HITACHI

Skills are all you need

A capability-first architecture for enterprise agent ecosystems

Centre for Architecture & AI (CAAI) – Vitor Domingos

March 2026

Hitachi Digital Services

Table of contents

POWERING WHAT'S NEXT

Executive Summary

Organisations are shipping agents that automate triage, coordinate remediation, generate reports, and manage infrastructure.

The pressure is to deliver quickly. The architectural risk is that each agent becomes a self-contained product, carrying its own connectors, business rules, and governance logic.

This CAAI whitepaper proposes that the unit of scale should not be the agent. It should be the **skill:** a governed, callable capability with a stable contract, enforceable policy, and operational telemetry. Agents remain valuable for interpreting intent, planning multi-step work, and managing dialogue. But execution and business logic belongs in skills, where it can be reused across organisations and teams, properly audited, and costed at the unit level.

When skills are made the unit of delivery, capability becomes composable: a skill built once can serve multiple agents, workflows, and applications without re-implementation. Governance becomes systematic, because controls sit at the skill boundary rather than scattered across prompts and bespoke middleware. Economics become legible, because bounded execution produces bounded cost - which is what managed services packaging and portfolio investment decisions actually require.

> **Core approach:** treat skills as enterprise capabilities with contracts, policy, and evidence. Use agents for orchestration, not for embedding integrations and privileged actions.

A note on the title: it is a deliberate nod to "Attention Is All You Need" (Vaswani et al.), the paper that established the transformer architecture and set the direction of travel for today's GPT-class models. The intent here is not to claim a similar scientific breakthrough, but to signal the same design instinct: pick the right primitive, and scale becomes manageable.

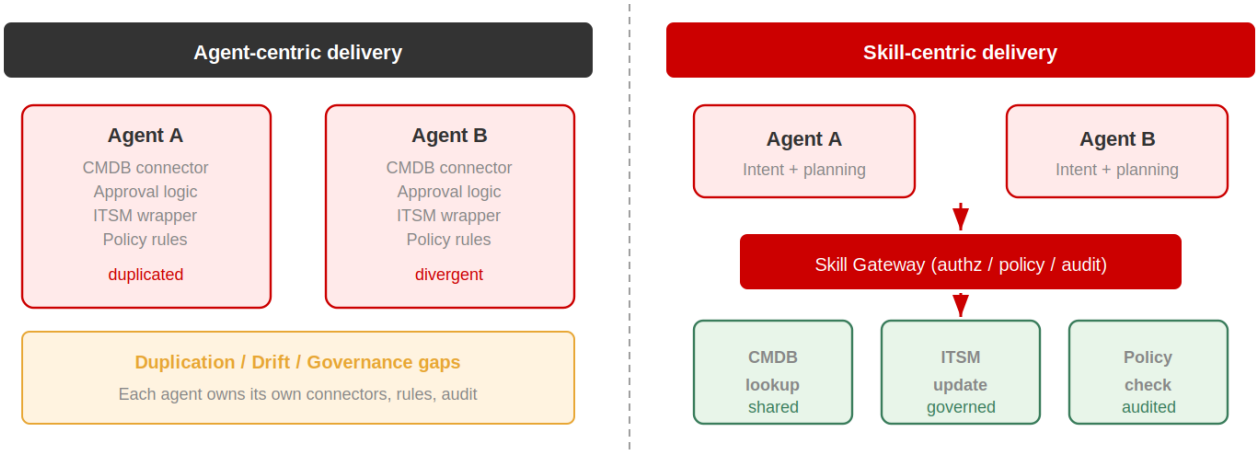
CONTEXT

The Agentic Scaling Problem

Agent frameworks make it easy to build an assistant that calls tools, retrieves knowledge, and takes action. That convenience obscures a structural trap: each agent typically bundles its own API connectors, domain rules, approval logic, and error handling. When organisations deploy a second agent that needs the same capabilities, the work is repeated. When a third arrives, divergence compounds.

