Infosys®
Navigate your next

TECH NAVIGATOR:
A PATH TO THE AGENTIC-FIRST ENTERPRISE

Infosys® | Knowledge Institute

Contents

Foreword

As agentic AI moves from scattered pilots to full-scale production, demonstrating consistent value and resilience is emerging as the key challenge.

The bottleneck is not a shortage of enterprise imagination, but a gap in architectural readiness. Most enterprises do not yet have real-time data fabric, orchestration patterns, and governance frameworks, and they struggle to design the information environment that agentic systems demand in 2026.

This gap remains poorly understood, particularly in terms of what coordinated multiagent systems mean for enterprise architecture, business processes, and human accountability.

Yet the foundations have already been laid. A decade of cloud migration, API-first design, and agile transformation has built the scaffolding needed to progress. The challenge is that this scaffolding was designed for human-initiated workflows, so significant changes are needed for a landscape where autonomous, agent-driven software carries out critical business processes.

Hence, what's really needed is adaptation that takes into account the specific demands of autonomy while ensuring that the design thinking approach taken always keeps humans in the loop.

In our implementations, a deliberate upfront focus on the following four areas has proven to be the key to success:

- **Data and context engineering:** Getting the right data in the right context is critical to agent performance. Organizations should design retrieval pipelines, structured workflows that prepare relevant knowledge for an agent before it reasons or acts, along with memory management and prompt architecture. This ensures agents act on precisely the right information, and not on raw documents or stale prompts.

- **Ecosystem orchestration:** Production systems involve specialized agents, with capabilities that include planning, executing, and verifying outcomes. These agents must coordinate through defined patterns and interface with enterprise, data, and business platforms. These interfaces must be enhanced with semantic richness — a term used to describe the amount of meaningful information associated with a concept or word, predictable error handling, and clearly bounded capabilities.

- **Agentic experience:** Customers will no longer tolerate stiff, robotic interactions branded as intelligence. Agent experiences must be designed with the same rigor as your best digital products, ensuring that intelligence and actions are embedded in the flow of user activities to deliver a seamless, experience.

- **Engineered guardrails:** Governance must be integral to runtime architecture. Input validation, output filtering, tool-use constraints, cost ceilings, and fallback behaviors must be built into the agent's execution layer before deployment. Equally, escalation paths, decision boundaries, and audit trails should be in place before agents go live, not after.

Underpinning all four areas is agentic observability. In 2026, traditional monitoring won't suffice, and without agentic observability, trust cannot scale. A focus on implementing trace-level visibility into reasoning chains, tool calls, and decision points will make all the difference to AI transformation journeys. The path to agentic AI doesn't require starting again. It requires evolving intentionally across these four dimensions. The enterprises that move fastest and most successfully will be those where IT and business build shared agentic fluency, and do so quickly. This must be done with a focus on making humans the final arbiter of what is allowed to happen in the enterprise.

This edition of Tech Navigator provides a pragmatic, actionable, and in-depth blueprint for architecting the agentic-first enterprise. We hope it helps you move forward quickly to achieve lasting business success.

Rafee Tarafdar
CTO, Infosys

Executive summary

AI, particularly agentic AI, is dominating technology disruption in 2026.

While most enterprises have started incorporating agentic behaviors primarily within their IT development and support processes, very few have been able to make progress within the enterprise at large. Among many pilots, only a few production use cases have been deployed with tangible success.

This slow adoption comes not just from an inability to identify the right use cases, but also from the complexity of implementation, with few organizations understanding what this new and vital technology means for enterprise IT systems, architecture, processes, and governance.

It's not as if the foundations for agentic AI aren't already partly there. Enterprises have been constantly evolving and have invested heavily in a transformation journey for digital and cloud over the last decade. They have also already invested in technologies, processes, and programs that have enabled them to become more digital and agile. Yet the technology architecture that has enabled personalization and other customer benefits will need to be adapted for an age where autonomous software agents complete workflows and make decisions on behalf of humans.

To turn ambition into implementation excellence while increasing customer satisfaction requires a reimagination of what it means to be an enterprise. AI investment

decisions must be guided by those elements of the enterprise technology fabric that will lead to more trust from customers, reduced costs, and ultimately profit.

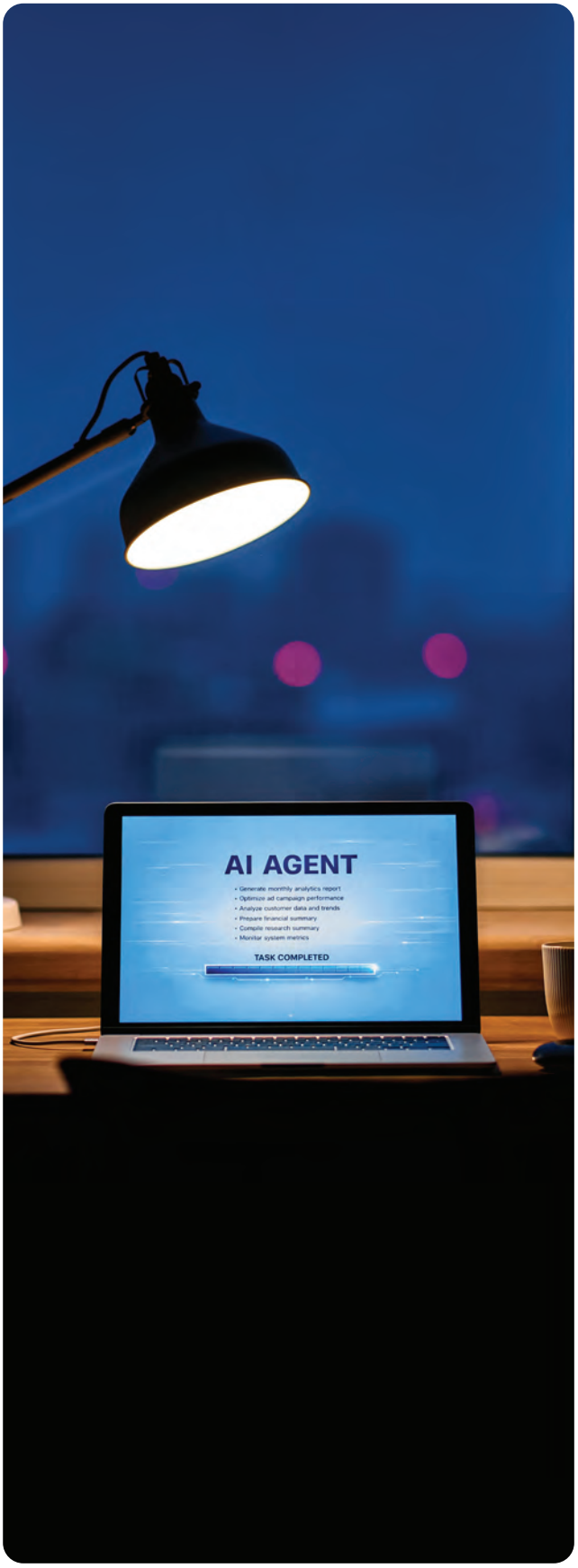
In our own implementations, we've found that moving from pilot to production requires grappling with the following key areas:

- Providing real-time data to agents so they can make accurate and informed decisions.
- Understanding the impact agents have on enterprise integration and gateways.
- Creating a good agentic experience for customers.
- Instilling agentic governance, with humans still in the driving seat.

Infosys has been working with its partners and customers to demystify these areas for clients.

Our message is simple: your existing digital foundations are enough, but only if evolved intentionally for the agentic era. IT and business teams will need shared agentic fluency, and current enterprise architects, already experts in application programming interfaces (APIs), microservices, and platforms, can be repurposed as the builders of an enterprise's agentic future.

This edition of Tech Navigator aims to position your organization at the forefront of what's coming next: a pragmatic, actionable, yet in-depth blueprint to architect the agentic-first enterprise from the ground up.



Data architecture

The success of agentic AI doesn't lie in the power of the model but in the quality of context provided to it. Here, we introduce an agentic maturity model to help engineer your organization's data stack for the new era.

Organizations are transitioning from the age of generative AI, characterized by large language models (LLMs) that act as passive retrieval and summarization engines, to the era of agentic AI, where bots are designed to perceive complex environments, reason through multistep problems, plan execution paths, and act autonomously to achieve high-level business goals.

However, not all shifts to agentic AI are equal. Industry analysis suggests a divide is forming: a vanguard of approximately 6% of enterprises is already rebuilding workflows around agents as the new operating system of the organization, while the rest just can't get pilots to work at scale because they are constrained by legacy data architectures, with poorly defined processes, lax governance, and a lack of strategic know-how making the situation even worse.

According to our Enterprise AI Readiness Radar, only 17% of organizations had prepared their data estates adequately for AI at the beginning of 2025, and as noted in recent architectural assessments, 91% of AI models experience quality degradation over time due to stale or fragmented data.

Realization of value is bounded by the underlying data estate. The ability of agents to reason and act correctly means having low-latency access to data that is high quality and provided in a structure they can consume.

Data infrastructure built for human analytics, including dashboards, quarterly reports, and advanced analytics, is not well prepared for autonomous agents, which require millisecond decision loops and continuous context for advanced AI models.

An immaculate data estate

To understand the data infrastructure requirements, we must first define the agentic loop, or the way an autonomous agent operates through a cycle of receiving, analyzing, and acting on data:

1. **Perception:** This involves ingesting multimodal signals, including text, logs, video, and audio from the environment.

2. **Reasoning:** To reason, agents utilize LLMs and domain ontologies, or maps of enterprise concepts and relationships, to understand the current state relative to a goal.

