MONO BY DUSTY · WHITEPAPER

Agentic design.

Designing agents the way you design a team. A working approach to designing agentic systems with clear roles, context, boundaries and responsibility.

AUTHOR	PUBLISHER	DATE	READING TIME
Dusty Baars	Mono by Dusty	June 2026	18 minutes

CONTENTS

SUMMARY

What this document argues, in four points.

Most organisations now have some experience with AI. Copilots are running, automations are in place, and the first agents have been built. What is usually missing is a clear way of thinking about how those systems become safe, manageable and genuinely usable. The conversation is about models, when it should be about design.

This document describes how I design agentic systems. Not from the question of which model performs best, but from the question of how work actually flows through an organisation. The approach rests on four connected choices.

01

A design problem, not a model problem

The value does not come from the model, but from the system around it. Roles, context, boundaries and responsibility decide whether an agent becomes useful in practice or stays impressive in a demo.

02

Design roles explicitly, the way you design a team

An agent without a clear role becomes a generic assistant, vaguely involved everywhere and accountable nowhere. Specialised roles make a system clearer and more reliable.

03

Treat context as the foundation, not an extra

Access to the right information, knowledge of earlier steps, understanding of business rules and a view of the current state are requirements, not nice-to-haves. When that layer is weak, every implementation stays dependent on luck, prompts and manual correction.

04

Keep autonomy tied to responsibility

The moment an agent touches processes, communication or decisions, the boundaries have to be explicit. What runs autonomously, what needs approval, what gets logged, and where a human steps in. That is not a brake on innovation, it is the condition for using AI seriously and durably.

The rest of this document works those four choices out. First, why agents so often stall in practice. Then what I mean by agentic design, followed by the principles I hold to. After that I show what it looks like through my own systems, and what adoption asks of an organisation. Finally, where this is heading.

PART I

Why agents stall.

The technology is rarely the problem. Models are good enough, the tooling is mature, and putting an agent together can be done in an afternoon. What is missing is the design around it. That is where agents stall, on the same few points, in organisation after organisation.

It often starts with isolated solutions built without underlying design principles. An assistant for content, a flow for support, an agent for internal knowledge. Each useful on its own, but without coherence the complexity grows faster than the value. No one can see who does what anymore, and maintenance becomes heavier than the work it saves.

A second pattern is autonomy at the wrong moment. Agents get too much room too early, or too little context to do good work. Without boundaries you get risk. Without context you get noise. Either way, AI becomes something that needs extra oversight rather than something that lightens the load. The promise inverts.

The third pattern is an approach that is too technical or too fragmented. Something works in one place, in the hands of whoever built it, but it cannot be rolled out reliably any wider. The step from a working proof to a system the organisation can lean on is underestimated. That is where most initiatives fail, not on the technology but on the design.

THE THREAD

Agentic AI gets treated as a technical question when it is a design question. Frame it that way and the first question shifts. Not which model, but how the work flows, where context is lost, and where responsibility has to stay explicit.

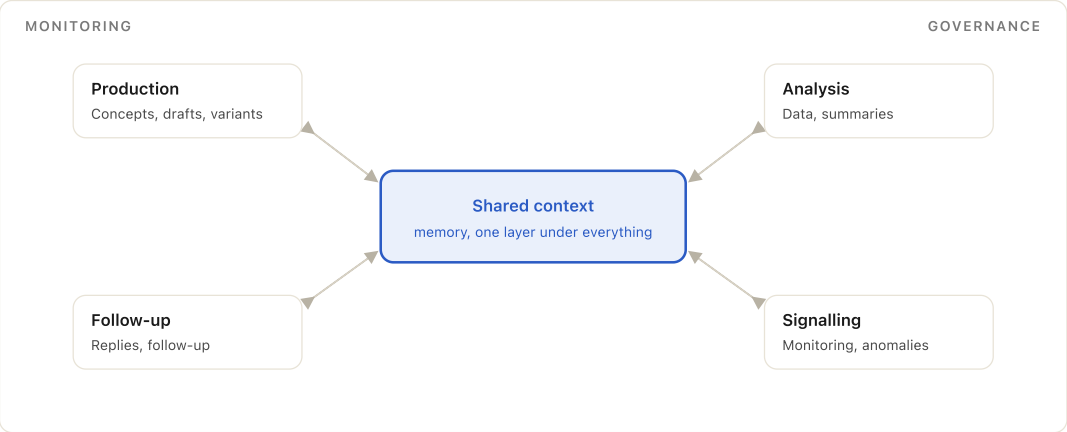
PART II

What agentic design is.

