mai

Designing Intelligent Futures

Convergence of AI Protocol and Proxy: MCP and AI Gateways

Author- Pav Dhillon

mai
Designing Intelligent Futures

TABLE OF CONTENTS

INTRODUCTION:

Bridging the AI Integration Gap

The artificial intelligence landscape is at an inflection point. We're moving from isolated, chat-based Large Language Models (LLMs) towards interconnected agentic systems that can perform tasks across enterprise tools and data. In this transition, a critical infrastructure gap has been exposed: the "contextual disconnect" between probabilistic AI reasoning and the deterministic APIs of business software. In simpler terms, today's AI models are great at understanding language and intent, but they don't naturally speak the structured language of our databases, CRMs, or internal APIs. This disconnect can slow down AI adoption and limit the value businesses get from AI.

To bridge this gap, two converging technologies have emerged as foundational pillars of modern AI infrastructure: Anthropic's Model Context Protocol (MCP) and specialised AI Gateways. As the CTO of **Mai**, an AI consultancy, I want to share why these technologies matter for both technical and business leaders, and how they can enhance your company's technical moat in the age of AI. Put simply, MCP provides a universal "language" for AI-to-system interactions, while AI Gateways act as the control towers managing those interactions at scale. Together, they promise a secure, scalable way to weave AI into the fabric of your business, enabling internal AI agents and customer facing AI features that provide enhanced value for both businesses and customers.

The Context Crisis: Why we need MCP

Early enterprise AI projects often ran into an integration crisis. Each AI assistant or LLM needed custom code to talk to each internal system, leading to a tangle of bespoke integrations. If you had '*M*' AI models and '*N*' tools or data sources, you might end up building *M* × *N* connectors .

This is commonly called the "N×M problem," where adding new models or new tools, multiplies integration complexity linearly on both axes. The result was massive technical debt and slow progress. Switching an AI model (i.e. from one vendor to another) meant rewriting all the integration glue for that model, and scaling up to more use cases meant an explosion of custom code for each combination.

Anthropic's Model Context Protocol (MCP) was created to solve this integration nightmare. MCP is an open standard that defines how AI applications can connect to external data and services in a consistent way.

Think of MCP as the *"USB-C of AI integration"*: a universal adapter that lets any compliant AI model talk to any compliant tool or database without custom engineering each time. Instead of building one-off bridges between every model and every system, developers can implement MCP once for each side.

This shifts the complexity from *M*×*N* to *M*+*N*: each model and each tool integrates with MCP *one time*, and then everything can interoperate. For businesses, this standardisation dramatically reduces integration effort and maintenance costs.

New AI capabilities can be introduced faster and with less risk of breaking when APIs change. Essentially MCP acts as a stable interface layer.

Inside MCP: How it works and what it offers

MCP introduces a simple yet powerful architecture for AI-tool interactions. It breaks the system into three roles (Host, Client, Server) to separate concerns and ensure modularity:

- **MCP Host:** The host is the AI application or agent that the end-user interacts with. For example, a chatbot interface, a virtual assistant in an IDE, or a custom enterprise AI app. The host is essentially the "brain's environment". It maintains the conversation with the user and decides when to hit external knowledge or actions. The host does not directly implement connections to tools, it delegates that to the client. This design keeps the host focused on AI logic and user experience.

- **MCP Client:** The client is embedded in the host application as a protocol translator and router. It's responsible for managing the connection to one or more MCP servers. The MCP client opens communications, negotiates protocol details, and routes requests/results back and forth. You can think of it as the adapter that speaks the MCP standard on behalf of the AI model. If the AI decides to call a tool (i.e. a "get customer record" function), the client packages that request in the MCP format and sends it out. It listens for any incoming data or notifications from servers, and each connected server typically gets its own dedicated client instance to keep things sandboxed and secure.

> ***"MCP prevents your AI from becoming a monolithic mess by splitting the job in two: The Host handles the 'Why' (logic and experience), and the Client handles the 'How' (routing and security). By sandboxing these connections in the Client, you ensure that adding new tools doesn't break the brain of your application."***

- **MCP Server:** An MCP server is a lightweight service that wraps around an external resource. This could be a database, an API, a file system, or any tool. The server advertises its capabilities and handles executing the actions or queries on the actual resource when asked. The key here is that an MCP server is model-agnostic. For example, you might have an MCP server in front of your CRM system. Whether the request comes from an OpenAI model, Anthropic model, Gemini model, or a fine-tuned open-source model, the server's interface is the same. This decoupling makes tools durable and re-usable across different AI solutions.

By enforcing this 'host → client → server' structure, MCP creates a clear separation of concerns. AI developers (working on the host side) don't have to write custom API calls, and tool owners (providing the server) don't have to worry about prompt engineering.

Each side adheres to the MCP contract. This greatly improves scalability and maintainability for complex AI systems.

"At its core, MCP is an insurance policy against rapid AI changes. It decouples your valuable data from the specific AI model of the month. Whether you use OpenAI, Anthropic, or open-source, your tools stay the same, keeping your system secure, modular, and ready for whatever comes next."