The consequences are familiar to anyone who has managed microservice sprawl. Duplication inflates engineering effort because the same integration is built and maintained in multiple codebases. Drift creeps in as each team iterates on its own version of a shared capability, producing subtly different behaviour. Governance fragments: approval logic ends up living in prompts, tool wrappers, and bespoke middleware rather than behind a single enforceable boundary.

The diagram below illustrates the contrast. On the left, each agent embeds its own duplicated connectors and rules. On the right, agents become thin orchestrators that invoke shared, governed skills through a common gateway.



In enterprise environments, the same capabilities recur across agents repeatedly: identity resolution, asset lookup, policy checks, knowledge retrieval, ticket creation, data quality validation, and integration error handling. If each agent owns its own implementation, the organisation accumulates technical debt at the pace of agent adoption.

OUR APPROACH

Skills as the Unit of Capability

A skill is a callable enterprise capability with a stable contract and an enforceable control surface. The contract defines inputs, outputs, error codes, policy requirements, and telemetry expectations. A skill may wrap an API, a workflow, a data pipeline, or a model endpoint. It may use language models internally, but its external behaviour must remain bounded and testable, with a clear audit trail.

What separates a skill from a tool is governance. A tool is typically a thin wrapper over an API: useful for prototyping, but without the contract, policy enforcement, or lifecycle management that enterprise operations demand. A skill takes that same wrapper and promotes it to a managed capability with ownership, versioning, telemetry, and a defined retirement path.

Where a skill differs from an agent is scope. An agent interprets intent, plans sequences of actions, handles dialogue, and manages exceptions. A skill executes a bounded operation where the need for determinism, auditability, and cost predictability is key. The architectural move is to separate reasoning from execution.

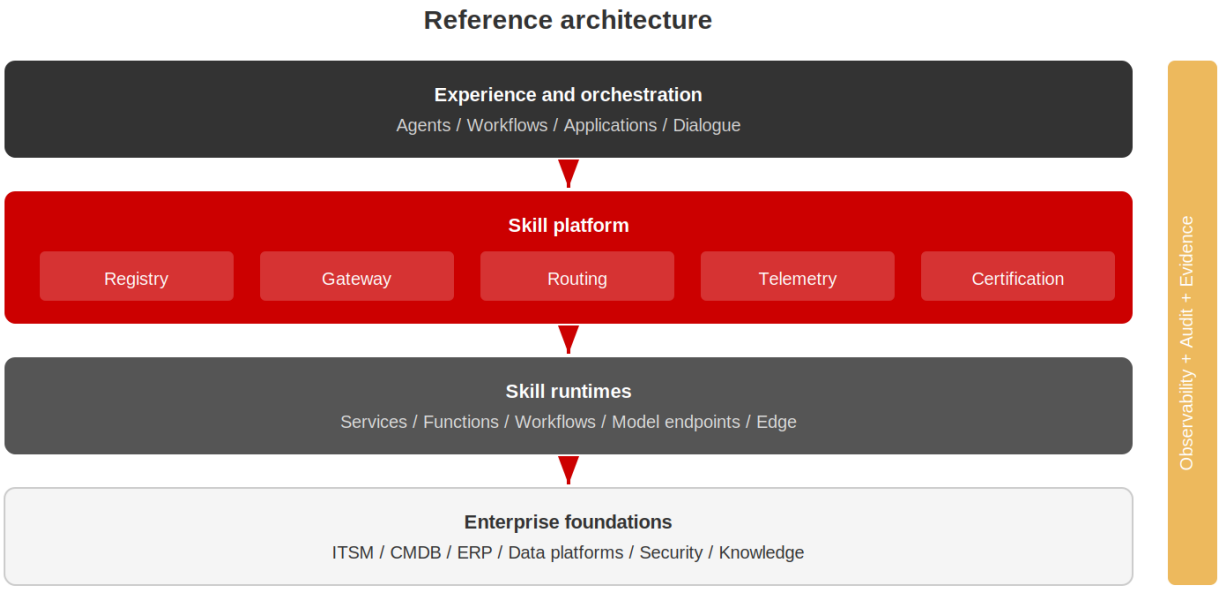
> If a capability can be executed deterministically, execute it as a skill. Reserve open-ended reasoning for work where deterministic execution is not available or not economical.