Agentic design means designing agents the way you design a strong team. Each agent has a clear role, a defined goal, access to the right information and tools, and clear rules about what it may do on its own and where a human is needed. The same things that make a team work make a system of agents work.

That asks for more than a good model. It asks for a system that accounts for collaboration, context, memory, information flows, monitoring and governance. The model is one part. The rest decides whether the whole holds up in practice. Only then do you get a form of AI that does not hover above the organisation, but becomes part of it.

AGENTIC DESIGN IN ONE VIEW



At the centre, a shared context and memory layer. Around it, agents each with their own role, drawing from that layer and adding to it. Around that run monitoring and governance, seeing what happens and where a human decides. Not a collection of loose tools, but a whole with an inside and an outside.

The image above sums it up. At the centre sits a shared context and memory layer. Around it, agents each with their own role, drawing from that layer and adding to it. Around that run monitoring and governance, seeing what happens and where a human decides. It is not a collection of loose tools, it is a whole with an inside and an outside.

The difference from the common approach is the order. Most implementations start at the model and the prompt, and hope the rest forms itself. Agentic design starts at the roles and the information flows, and chooses the model only once the work is clear. The technology follows the design, not the other way around.

PART III

The design principles.

Five principles carry this approach. None is remarkable on its own. The combination is what makes a system in production deliver real value.

Start with the work.

Not with tooling, but with the process itself. Where is the delay, where is context lost, where is the repetition, and where do people spend too much time coordinating instead of doing the work. Not every task suits agentic AI. The best first applications sit in processes with clear outcomes, several handover moments, and a strong need for consistency, speed and good context.

Design roles explicitly.

An agent without a clear role sinks into vagueness. Specialised roles, each with its own task and a defined place in the whole, make a system clearer and more reliable. Once it is clear who or what is responsible for what, the collaboration between people and agents becomes steerable.

AN AGENT, DESIGNED AS A TEAM MEMBER

ROLE IN THE TEAM

An agent as a team member

ROLE	What it is, one clear task
GOAL	A defined outcome
ACCESS	The right context and tools
BOUNDARIES	autonomous · approval

An agent reads like a team profile. The role says what it is, the goal gives a defined outcome, the access arranges context and tools, and the boundaries make explicit what runs autonomously and what needs approval.

Treat context as the foundation.

An agent is only usable when the context is well organised. Access to the right information, knowledge of earlier steps, understanding of business rules and a view of the current state are not extras but requirements. That is why I pay close attention to how information moves through a system. When that layer is weak, every implementation stays at the mercy of chance.

Keep autonomy tied to responsibility.

The moment an agent touches processes, communication or decisions, the boundaries have to be explicit. What runs autonomously, what needs approval, what gets logged, and where a human steps in. To me that is not a brake on innovation, it is the condition for using AI seriously and durably.

Build for scale, but start small.

Start small, with a process that matters directly to the business. Not because the ambition should be small, but because good design has to prove itself in practice first. A strong first system shows that the technology works, and that the organisation is ready to work with it. Scaling then becomes a logical step instead of a leap in the dark.

PART IV

In practice.

This approach did not come from a book. I built it, in systems that run. Four of them each show a different part of the principle.

Mono Dash, roles and orchestration

Mono Dash directs teams of agents, each with its own role and mandate. Not one model trying to do everything, but specialised roles that carry a process together, with a coordination layer above them. It makes visible what the second principle means in practice: a system becomes more reliable the moment responsibility is distributed instead of piled up.

