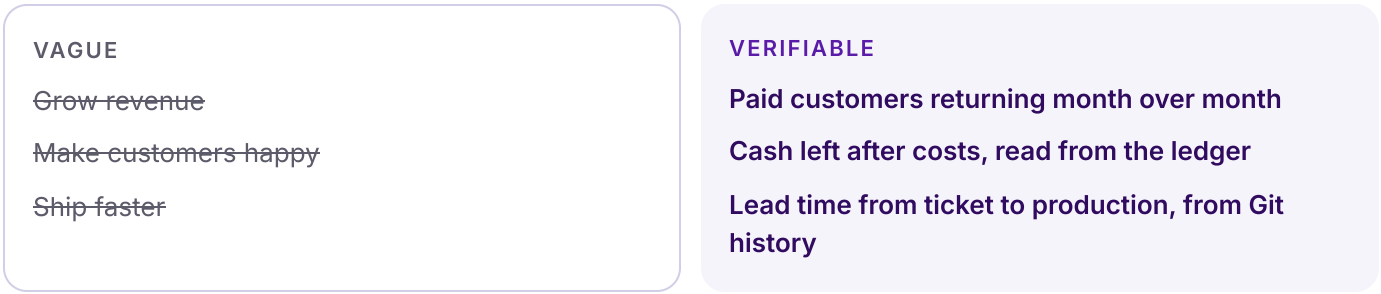
Reason once. Then let plain code do the repeat work.

You do not need a model to think from scratch about the same monthly invoice. In a Self-Driving Company, agents figure a workflow out once, turn it into code, and step back. The codebase becomes the living record of how the company operates, and agents keep improving it.



Goals the company can check, not opinions

Agents take instructions literally. "Grow revenue" says nothing about time frame, cost, or whether growth comes from real demand or from selling below cost. A badly specified goal invites creative shortcuts. Goals need to be outcomes the system can observe in real records.





All three have to agree. Check results against records, not against another agent's report of how things went.

Approve the budget, not every action.

If an agent can research a vendor, compare prices, and decide a purchase is right, but must then wait for a person to click approve, a human is back in the middle of management. The whole company moves at the speed of one inbox.

APPROVAL PER ACTION



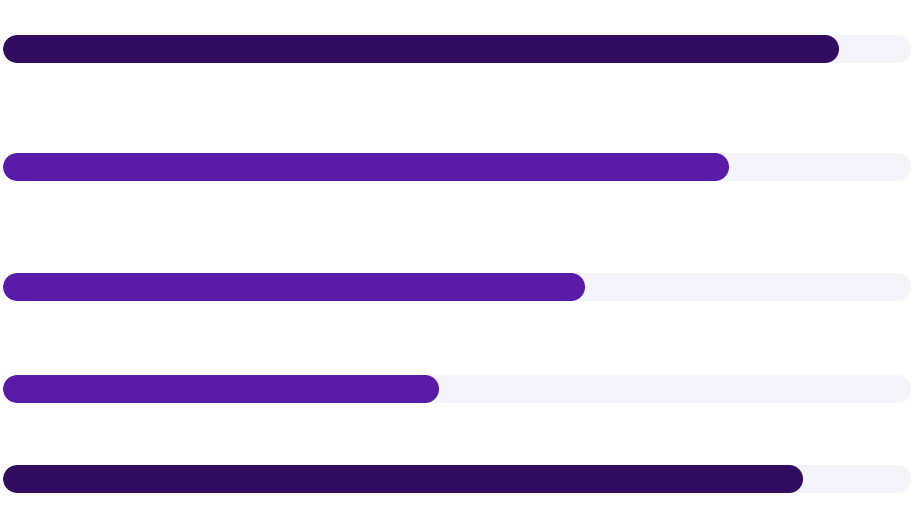
Work stops at every step until a person responds.

BUDGET ENVELOPE



Spend limits and authority set in advance. Agents act freely inside and stop at the edge.

