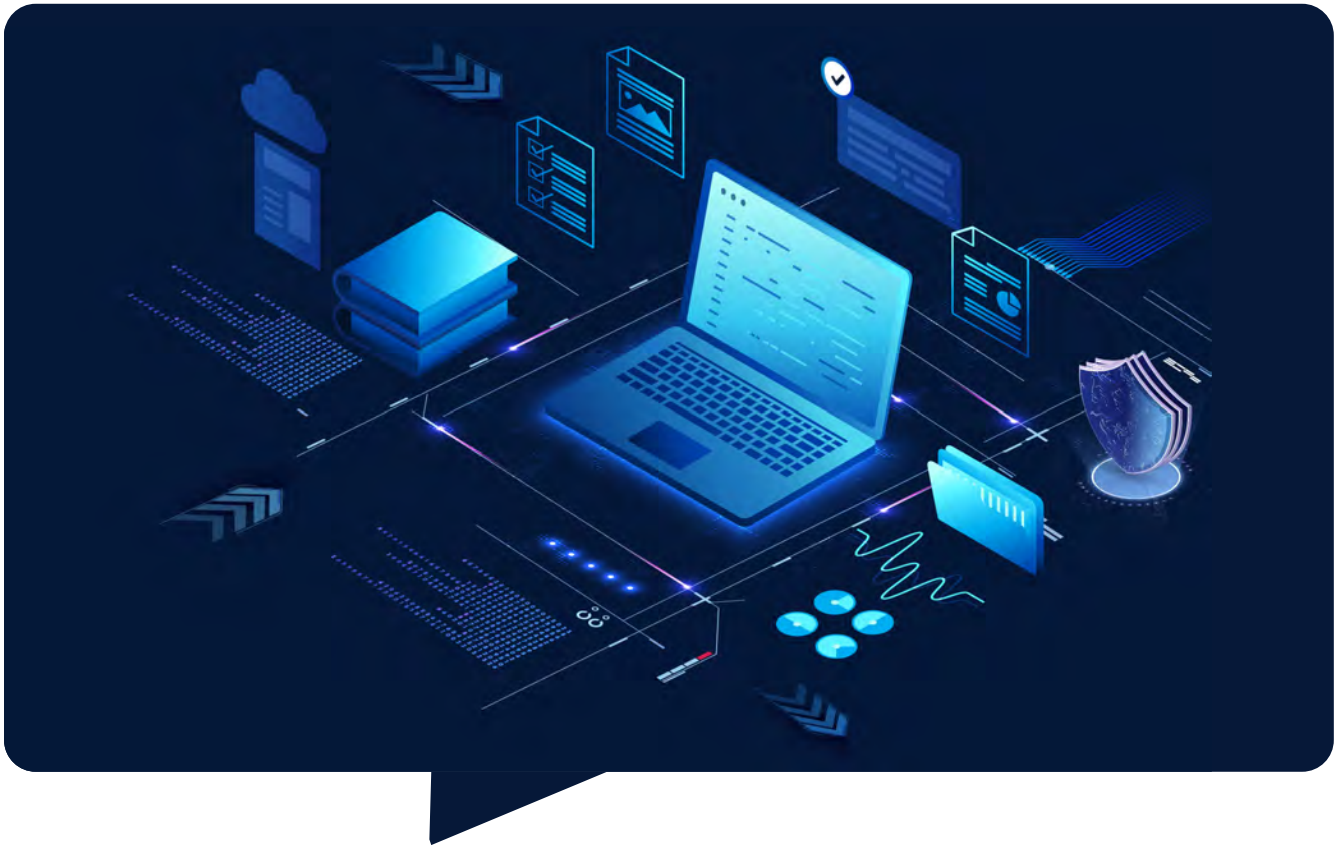


# TECH NAVIGATOR: AGENTIC ENTERPRISE AI PLAYBOOK





# CONTENTS

Executive summary	4
Agentic AI systems	6
Agentic AI architecture and blueprints	18
AgentOps and agentic life cycle management	36
Advanced agents	48
Responsible agentic AI	58



## Executive summary



The transformative potential of artificial intelligence (AI) has captivated everyone for decades, evolving from a futuristic concept to the pinnacle of the hype cycle. Now, AI is on the verge of its next level of evolution: Agentic AI. Unlike traditional AI, which enhances human tasks through insights and automation, agentic AI redefines both expectations and capabilities. It goes beyond supporting existing workflows, instead reimagining and redesigning processes from the ground up to create truly AI-native systems.