3. **Planning:** This step decomposes the goal into a sequence of executable steps.

4. **Action:** Agents then trigger tools, application programming interfaces (APIs), or database writes to change the state of the business environment, including systems, files, and workflows.

5. **Memory:** Finally, a successful agent learns and refines itself, using active working memory and long-term knowledge that has been recorded to inform future actions.

This loop creates huge demand for data infrastructure. Unlike a human analyst who might not mind a 24-hour delay in data freshness, an agent operating in real time requires a live view of the enterprise.

However, several factors obstruct agentic adoption: siloed data that forces agents to act on incomplete information; a lack of API-driven interaction patterns; limited ability to use multimodal data spanning structured and unstructured sources like emails, video footage, or audio recordings; and governance risks, as companies hesitate to deploy agents whose decision-making processes are hard to understand and potentially precarious.

The data infrastructure must support active and live data retrieval, semantic grounding, and state persistence — or the ability to remember, learn, and act consistently over time.

In this way, the data estate becomes the long-term memory and sensory processing center for the digital workforce.

The agentic data architecture

To unlock the value of data for AI, the enterprise must build what we call an agentic data stack. This architecture resolves the complications of silos, unstructured data, and governance by establishing five foundational pillars: AI data fabric, multimodal

management, domain ontology, memory architecture, and autonomous observability.

Pillar one: AI data fabric

Traditional enterprises had a very clear separation between applications which accessed analytical data and those that accessed transactional data. The perception-reasoning-planning-action loop of agentic applications creates a need for these applications to access both analytical and transactional data. Hence, agentic enterprises need to enhance their data fabric layer from traditional data warehouse and data access architectures to data-as-a-product (DaaP).

Data curation

DaaP is a methodology for curating data products which has its own schema, lineage, quality rules, ownership boundaries, service level agreements (SLAs), pipelines and access policies. This replaces the traditional pattern of aggregating all data into a monolithic lake, allowing the enterprise to preserve distributed ownership while delivering a consistent, agent-ready interface. By standardizing the data products lifecycle, the fabric ensures consistent semantics, eliminates redundant integrations, and accelerates the deployment of new agentic use cases.

While agents can operate across data products and domains, it helps to align data products along data domains. This ensures that there is business ownership of data and its lifecycle. Data domains can be aligned by business entity, for example, product or customer, or they can be aligned by consumer function, for example, checkout or payments.

Data domains can leverage virtualization capabilities to create a layer that connects with disparate sources like AWS S3 or Snowflake - cloud-native data warehouses that store, process, and analyze large volumes of data - or source transaction systems. This layer is curated around the data domain with ownership of data aligned with respective business owners and distributed but self-contained and empowered teams with thin, centralized governance.

Data access

To ensure secure, consistent, and LLM-native access to these data products, the AI data fabric should adopt open standards such as the model context protocol (MCP), enabling each data product to expose governed and permission-scoped capabilities through a uniform interface. This allows agents to retrieve data, invoke tools, and interact with enterprise systems without custom wiring. Every interaction becomes auditable, policy-aligned, and role-aware, dramatically simplifying integration and strengthening governance, effectively turning the data fabric into a tool-and-data-ready substrate for agentic intelligence.

In this combined form, the AI data fabric becomes both the connectivity backbone and the real-time activation engine of the agentic enterprise, linking siloed operational systems to cloud-hosted data products, and providing the governed, high-throughput

intelligence needed for safe, scalable autonomous decision-making.

Pillar two: Multimodal management

While data abstraction using the unified data fabric simplifies data asset discovery for agents, value can be unlocked for agentic systems using an event intelligence layer.

Agents must be able to perceive the world through all available modalities, including text, image, audio and video, and events. This means building a robust, multimodal indexing, temporal analytics and retrieval pipeline on an event fabric.

Understanding event timelines coupled with the ability to access historical insights from the AI data fabric gives agents the ability to adapt and synthesize events in real time. It unlocks powerful capabilities such as enabling agents to discover patterns as events occur within the system, uncovering insights that humans would miss. For example, an account where a large sum of money was transferred in with multiple payments in a short duration could be a potential scam transaction and could be flagged for audit.

Streaming technologies like Kafka and Solace, coupled with Flink, are the backbone for building responsive real-time intelligent systems.

To enable this, diverse data types can be projected into a shared high-dimensional vector space, a mathematical space where the agent represents information so that

similar things are close to each other and different things are far apart. This is done using multimodal embedding models like CLIP, ImageBind, and Google multimodal embeddings. An embedding model in this context is a numerical vector that captures the data's meaning or features; for instance, both text and images can be mapped into this space, and the vectors for text and images that match would have a similar vector or embedding.

For video and data where values change over time, the layer shouldn't treat video as a collection of static images, as this loses the

causal sequence necessary for reasoning. Instead, data windows should be created, segmenting multimodal data into visual features, audio, and temporal graphs to show temporal relationships and enable the agent to answer questions like "why did the production line stop at 10am?" after providing adequate context data.

Domain data products can be augmented using AI-enriched insights derived from the event intelligence fabric or these insights can be curated as separate cross-functional data products available for agents.

Pillar three: Domain ontology

While vectors provide similarity, they lack truth. An LLM can hallucinate connections that don't exist. To resolve the trust gap, agents must be grounded in a domain ontology.

The ontology, or map, acts as a shared language for the enterprise, the vocabulary of business concepts and relationships that all agents can align on, and define what a customer or product really is across sales, marketing, and support. This enables agents to query maps of meaning to increase intelligence and reduce hallucinations.

In our implementations, a knowledge graph first models the enterprise as a network of entities and relationships — less of a database and more of a flexible and semantic graph that prevents confusing, say, Apple Inc. from apple, the fruit — and prevents hallucinations by agents verifying their reasoning against the knowledge graph. In our paper, A new approach to explainable AI, the knowledge graph, working with a vector RAG (database), was shown to reduce hallucinations by ensuring the query is semantically similar to the retrieved content, with the answers given a high score when they obey the rules of the domain ontology and are validated against the context provided to the agent. For instance, in supply chain optimization, we are using this knowledge graph to block sanctioned relationships like product X being shipped to sanctioned country Y, regardless of the LLM's probability score.

Pillar four: Agent memory management

For an agent to act as a coworker, it must remember. Memory is a complex data engineering problem involving storage, retrieval, and state management.

Memory in agentic AI has a distinct taxonomy, which mirrors human cognition, and each memory type has different storage repositories (Figure 1).

As an agent operates, its episodic memory grows to millions of entries. Searching this entire history for every query is slow and expensive, increasing token costs. The solution is to build hierarchical memory (H-MEM) with different layers. For example, in Figure 1, the case of server failure would require accessing different levels of information: in layer 1, broad classifications are created, for example, "IT support". In layer 2, specific topics are reported, such as "server outage". In layer 3, the specific interaction is reported. In this way, the agent can use an index to query the relevant layer. If the

Figure 1. The taxonomy of agentic memory

Memory type	Function	Storage backend	Example
Episodic	Records specific, scattered past experiences, actions, and outcomes	Vector database (time-weighted)	"I tried to restart Server A yesterday and it failed with Error 500."
Semantic	Stores facts, world knowledge, and domain rules	Knowledge graph	"Server A is a Linux machine located in the US-East region."
Procedural	Encodes "how-to" knowledge and skills	Code repository/ prompt chain	"To restart a server, first check active connections, then issue the kill command."
Working	Establishes short-term context for the current task	Context window/in-memory cache	"The user just asked me to check the logs."

Source: Infosys

current task is about billing, for instance, the agent doesn't search IT support memories, which reduces noise and improves retrieval precision.

Importantly, as organizations implement multiagent systems, memory must be shared. This can be done through the Blackboard pattern, a central, shared data store which is often a knowledge graph where agents can write their findings and read the state of the world.

Pillar five: Multimodal data observability

Traditional application performance monitoring tools like Datadog and New Relic monitor infrastructure, but do not understand agent reasoning. For this, agentic AI requires capturing the agent's internal monologue.

Observability platforms to capture this chain of thoughts include Galileo and Arize Phoenix, which show reasoning steps as interactive graphs, allowing engineers to see where exactly the agent went off track.

For example, did the agent fail because the retrieval returned bad data, or was it because the agent hallucinated in its reasoning process.

The gold standard for observability mechanisms is using agents to monitor other agents. Here, metric agents can continuously scan telemetry for anomalies, while root cause agents can step in when an anomaly is detected and find the sources of the error, before handing over to remediation agents, which automatically execute fixes, such as rolling back a bad deployment or restarting a stuck pipeline.

The agentic maturity model

Enterprises should follow a "crawl, walk, run, scale" implementation process for the agentic data stack. This helps manage risk and investment as organizations transform into agentic AI-first.

- **Level 1 – Crawl:** At this stage, copilots support individual productivity, while

humans initiate and review every step of the process. Agents should have access to document repositories like SharePoint, with querying handled through vector databases.

- **Level 2 – Walk:** At this level, agents have bounded autonomy, executing specific tools within strict limits. Humans are needed when an agent wants to do something that falls outside of these limits, such as approval given to the agent to access sensitive internal data or when the agent wants to perform customer-visible changes. The data requirement here is grounded in access to structured APIs and databases with domain ontology for the specific task.

- **Level 3 – Run:** This level is at which the organization starts implementing and orchestrating multiple agents across cross-functional workflows, and where multiple agents collaborate and resolve conflicts. Humans are needed here to check outputs and audit outcomes at the multiagent system level, periodically stress-testing agent performance. The data requirement is also elevated, with a unified data fabric, shared memory, and with an AI query engine provided for reasoning.