ARCHITECTURE

Reference Architecture

A skill-centric architecture has four layers. The experience and orchestration layer is where agents, workflows, and applications interpret user intent and coordinate multi-step work. The skill platform layer provides the control plane: a registry for discovery, a gateway for enforcement, routing for locality, telemetry for observability, and certification for lifecycle governance. The skill runtime layer is where capabilities execute as services, functions, workflows, model endpoints, or edge processes. The enterprise foundations layer comprises the systems of record that skills connect to.

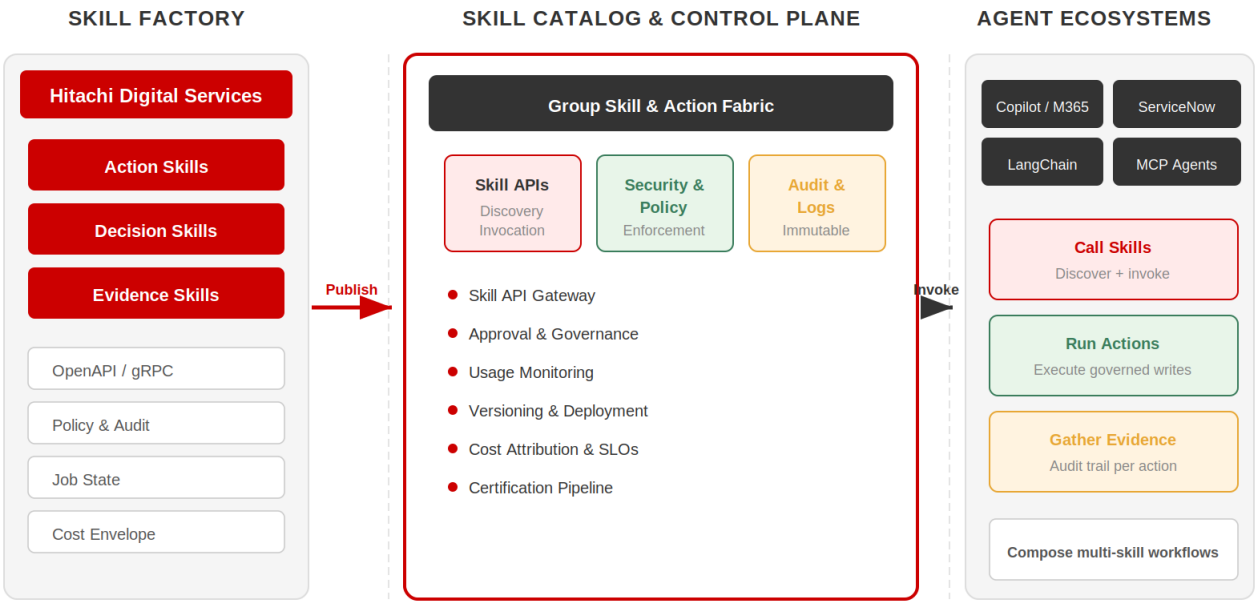
Observability and audit span all four layers. Every invocation emits traces, metrics, and structured evidence linking the agent's request to the skill's execution and the enterprise system's response.



This architecture materialises as three operational zones:

- **A skill factory** is where capabilities are authored, tested, and packaged with their manifests, schemas, policy rules, and cost envelopes.
- **A skill catalogue and control plane** provides the runtime fabric: discovery via APIs, enforcement through security and policy gates, and observability through audit logs.
- **Agent ecosystems** on the consumption side invoke skills through the control plane, regardless of whether the agent is built on Copilot, ServiceNow, LangChain, or a bespoke MCP runtime.