In its early stages, generative AI primarily served as a tool to enhance efficiency and accuracy for individuals and teams. However, as AI models have matured and become increasingly commoditized, the focus is shifting from augmentation to reinvention — transcending incremental

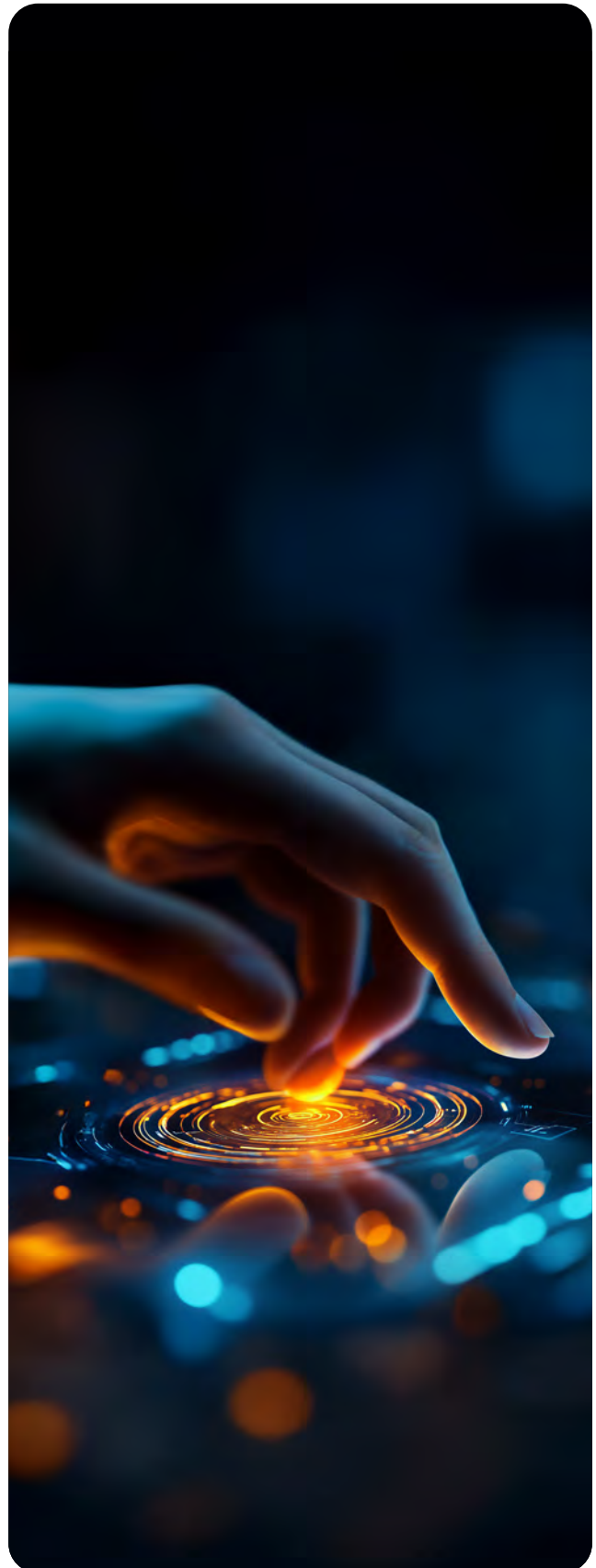
improvements. This transformation echoes past technological revolutions, such as the shift to digital-first processes during the rise of digital transformation. Just as businesses once reengineered processes and business models, rather than merely digitizing analog workflows, agentic AI demands a similar rethinking of processes to unlock its full potential. Achieving this requires deep integration of cognitive capabilities into process engineering and experience design — not just at the periphery of business but at its very core.

The potential of agentic AI extends far beyond its early applications in customer service and IT operations. It has the power to revolutionize mission-critical domains, such as customer onboarding and credit decisioning in banking; supply chain management in retail, consumer goods,

logistics, and manufacturing; sales and marketing life cycles; and product design and development. By embedding intelligent agents across these functions, organizations can unlock higher levels of efficiency, adaptability, and innovation, fundamentally transforming how they operate and compete in an AI-driven world.