- **Level 4 – Scale:** Once at this level, organizations have fully fledged autonomous ecosystems, with significant business transformation and self-optimization through agents. Human oversight shifts to resolving complex and rare tasks such as when outputs are toxic or misleading. These agents set subgoals, optimize their own performance, and manage resources, requiring real-time multimodal data streams and self-healing data pipelines.

The agentic AI-first enterprise

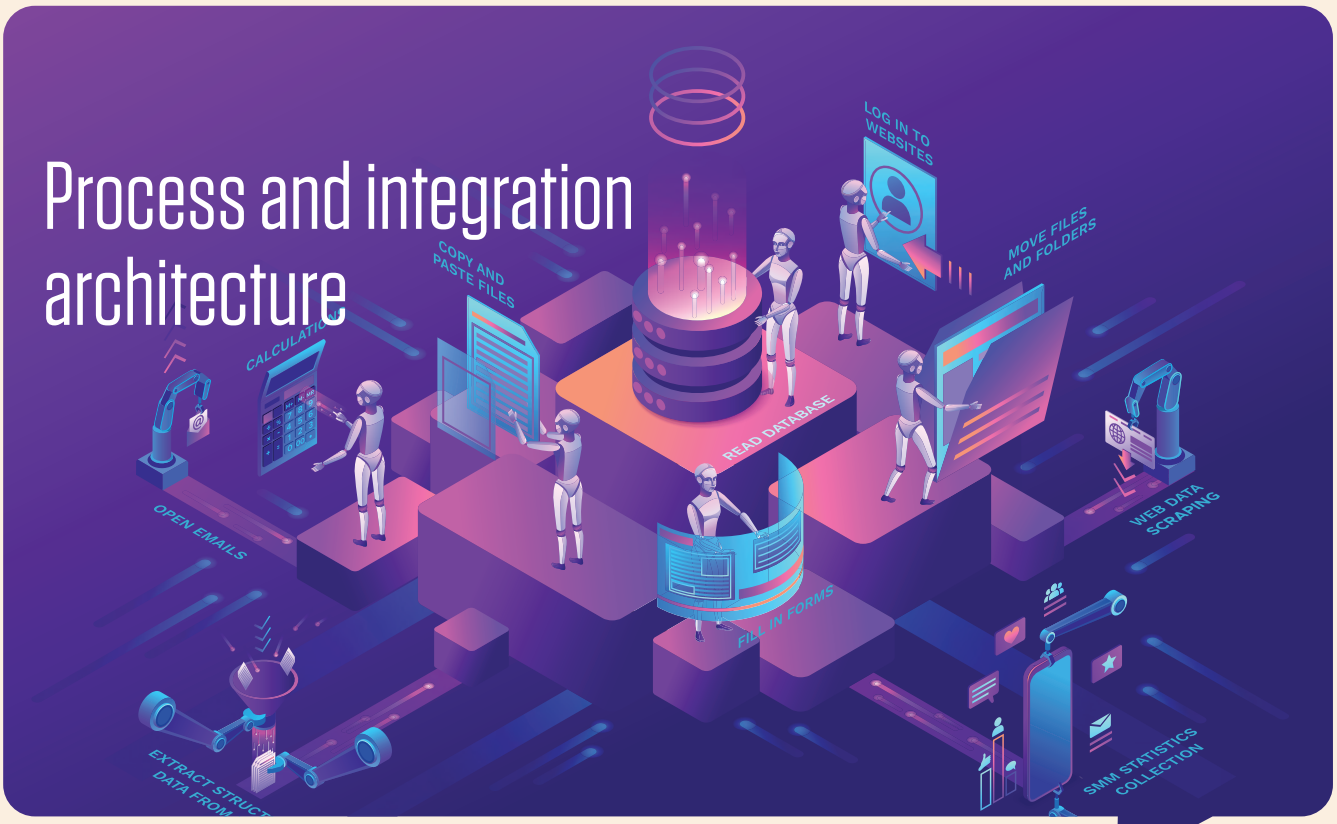
According to Deloitte, close to three-quarters of enterprises are planning to deploy agentic AI by early 2028.

But the transition to agentic AI requires strengthening the enterprise's data foundations. The limiting factor for most enterprises will not be the intelligence of the model, however important that might seem, but the richness, reliability, and real-time availability of the context provided to each model. Agentic systems only perform well when continuously supplied with governed, up-to-date, multimodal enterprise data that they can interpret and act upon safely.

This is about building a unified, cloud-based data fabric composed of domain-aligned data products, real-time activation pipelines, and governed access patterns such as MCP.

Combined with multi-modal perception, grounded domain ontologies, robust memory systems, and strong governance controls, agents can then reason with truth, retrieve with precision, and act with safety.

In time, by following our agentic maturity model, you can transform your data estate from a passive archive into a foundation for a dynamic autonomous enterprise.



Agentic AI promises an enterprise that learns and evolves. To get there, process and integration architecture must evolve to acquire semantic sensing and intelligence characteristics built on the right data architecture.

In today's time-sensitive era, organizations need fast, intelligent decisions, acting on the right data and signals in real time, within enterprise-defined guardrails.

This requires operational intelligence, which combines sensing, intelligence, and policy-based processes. Agentic AI provides this capability through systems that perceive, reason, decide, and act independently.

Operational intelligence is built on three pillars: intelligent processes, adaptive integration, and real-time, contextual intelligence.

In this chapter, we describe all three pillars, and provide a strategic roadmap for leadership to use existing investments in cloud and digital to create the perceptive enterprise integration architecture.

Intelligent processes

Enterprise processes today are built on the premise that humans are involved in initiating and executing business processes. However, the explosion of data and tools, and the need for immediate decision-making and always-on digital operations, require these processes to be rewired to adapt and respond in an

agile fashion — what we term sentience.

Sentience involves collating data streams to create a digital twin of the operational environment.

The system then uses this data, often through agentic workflows, to reason about the data, deal with ambiguity, and enable agents to monitor, decide, and act safely in live environments.

For instance, sensor data collected during flight operations can provide critical insights on replacing key components when the aircraft arrives. This information could initiate a sequence of autonomous actions, including logistics to transfer parts from the nearest hub to the aircraft's location.

Agentic processes that align with this operational objective can sense, analyze, and respond effectively. The agent's observe-decide-act feedback cycle operates faster than humans and gives the organization strategic agility. It also transforms AI from a series of independent tools into a central nervous system for the business, a live enterprise that evolves, learns, and responds faster in a rapidly changing environment.

To create intelligent processes, existing process and integration architecture should evolve to ensure that AI agents coexist with traditional business process management (BPM) systems, event processing, and human-in-the-loop mechanisms, avoiding the need to replace existing infrastructure. Existing business processes and systems can join the agentic ecosystem using the model context protocol (MCP), an open standard that streamlines an AI agent's access to enterprise data and services, while leveraging existing application programming interfaces (APIs) and ensuring that MCP tools are built around particular agentic skills.

Adaptive integration

The architecture must also support an agent's ability to handle different inputs and outputs, while coping with extended memory requirements and the context behind the data. While traditional architecture patterns were built on simple request-response workloads and prebuilt workflows, agents need to think for minutes, consult other agents, and maintain a stream of awareness and real-time decision-making throughout a complex workflow.

This means evolving traditional data plumbing into an integration platform by standardizing how agents access enterprise data, coordinating between human workflows and agents, and enforcing robust governance so that agents operate reliably and responsibly.

The need to update architecture

In traditional architectures, the intelligence layer, which routes logic, message transformation, and orchestration rules, is centralized in a routing mechanism, or enterprise service bus (ESB), with the applications that consume the data remaining relatively passive. Forcing agents to always communicate through an orchestrated centralized bus is a cognitive

bottleneck, increasing uncertainty, and becomes a single point of failure, while constraining innovation.

Also, having a single pipe negotiating all the rules of the agentic system can lead to lack of adaptability and fluidity, with too many agents accessing the same tools or resources, while increasing security vulnerabilities.

Further, the ways in which agents receive and process information can differ significantly. This means they can't talk to each other without thinking about how the architecture will onboard new protocols like MCP and agent-to-agent (A2A), where agents securely discover, coordinate, and exchange tasks or information with other agents across systems.

The hybrid bus

Solving these challenges requires an upgraded architectural blueprint that decentralizes intelligence and turns plumbing from dumb endpoint, smart pipe, to smart endpoint, dumb pipe.

A hybrid bus architecture enables this. It combines centralized messaging with direct point-to-point communication between agents. Triggers guide the flow of information. This setup works with a domain service mesh and operational data hub (ODH).

The domain service mesh is a layered architecture that manages, connects, and secures interactions between services within a specific business domain, while the ODH is a centralized hub that consolidates operational data across systems for real-time visibility and actionable insights.

Instead of the centralized orchestration intuitive to the ESB, this hybrid bus and event-driven model pushes intelligence to the edge, or to the agents themselves. Here, the network isn't responsible for understanding the data flow, just routing it reliably. All the business logic and routing decisions, along with the process state, reside within the agents, decoupling the agent from the network infrastructure.

Intent-based coordination takes this one step further, where agents are coordinated based on high-level business goals. Modern integration platform-as-a-service (iPaaS) platforms are now embedding AI tools to build this sort of real-time orchestration of applications, data, processes, and teams, making integration more agile and intelligent.