The diagram below shows this end-to-end flow. Skills move left to right: authored in the factory, published to the catalogue, and invoked by agents. Governance and telemetry concentrate in the middle layer rather than being scattered across each consumer.



Skills are published once, governed centrally, and consumed by any agent ecosystem

INTEGRATION

Relationship to MCP and Function Calling

The closest relatives to skills are MCP servers and the function-calling interfaces offered by model providers. Skills are not a competing protocol - they are what we certify and operate; MCP and function calling are transport surfaces through which agents invoke them.

Function calling defines how a model requests an action. MCP defines a protocol for exposing capabilities to agent runtimes, with conventions for tool schemas, resource access, and prompt templates. A skill platform sits behind both: providing the manifest (the platform contract, covering ownership, policy, lifecycle, and cost), the gateway (the enforcement point for authentication, authorisation, quotas, and audit), and the runtime (where execution happens).

In practice, the alignment is clean: skills are exposed through MCP by default so that any compatible agent runtime can discover and invoke them. The MCP tool schema is derived from the skill manifest to prevent drift between what the platform governs and what the agent sees.

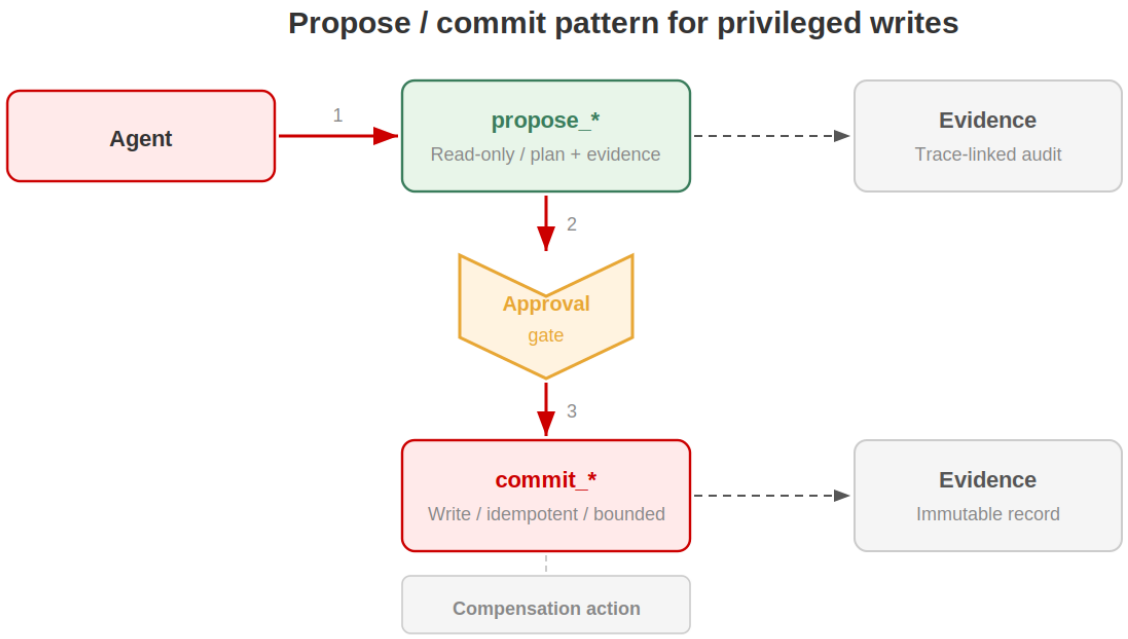
> MCP is the interoperability layer. The skill gateway is the control plane. The manifest is the single source of truth.