Yet this journey is not without its challenges — the inevitable barriers, frustrations, and setbacks that mark all progress. The technology is still evolving, and enterprises face critical questions about how to strategically implement agentic AI at scale. How should organizations architect systems that can accommodate hundreds — or even thousands — of intelligent agents? Which platforms and models should they adopt? How can they ensure interoperability among diverse AI systems while maintaining flexibility for future advancements?

This report explores these critical questions through a pragmatic lens, cutting through the hype surrounding agentic AI to provide actionable insights for enterprise leaders. It offers practical guidance on navigating this complex landscape and implementing AI-driven strategies effectively. By adopting a poly-AI and poly-agent architecture — one that integrates the best models, providers and agents while still ensuring interoperability — organizations can stay agile, future-proof their investments, and gain an early lead on competitors.







# AGENTIC AI SYSTEMS



Artificial intelligence (AI) has steadily surpassed human cognition in fields once thought to be uniquely ours — from image recognition and speech processing to algorithm design. Yet, despite its astonishing power, computer scientists are still trying to make AI behave more like humans: Intuitive, adaptable, and independent.

People are naturally skilled at recognizing patterns and making sense of chaos, even when information is disorganized or incomplete. However, comprehension doesn't always strike immediately: We rely on books, online searches, and the wisdom of others to make more informed decisions that lead to better results. Generative AI follows a similar trajectory by retrieving information, generating insights, and sometimes taking actions — whether that's analyzing a customer's purchase history to recommend

tailored products or automating essential tasks, such as sending emails and processing transactions.

The expectation that AI will deliver highly personalized experiences stems from the level of tailored care humans have come to expect across the services they consume. Businesses have refined their hyperpersonalization efforts, leveraging streamlined communication and interaction history to enhance customer engagement. Basic and routine inquiries were efficiently managed by chatbots that respond quickly and reduce human intervention.

However, while early chatbot implementations served their purpose, their lack of context awareness and empathy soon became clear, exposing a gap in user experience. The shift from static

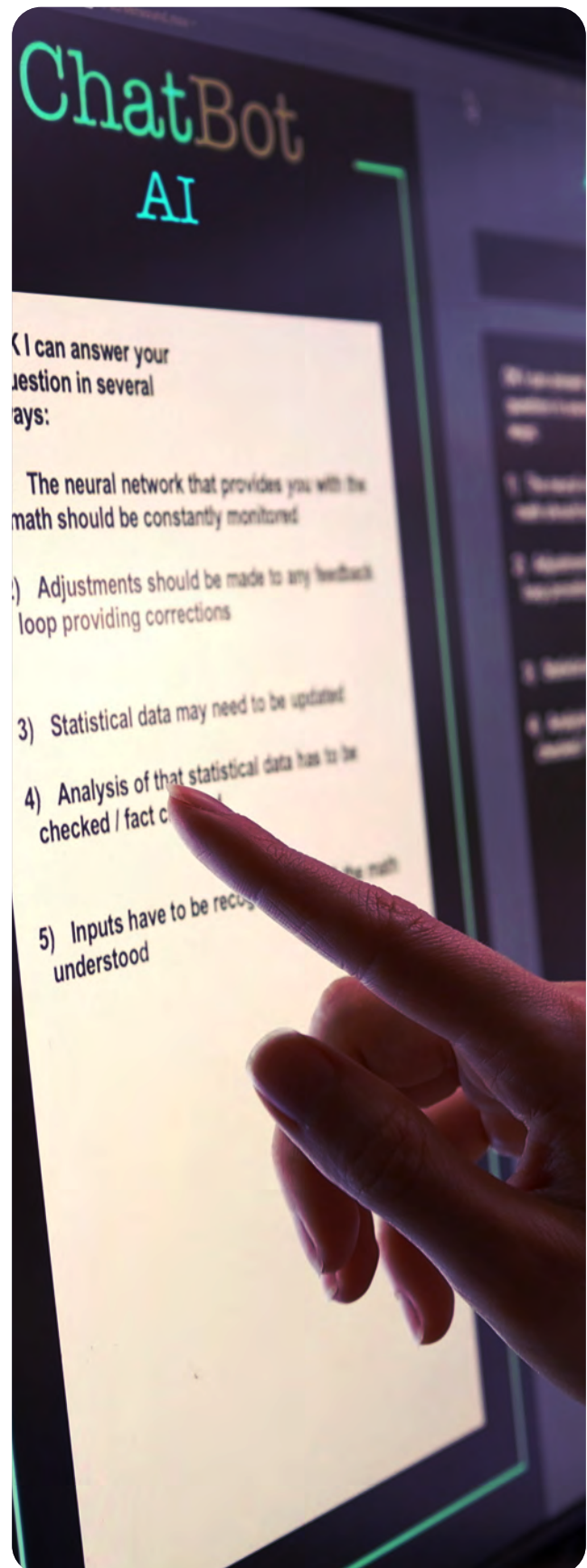
acknowledgments to dynamic, adaptive responses — capable of inheriting context from the industry domain and acting on insights — represents a significant opportunity for AI advancement. Developing systems that can understand, adapt, and respond intelligently in real time is the next frontier in AI-driven personalization.