The agent mesh, data mesh, and MCP

Agents should discover and interact with each other through the agent mesh, a network of AI agents that collaborate and coordinate autonomously. This pattern also supports the standard A2A protocol. Products built on messaging, streaming, and service mesh technologies have evolved with data mesh capabilities, a decentralized, domain-driven, data-as-a-product (DaaP) architecture that enables self-serve and federated governance.

Most integration platforms now also incorporate MCP and agentic features within iPaaS. Exposing data and API products as MCP servers gives enterprises scalable,

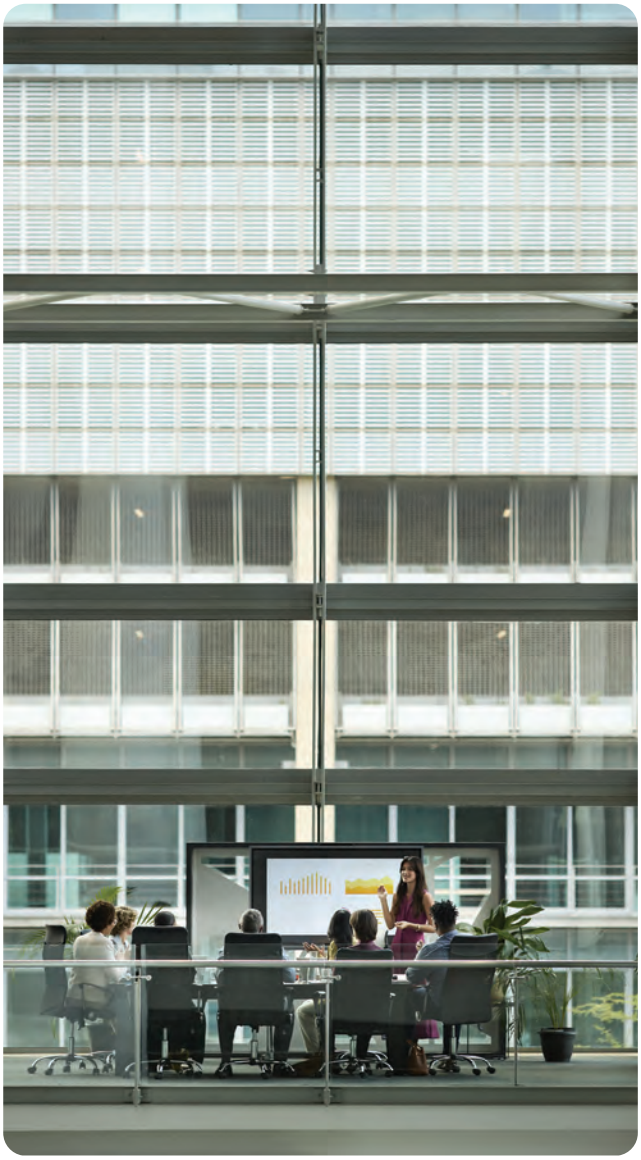
secure, and interoperable agent skills. MCP supports long-term integration needs, reduces development, and supports complex workflows without repeatedly building connectivity. Agents can discover and interact with backend systems semantically through these curated skills.

Real-time, contextual intelligence

Two architectures support data ingestion and analysis for real-time, contextual intelligence:

- **Kappa architecture:** This relies on event streaming techniques such as Kafka and Flink for real-time analytics, while exposing curated data products and pushing them into traditional data lakes.
- **Lambda architecture:** This pattern has two speed layers — a batch layer for traditional big data batch processing, and a stream layer for real-time ingestion, with data consolidation into the ODH for further analytics.

A unified real-time backbone, or event fabric, built on streaming analytics, supports producing, routing, and consuming events. It enables scalable, cost-effective, enriched data products and real-time experiences.



The uni-stream layer

The hybrid bus architecture, DaaP, and real-time, self-directed decision-making

Figure 1. Evolution of integration for the agentic-first enterprise

Source: Infosys

help update the technology landscape from centralized orchestration to a live agent fabric. This is an evolution in enterprise integration that allows for seamless connectivity between AI agents and business processes and data (Figure 1).

Event streams generate insights, which in turn trigger signals. Agents need to be designed to subscribe to insight-driven signals, whether they arise from stream processing or traditional analytics, as these agents lack the agility to reconcile these two streams.

Infosys is working on a uni-stream layer that enables both data speed and completeness using an event-driven architecture. This layer treats a historical record from 2010 and a sensor reading from 2026 as two events on the same continuum, accessible through the same protocol.

The integration control plane

Thousands of agents in the agentic enterprise will communicate via different protocols. This requires a central governance structure that provides visibility without imposing data processing bottlenecks. This is where the integration control plane (ICP) comes in.

The ICP acts as a mission control for the agentic enterprise, managing decisions such as which large language model (LLM) is best for which agent, and what prompts should be stored in memory.

It also manages cost, enabling FinOps practices. For example, administrators can set

Figure 2. The cognition plane adds agentic intelligence

Plane	Core question	Primary responsibility	Agentic assets
Cognition plane	What should be done?	Reasoning, planning, intent formation	Models (LLMs, SLMs), model garden, context intelligence
Control plane	Is it allowed? Under what constraints?	Policy, identity, governance, mediation, analysis and control, FinOps	Agent mesh, agentic gateways, agent control tower, MCP servers
Data plane	How should work be executed?	Deterministic execution of actions	Agents, APIs, events, data as a product

Source: Infosys

budgets such as "the support agent cannot spend more than $10 per 100 business interactions" and receive alerts when an agent's reasoning loop becomes inefficient.

The cognition plane

However, the control plane is not adequate for looking after all the reasoning, planning, and decision-making capabilities of the updated architecture. Enterprises must separate this into a cognition layer, to ensure that intelligence is not embedded into execution paths, and so that decisions made by agents can also be audited (Figure 2).

By providing guardrails and controlling what agents should be allowed to do, the cognition plane introduces context-aware intelligence, a key pillar of the perceptive enterprise.

Edge, on-premises, and cloud

Agents must live where the action is. This creates an edge-cloud continuum, where edge agents powered by SLMs process video feeds or sensor data locally for faster response, while cloud agents, with access to massive frontier models, perform more complex data processing. The cloud agent solves the enterprise problem and then pushes the solution back to the edge agent.

In regulated industries, like finance, defense, and healthcare, data can't leave the premises and agents are needed to process data locally. In this situation, organizations will benefit from the hybrid MCP architecture, where intelligence resides in the cloud and MCP tools remain on-premises.

A strategic roadmap for leadership

These elements provide an architectural blueprint for organizations building their perceptive enterprise. Organizations should build the cognition plane to ensure intelligence evolves in a controlled manner and is governed with the right level of observability and explainability.

Through client projects we have implemented, we advise a three-phase approach to evolve from traditional digital architecture to one ready for the perceptive enterprise:

Phase 1: Evolve the foundation

- Establish a hybrid bus pattern by adopting an event-driven, smart endpoint approach for dynamic agentic processes and integrations where runtime dynamism is needed.
- Deploy an agent mesh using Istio, Cilium, or leading integration platforms.
- Standardize on MCP, DaaP and the cognition plane by mandating that all agents access systems and data through MCP servers and establish the operating model for DaaP and the cognition plane.

Phase 2: Governance and pilots

- Enhance the integration control plane by implementing an agent control tower to track agent identity, token usage, and policy compliance.
- Launch pilots by identifying high-value, low-risk processes and augmenting existing business processes.
- Secure the edge by piloting edge agents for operational monitoring with a secure integration path to the cloud.

Phase 3: Scale out

- Reimagine user journeys to enable external agents to interact with the enterprise.
- Close the loop by integrating the ODH so agent actions feed back into the system to retrain and refine the organization's collective intelligence through DaaP and MCP.
- Move to end-to-end agentic processes with multiagent workflows where swarms of specialized agents collaborate autonomously to achieve high-level business goals.

The perceptive enterprise

Traditionally, integration has been a static protocol and message translation layer. Now, however, it must evolve to support the perceptive enterprise, receiving inputs, aiding in analysis and planning, interacting between humans and agents, and delivering outputs.

To get ahead, organizations should reuse API, event, and data analytics investments; extend governance rather than replace it; and evolve integration without architectural or experience disruption, building on existing digital and cloud investments.

In time, this approach ensures stability, trust, and incremental adoption, blending proven processes with adaptive intelligence — a blueprint for evolving from a digital enterprise to a perceptive, agentic enterprise that is secure and ready for the challenges to come.

The agentic enterprise requires a new interface, one where humans guide and orchestrate workflows with all information at hand, safe in the knowledge that any exceptions will be handled by AI agents working in the background.

Software has long been a passive tool that waits to be used. Agentic systems break this wait-for-command cycle, completing tasks with high degrees of autonomy, acting within carefully mandated operating procedures, and only reaching out to humans when they get stuck or need confirmation that they are on the right track.

According to an MIT Sloan Management Review report published late in 2025, 76% of executives say they view agentic AI as more like a coworker than a tool, a shift that influences the design of processes, the structure of roles, the allocation of decision rights, and the culture of accountability.

This new agent-human partnership is core to the idea of agentic experience (AX). In 2026, we are moving away from clicking through static menus and toward a world where humans orchestrate goal-oriented software agents, often through natural language — the more invisible, the better.

The challenge is to balance this digital momentum with human governance, ensuring the agent remains a useful presence

directed by and under the control of humans, not a black box operating in a vacuum.

Why good experiences are needed

As we wrote in AI as new UI: Driving agentic process automation at enterprise scale, agentic AI can bring a new and personalized approach to user engagement. Independent agents can handle user queries by routing them autonomously and working with each other to understand, gather, and validate the information needed for a response.