Inside MCP Primitives: Tools, resources and prompts

To enable rich interactions, MCP defines a few core ‘primitives’. Primitives are the vocabulary that AI agents and tools use to communicate. MCP standardises three key primitive types:

- Tools (executable actions): These are functions like: ‘create_ticket’, ‘get_forecast’, ‘run_sql’. Each comes with a JSON schema for arguments and results. The model emits a structured tool call, the client forwards it, the server runs the logic and returns a result.
- Resources (contextual data): Read-only objects exposed via URIs (e.g. ‘file:///…’, ‘db://customers/1234’). The host can fetch them or subscribe for updates. Enabling agents to react to events (new logs, new orders, changing KPIs, etc) without constant polling.
- Prompts (reusable templates): Server-defined “recipes” such as “Review Pull Request” or “Summarise Incident”. They bundle system instructions plus relevant context, so every client gets consistent behaviour.

Together, primitives give models a grammar of action and context that works across all systems.They let the AI model’s probabilistic reasoning translate into deterministic operations on tools and data. By standardising this “grammar”, MCP ensures that whether your AI is doing customer support, writing code, or analysing data, it speaks the same language to invoke operations. This is crucial for building complex workflows that might involve multiple tools in sequence.

"Think of MCP Primitives as grammar school for your AI. It gives the model a strict vocabulary: Tools are the verbs (actions), Resources are the nouns (data), and Prompts are the rules. This structure forces the AI to stop just 'guessing' probabilistically and start executing deterministically."

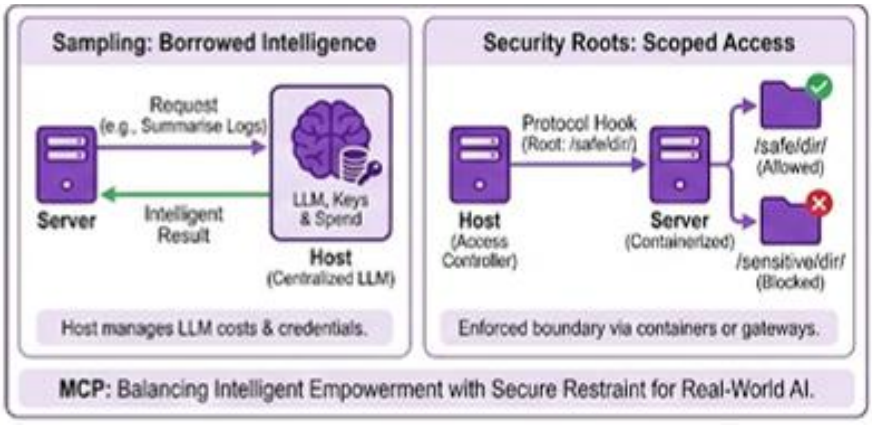
Advanced MCP Features: Sampling and Security Roots

Beyond basic request-response, MCP has some advanced capabilities that enable more intelligent and secure behaviour:

- **Sampling:** Servers can ask the host's model for help (e.g. "summarise these logs before I return them"). This keeps LLM keys and spend centralised in the host, while letting tools borrow intelligence.
- **Roots:** A way for the host to tell servers where they're allowed to look (e.g. specific directories). It's a protocol-level hook for scoped access, which you then enforce with containers, sandboxes or gateways.

By addressing intelligent behaviour through sampling and tightening security with roots, MCP is evolving to cover the nuances of deploying AI agents in the real world.

These features help maintain a balance between empowering AI (to do more on our behalf) and restraining AI (to prevent unintended consequences or data leaks).



Securing the MCP Ecosystem

Modern Model Context Protocol (MCP) implementations unlock powerful AI capabilities, but they also introduce a new class of security risks. For European enterprises under GDPR, NIS2 and the emerging AI Act, MCP must be treated as critical infrastructure, not just developer tooling. Key risk areas as follows:

- **Prompt injection & tool abuse:** With MCP, a poisoned document or ticket can instruct the model to call sensitive tools or disclose data. This turns prompt injection from a model quality issue into remote code execution by proxy. If personal data is accessed or exfiltrated, you're looking at a likely GDPR incident alongside other infringements of governing legislation acts.

- **Over-privileged servers & confused deputies:** One MCP server often holds broad credentials for GitHub, Jira, prod DBs and more. The model isn't good at least-privilege reasoning, so it acts as a "confused deputy" with the server's full rights. Without per-user scoping and strict permission boundaries, a single compromised session can become full-scale cross-system access.