Composition and Transaction Semantics

Enterprise and mission-critical work is rarely a single call. A vulnerability remediation workflow involves asset lookup, severity assessment, change-window validation, patch deployment, and status update. A predictive maintenance flow chains signal analysis, anomaly detection, failure forecasting, and work-order creation. Skills must compose.

Four composition patterns cover most enterprise scenarios:

- **Read chains** assemble context from multiple sources; they require read-only scopes and evidence linkage but no transactional controls.
- **Propose/commit** separates planning from execution for privileged writes: the agent calls a read-only skill to generate a plan, then a governed write skill to execute it, with an approval gate between the two.
- **Workflow skills** handle long-running or branching processes where state, retries, and compensation logic justify a dedicated orchestrator.
- **Break-glass routes** provide emergency access with tighter scopes, mandatory review, and enhanced evidence capture.



Transaction semantics in skill chains are typically compensating rather than ACID. When a chain involves writes, each write skill should support idempotency keys, bounded retries, and a compensation action that can reverse its effect. Audit coherence across the chain is maintained through trace propagation and invocation IDs that link every step to a single orchestration context.

GOVERNANCE

Security and Trust

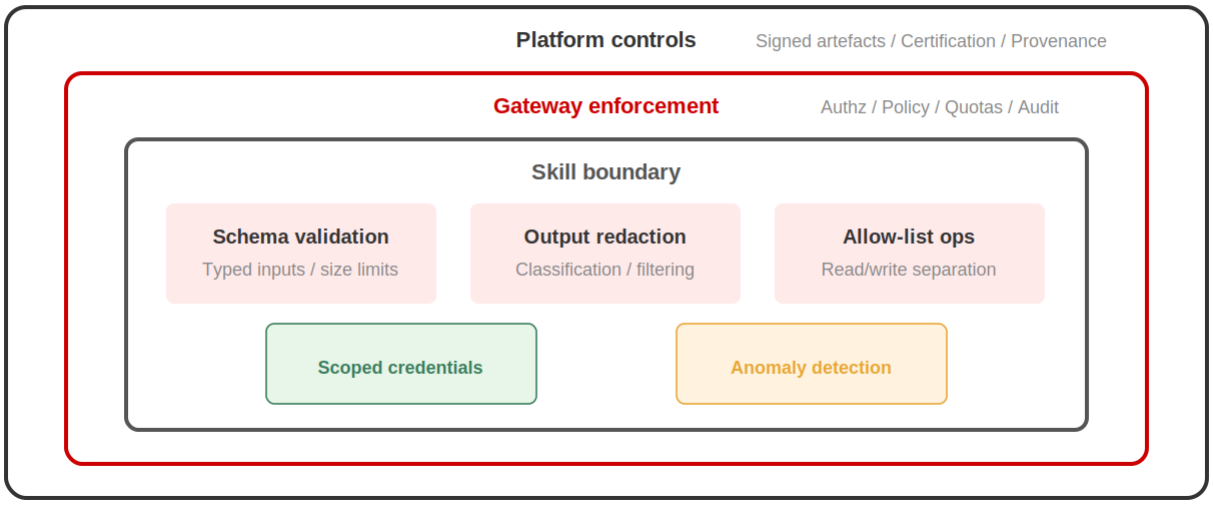
A skills-first architecture improves security posture, but only if the control plane is designed for real threats rather than aspirational compliance. The relevant threat categories are injection, lateral movement, data exfiltration, privilege abuse, and supply-chain compromise.

Injection is the most discussed risk in agentic systems. A skill-centric design mitigates it by moving privileged actions behind a gateway that enforces authorisation regardless of what the agent requests. Even if a prompt injection manipulates the agent's reasoning, the gateway independently evaluates whether the requested skill, scope, and parameters are permitted.

Lateral movement is less discussed but equally important. If an agent can invoke any skill in the catalogue, a compromised agent becomes a pivot point. Graph-aware guardrails restrict which skills an agent is permitted to chain.