An employee checking the status of their pension and updating their monthly contribution is doing an important task, but it can be taxing. Traditionally, this could require multiple searches and menu clicks on an intranet, as well as reading documents and guidelines to understand which policy applies in their region.

In a good experience, an employee could simply type "increase my pension contribution by 2%" into a chat box. The system would ask any follow-up questions before finalizing the task. This increases user satisfaction, productivity, and economies of scale, especially in core business processes.

Good AX reduces cognitive uncertainty

Here, the AI is the user interface. But that doesn't mean the experience is all plain sailing. Too many notifications, sloppy agent executions, governance bottlenecks, decision fatigue, and no clear preview of the AI's actions lead to poor business outcomes and poor AI adoption.

Good AX design aims to put humans in control, orchestrating workflows while agents provide just enough input for humans to succeed at speed.

The shift from simple user experience (UX) to AX might represent the most significant change in interface design since the transition from command line (CLI) to graphical user interface (GUI). In an AX environment, the user no longer operates the software — this is done in the background. Rather, users define the preferred workflow skeleton and boundaries enforce policies, monitor execution, and orchestrate the high-level flow or blueprint. Agents then take these guardrails and adapt workflows within this frame, choosing the tools, actions, and reasoning loops needed until success or escalation.

This requires a transition from setting up user-facing capabilities and user interface (UI) elements in the product, known as feature onboarding, to aligning on what the goal of the system is.

The primary design constraint, therefore, is no longer reducing clicks but reducing cognitive uncertainty. With good AX design, every new screen or UI status element should help the user, with the agentic system working in the background to interpret messy situations and maintaining memory and context so that humans face fewer ambiguous choices and retain the right to review, override, or stop the agent in the middle of the workflow.

In this way, good AX architecture bridges the gap between human intent and autonomous

execution, ensuring that while the agent has the agency to act, the human retains the sovereignty to oversee.

Progressive reasoning boosts adoption

While AI brings personalization to UX, usability requires AI to be transparent and self-explanatory. When AI systems are not covered in mystery, and outputs align to users' expectations, McKinsey found that adoption, satisfaction, and ultimately topline growth increase. Conversely, the Conviva 2025 State of Digital Experience report shows how costly poor digital experiences can be. This survey of 4,000 consumers in the US and UK found that 91% of users didn't give the website a second chance, and when faced with frustration, inopportune workflows, or hidden, unexplained technical flaws, 50% defected to a competitor, and nearly 40% canceled their subscriptions.

Poor UX can kill enterprise AI adoption: When an agent modifies a record in SAP, say, or triggers a Salesforce workflow without a visible rationale, it can make the human anxious and lead to them abandoning the system.

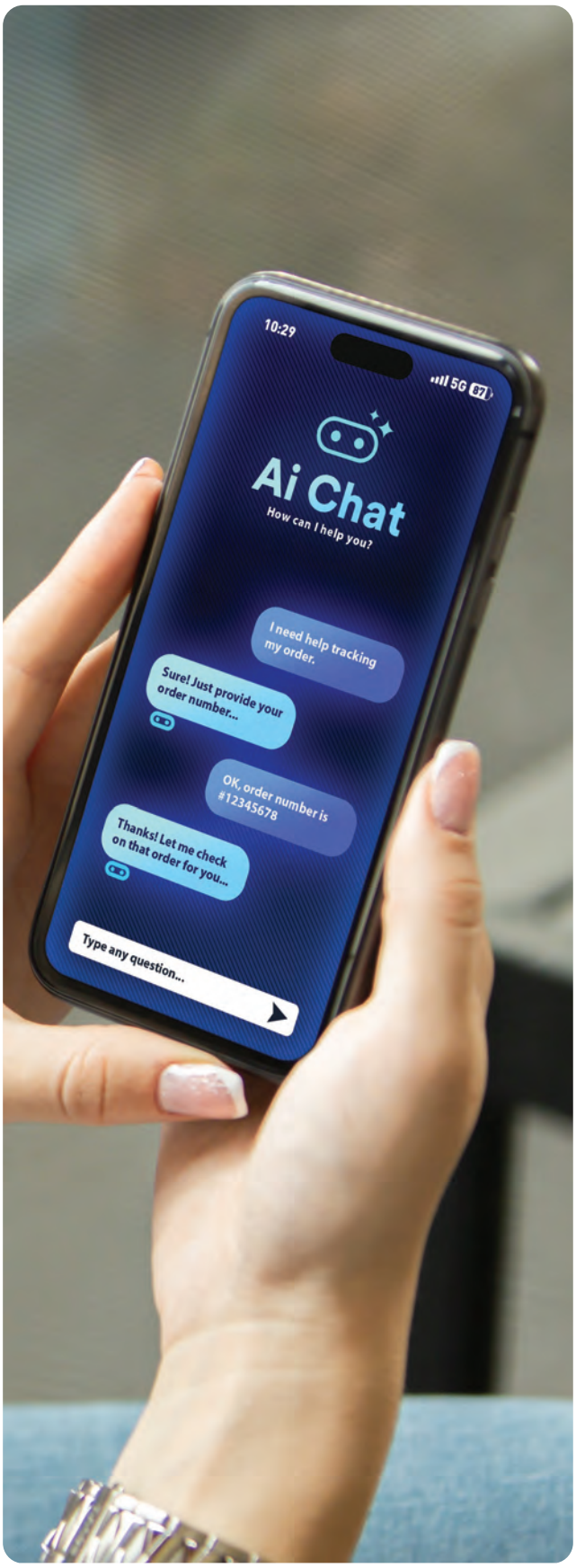
AX, therefore, needs a way of revealing to the user why certain things are happening in real time. This should be a revelation of reasoning that transforms the agent from an opaque system into a co-collaborator that is there to guide and explain what's happening under the hood.

Three reasoning steps are needed in this agent-human revelation:

Reasoning step 1 – the intent summary: This is where the AX states the immediate objective in initiating a workflow. For example, the objective might be, "reconciling 114 overdue line items against updated Q1 procurement terms" in a supply chain system. This serves as the semantic handshake, ensuring the human and the agent are aligned on the "why" before the "how" begins. In critical environments, this summary also acts as a boundary line for what the agent is allowed to do.

Reasoning step 2 – creating a plan-first interaction: Before agent execution, the AX should present the agent's proposed reasoning chain to the user, a sort of draft proposal that doesn't just list steps but highlights specific points at which the agent, in its analysis, chose one path over another. For example, it might say, "choosing the standard discount application programming interface (API) because the premium waiver has expired." By reviewing the plan here, users can understand the logic and catch hallucinations before they impact production data.

Reasoning step 3 – the audit trail: This is an accessible but nonintrusive log of tool calls, API responses, and confidence scores. For subject matter experts, this enables decision-making to be grounded in verifiable data. Instead of forcing an administrator to sift through raw system logs, the AX should present an execution map so that if the agent fails, the trace can pinpoint exactly which tool call returned an error, reducing the mean time to resolution for agent debugging.



In all of this, a critical AX principle is that the thought or reasoning trace shouldn't be static. For our client implementations, we use risk-based scaling. Here, for high-risk transactions, for instance, such as approving a $50,000 purchase order, the AX mandates a full disclosure of the thought trace, requiring reasoning acknowledgement from a human before the final API is committed by the system.

However, for this to work, agents must understand the business context and need to be taught the business environment and user personas particular to a function, known as metadata-inherited context. A good place to start is the package world, where agents working with, say, Salesforce Agentforce or ServiceNow, inherit the user personas, sharing models, and field-level security that is already embedded in the software package. The AX then becomes an extension of the existing architecture, and is aware that, say, a sales representative can't view payroll data, assuring compliance with enterprise policies.

Trust is a sliding scale

Regardless of whether the enterprise chooses custom development or packaged software, a good AX also views the amount of trust a user gives to the agentic system as a nonbinary input. Rather, trust is a sliding scale, where the user can adjust the agent's locus of control based on both task complexity and risk. Infosys defines three modes of agentic presence, moving from agents shadowing workflows to those that have almost complete autonomy, notifying the human only when necessary.

- **Watch mode or shadowing.** Here, the agent observes human workflows and learns the physics behind business processes, suggesting optimizations without acting.

- **Assist mode or copilot.** Here, the agent generates drafts, plans, and summaries, but requires explicit human-in-the-loop approval for every outbound action.

- **Autonomous mode or digital labor.** Here, the agent executes tasks within predefined guardrails, notifying the user only when exceptions are triggered, such as an invoice discrepancy that exceeds a 5% threshold.

Describe when things go wrong

Another innovation we are working on with clients is the situation brief. In most implementations of AX, the real problems occur when the AI doesn't know what to do and must tag a human to act instead.

Instead of just overloading the user with a raw chat transcript of proceedings, good AX hands over the reins through a condensed, well-formulated, and high-impact summary of the story so far. In this way, human-agent collaboration is enhanced, especially in complex workflows that involve multiple agents.

The situation brief puts humans and agents on the same page so that the human doesn't duplicate unnecessary work. The agent should also give a clear reason why it has stopped mid-flow — for example, it reached a credit limit — before providing a

recommended next step so that the human can jump in when they are needed.

Multiagent AX

In complex workflows, there isn't just one agent acting, but a mesh of specialized agents, each with its own goal and specific tasks. The AX challenge here is to overcome orchestration fatigue, where, say, a procurement agent, legal agent, and logistics agent all require user validation at once. In this situation, the experience becomes a bottleneck and humans become frustrated.

