m2consulting

WHITE PAPER · THE THREE PILLARS OF ENTERPRISE AI · PILLAR 02

Engineered Agent Systems: From Prompt Wrapper to Production

Why one giant prompt falls apart, what a production-grade multi-agent architecture actually contains, and where the real engineering work sits: gateways, identity, governance, and the security of the tool chain.

Marcel Meyer · m2-consulting

July 2026 · m2-consulting.io

A prototype that shines in a demo and a system that holds up day after day in production are not the same thing. Most "AI systems" in enterprise environments today are prompt wrappers with a pleasant UI — one large model, one oversized prompt, unrestricted tool access, and hope. This paper describes what replaces that: a purpose-built multi-agent architecture with the layers, guards, and audit trails that let you stand behind the system in front of an auditor.

01 Why the one-giant-prompt approach falls apart

The instinct in every AI project is to start with a single model, a single prompt, and every tool wired in. It works in the demo. Then reality arrives: behaviour becomes unpredictable because one prompt is trying to do everything; nobody can explain afterwards why the system did what it did; every user implicitly has every permission the system has; and each new capability added makes all previous ones slightly worse, because they compete for the same context.

> **The alternative — one giant prompt trying to do everything — falls apart fast. Predictability comes from separation of concerns, not from better prompting.**

Enterprise-grade agent systems are not configured. They are **designed** — for the specific use case, with the right layers, security mechanisms, and quality assurance. The difference between an agent being tried out and an agent going to production is system engineering.

02 The anatomy of an engineered agent system

The orchestrator — traffic control as a separate concern

The orchestrator reads what the user is after, hands the request to the right agent, and decides which tools that agent may use. Keeping this job separate is what makes the system's behaviour predictable — and it is what lets you see afterwards why it did what it did. For larger tasks, the orchestrator decomposes work into subtasks, routes them to sub-agents in parallel, and reasons over the combined results before answering.

Specialised agents — narrow by design

Separate agents for research, for sales support, for marketing, for internal helpdesk. Each stays narrow. Narrowness is not a limitation; it is the property that makes agents testable, and it means you only ever grant an agent the access it genuinely needs. A research agent has no path to the invoicing system — not because a prompt forbids it, but because the architecture never gave it one.

The MCP gateway — one door to every tool

Every time an agent reaches for a tool or data source, it goes through a single gateway. This is probably the single most important security decision in the design:

- **One entrance.** All tool access flows through one authenticated, authorised point — OAuth2-based, with per-agent tool allowlists.
- **Scoped access.** Agents are created for a purpose and see a curated selection of tools, not a universe of options. This saves token cost, cuts latency, and improves quality — agents are measurably worse when overwhelmed by unnecessary choices.
- **Separate tracks.** Internal and external tools run on separate tracks. In a dev system, an agent calls whatever it likes; in production, this is the point where access is configured, granted, logged, and can be audited later.

The LLM gateway — the same idea, for models

A second gateway handles the models: authentication, model routing per request and per user, and safe credential exchange with the tool side. In practice it means you can run on a managed platform today and bring in a private model later — when there is a real reason to — without losing your grip on cost or usage. The model becomes a routing decision, not an architectural commitment.

Identity and delegation — the agent acts as the user

Data does not equal data. Some is public, some internal, some confidential. Users must not have uncontrolled access to all of it — and neither must agents. An agent should work **in delegation of a user**: it carries a delegation token derived from the user's access token, and every downstream system grants access based on what that user is allowed to see. Your CRM, trading, and finance data stay inside the wall; market data, news, and web search sit outside it. Keeping those two worlds apart, per user, matters a great deal once sensitive records are in play — and it is painful to retrofit, so it belongs in the design from day one.

Governance, compliance, and observability — the wrapper around everything

Every conversation, every decision, every tool call, and the cost of each is recorded and reviewable. For a regulated business this is not a nice-to-have: it is what lets you stand behind the system in front of a regulator or a SOC 2 auditor, and answer the data-conduct questions that come with the territory. It also pays for itself twice over: usage data becomes the input for systematic improvement, and you gain full visibility of how data flows — from whom, to where, and why.

03 The tool chain is an attack surface

Wiring agents up to tools — external ones especially — opens a real door for trouble. The threat model is specific to agentic systems and still unfamiliar to most security teams:

THREAT	WHAT IT LOOKS LIKE	ARCHITECTURAL ANSWER
Poisoned tools	A compromised or malicious tool returns manipulated data that steers agent behaviour.	Tool allowlists, provenance checks, internal/ external separation at the gateway.
Instruction injection	Instructions smuggled in through tool descriptions or retrieved content hijack the agent.	Treat all tool output as untrusted input; guard rails at the orchestrator, not in the prompt.
Silent exfiltration	Data leaves quietly through an over-permissioned external tool.	Per-agent tool scoping, delegation tokens, full egress logging at the gateway.
Supply-chain weak links	A dependency of a tool server changes behaviour underneath you.	A curated tool registry with review gates — the gateway is a genuine boundary, not a label on a box.

04 Running it, not just launching it

Go-live is the easy part. Staying reliable, monitored, compliant, and used — month after month — is the real job. Three disciplines separate systems that keep earning their keep from those that fade:

- **Cost discipline.** It is easy to overbuild — standing up your own models too early, or losing control of token spend. Route the large majority of traffic to mid-tier models, reserve frontier models for tasks that demand them, and watch both usage and the bill continuously.
- **Quality loops.** Feedback, evaluation sets, and regression testing for agent behaviour. An agent change is a deployment like any other and deserves the same rigour.
- **Operational habits.** Monitoring, an explicit change process, and clear ownership. The systems that fail rarely fail technically first — they fail organisationally, when nobody owns the running system.

> **THE HONEST TEST**
>
> Can you answer, for any answer your system gave last Tuesday: which agent produced it, which tools it called, what data it saw, whose permissions it ran under, and what it cost? If yes, you have a system. If no, you have a prompt wrapper — technical debt waiting for approval.

05 The pillar in context

An engineered agent system is Pillar 2 of our three-pillar architecture. It is the part that acts — but it acts well only when the other two pillars carry their share: the semantic layer (Pillar 1) grounds every agent in what the data means, and the memory (Pillar 3) lets the system improve with use instead of meeting the thousandth request as if it were the first. The companion papers in this series cover both,

and a fourth paper describes how the three interlock — and why, once they do, the model itself becomes a swappable component.

Where m2-consulting comes in

We design and build multi-agent systems for regulated environments — with the gateways, delegation model, and audit trail in place from the start. If you have a prototype that needs to become a system, that is exactly the gap we close.

m2-consulting.io · marcel.meyer@m2-consulting.io · Book a strategy call at m2-consulting.io

---

© 2026 m2-consulting. This paper is part of the series "The Three Pillars of Enterprise AI." All papers: m2-consulting.io/thinking