Supply-chain risk arises when skills are shared across teams or published to a marketplace. Skills should be treated as software supply-chain artefacts: sign packages, require provenance metadata, and enforce certification tiers.

Defence in depth: security control layers



COMMERCIAL MODEL

Economics: Replacing Token Spend with Unit Cost

Cost volatility remains a practical barrier to agentic adoption. An agent that reasons in open-ended loops consumes tokens unpredictably, and the cost of a single task can vary by an order of magnitude depending on the model's reasoning path. Skills introduce a different economic primitive: bounded execution with measurable unit cost.

Each skill invocation can be metered at the boundary: compute time, backend calls, model tokens (if used internally), cache hits, and evidence storage. A cost envelope attached to the skill manifest declares the expected p50, p95, and maximum cost per invocation. If bounded cost cannot be guaranteed, the capability stays experimental.

This creates a foundation for commercial packaging. Managed services can offer: consumption models (charged per invocation), outcome models (charged per resolved incident or completed work order), capacity models (charged for platform runtime and SLOs), or tiered catalogue access. The common requirement is that cost is attributable, predictable, and defensible - which skills provide and agent-level token accounting does not.

PLATFORM

Governance and the Skill Platform

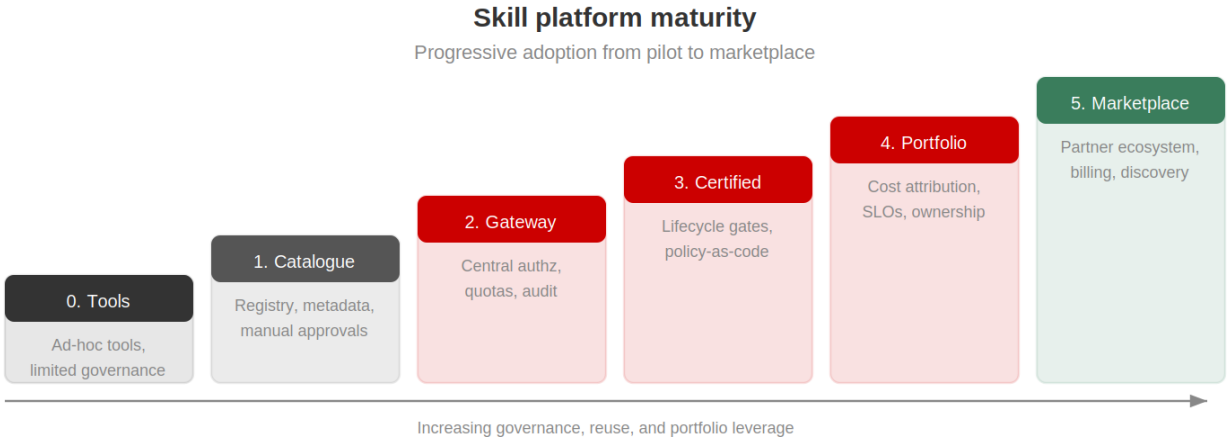
In a skill-centric architecture, skills are the action boundary, so that is where governance belongs. The skill platform makes this enforceable in practice. It provides a registry for discovery and metadata, a gateway for authentication, authorisation, policy evaluation, and quotas, routing for locality and resilience, telemetry for traces, metrics, and cost attribution, and lifecycle tooling for versioning, certification, and deprecation.

Without this platform, skills become another set of bespoke services with inconsistent contracts. The platform is what turns individual capabilities into a portfolio, and the portfolio is what creates compounding value: each new skill added to a curated catalogue reduces the marginal cost of the next agent or workflow that uses it.

ADOPTION ROADMAP

Maturity and Adoption

Organisations typically move through five stages, from ad-hoc tool usage to a fully externalised marketplace. Each level adds governance, reuse, and portfolio leverage.