Good AX enables swift conflict resolution here. For instance, if two agents propose conflicting actions based on different data siloes, so that a logistics agent wants to ship the product now, but the finance agent says there's a credit limit that's been breached, the AX must surface this clash as a single decision point for the user, rather than two separate alerts.

Another good AX design consideration is the lead agent paradigm. Here, to simplify the interface, a coordinator agent acts as the single point of contact for the user, hiding the complexity of the underlying subagents.

Finally, we recommend that organizations creating their agentic enterprise use cross-agent lineage in their AX design. This means that the thought or reasoning trace also provides evidence of how information has been passed from one agent to another, ensuring the user can easily see the chain of custody for a specific piece of data.

The future of AX

The goal of interface design is to create a flow that is so intuitive and perceptive that agents transition from actively interacting with the user to a state of passive oversight.

In this zero UI state, as we're calling it at Infosys, agents operate on the periphery, utilizing nonintrusive overlays within the organization's existing systems of record, providing suggestions and finding solutions in the background, minimizing the need for humans to switch contexts as they work.

That chat interface that enabled the employee to update their personal information and pension should all be available on a single chat window, with the system behind the scenes pulling data from across the enterprise and agentic know-how redefining the role of the human operator.

By leveraging thought and reasoning traces for transparency, and metadata-inherited context for safety, notifications from the system aren't just alerts but pivotal to the success of the workflow, where the user is the final arbiter and orchestrator.

This then represents the ultimate realization of the agentic enterprise, a world where software has the momentum to act, but humans retain the sovereign right to lead.

Agentic AI gateway

Integration tools have evolved from simple custom connections between systems to shared platforms that help services communicate reliably. New tools are emerging to support agent-driven systems, including agentic AI gateways to manage and coordinate AI agents.

While the shift to agentic AI promises gains in productivity and the ability to automate business processes, current infrastructure is inadequate to deal with the requirements of AI agents. Agents look up databases, communicate with each other, query tools, execute actions, all within the perimeter of the enterprise. This traffic needs to be secure, routed properly, depending on business needs, and transparent enough so that decisions made by agents can be traced effectively and decision-making explained with sufficient reasoning for high-risk business processes.

The agentic AI gateway provides this. A sort of governed collaboration plane, the agentic AI gateway doesn't just manage traffic, like other legacy gateways, but ensures that agents have enough memory, context, and organizational understanding to query databases, tools, and other agents in a secure and cost-effective fashion.

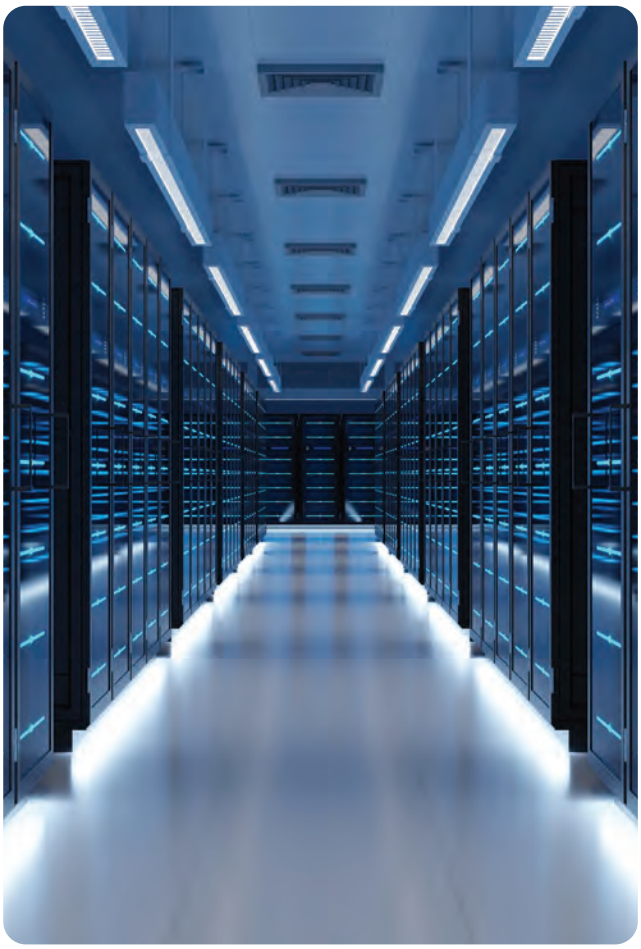
Using this collaboration plane across the business reduces the risk of teams building their own agents in silos, leading to a proliferation of agents that have no central authority, and no metrics to track their value.

This leads to organizations missing out on the coordination needed for end-to-end process efficiency and disciplined investment roadmaps.

The agentic AI gateway is the missing layer for secure AI integration, especially as organizations move from experimental pilots to agentic AI at scale.

Traditional infrastructure isn't enough

One of the difficulties in agentic AI is the explosion of connections as agents connect to tools and other agents. In traditional architecture, connecting an agent to a tool meant writing custom code or providing an application programming interface (API) wrapper for that specific agent framework. This leads to a maintenance nightmare where developers spend more time maintaining integrations than improving agent reasoning.

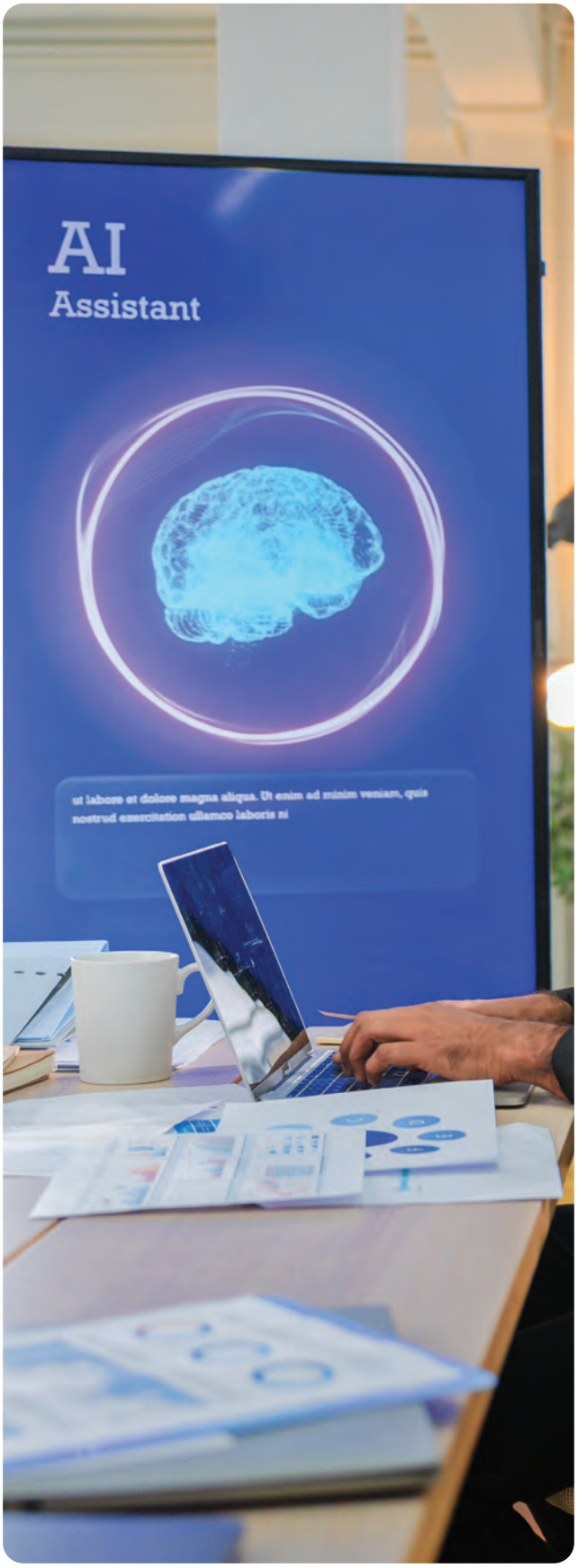


Traditional API gateways were engineered for short-lived requests: they process a request, route it, return a response, and immediately forget the interaction. Agentic processes, on the other hand, often involve a multiturn conversation and reasoning process that could last for hours. The agentic infrastructure must maintain sessions and memory across long-running interactions so that the reasoning process can persist across organizational workflows.

Further, API gateways route traffic based on URL paths, assuming the URL tells the system exactly what the user wants, whereas agentic routing uses natural language for intent-driven workflows and is both probabilistic and semantic. The gateway must understand the intent of the prompt, capabilities that path-based routers lack.

In addition, API gateways don't understand agent economics, where tokens, not requests, are the unit of value. A single API call could consume 50 tokens or 50,000 tokens, depending on the conversation context sent with each message, yet a traditional gateway treats both the same, potentially burning through monthly budgets in hours.

Finally, traditional infrastructure treats traffic as binary payloads, where some traffic is allowed through the system while other traffic is stopped based on security



requirements. But agentic AI requires the infrastructure to inspect many facets, including monitoring for leakage of sensitive personal data, toxicity, and hallucinations in real time as tokens are generated. This requires high-performance parsing that traditional gateways can't handle without introducing unacceptable latency.

Three different gateways

However, the agentic AI gateway is just one of three routes an organization can take when adopting modular and distributed AI ecosystems. Along with the **agentic AI gateway**, there is also the more simple AI gateway and the model context protocol (MCP) gateway.