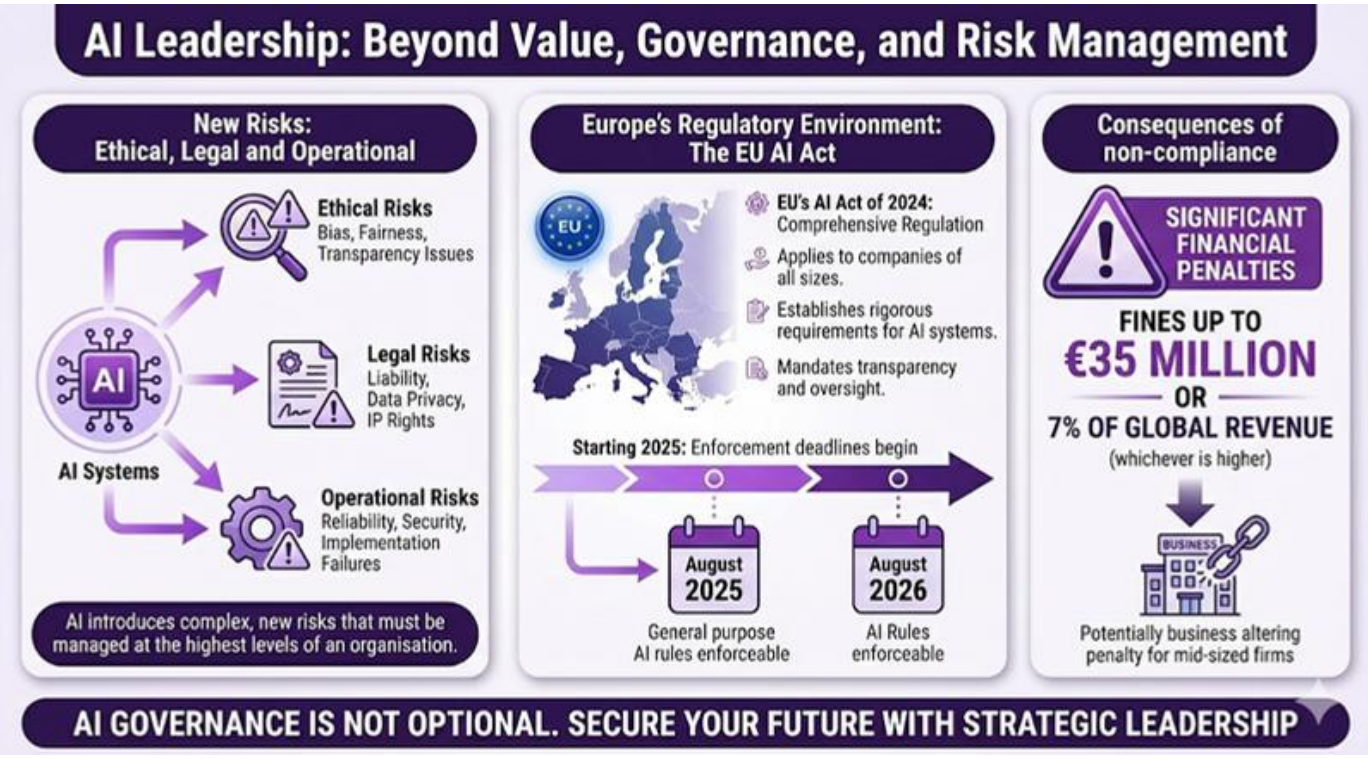
- **Secrets, tokens & credential sprawl:** MCP servers store API keys, OAuth tokens and DB credentials, making them prime targets for bad actors. Plaintext configs, shared long-lived tokens and over-permissive roles mean compromising one server can give indirect super-admin access to many systems, undermining GDPR's "security of processing" requirements.

- **Data exfiltration & privacy leakage:** Agents can read files, query internal databases and hit internal HTTP endpoints. A tricked or misconfigured agent can pull sensitive information into its context and leak it to users, third-party APIs or logs. Uncontrolled data flows like this sit squarely in GDPR breach territory and clash with NIS2's expectations for essential service resilience.

- T**hird-party MCP supply chain:** Marketplace or open-source MCP servers run with your permissions and access. If malicious or compromised, they can exfiltrate tokens, inspect files or inject backdoors. Under NIS2 and the Cyber Resilience Act, this becomes a formal software supply-chain risk that must be managed and evidenced.

- **Transport, sessions, logging & multi-tenancy:**
  - Unsecured long-lived connections (weak TLS, poor auth, leaky session tokens) invite MITM and hijacking.
  - Verbose logging can create a shadow data lake full of PII and secrets if not carefully sanitised and accessed-controlled.
  - In shared IDEs, SaaS agent builders or internal platforms, weak isolation and shared caches risk cross-tenant data exposure.

- **Governance, compliance & auditability:** Boards, CISOs and regulators now ask: "Who authorised this AI action?", "What data moved where?", "Can we replay the decision path?". MCP platforms need structured, privacy-aware audit trails of tool calls, parameters and outcomes, plus clear maps of cross-border data flows.



Across MCP deployments, **Mai** implements the following mitigation playbook for our customers:

- **Least privilege by design:** Small task specific MCP servers, narrowly scoped, short-lived tokens & regular permission reviews.
- **Tool-level and user-aware access control:** Per-user/role scoping, allow-lists and human approval for high-impact actions (HITL: Human-in-the-Loop).
- **Prompt-injection defences:** Strong system prompts, input/output filtering, schema validation and sanity checks on tool responses.
- **Ecosystem hardening:** Security review and allow-listing of MCP servers, signing and checksums, sandboxing, vulnerability monitoring.
- **Shift-Left-Testing:** Shifting the development testing of left within the SDLC process. This ensures testing is done at early stages within the dev process and carries through the entire SDLC process.
- **Network & platform security:** TLS and strong auth on all MCP transport, network segmentation, egress controls, and use of AI/API gateways as choke points.
- **Monitoring & audit:** Structured logging of tool usage, anomaly detection, red-teaming of MCP scenarios, and "circuit breakers" to disable tools when behaviour looks unsafe.