Open Brain, context as the foundation

Open Brain is a memory layer that remembers across applications and sessions. Thoughts, decisions and patterns are stored and retrieved by meaning, not by keyword. It is the practical form of the third principle. Agents draw from a shared context and add to it, so knowledge does not stay trapped in separate tools or in someone's head. Built on MCP, so the same layer is reachable from several applications.

Memortium Studio, autonomy with a limit

Memortium Studio edits portrait photos of the deceased. A pipeline generates variants automatically, and a validation step guards that the identity does not change. The rule is firm: when in doubt, do not deliver. If the automatic result is not good enough, the work escalates to a person. AI-first, with a human fallback. The fourth principle in its sharpest form, because the stakes are high and a mistake should not rest with an agent.

The Memortium marketing machine, a team of agents

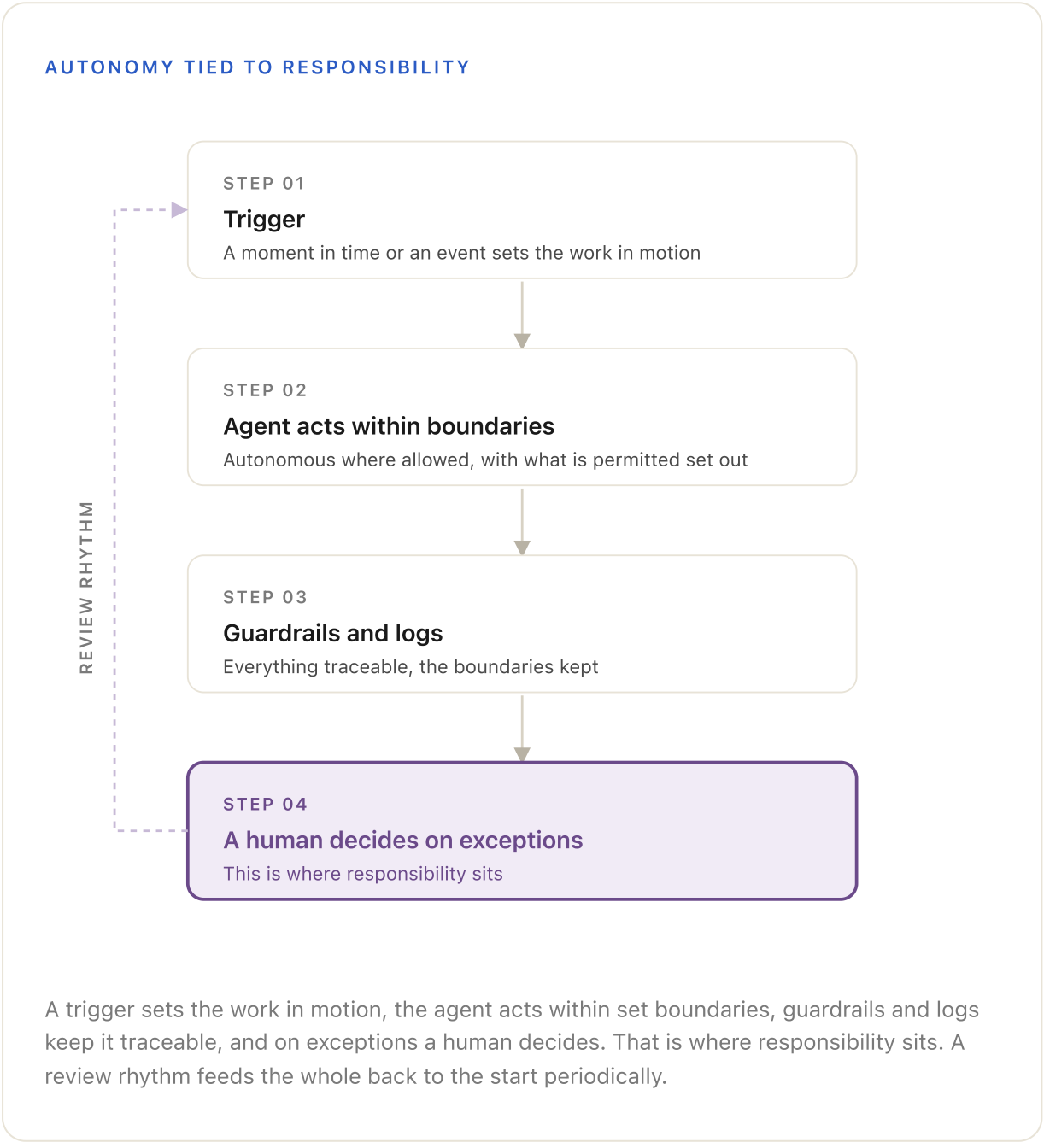
Here several agents with their own roles work together on content, outreach and follow-up, with a person who sets the boundaries and decides on exceptions. It shows how the principles come together: roles explicit, context shared, autonomy bounded, and started small before it went wider.