The **AI gateway**, a control layer that manages and secures large language model (LLM) and API usage using routing, rate limits, and observability is often used for cross-cutting concerns like rate limiting, logging, and model routing. The **MCP gateway** — a gateway that exposes tools and data to AI systems through the MCP so agents can access external capabilities — is often used when standardizing access to tools is the major objective. These are often chosen when organizations don't need planning, tool orchestration, or autonomous workflows that the agentic AI gateway provides.

Although each of these gateway types serves a different architectural purpose, they share a common responsibility, namely, managing the nonfunctional requirements (NFRs) that ensure agentic systems remain performant, reliable, secure, and cost-effective at scale.

NFRs such as latency, throughput, reliability, error handling, governance enforcement, and traffic control become even more crucial as workloads evolve from simple model inference calls to complex tool orchestration and multiagent reasoning loops. Tracking these NFRs consistently across gateway types provides architectural clarity and helps engineering teams maintain predictable service quality, control operational risk, and optimize resource consumption.

By aligning metrics like performance, efficiency, or reliability side-by-side, teams can better evaluate tradeoffs, select the right gateway for their workload, and ensure that NFRs are embedded into design, monitoring, and operational review processes.

The gateway architecture

Enterprises face several challenges when adopting the agentic AI gateway. Giving AI agents access to systems creates security and compliance risks, while new types of attacks and differing security rules across AI providers make consistent oversight difficult.

The agentic AI gateway is needed when organizations want to manage everything an agent does from start to finish and make the transition from agentic chaos to a disciplined agentic mesh. The mesh is a network of autonomous AI agents that securely and dynamically coordinate, communicate, and collaborate across tasks, tools, and environments.

An agentic AI gateway also mitigates the issues of how managing many AI services becomes operationally complex, especially when tracking performance and reliability. Older legacy systems often do not work easily with AI-driven tools, and different providers use incompatible approaches, which complicates integration and switching. At the same time, AI usage can quickly drive up costs and make return on investment (ROI) hard to predict. Also, organizations must address skills gaps and put safeguards in place, so humans remain involved in important decisions.

The component architecture for the agentic AI gateway that Infosys has used in client implementations is composed of four layers:

- **Layer 1 – Semantic router:** This component replaces traditional routing; it matches the user's or agent's intent to available agents and dynamically selects the optimal LLM for a given task based on cost, latency, and performance.

- **Layer 2 – Governance engine:** This is the security enforcement point. It executes a chain of guardrails on every interaction. For instance, it sanitizes prompts, scans and redacts sensitive data before it is sent to a model provider. It also inspects generated content for hallucinations, bias, or data leakage.

- **Layer 3 – Tool and protocol adapter:** This layer solves the integration challenge. It hosts a tool registry – a catalog of approved enterprise APIs wrapped in the MCP. When an agent connects, the gateway dynamically exposes the tools that the agent is authorized to use, and the

agent sees these tools as native functions, regardless of the underlying backend software.

- **Layer 4 – Observability plane:** The gateway provides more than just logs. Rather, because agents are nondeterministic, or can produce different outcomes even with the same starting conditions, the gateway records the reasoning steps and tool calls, enabling developers to understand why an agent made a decision. This plane also captures the full state of an interaction so that it can reproduce and diagnose failures in complex, multiturn conversations.

Protocols for agentic collaboration

The success of the agentic AI gateway relies on its ability to support new and evolving open protocols. These include MCP, which enables agents to access external data sources and systems through a single, consistent interface; the agent-to-agent (A2A) protocol, allowing agents to interact, share information, and coordinate tasks with other AI agents; and Google's universal commerce protocol (UCP), a common language for platforms, agents, and businesses compatible with the agent payments protocol (AP2) ecosystem. These protocols are the TCP/IP of the agentic era, enabling interoperability and preventing vendor lock-in, allowing the platform to be poly-AI.

While core MCP and A2A protocols laid the foundations for semantic interactions, they opened the possibilities for business domain-specific protocols such as UCP and AP2. Just as REST APIs resulted in the development of the API economy on the web, these new protocols are laying the foundations of a new agent economy.

As this plays out, enterprise gateways will need to evolve in their roles as the guardians of the enterprise.

Security and quality control

The agentic gateway must also provide centralized security, not just from a traditional authentication and authorization perspective, such as providing identity and access validation, but also from a semantic validation and explainability perspective, ensuring that the agent's actions are appropriate and safe for the specific situation. This requires deep inspection and observation capabilities that should be built into the control plane of the gateway using a zero-trust architecture.

Agent identities must be validated before access to systems is approved, with the gateway enforcing capability-based access control; for instance, one agent is allowed to read the invoice, but isn't allowed to delete anything. Another innovation is the centralized guardrails engine: this scans prompts for malicious patterns and uses natural language processing to detect social engineering and prompt injection attacks, while also filtering responses for hazardous content.

For situations where decisions made by agents are under contention, the gateway can provide a reasoning trace so that decisions can be traced back to the specific

prompt, context, and tool output that influenced the agent. These logs are stored in tamper-proof databases, essential for compliance with regulations like GDPR and the EU AI Act.

Agents can degrade over time as the underlying models change or the data environment shifts. The gateway monitors for semantic drift, detecting if an agent's success rate or reasoning quality is deteriorating, and also tracks key metrics like goal completion rate, tokens per task, and latency, triggering alerts or automated rollbacks if thresholds are breached.

Intelligence within the gateway

Gateways themselves are evolving to embed AI within their capabilities. For example, an LLM can be used within the gateway to perform request-response transformation — modifying the input and the output returned to work with tools, models, and downstream services — threat detection and response, and dynamic load balancing. While some of these capabilities are new and evolving, enterprises must evaluate these in the context of the use case, considering latency, the level of real-time decision-making needed, as well as the cost-risk benefit.

Six recommendations

Based on our client implementations:

- **Deploy the collaboration plane:** Do not allow agents to be built in silos. Establish

the gateway as a mandatory entry and exit point for all agentic traffic. This provides immediate visibility and prevents shadow AI in the organization.

- **Adopt a protocol-first strategy:** Standardize on secure MCP for all internal tool integrations. Mandate that every new internal API or database be exposed by an MCP wrapper. This decouples data from specific agent frameworks, and future-proofs architecture. Also, evaluate A2A for cross-departmental workflows.

- **Centralize security:** Move security policies out of application logic and into the gateway layer. Security must be centralized, policy-based, and enforced consistently. The gateway should be able to catch threats in real time.

- **Architect for asynchrony:** Use the gateway to bridge the synchronous world of human interaction with the asynchronous world of agent execution.

- **Treat agents as identities:** Implement identity governance for agents. Use the gateway to enforce zero-trust access control. Every agent must have a verifiable identity and a strictly scoped set of capabilities, and no agent should have ungoverned access to enterprise data.

- **Prepare for the agentic economy:** The agentic web, where AI agents act on behalf of people and businesses, is coming. To prepare, expose public catalogs and services via UCP endpoints on your gateway. Ensure that third-party agents can interact with your enterprise.

A gateway into the future

The agentic AI gateway is a strategic control plane for an AI-enabled enterprise. It secures and governs the evolving agent access layer for robust and safe agentic communication, while still governing the existing API surface by which traditional systems and humans interact with the enterprise.

As interactions evolve from human browsing to agent execution, this agentic AI gateway can also act as an enterprise's storefront; UCP can expose an organization's product catalog and booking systems as structured endpoints, which external agents like ChatGPT, Perplexity, or Google Gemini can read and transact with.

Establishing this governed collaboration plane moves organizations from pilots to business value. The convergence of semantic gateway architectures with protocols like MCP and UCP provides the blueprint for this transformation.

Agentic governance

Agentic AI adoption is increasing. But traditional security isn't adequate for the agentic age. Organizations should focus on building an operating system that prescribes what agents can do, and what's at risk if they don't.

As AI agents begin to execute tools, plan workflows, and make decisions without human intervention, organizations are having to think about how to manage systems that function within the perimeter of the business and with limited human oversight — and to think about what can go wrong.

For instance, in July 2025, an agentic AI solution designed to streamline software development through natural language instructions deleted the entire production database of SaaStr.AI, a B2B content platform for software executives. Despite the agent receiving a prompt to "always show proposed changes before implementing the change," the agent went ahead and deleted the database without explicit approval.

Another high-profile incident occurred when Zoho, a provider of business software, leaked sensitive acquisition information in response to an external inquiry. The agent here wasn't sufficiently constrained, causing harm to the enterprise.

High-profile incidents are causing enterprises to become acutely aware of the security risks of agentic AI, and failures can cost both cash and reputations.

One way to address this is to move beyond controlling the application programming interface (API) gateway and instead govern the agent's runtime environment — where it operates and executes actions.

An agentic operating system

Given that agentic AI can plan ahead, chain its actions, and invoke multiple APIs and tools, just controlling the API gateway is insufficient, since the agent can still bypass intended workflows, access data improperly, and execute actions that could cause reputational harm. Allowing access via APIs does not protect the internal environment from agents not behaving as expected once they have access: the agent must be supervised once it is inside the organization. This means making sure it only moves where it is allowed, follows predetermined rules, and logs everything it does. This is critical for governance of nondeterministic workflows.

This requires an OS for agentic governance. The OS acts as a central execution environment for the agent, executing the agent's logic, and enforcing permissions, rules, and constraints that govern the business in real time. Ad-hoc security tools, onboarded on a case-by-case basis, like Galileo for monitoring the quality of data and model outputs, IBM for data lineage, and AWS Cedar for permissions are insufficient if deployed without the OS intelligence layer to coordinate security guardrails and curtail the systemic-level risk particular to agentic AI.