When AI gains the ability to reason, coordinate tasks, and act with purpose, it transcends being a mere tool and becomes an agent. And as companies seek more adaptable, self-sufficient solutions, these emerging AI agents will evolve from technology tools into indispensable business partners.

## The evolution of AI agents

Agentic AI's arrival is accelerating — reminiscent of the rapid rise of generative AI just over two years ago. Gartner considers agentic AI to be the [top technology trend for 2025](#). And Deloitte forecasts that by 2025, one-quarter of companies that use generative AI will initiate agentic AI pilots or proof-of-concept projects, with adoption increasing to half by 2027. The consulting firm also projects that in certain industries and use cases, agentic AI applications could begin integrating into existing workflows in late 2025.

At their core, agents are autonomous software entities that use a simple yet potent operational loop: They observe their environment via sensors, process this input, and use either mechanical or digital actuators to change their environments and achieve



specific objectives. This traditional sense-plan-act cycle remains as relevant in the AI era as when it was initially conceptualized for robotics.

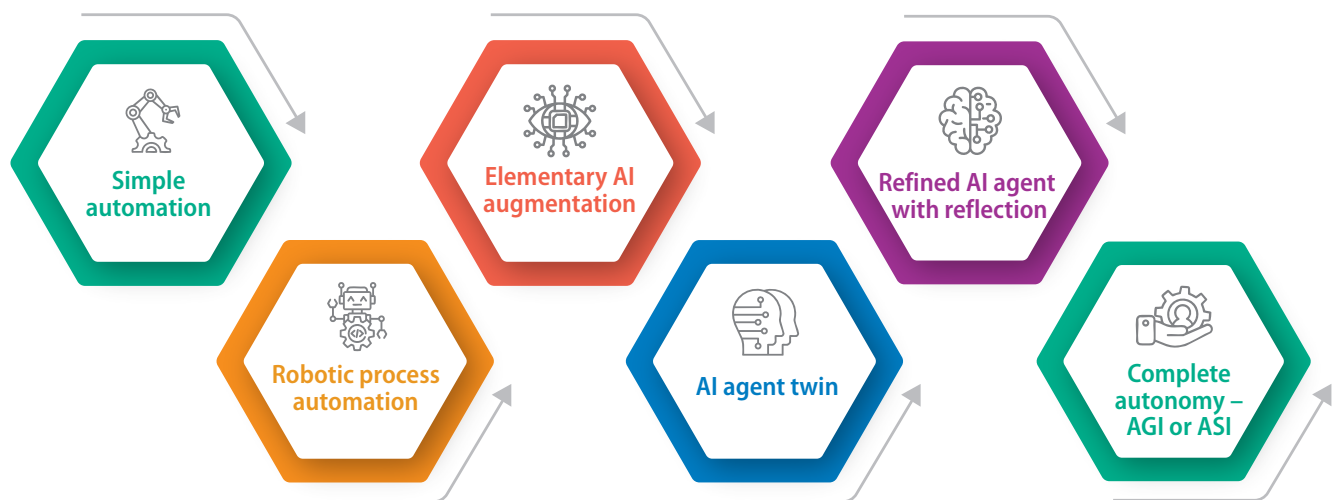
As agents evolve through each stage of enterprise expansion and adoption, their capabilities progressively advance. Traditional robotic process automation relies on rules-based or instruction-driven configurations, operating strictly within predefined parameters. These systems respond only to programmed rules, lacking adaptability or contextual reasoning.

In contrast, agentic AI systems introduce adaptive, autonomous decision-making, bridging the gap between rigid automation and human-like reasoning. With minimal to no human intervention, AI agents perform specific tasks using capabilities that are integrated across multiple layers in the overall agentic system.