Most organisations should target Level 2 (gateway enforcement) within the first 90 days and Level 3 (certified skills) within six months. The practical adoption sequence:

- Weeks 1–2: inventory duplicated tools across existing agent implementations.
- Weeks 3–4: define the minimal manifest schema and gateway policy templates.
- Weeks 5–8: stand up the registry, gateway, and telemetry infrastructure.
- Weeks 9–12: migrate five to ten high-reuse capabilities into certified skills while retiring duplicates.

> If organisations cannot describe a bounded cost envelope and a rollback path, they should not treat the capability as a production skill.

WORKED EXAMPLE

Vulnerability Remediation

Consider a hybrid vulnerability management programme where remediation requires change control. The baseline architecture uses three agents; triage, patch coordination, and reporting, each embedding its own connectors to the vulnerability scanner, CMDB, ITSM platform, and knowledge base. Nine connectors are maintained across the three agents, with inconsistent approval enforcement and manual audit assembly.

A skill-centric redesign publishes six certified skills: CMDB lookup, vulnerability prioritisation, change-request approval, patch deployment, ITSM update, and knowledge retrieval with citations. Write skills enforce propose/commit with approval tokens. All invocations route through the gateway, which handles authentication, policy evaluation, and trace propagation.

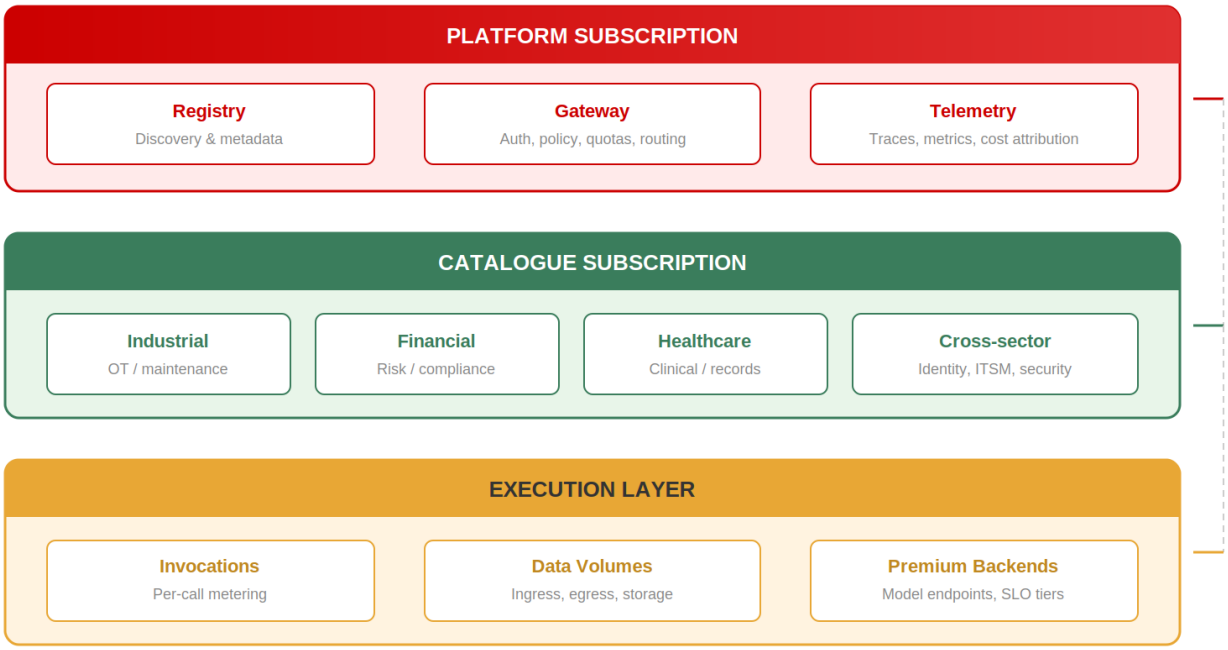
Illustrative Metrics

Metric	Before	After	Comment
Connectors maintained	9	3	Shared skill library removes duplication
Write approvals enforced	Inconsistent	100%	Approval tokens enforced at gateway
p95 cost per case	1.00x	0.55–0.65x	Reduced loops, caching, bounded execution
Audit completeness	Manual	Automatic	Evidence emitted at skill boundary