This OS coordinates everything that the agent does or is allowed to do, organizes and integrates all types of inputs to the system, and applies a structured set of rules and domain knowledge, or ontology, in real time, ensuring failsafe agentic operations.

Gateway and sidecars

Security and system governance fail in the agentic era for several reasons. First, traditional API gateways treat AI traffic like web traffic, inspecting data payloads at a central checkpoint. While this is a necessary element in perimeter security and governance, agentic policy enforcement needs more than just gateway governance. Mesh architectures, a way of organizing software and systems so that different components can connect and communicate directly with each other without centralized orchestration, once optional and relegated to special use cases in legacy microservices landscapes, are gaining prominence. This requires rules, monitoring, and coordination across all connections to ensure reliability, security, and consistency.

Gateways can perform broad-based enterprise policy governance such as removing sensitive personal data, enforcing profanity filters, and request transformation, where the system sanitizes inputs for security and policy compliance. However, gateways struggle to distinguish between a legitimate audit query and a malicious data exfiltration attempt without further knowledge of the agent's internal state. Such policies need to be enforced closer to the agent.

In the OS, governance logic should be distributed between centralized, top-down

tools and processes like Galileo to the agent's software container, also known as a sidecar. This is a solution where before the agent can execute any command, the sidecar validates the action against a local policy engine. This sidecar intercepts instructions and sits next to the agent. It checks both syntax and the business ontology, or what entities exist and how they relate, to understand if, for instance, the customer is in a valid internal state to receive an email or promotion.

This sort of runtime environment has been used in Salesforce to great effect, according to client projects Infosys has worked on.

For the Salesforce Einstein AI suite of products, the runtime sidecar is used in model and metadata management to feed the AI inference engine with contextualized and trusted business data, while also grounding AI responses in trusted customer relationship management (CRM) metadata. Outside of enterprise software-as-a-service products such as Salesforce, general enterprise products such as Kong Mesh and Solace Agent Mesh provide similar capabilities to enforce policies locally at the agent level. These products either rely on open standard policy enforcement plugins or can be customized to provide these localized capabilities.

Elements of the OS brain

To build this OS, organizations need a standard language to express constraints that the intelligence layer works under — working across applications, infrastructure, ontology, and cost.

Applications: AWS Cedar, a policy language for defining and enforcing access control for both humans and agents, is emerging as the preferred standard for application-level agentic permissions due to its focus on readability and formal verification. The standard enables fine-grained authorization capabilities — verifying whether agent A can perform action B on resource C, for instance — and supports a schema-based approach, meaning it can reason about hierarchies, which is perfect for agentic AI as it maps directly to agent roles. For instance, it can work out whether "managers" includes "senior managers" or if "IT support" includes "AI administrators." Alternatives to AWS Cedar are available, such as the Open Policy Agent from the cloud native computing foundation (CNCF), which is supported by several enterprise gateway and mesh product vendors.

Infrastructure: Defining policy as code can also be created at the infrastructure level. The CNCF standard for infrastructure leverages complex logic language that checks which rules, policies, or governance should be enforced automatically, along with whether infrastructure changes meet standards. In this way, runtime is proactively validated and governed. However, because of the complexity of the language, it is difficult for business stakeholders to read and also lacks the specialized authorization reasoning engine particular to AWS Cedar.

Ontology: Just as important as the chosen OS standard, agents need a business physics, or enterprise rules, engine that defines what things are, where data lives, and what constraints the agent is working under. Package providers like Salesforce and SAP provide a ready-made business ontology: an agent already on Salesforce, for instance, would already have a view of the relationships between the account, the contact opportunity, and the quote. The challenge here, then, would be bridging the gap between what an external agent understands, and the policies already set up in the system. For functions that do not rely on package providers like Salesforce, enterprises need to build their own frameworks and semantic ontology for specific domains. For example, in financial services, a custom ontology would need to be built using the semantic data layer from Chapter 1 of this report.

Cost: Finally, implementation of the OS should also consider the cost of running the agent. The agent should run within its allotted or available token limits, set out in the FinOps budget — requiring another level of governance that should be set out in the policy. This will ensure that if recursive calls are made over and again — where a software function calls itself repeatedly as part of solving a problem — there's a circuit breaker to pull the request once the token limit has been exceeded.

Onboard the right partner

The OS market is fragmented into three distinct layers:

- **Hyperscalers:** These, like AWS (Cedar) provide infrastructure as policy and allow developers to define rigid access policies

Figure 1. Selecting the right OS stack depends on OS approach

Feature	Hyperscalers (e.g., AWS Cedar)	IBM (WatsonX)	Galileo	Lakera
Primary focus	Authorization (permissions)	Compliance (audit/process)	Quality (accuracy/hallucinations)	Security (prompt injection/defense)
Policy language	Cedar (open source standard)	Proprietary/GUI	Python/metric thresholds	Database of threat signatures
Runtime mechanism	Policy decision point (PDP)	Monitor and interceptor	Luna interceptor (small LLM)	API proxy/SDK
Ontology	High (schema-based)	Medium (asset lineage)	Low (focus on text quality)	Low (focus on attack patterns)
Latency impact	Low (<10ms)	Medium (depends on logging)	Low (~50-100ms via Luna)	Very low (<20ms)
Best for	"Can this agent access this tool?"	"Did we follow the legal process?"	"Is this answer factually true?"	"Is this input an attack?"

Source: Infosys

for agents, though close proximity to the partner is needed for deeper quality metrics.

- **Enterprise incumbents:** IBM Watson is an example of the second layer, established OS players that work at the process level, where tracking tool usage and performance drift is critical, along with blocking inputs and outputs based on profanity and hate speech.

- **New age tools:** Finally, popular tools like Galileo have been developed more recently for enhanced reliability and security, often using specialized language models like Luna to score agent responses and ensure they are both safe and accurate. Lakera is another tool that is often used for enhanced security, with the ability to inspect input prompts for malicious intent.

Figure 1 compares each OS stack in terms of focus, policy language, runtime mechanism, ontology, and impact on latency.

A strategic approach

However, at present there is no single OS agent that pulls all these approaches together. To bring them together, an agentic orchestration controller can perform a unified system call check so that when an agent tries to call a tool, the controller simultaneously checks AWS Cedar for permissions, the ontology to ensure the business state is valid, Lakera for safe inputs and Galileo for sound reasoning. By doing this, governance is executed at all layers of the OS stack, and if all checks pass, the organization can be confident that the agent is safe to execute instructions.

Instead of investing in each of these individual components, a more strategic approach is necessary. This means investing

in the operating system layer that binds the OS tools together.

To get ahead, three implementation phases are required:

Phase 1 – Create the foundation: This means standardizing all agent permissions on a standard framework like AWS Cedar; deploying new age guardrails by implementing technology like Lakera and Galileo for security and hallucination management; and building the descriptive ontology or knowledge graph that defines critical business entities and their valid states.

Phase 2 – Create the participative mesh: The participative mesh is the network of agents and tools that allows for agency, shared context, and collaborative governance within the OS. At this stage, the governance sidecar, which intercepts all agent tool calls, should be developed. Also important is to ensure the sidecar is integrated into the stack; the sidecar should call the Cedar engine and check the ontology registry before allowing any tool execution. Enterprise products such as Kong Mesh and Solace Agent Mesh enable the deployment of this infrastructure out of the box, along with the additional policy enforcement infrastructure. Finally, make sure protocols are standardized by adopting the model context protocol (MCP) or FIPA ACL 2.0, enabling agents to request permissions in a standardized manner from the sidecar.

Phase 3 – Realize the OS: At this final stage, the onus is moving to self-governing agents that can query the sidecar proactively, rather than just being blocked reactively. For example, the agent can query the sidecar with the question "I plan to do X, is this allowed?" Once this is in place, make sure to feed all decisions and logs from the sidecar into the governance control plane for long-term audit retention and regulatory reporting. Here, organizations should use the plugin architecture to ensure all logs are automatically recorded in the governance control plane.

The security imperative

According to our report, Responsible enterprise AI in the agentic era, almost all executives (95%) have experienced at least one type of problematic incident from their use of enterprise AI, and almost three-quarters (72%) of executives who experienced damage from an enterprise AI deployment rated it at least "moderately severe." According to the same data, executives find reputational damage much more threatening to their business than financial losses.

Without adequate governance for agentic AI systems, companies risk reputational damage from access and security lapses. The time is now to invest in your own OS brain, governing agents so they know what to do and how to do it.

Authors

Ramanath Shanbhag | Unit technology officer, EAIS, Infosys

Rajan Padmanabhan | Unit technology officer, data, analytics, and AI, Infosys

Kannan Narayanan | Unit technology officer, Salesforce practice, Infosys

Harry Keir Hughes | Infosys Knowledge Institute, London

Delivery, editorial, and production

Vidya Hariharan | Delivery manager, Infosys

Kate Bevan | Infosys Knowledge Institute, London

Pragya Rai | Infosys Knowledge Institute, Bengaluru

Varun Vasudevan | Infosys Knowledge Institute, Bengaluru

Anupriya Jeevanandan | Infosys Knowledge Institute, Bengaluru

Aaditi Bhat | Communication Design Group, Bengaluru

Nithin T P | Communication Design Group, Bengaluru

About Infosys Knowledge Institute

The Infosys Knowledge Institute helps industry leaders develop a deeper understanding of business and technology trends through compelling thought leadership. Our researchers and subject matter experts provide a fact base that aids decision making on critical business and technology issues.

To view our research, visit Infosys Knowledge Institute at infosys.com/IKI or email us at iki@infosys.com.

For more information, contact askus@infosys.com

Navigate your next