Now, the rise of large language models (LLMs) has transformed the agentic world by acting as increasingly sophisticated brains behind the agents. Agents and LLMs function as partners with agents orchestrating LLMs as needed. However, the agent retains control over when to call and execute an LLM, ensuring structured, task-driven interactions that align with the system's objectives and security requirements. From simple automation to the potential for artificial general intelligence (AGI), AI has steadily progressed from a world of rules-based actions to greater independence (Figure 1).

**Tier 0:** Simple automation: Localized automation applies to a specific segment of a broader process. It lacks agentic behavior and operates through rules-based, deterministic automation. A common example is component test automation, where scripts execute predefined tasks.

Figure 1. The progress of AI agents



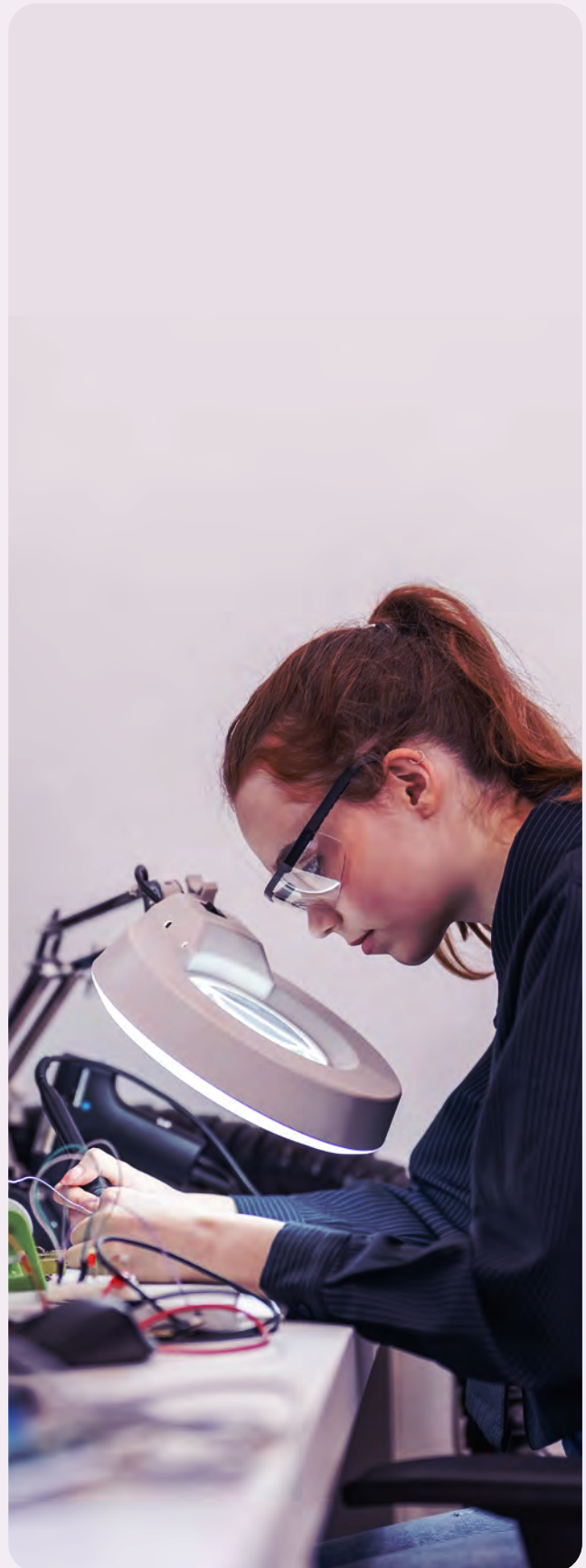
Source: Infosys

**Tier 1:** Robotic process automation: An advancement over basic automation, this approach extends beyond point solutions to cover entire process segments using predetermined rules-based logic. While AI or agentic behavior remains minimal to nonexistent, it enhances efficiency in tasks such as screen scraping or automated form completion.

**Tier 2:** Elementary AI augmentation: This stage introduces the first meaningful presence of agentic AI, offering an opportunity to replace tier 1 automation, particularly in areas requiring human oversight. By leveraging a language model, agents enable intelligent interactions while maintaining a limited, yet impactful, role in automation. Examples include sentiment analysis or ticket data labelling, where agents classify information into appropriate categories.