What connects these systems is not the model. It is the design around it. In each case it began with the work, the agents were given a clear role, the context was arranged, and the final responsibility stayed with a person.

PART V

What it asks of an organisation, and what stays human.

An approach rarely works just because it is sound on paper. Adoption asks something of the organisation itself, and it asks that a few things stay deliberately human.



A trigger sets the work in motion, the agent acts within set boundaries, guardrails and logs keep it traceable, and on exceptions a human decides. That is where responsibility sits. A review rhythm feeds the whole back to the start periodically.

What stays in human hands.

Strategy, the client relationship, concept and direction, final approvals and ultimate responsibility stay deliberately human. An agent can prepare a proposal, a person signs for it. The responsibility that comes with choices when something goes wrong belongs to people, not to a system. Making this distinction explicit does more than protect what is human, it brings calm. When you know where the human stays indispensable, you look with more confidence at the places where an agent can land. The conversation becomes about setup, not about fear.

What the organisation has to arrange itself.

An internal owner, someone with a mandate over scope, quality and further development, so an agent becomes part of the company rather than a dependency on the builder. An agreement on governance: who sees which data, where it ends up, who has access to the logs, and who signs for exceptions. And a review rhythm, because an agent in production needs to be held up to the light periodically. Does it still work as agreed, does it still deliver the outcome it was built for, and are the boundaries still right.

> For most organisations this is not a complex document. It is a subject that belongs on the table beforehand, because it makes privacy, traceability and accountability concrete instead of abstract.

PART VI

Where this is heading.

The field moves fast, and the direction is clear. Isolated agents become collaborating agents. Protocols like MCP make context and tools broadly reachable, and agent-to-agent communication lets systems hand each other work. Guardrails and evals are maturing, so reliability becomes measurable instead of a feeling. The question shifts from "can an agent do this" to "can we trust that it does it well, time after time".

That is exactly where design makes the difference. As agents do more on their own, the setup around them counts for more, not less. Roles, context, boundaries and responsibility are not temporary conditions that disappear once the models improve. They are the lasting core. A better model does not make a badly designed system reliable, it only makes the mistakes faster.

I build with agents, orchestration and automation, and push at the edges to see what becomes possible. Not to replace people, but to lighten the work and make room for the work that truly matters. That is the direction I want to work in, and the one I want to lead from.

CLOSING

An invitation.

Agentic AI only becomes valuable when it is not about loose prompts or clever demos, but about design. I implement agentic systems the way I would design an organisation: with clear roles, good information flows, explicit boundaries and responsibility in the right place.

For anyone who wants to think this through, for a team or an organisation that wants to take AI beyond isolated experiments, I am glad to talk further.

About the author

Dusty Baars is a designer turned AI builder. With AI he builds solutions that people actually use, from one conviction: good technology makes people better instead of replacing them. Over twelve years of enterprise UX at Shell, Roche, Philips and ING, through Plat4mation. The last two years building AI in production: Memortium, Open Brain, Mono Dash. He works from Amsterdam.

Mono by Dusty · Amsterdam · contact@dustybaars.com
For a first exploration or to get acquainted, a short email is the quickest starting point.