Handled this way, MCP becomes a powerful enabler rather than a new attack surface. But only if security, compliance and platform engineering move in harmoniously from day one.

AI Gateways: New control plane for AI Operations

While MCP standardises how AI and tools connect, it doesn't by itself handle broader operational concerns. This is where AI Gateways come into play. An AI Gateway is like a traditional API gateway or reverse proxy, but purpose built for the unique demands of AI and LLM applications.

As companies start putting AI into production, they face challenges that classic web infrastructure wasn't designed for. Things like unpredictable query costs, malicious prompts, or multi-model orchestration. An AI Gateway addresses these by acting as a central control plane for all AI model traffic.

Think of an AI Gateway as a smart traffic cop that sits between your AI-driven applications (including MCP hosts) and the various AI model APIs or inference servers they use.

Unlike a generic proxy that treats requests as opaque blobs, an AI Gateway actually understands the contents of AI requests: it knows about prompts, tokens, model parameters, etc. This awareness enables a range of AI-specific capabilities that improve performance, safety and cost-efficiency.

"If MCP is the wiring, the AI Gateway is the smart traffic cop. Unlike a standard proxy that blindly waves data through, the Gateway actually inspects the traffic. It spots malicious prompts, caps runaway costs, and manages the flow before it ever hits your expensive models."

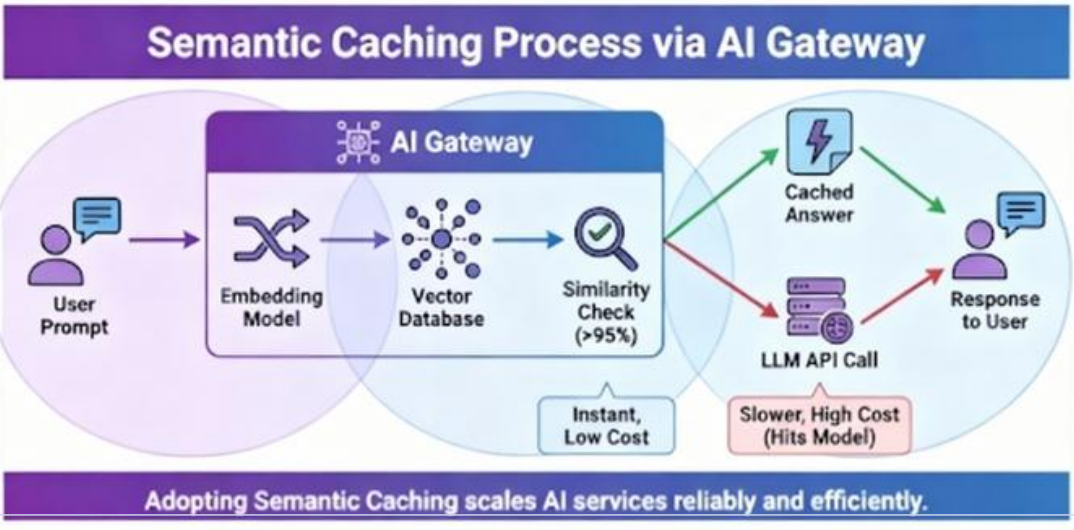
Smarter caching with semantic understanding

Traditional web caching (like what CDNs or HTTP proxies do) works by exact matches; if the URL and headers match a previous request, return the cached response. That doesn't work well for natural language queries, as the same question can be phrased in countless ways. For example, "Who is the CEO of Apple?" vs "Name Apple's chief executive.", two different strings that likely produce the same answer. A naive cache would treat them as misses.

An AI Gateway implements semantic caching to solve this. This means using vector embeddings of the queries to detect similarity. When a new prompt comes in, the gateway computes an embedding (a numerical representation of the query's meaning) and compares it to embeddings of past queries in a vector database. If it finds a very similar one (for example 95% similarity or above), it can confidently reuse the cached answer.

This can cut both latency and cost dramatically, users get instant answers (because it doesn't even hit the model. Just like a traditional cache) and you save on expensive LLM API calls. In practice, this is like an LLM-specific memorisation strategy. Some open-source tools (like GPTCache or Upstash's Semantic Cache) do this, and enterprise AI gateways are adopting it too.

The result is an AI system that learns once, answers many, a crucial trait when scaling customer facing Q&A or internal knowledge bots where many users might ask variants of the same questions.