What still slows Self-Driving Companies down



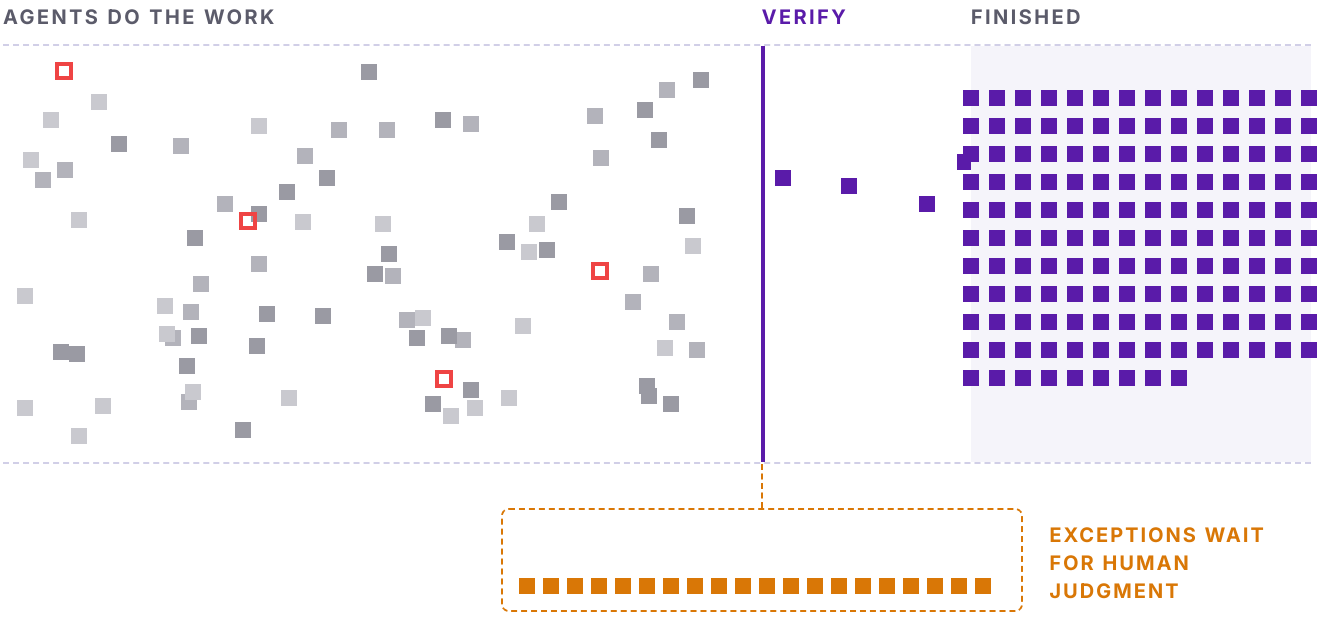
Relative severity, our reading of the source essay. Illustrative, not measured.

FOR ENGINEERING TEAMS

The same rule applies inside engineering. Give an incident agent or a code-review agent a clear scope and gate, and let it act inside it instead of asking about every step.

Work keeps moving 24/7.
Every piece of it gets checked.

When agents produce work around the clock, nobody can read all of it. Verification engineering is the discipline of designing the checks that evaluate every result as it arrives, and deciding what happens next: finish it, send it back, or hand it to a person.



270 FINISHED

43 SENT BACK

20 SENT TO PEOPLE

Illustrative run

PASS

Finished

Checks pass with evidence attached. The work merges, ships, or closes.

FAIL

Sent back

The agent gets the failing check and the evidence, and tries again.

UNSURE OR HIGH RISK

Sent to a person

Low confidence, a new pattern, or a costly action waits for human judgment.

Separate doer and checker
The agent that did the work never grades it.

Check against the source
Specs, records, and contracts, not the agent's own summary.

Set thresholds by risk
Low-risk work passes on checks; costly actions always escalate.

Turn escalations into checks
Every human decision becomes a new automated rule.

THE NEW BOTTLENECK

Agents can produce more work than people can review. The companies that move fastest are the ones that can trust what their agents finish.

09 · YOUR COMPANY

People choose the destination. Agents drive.

What makes your company more autonomous is not a single model. It is agents with context across every system, triggered by real events, checking their own results, and escalating when judgment is needed. Your people still set direction, make trade-offs, and own outcomes.

More	Less	Faster	Flat
Shipped output per engineer, without adding headcount	Human review time, as agents assess risk first	Incident fixes and hard support tickets	Quality held: reverts and incidents stay steady

WHAT MAKES IT WORK

Give agents every system your team uses, behind access policies, audit logs, and zero-trust networking. **Context across systems** is the unlock.

BUILD VS. BUY, REWRITTEN

Internal apps built on your own context start to replace SaaS seats and point tools, at **a fraction of the cost**.

