

**JarkoMes.**

JARKOMES BV · FIELD GUIDE

# From AI pilot to production

The questions you answer before you build, and why most AI projects stall on exactly those.

# Most AI projects never reach production

Not because the model falls short. Models have been good enough for a while now for the kind of work businesses put them on: moving data, checking records, assembling reports, catching signals.

They stall on organisation. On who owns the process underneath. On what happens when the agent gets it wrong. On the fact that whoever built it didn't know the business process, or whoever specified it didn't know the technology.

This guide is what I use before I start building. No theory, no vendor talk. The questions that separate an agent that stalls after three months from one that's still running in two years.

You can use this document without me. That's the intent. If you'd rather do it together afterwards, my number is at the back.

# Three questions nobody answers up front

I've watched enough pilots up close to know where they fall over. Almost always on one of these three.

01

## **Who owns the process?**

Not the AI project — the process itself. If the agent disappears tomorrow, who does the work? If the answer is "we'll figure that out then", there is no owner, and nobody will notice when the agent gets something wrong.

02

## **What happens when it hesitates?**

An agent that only knows the happy path is a demo. There has to be a threshold above which a human decides, and that threshold has to be fixed before the build, not after. Otherwise it gets decided ad hoc in production, usually by whoever happens to find the error.

03

## **Who can change it when the business changes?**

Processes change. If nobody on your own team understands how the agent works, you didn't buy automation, you bought a dependency. Handover isn't aftercare, it's part of the job.

# Twelve questions before you build

Answer these twelve and you know whether you're ready to build. If three or more have no answer, the process itself isn't sharp enough yet, and automating it will magnify the problem rather than solve it.

## The process

---

- ☐ Can you write the process down in five steps, without "and then someone looks at it"?
- ☐ How many times a week does it happen, and how long does each one take?
- ☐ What does an error cost today, in money or in time to put right?

## The decisions

---

- ☐ Which decisions may the agent make on its own?
- ☐ Which must it escalate, and to whom exactly?
- ☐ Which may it never make, even when it looks certain?

## The data

---

- ☐ Which systems must it read from, and do you have read access?
- ☐ Where does it write, and is that reversible?
- ☐ Is there personal or sensitive business data in it, and may that leave the building?

## The safety net

---

- ☐ Where do you see what the agent did, without asking it?
- ☐ How do you notice it has stopped, and within what time?
- ☐ Who gets the signal when something goes wrong, and on which channel?

# What “in production” actually means

This is the word most misunderstandings hide in. A demo that runs on your laptop is not in production. A script someone starts by hand is not in production. A proof of concept on a test environment certainly isn't.

## **It runs without anyone starting it**

On a schedule or an event, on infrastructure that isn't someone's laptop.

## **It survives an error**

A system briefly unreachable, a field that turns out empty, a format that changes. It stops, reports, and picks the thread back up.

## **You can see what it did**

Logging a human can read, not just a stack trace. You have to be able to reconstruct a decision afterwards.

## **Someone is responsible**

A name, not a department. That person gets the escalations and is allowed to switch the agent off.

## **Your team can change it**

Documentation, access to the code, and someone who understands the logic. Without that, every change is a new project.

My own rule: what I deliver runs in production or it isn't finished. An agent that only works in a demo isn't an agent, it's a presentation.

# Processes that suit this well

Not everything should become an agent. These are the patterns that pay off fastest in practice: repetitive, rule-based enough to scope, and tedious enough that nobody enjoys doing them.

## **Reconciliation and checking**

Billing against CRM, payments against accounting, contract against delivery. Dull, repetitive, error-prone.

## **Monitoring and follow-up**

Catching signals across systems, connecting them, and escalating before anyone notices by themselves.

## **Onboarding and customer follow-up**

Who is stuck, where, and for how long. Followed up automatically instead of discovered monthly.

## **Reporting and synthesis**

Weekly overviews, board packs, status reports. Pulling data from five systems and turning it into something readable.

## **Back office and admin**

Creating records, moving data, updating statuses, chasing missing information.