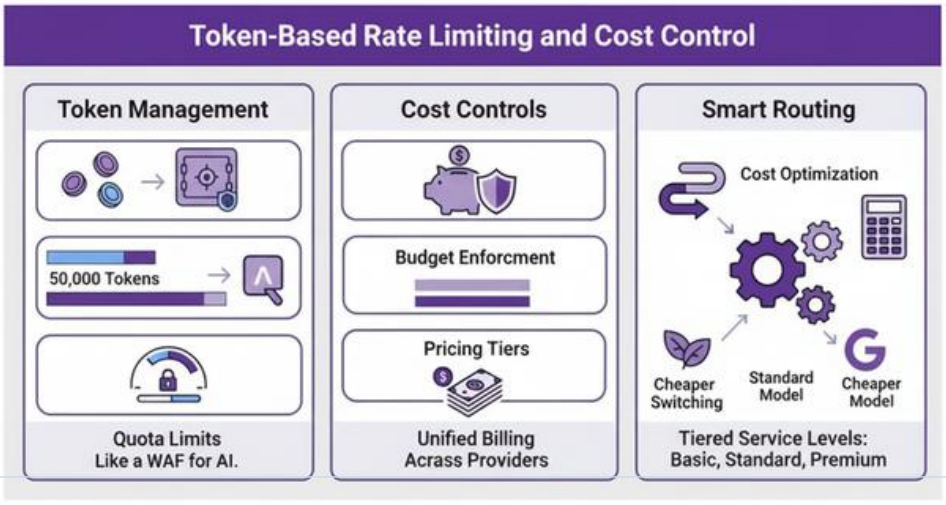
Token-based rate limiting and cost controls

In the world of APIs, we usually rate-limit by requests per second or minute. But AI models have a different cost metric: tokens. One request might be a short prompt that uses 50 tokens, another could be a long document analysis using 50,000 tokens. From a cost and load perspective, these are wildly different.

AI Gateways introduce token-based rate limiting to manage usage in a fair and predictable way. For example, you could set a policy that each user can use up to 100,000 tokens per day, or that a given backend model is limited to 1 million tokens per hour to control spending. The gateway will count tokens in real time (input + output) and enforce those budgets.

If a user exceeds their quota, the gateway can queue or reject further requests, or switch to a cheaper model. This granular control prevents one heavy user or runaway process from blowing the monthly AI budget or monopolising the model's capacity. It's essentially LLM throttling based on actual usage rather than arbitrary call counts. For a business, this means predictable costs and the ability to offer tiered service levels.

Beyond rate limiting, AI gateways often provide unified billing and monitoring for usage. If you're using multiple model providers (OpenAI, Anthropic, Gemini, etc.), the gateway can aggregate usage stats and costs in one place. This is invaluable for FinOps, you get visibility into how AI is being used across the company, which teams or features are driving usage, and where you might optimise prompts or caching to save money.



Dynamic model routing and resilience

Another strength of AI gateways is the ability to do dynamic routing of requests. In a production environment, you might have multiple AI models that could serve a request some might be faster, some cheaper, some more accurate for certain tasks. The gateway can act as a smart dispatcher. For instance:

- Fallbacks: If the primary model fails (maybe the API is down or it hits a timeout), the gateway can automatically retry the request on a backup model. Your application doesn't see an error; it still gets a response, perhaps with a slight delay. This increases reliability. For example, if OpenAI's API has an outage, the gateway might route to an internal model or another provider like Gemini, Azure or Anthropic.

- Multi-Model A/B Testing: You can configure the gateway to send 10% of traffic to a new model while 90% goes to the default. This way you can compare outcomes (quality, user feedback, costs) before fully switching. It's an easy way to do continuous improvement on your AI stack. Some gateways allow conditional routing too, like using a smaller, cheaper model for very simple queries and a bigger model for complex ones (determined by prompt length or other properties). All of this is abstracted behind a unified API surface, so your developers don't need to change application code to switch models.

- Unified Endpoint: AI gateways often expose a single API endpoint (for example, mimicking the OpenAI Chat API schema) to your applications. Behind that, they can route to many different underlying models or even chains of models. This means your developers code to one interface, and the AI ops team can manage providers and updates in the gateway config. It decouples the application from direct dependence on a specific model provider, which is great for avoiding vendor lock-in and mitigating single vendor risk.

"The Gateway is a ruthless logistics manager. It looks at every request and decides: 'Does this need the expensive Ferrari model, or can I send it via the cheap scooter model? It routes traffic based on cost and complexity, and if one road is closed, it automatically finds a detour. Your users get uptime, you get budget control."

Safety and governance filters

When deploying AI at scale, especially customer-facing, you have to worry about things like prompt injection attacks, data leakage, or inappropriate content generation.

AI gateways serve as a checkpoint to enforce safety rules. For example, they can integrate prompt sanitisation and guardrails: if a user input contains known prompt injection patterns or disallowed content, the gateway can refuse or alter the request before it even hits the model.

Similarly, the gateway can inspect model outputs for compliance, using AI classifiers or keyword checks to ensure nothing toxic or confidential is returned. This is analogous to a web application firewall (WAF) but for AI conversations.

From a governance standpoint, the gateway becomes the central audit log and control point for AI usage. All requests and responses can be logged and monitored (with appropriate privacy controls).

This is crucial for regulated industries or just for debugging. If an AI makes a strange decision or says something it shouldn't, you have the trace in the gateway's logs to analyse what prompt led to that and which model produced it.

It also simplifies compliance with data protection rules, e.g. you can configure the gateway to redact certain sensitive fields or to ensure data stays within certain regions (especially important for EU data residency concerns).