COMMERCIAL PACKAGING

HDS Managed Services and Commercial Transparency

Our practical packaging model uses three layers. The platform subscription covers the registry, gateway, and telemetry infrastructure. The catalogue subscription provides access to curated sector skill libraries. The execution layer charges for invocations, data volumes, and premium backends. Enterprises can implement the same controls internally; the architectural requirements remain the same regardless of operating model.



Enterprises can implement the same controls internally; the architectural requirements remain the same regardless of operating model

CONCLUSION

Executive Conclusion

Enterprise AI is heading towards agents with greater scope and greater autonomy. The architectural question is where to put the capability that makes that work safe, repeatable, and economical.

This paper's answer is: in skills. Build skills as enterprise capabilities with contracts, policy, and evidence. Expose them through MCP for portability. Govern them through a platform that enforces controls and emits telemetry. Use agents to coordinate skills, not to embed them.

The portfolio that results is a strategic asset. It compounds with every new skill added and every new agent that reuses existing capability. Models will change. Agent frameworks will change. A well-governed capability catalogue endures.

APPENDIX A

Skill vs Agent Decision Matrix

When deciding whether a capability should be implemented as a skill or left within an agent, the following criteria provide a practical framework.

Question	Prefer a skill when...	Prefer an agent when...	Prefer a hybrid when...
Is the operation privileged or changes state?	It writes to systems of record, affects production, or needs approvals	It never changes state; it only reasons, explains, or drafts	The agent proposes and routes; a skill commits under policy (propose/commit)
Is the intent ambiguous or exploratory?	Inputs are clear and structured; the user knows what they want	The user's goal is fuzzy; needs dialogue, clarification, or hypothesis-building	The agent clarifies intent then calls skills with clean parameters
Is the outcome deterministic?	Rules/contracts exist and are testable	Outcome depends on judgement, trade-offs, or incomplete context	Agent makes the decision; skills perform bounded checks/actions
Is the work repeatable at scale?	High-frequency capability shared across teams/domains	Low-frequency, highly situational work where building a skill is overhead	Reuse exists, but variability is high: skill handles the stable core; agent handles the edge cases
Is evidence mandatory?	You need audit artefacts, traceability, and signed decisions	You need a human-readable narrative, rationale, and stakeholder comms	Skills produce evidence; agent produces the explanation and the "why now"
Is latency critical?	Low-latency execution, edge/local routing, predictable runtime	Latency can be absorbed by interaction and human-in-the-loop	Agent keeps the user engaged; skills run in parallel, with time budgets
Is the workflow multi-step and stateful?	It's a bounded transaction or a single step with clear rollback	It's mostly cognitive sequencing: gather, compare, decide, summarise	Agent orchestrates the chain; workflow/skills handle state, retries, compensation
Is the dependency fan-out high?	Many backends require consistent auth/policy/rate limits	The main work is choosing *which* path to take (triage/routing)	Agent selects the route; skills encapsulate each backend integration
How fast does the logic change?	Stable logic and interfaces; versioning is manageable	Rules change weekly; the "shape" of the work is evolving	Skill contracts stay stable; agent prompt/strategy iterates quickly
Who owns this long term?	A platform/domain team can own SLOs, tests, and lifecycle	It's owned by a product/team as a capability experience	Platform owns skills; product owns agent experience and guardrails

HITACHI

Hitachi Digital Services

Hitachi Digital Services, a wholly owned subsidiary of Hitachi, Ltd., powers mission-critical platforms with cloud, data, IoT, and ERP solutions, underpinned by advanced AI. With over 110 years of expertise, we drive innovation and growth for a more sustainable future.