**Tier 3:** AI agent twins: These systems function as digital twins to users, interpreting intent and autonomously taking action to achieve specific outcomes. Some of the best-known instances of AI agent twins include GitHub Copilot and Microsoft 365 Copilot, which assist users by generating code, automating tasks, and enhancing productivity through intelligent decision-making.

**Tier 4:** Refined AI agents with reflection: These systems represent an advanced class of AI agents that many organizations are eager to implement. They can decompose tasks from a given objective, formulate plans to achieve the intended outcome, and analyze results to adapt their approach in response





to failures or unexpected events through complex reasoning sequences. This tier can be seen in credit decision-making systems, where agents process loan applications, extract and analyze documents, and match them against stored information to ensure accuracy and compliance.

**Tier 5:** Complete autonomy, AGI or artificial superintelligence: When this tier arrives in the future, these agents will possess the capability to conduct entirely original research, independently reason through complex problems, and develop innovative solutions beyond their initial training data. With advanced logical reasoning and adaptive learning, they will continuously acquire new skills, refine their methodologies, and tackle previously unsolved challenges, pushing the boundaries of AI-driven discovery and problem-solving.

Although [tier 5 is still somewhere in the future — with optimistic predictions ranging from 2026 to 2029](#) — AI has already shifted possibilities and expectations. This technology has evolved from rules-based automation to intelligent, self-improving systems capable of operating independently, dynamically responding to their environments, and optimizing decision-making in real time. Organizations can now deploy these agentic solutions across domains — such as software development, IT operations, and customer care — to drive unprecedented efficiency and adaptability.

## The blueprint for agentic AI

Business leaders recognize the value AI has already delivered and see its potential



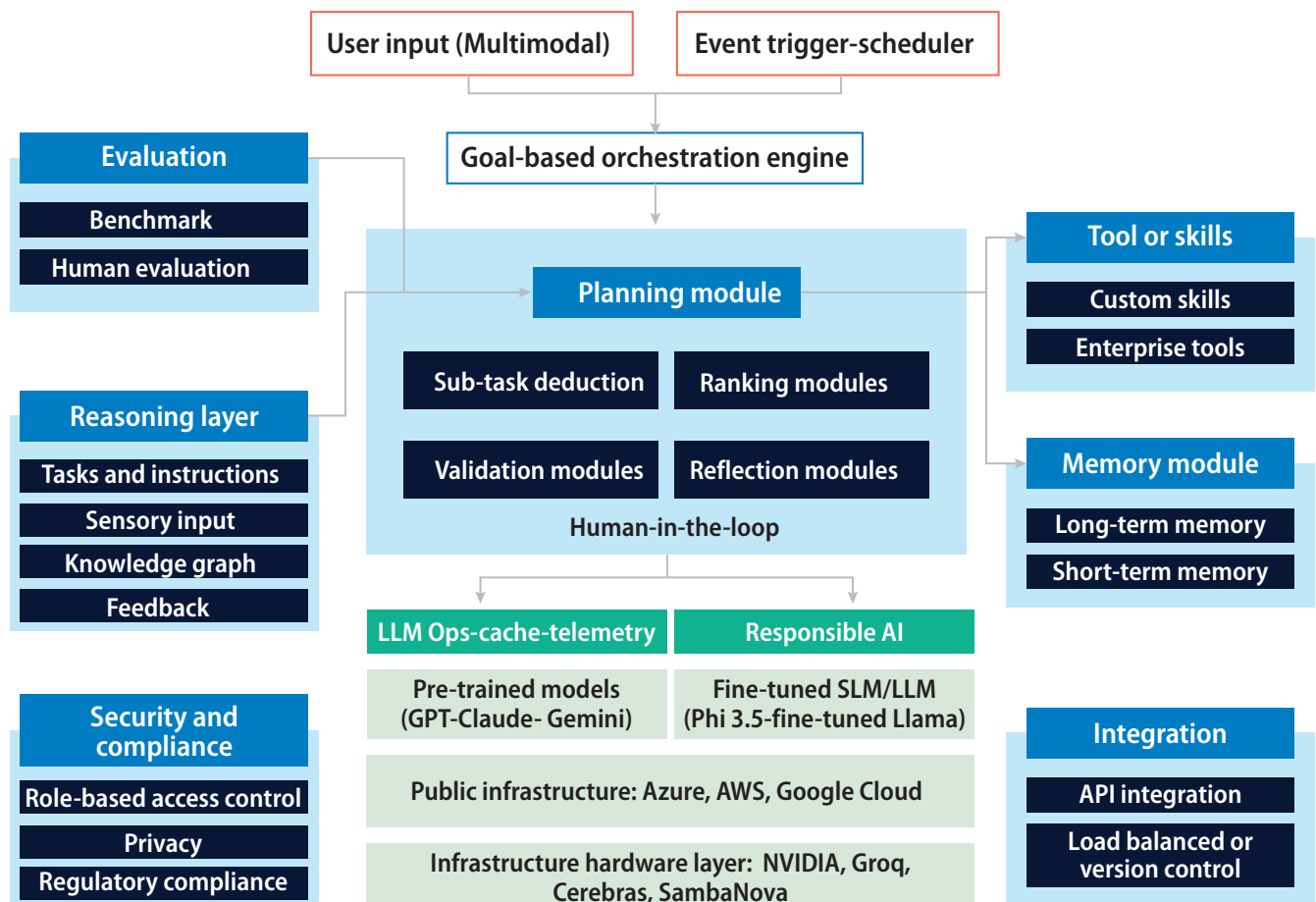
to drive further transformation. However, fully capitalizing on AI-driven opportunities requires a strong foundation in both architecture and operational layers of advanced AI systems. The blueprint for agentic AI defines the core components and processes that drive their functionality. By analyzing this framework, it becomes clear how user inputs, orchestration engines, planning modules, reasoning layers, utilities, memory, integration interfaces, human