In summary, an AI Gateway is an essential operational layer for any AI deployment. It brings the maturity we expect in traditional software (caching, rate limiting, routing, security) into the world of generative AI. By doing so, it allows organisations to scale AI services reliably, safely & securely, turning what could be a chaotic set of model calls into a managed, observable service.

"Think of the AI Gateway as the bouncer at the door of your data. It checks every request entering the building and kicks out the troublemakers (like prompt injections or toxic queries) before they cause havoc. It ensures that nothing dangerous gets in, and more importantly, that no sensitive secrets slip out."

The MCP gateway pattern: Convergence of protocol and proxy

The most exciting development is how MCP and AI gateways intersect to form a new architectural pattern: the MCP Gateway.

When you start deploying AI agents that use many MCP servers, you might connect your agent (MCP host) to dozens of servers: one for Salesforce, one for your database, one for Slack, GitHub, etc. That works, but it can become an operational challenge. Each server might require separate credentials, logging, monitoring, and each one might stream back events that need coordination. Without a hub, you run the risk of having tangled connections (the return of an N×M problem).

An MCP Gateway sits in the middle to simplify this. It's essentially an AI Gateway that "speaks" MCP on both sides. It presents a single MCP interface to the AI agent and connects to multiple backend MCP servers. From the AI's perspective, it's just one big server with many tools; from the backend perspective, it's a client aggregating requests. This has several benefits:

- **Consolidated Tool Discovery:** The gateway can aggregate the tools from all your internal servers and present them as a unified catalog. For example, instead of the AI having to connect to 5 different servers and list tools from each, the gateway could list all tools in one go. Under the hood it knows which backend to route each tool call to. This not only is simpler for the AI, but it lets the gateway filter and control that list. If certain agents shouldn't use certain tools, the gateway can simply omit them. This acts as a capability firewall, very useful for enforcing least privilege for different AI agents.

- **Single Connection & State Management:** MCP sessions can be long-lived with streaming responses and such. Managing a dozen open SSE streams (one to each server) can be heavy for a client. The gateway can maintain those connections on the backend, but the AI host only deals with one connection (to the gateway). The gateway can be stateful, remembering the session context, handling reconnects if needed & buffering messages. Essentially it offloads the complexity of juggling multiple async tool streams from the AI app. This is important if an agent is running in a constrained environment or if you want to simplify client-side logic.

- **Centralised Observability:** When all MCP traffic goes through one gateway, you get one log stream to watch. The gateway can log every tool invocation, every resource access, how long they took, etc. Your ops team can monitor this single point for anomalies or performance issues, rather than aggregating logs from many servers. This means easier auditing: if you need to review what actions an AI took over a session (which tools it called with what arguments), the gateway log provides a chronological record. This is extremely useful for debugging agent behaviour or investigating incidents.

- **Security Enforcement:** An MCP gateway can enforce security at a higher level. For example, it might examine the arguments going to a tool, if an AI is calling a database query tool and the argument is a SQL string that attempts 'DROP TABLE', the gateway could intercept that and block it before it reaches the actual database. Similarly, the gateway can scrub or validate outputs from servers if needed. It becomes a choke point where security policies can be applied uniformly, instead of hoping every MCP server implements them correctly. Essentially, the MCP gateway can serve as a policy enforcement point for all agent actions, which is much easier to manage.

Implementing an MCP Gateway is non-trivial because it must maintain state and handle streaming, but several solutions are emerging. Kong's AI Gateway is one. There are also open-source efforts and proxies being developed to fill this role. The pattern to note is that protocol (MCP) and proxy (gateway) are converging. The ultimate goal is that AI agents will plug into an internal "AI mesh" in an enterprise where one or a few gateway services mediate all AI interactions with tools and data. This mesh will allow thousands of AI agents and tools to discover each other and interact safely, without building custom links for each pair.

From **Mai's** perspective, adopting the MCP Gateway pattern means you're setting up an internal AI platform, a layer that abstracts and manages AI's interface to your business systems. It's like how enterprises moved from individual database connections everywhere to centralised data warehouses and data lakes. Here we're moving from scattered AI experiments to a cohesive AI integration layer that is governed, scalable, secure & observable.

Business Impact: Building a sustainable AI moat

Adopting technologies like MCP and AI gateways isn't just an AI modernisation/adoption, it directly contributes to a company's competitive moat in the AI era. Here's why:

- **Acceleration of AI Deployment:** Standardising on MCP can speed up development by eliminating integration work. Engineers spend less time writing plumbing code and more time on actual AI logic and user experience. This faster iteration means you can deploy new AI-driven features to customers quicker and respond to market opportunities in weeks instead of months. For internal use, it means you can empower more teams with AI assistants without needing a huge platform team to custom connect everything. Early adopters of MCP have experienced substantial reduction in integration effort. This efficiency is a force multiplier for your R&D investment.

- **Reduced Maintenance & Flexibility:** With a decoupled MCP architecture, changes are easier to manage. If a tool API changes version, you update the MCP server once and all AI agents benefit immediately, with no prompt re-engineering or new fine-tunes needed. If a better LLM comes along, you can swap the AI in the host without rewriting how it talks to tools. This modularity cuts maintenance costs. It also means you're less prone to vendor lock-in, your whole AI ecosystem isn't tied to one provider's SDK or plug-in format. Strategically, that puts the power back in your hands to choose the best models and services over time without being stuck.