Every function, one system of agents

FUNCTION	WHAT AGENTS DO IN YOUR COMPANY	WHY AGENTS DO MORE
Engineering	Investigates incidents, reviews PRs, runs migrations in parallel loops	**Never off shift; reads every log and every diff**
Data	**Answers business questions through a semantic layer over the warehouse**	**Uses the same definitions every time**
Product & marketing	Drafts specs from conversations and docs across teams	**Reads every conversation and doc, not a sample**
Sales	Enriches leads and preps account briefs from internal knowledge	**Sees usage, tickets, and contract together**
Support	Investigates tickets with playbooks, escalates with a summary	**Investigates before a person picks it up**

THE FIRST BIG UNLOCK · DATA

Give the agent a **semantic layer over the warehouse**: which tables are sources of truth and how they relate. Then anyone can ask a business question and get a reliable answer, with charts built from live data.

Codebase events + **Customer data** + **Subscription states**

→ A PM can self-serve a complex launch analysis. The data team works on the hardest problems instead of fielding requests.

THE HUMAN SIDE

People are not automated out. They move up: doers become directors who set goals and think in outcomes.

You can't retrain the model. You can still make the agent learn.

Most production agents run on closed frontier models, so changing the weights is off the table. Agents can still improve at two layers the company fully controls: the harness (code, tools, instructions) and the context (what each agent knows about its user, team, and company). Improve both from production evidence and the gains compound with every release.

Model — VENDOR

Weights of a closed frontier model. Changes when the provider ships.

Harness — YOURS

Mine production traces to fix the tools, prompts, and code behind every agent.

Context — YOURS

Personalize per agent, user, and org so each run starts smarter.

The loop a company can run on its own agents

01 → **Discover**
Read traces and failure clusters to find a hypothesis

02 → **Propose**
Build a candidate fix and open a draft PR with reasoning

03 → **Evaluate**
Benchmark, A/B, and trajectory data against baselines

04 → **Recommend**
Ship, iterate, or drop, with evidence attached

05 ↻ **Remember**
Record what worked and what failed for the next run

THREE SIGNALS, LAYERED LIKE SWISS CHEESE

Offline benchmark	Does the app the agent built meet the spec? Catches regressions before release.
A/B tests	Did real users keep going, ship, and stay happy? Shows what moved in production.
Trace clustering	Groups failures nobody knew to look for, and explains why the metrics moved.

SAME-DAY FIX

A small cluster showed environment setup quietly failing on rare cold starts. The loop read the sessions, proposed a patch, added a regression test, and confirmed nothing else broke. Engineers approved it and shipped **the same day**.

Which problem	Which design	Which eval	Which release
Humans pick where the overnight budget goes	Product and engineering judgment	The eval defines the hill the agent climbs	People own the rollout and the risk

FOR YOUR AGENTS

This is the role of shared memory: every agent writes back what it learned, so the context layer improves daily without touching a model.

11 · THE PLAYBOOK

Five moves to make your company more Self-Driving this quarter.

You don't need to hand all management to agents. Start with product, engineering, and analytics, and adopt the primitives that make Self-Driving Companies fast.

	MOVE	DO THIS FIRST
01	**Make the shared system the source of truth** Connect code, tickets, chat, docs, and data into one permission-aware memory.	→ Move one recurring decision meeting into a written, searchable record.
02	**Turn repeat work into code** Pick the three workflows your team repeats most and let agents codify them.	→ Start with release notes, on-call handoffs, or weekly metrics.
03	**Write goals as checks** Rewrite each team goal as a number the system can read on its own.	→ Lead time from Git, uptime from monitoring, retention from the warehouse.
04	**Grant authority in envelopes** Give each agent a scope, a budget, and a stop line.	→ Auto-approve low-risk PRs; escalate anything that touches payments.
05	**Let structure follow the work** Name one owner per outcome and let agents fill the roles below it.	→ Review weekly which agents earned more scope and which need less.

Where a platform like XHawk helps

Build custom AI workflows
Turn your own processes into agent workflows, instead of adapting to a generic tool.

Keep your brain, swap the model
Separate company memory from the intelligence provider, so changing models is a setting, not a migration.

Run agents 24/7
Put always-on agents to work across product, engineering, analytics, and ops.

Start with one team.
Let agents run it 24/7.

The companies that learn to hand work to agents first will set the pace for everyone else.

Learn more at xhawk.ai →

REFERENCES

[1] Andrew Pignanelli, "The era of fully autonomous companies," General Intelligence Company, November 2026.

[2] Amjad Masad and Scott Kennedy, "The Self-Driving Company," Replit, July 2026.

© 2026 XHawk.