oversight, and pretrained language models work together to enable intelligent and autonomous agent behavior.

The following are the core layers of an agentic system and an examination of how it operates (Figure 2).

- **User input or event trigger:** A predefined instruction set or event combination that activates an agent, prompting it to initiate

Figure 2. AI agent framework



Source: Infosys

actions or processes based on specified conditions.

- **Goal-based orchestration engine:** This serves as the foundation for decision-making and prioritization, orchestrating various actions to align with a system's goals.
- **Planning module:**
  - **Reasoning layer:** Algorithms designed to analyze inputs, develop strategies, and break tasks into subtasks to ensure efficient execution.
  - **Tools and skills:** Reusable software modules activated to achieve specific goals, such as document digitization, optical character recognition, and PDF generation.
  - **Memory:** Module maintains a record of current and past interactions, tool usage, and learned behavior, enabling context-aware planning.
  - **Integration module:** Module interfaces with the outside world, often through APIs.
  - **Human in the loop:** Humans approve certain critical decisions recommended by agents.
- **Language model:** Pretrained or fine-tuned models deployed responsibly on public or private cloud infrastructure.
- **Security and compliance:** This layer ensures strict compliance with data security, privacy, and ethical standards, making sure the agent aligns with responsible AI principles and regulatory requirements.

- **Evaluation layer:** This layer measures the effectiveness and the efficiency of the agent. This evaluation layer also provides feedback to the planning module for continuous improvement.

## Planning module

The planning module, often considered to be the agent's "brain," is the central orchestrator of decision-making, task prioritization, and adaptive execution in an agentic system. This planning module also plays a pivotal role in transforming reactive automation into proactive, intelligent decision-making.

Functioning alongside the orchestration layer, the planning module interprets input triggers and breaks down objectives into achievable goals. The orchestration layer refines these objectives into a structured sequence of subtasks. It also oversees execution, engages reflection modules for validation, and generates user responses.

Together, these layers form a continuous cycle that guides agent behavior through the plan, act, and reflect stages. This process involves gathering data from internal and external sources, analyzing it using predefined logic, reasoning frameworks, or learned patterns, and determining the optimal next step based on the current state and objectives.

By integrating these components, agentic AI systems achieve a high degree of autonomy, adaptability, and operational intelligence.

The layers that enable planning at scale include the following:

### Reasoning layer

This capability shapes interactions with the language model by refining prompts and evaluating or ranking responses. It plays a crucial role in identifying specific milestones within subtasks, processing environmental inputs, and capturing sensory data. Additionally, this reasoning layer leverages knowledge bases and reference frameworks, such as knowledge graphs, to narrow down results and enhance accuracy.

### Memory module

The ability to retain context from both current and past interactions, while continuously learning from ongoing and long-term activities, provides the reasoning layer with the necessary background and feedback for effective operations.