- **Operational Cost Savings:** AI Gateways directly help control costs. Semantic caching can drive down repetitive query calls. Token-based rate limits prevent budget overruns by design. Dynamic routing allows use of cheaper models where possible (e.g., use a smaller model for trivial requests, saving the expensive model for only the hard cases). All of this means you can deliver AI features at scale with a more predictable and optimised spend. In a boardroom context, that's moving AI projects from blank-check research to something that can be budgeted and tracked like any other service.

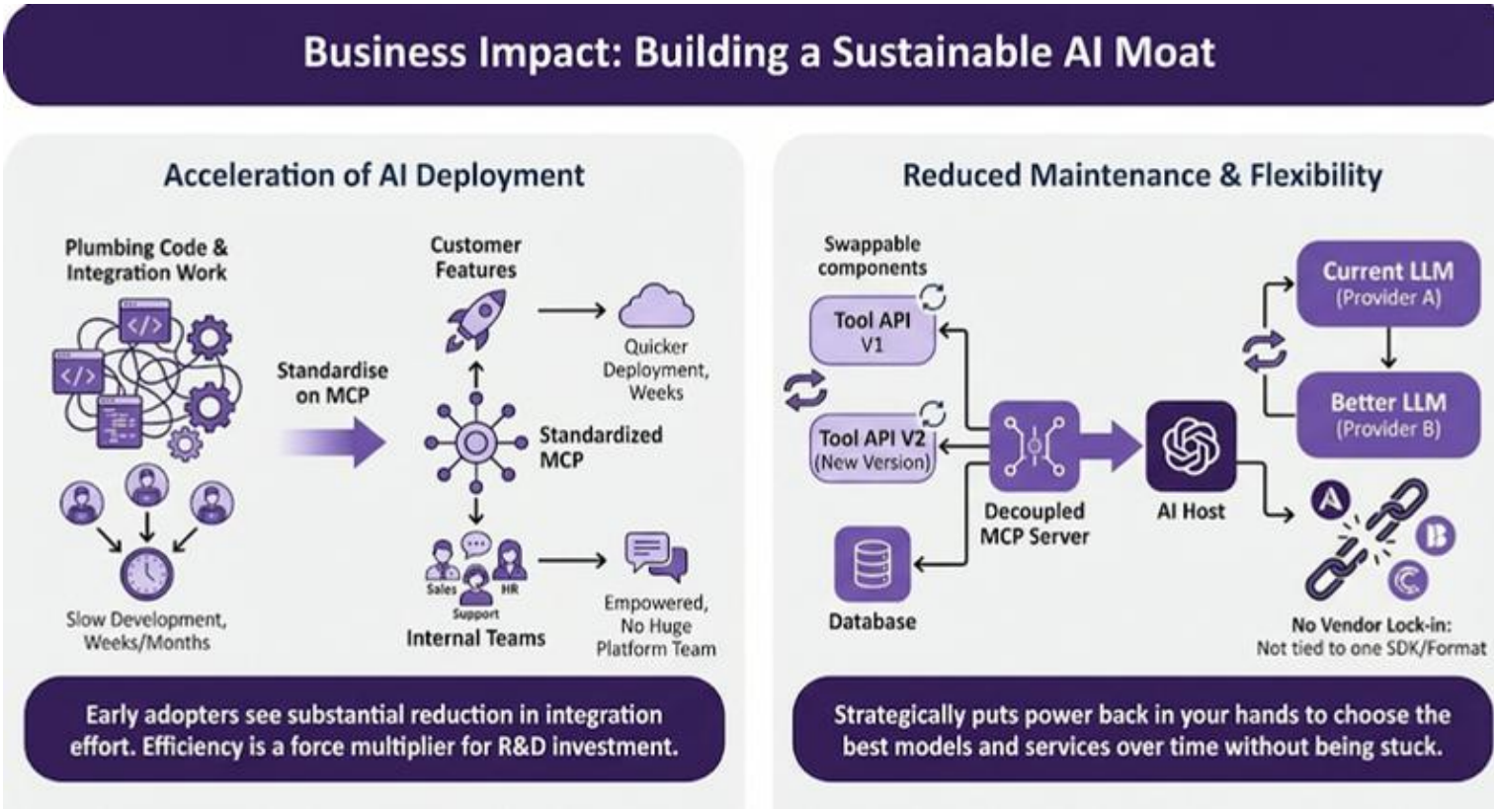
- **Security and Trust as Differentiators:** In the EU especially, consumers and regulators are very conscious of data privacy and secure AI usage. By implementing robust gateways and MCP with access controls, you're building AI systems that are secure by design. Data stays where it should, and every action an AI takes is authenticated, authorised, and logged. This not only reduces risk (avoiding headline-grabbing incidents of AI leaks or rogue AIs), but can become a selling point. You can assure customers that your AI features comply with GDPR, that their data won't be sent off to unknown third-parties, and that there are safety nets against inappropriate content. Companies that can build trust in their AI will stand out. As AI's adoption with customers will hinge on comfort and reliability.