## **Unlocking knowledge**

RAG over your own documentation, contracts and history, so the team stops searching.

# Three weeks of noise, and what it taught me

I built myself an agent that looks across all my systems and tells me what needs attention. Office 365, CRM, GitHub, Azure DevOps, product analytics, Notion, billing. Technically it worked from day one.

And for three weeks it mostly told me that things were happening. An email arrived. A commit was pushed. A deal moved. All true, all useless. I started dismissing its messages without reading them, which is worse than having no agent: you get used to it.

The problem wasn't the model and it wasn't the integrations. It was that it had no memory of what it had already said, and no threshold for what was worth saying. It had no idea what normal looked like, so everything was news.

✓ **A signal is only a signal when it deviates**

The agent had to learn the normal pattern before it could report anything.

✓ **It has to remember what it already said**

A persistent register of open loops, so the same alert doesn't arrive fresh every day.

✓ **Silence is a valid outcome**

A day without an alert means nothing is wrong. That's exactly what you want.

This is why I don't build agents without first measuring what normally happens. An agent that sees everything and means nothing costs you more attention than it saves.

# How I run an Agent Build

One process, tightly scoped, fixed price agreed up front. From scope to handover.

01

## **Fix the scope**

One process. We agree up front what the agent may decide, what it must escalate, and what it never does. That's on paper before there's a line of code.

02

## **Build against the real data**

No synthetic test set. I build against your actual systems, in an environment where mistakes break nothing. That immediately exposes what the brief didn't say.

03

## **Guardrails and observability**

Logging, error handling, escalation paths and a clear threshold above which a human decides. This is exactly the part pilots skip.

04

## **Rollout and handover**

In production, with documentation and a team that knows how it works and how to change it. I'm not a dependency.

# I build this for myself too

I'm Jarko Mesuere. Today COO of a SaaS scale-up, where I carry the full operation: commercial, onboarding, finance and engineering delivery. Before that, fifteen years in IT: systems administration, infrastructure lead on a national tolling platform, engineering manager and program lead on the largest parking app in the Benelux.

That order is the point. I ran Kubernetes clusters, wrote Terraform and rolled back releases at three in the morning, and only then learned how to steer a P&L and report to a board. The same person who chairs the meeting writes the code. That removes a translation step, and that translation step is exactly where most AI projects die.

None of the above is theory. Sentinel, my own monitoring agent, reads across seven systems, keeps a persistent register of open loops and classifies every proposed action into three levels: do it, ask first, or stay away. It runs self-hosted, behind Tailscale, because the data is sensitive.

---

**15 years**

in IT and operations

**7**

operating dashboards in one day

**4**

structural leaks exposed

# Three ways to start

Each with a defined scope and a fixed price we agree before you have to decide anything. No day rate that keeps running.

## Agent Build

Fixed price per process

One concrete process automated and in production. Running, with monitoring, error handling and ownership that transfers to your team.

- ✓ Process design and scoping
- ✓ Build, integration and rollout
- ✓ Guardrails, logging and escalation
- ✓ Handover and documentation

## Ops Scan

Fixed price, agreed up front

Knowing where it leaks before you automate. A working cockpit on your real numbers plus a prioritised list of what an agent could take over.

- ✓ Review of processes and systems
- ✓ Operating cockpit on your own data
- ✓ Prioritised list with impact estimates
- ✓ A proposal for what comes first

## Fractional COO or program lead

Limited availability

For when there's a gap at operations or program level. I take it over, get it under control and hand it back.

- ✓ Operating cadence and governance
- ✓ Steering on numbers that hold up
- ✓ Team structure and ownership
- ✓ Handover as an explicit goal

# Got one process in mind?

Tell me which process annoys you most. In thirty minutes I'll tell you whether it suits an agent and roughly what it takes. No sales pitch — if I'm not the right person I'll say so, and usually I know someone who is.

---

 +32 479 02 62 01

 [jarko@jarko.be](mailto:jarko@jarko.be)

 [jarko.be](https://jarko.be)

I usually reply the same day.