- **Technical Moat via Internal AI Network:** The biggest long-term advantage is what I call the "internal agent network". By laying down the MCP + Gateway infrastructure, you enable a future where hundreds of AI agents and automations collaborate across your organisation's data and processes. Think of it as an AI-powered nervous system for the company. Over time, you'll accumulate specialised agents, one for finance auditing, one for market research, one for DevOps monitoring, etc. Through MCP, they can all safely plug into the tools they need and even interact with each other. This kind of integrated AI capability becomes a self-reinforcing asset, the more you have, the more you can do. It's not something a competitor can buy off the shelf tomorrow; it's built through experience and data. In contrast, a company that merely uses a couple of external AI APIs here and there (without integration) won't achieve the same depth of automation or intelligence. In essence, adopting MCP and AI gateways early helps you lock in AI advantages, you're building the foundation on which more and more AI-driven efficiencies and innovations can pile on, and that foundation is hard for others to replicate quickly.

Lastly, consider the "USB-C moment" analogy for the industry. When a standard like USB-C became widespread, it commoditised connectors, the value moved to what you do with the connection (the devices, the data). Similarly, as MCP standardises AI-tool integration, simply having an integration is not the differentiator, everyone can have connectors.

The differentiator becomes what you do with it: the proprietary data you can feed your AI, the unique workflows you orchestrate, and the quality of the models or agents you train on your data. By embracing these standards, businesses ensure they're competing on insight and innovation, not on plumbing.

Those who don't adopt will find themselves stuck with fragile, bespoke systems ('shadow AI' projects that don't talk to each other), and they'll waste time catching up on integration basics while the leaders are iterating on higher-level solutions.



Adopting MCP & AI gateways is a strategic moat. Build a future proof AI Ecosystem.

Conclusion: A blueprint for AI enabled enterprises

The convergence of the Model Context Protocol and AI Gateways represents a new AI architecture for enterprises, one that marries the creativity of AI with the rigor of software engineering. MCP provides the lingua franca that allows AI to work with systems (not just chat with humans), and AI Gateways ensure this work happens safely, efficiently, and at scale. Together, they turn the sci-fi idea of a truly helpful, tool-using AI assistant into a practical reality in the workplace.

From a strategic standpoint, here's the blueprint we *(at **Mai Technology**)* recommend to our customers:

- **Standardise on MCP:** Encourage your teams to use MCP for any project that connects AI to data or tools. Instead of writing one-off integrations or relying solely on vendor-specific features, use MCP servers/clients. This creates a plug-and-play ecosystem where any new AI model can immediately leverage existing tools, and any new tool becomes available to all your AI use cases. It's a one-time investment that pays off in agility down the road.

- **Deploy an AI Gateway:** Just as you have API gateways for your services, put an AI gateway in front of your LLMs and agent platforms. This will give you immediate benefits: unified monitoring of AI usage, cost control, and protection against misuse. It's an essential piece of turning prototypes into production services. Whether it's a solution from Nvidia, Kong, Cloudflare, or others, the key is to have that control plane in place so you can manage AI, not just hope for the best.

- **Plan for MCP Gateways (Aggregation Layer):** As your AI usage grows, avoid the temptation to let it sprawl unmanaged. Design an architecture where there is a central MCP gateway (or a small number of domain-specific ones) that broker all agent tool interactions. This will be your strategic stronghold for applying security rules, compliance checks, and optimisations. It also abstracts complexity away from individual AI applications, making them easier to build. In effect, you're creating an AI integration middleware for your company, likened to an ESB (enterprise service bus) but for AI agents. This is forward-looking, but the earlier you consider it, the cleaner your expansion will be.

We are moving from a world where AI was a cool demo like “Chat with this PDF”, to a world where AI is an active participant in our business workflows. The Model Context Protocol and AI Gateways are the bridges making this possible. They enable AI to not only understand our requests but also to act on them securely across digital infrastructure. That means higher productivity internally and more powerful, responsive experiences for customers.

For European organisations, which often operate under strict data and security regulations, this approach ensures AI adoption doesn’t run on the wrong side of compliance. You can keep data local, enforce policies, and demonstrate control. This alignment of innovation with governance is key to unlocking AI’s benefits in a responsible way.

In conclusion, by investing in these foundational technologies today, enterprises can build a sustainable advantage. An AI enhanced infrastructure that is fast, safe, and adaptable. At **Mai**, we’ve seen firsthand how companies that take this structured approach to AI deployment quickly outpace those sticking to ad-hoc experiments. The future belongs to those who create the environment for AI to thrive in harmony with their existing systems and rules. MCP and AI Gateways are that environment. Let’s embrace the path to AI that truly works for us, not just with us, as we usher in the next generation of intelligent business.

"Think of this architecture as building a Universal Remote for your Enterprise. By standardising on MCP, you build the buttons once. By adding a Gateway, you control who gets to press them. This stops you from building a new remote every time a better TV (AI model) comes out, giving you a massive speed advantage."

MAI - DESIGNING INTELLIGENT FUTURES