### Tools, skills, and integration layer

This layer compensates for the inherent limitations of language models in directly interacting with the real world. It achieves this by leveraging reusable software components, such as PDF generation and document digitization, to accomplish specific goals. Additionally, this layer enables integration with specialized systems by accessing external web APIs, including retrieval-augmented generation (RAG) frameworks.





## Language model

The language model serves as the central decision-maker, consisting of one or more models that employ reasoning frameworks such as ReAct or chain-of-thought. These models can be general, multimodal, or fine-tuned for specific business objectives. For optimal performance, the chosen model should align with target requirements and be trained on data relevant to the integrated tools. While the model is typically not trained on the agent's specific configuration, its decision-making accuracy can be enhanced by providing contextual examples that highlight the agent's capabilities. This approach ensures more precise and context-aware outputs.

The planning and reasoning cycle continues iteratively until the goal is achieved, or a stopping condition is met. The complexity of orchestration depends on the agent and task, varying from simple calculations to advanced logic, such as chained reasoning or machine learning algorithms.

## Why do we need AI agents?

The rise of agentic AI is fueled by the growing complexity of modern businesses and the demand for intelligent automation. Agentic AI overcomes the limitations of traditional automation by integrating adaptability, contextual awareness, and autonomous decision-making.

Unlike conventional systems, which struggle with unstructured challenges and evolving conditions, agentic AI processes ambiguous

data through continuous learning and real-time analysis to optimize workflows. This shift from rules-based automation to context-driven intelligence allows organizations to service increasingly complex demands. The following features make agentic AI particularly relevant for the modern enterprise.

- **Operational efficiency:** Integrates cross-functional tools, skills, and APIs to detect inefficiencies and autonomously implement improvements, ensuring process agility.



- **Ability to scale rapidly:** Unlike the resource-intensive traditional systems, agentic AI enhances capabilities without requiring proportional staffing increases by taking advantage of self-optimizing architectures.
- **Dynamic problem-solving:** The ability to manage unstructured tasks through contextual reasoning allows agents to generate intelligent responses to complex requests without relying on predefined rules.

## When is agentic AI the best way?

Frameworks play a crucial role in the effective deployment and use of agentic AI, ensuring that agents operate efficiently, ethically, and in alignment with business objectives. And with the limited number of real-world case studies, these frameworks serve as blueprints that can guide organizations that want to move quickly and decisively — rather than waiting for competitors to define the way forward.

At Infosys, we use the SCOPE framework to assess the suitability of agentic AI for addressing business challenges. Each parameter is then scored based on its impact.

### Strategic alignment

- Does the opportunity align with long-term organizational goals?
- Are the workflows we use today largely manual?

### Complexity of task

- Is the task planning intensive and multistep?
- Is the planning largely deterministic, and if so, are the rules expected to increase exponentially to comprehensively cover the possibilities?

### Operational environment

- Is there a need for real-time action or decision-making, or a need to adapt to changing circumstances?
- Are there multiple data streams to integrate with and will the need grow to integrate with more systems?

### Performance requirements

- Are there latency constraints or impacts that affect the cost?
- What are the ethical and security constraints?
- Can the system's actions be adequately governed, controlled, and monitored?
- Are there ethical risks with autonomous decision-making or recommendations?

In addition to insights from a robust framework, enterprises should closely examine the growing number of agentic AI use cases that are applicable to complex, real-world scenarios. Below are the most promising uses for agentic AI, although not all have been tested at enterprise scale. (Figure 3). Remember that agentic AI is an emerging technology that is still building a track record.




While AI adoption is accelerating, its enterprisewide deployment remains

limited, with many organizations still in the experimental phase. Relatively few have successfully implemented and scaled agentic AI across specific operations. Below are examples of organizations that have made significant progress in integrating agentic AI into their workflows.

In one example, [CommBank](#) leverages

agentic AI to process about 15,000 payment disputes every day. Customers can describe their issues through an AI-assisted channel, which autonomously verifies eligibility criteria and lodges disputes without requiring manual intervention. This automation enhances efficiency, reduces processing time, and improves customer satisfaction.

**Figure 3. Agentic AI use cases by industry**

 Industry	 Use cases	 Capability
Financial services	Periodic risk review and renewals Risk and compliance reporting Watchlist reporting (adverse news, sanctions, PEP)	HSBC has built NOLA2.0, a cloud native solution on GCP to modernize credit risk management in compliance with Basel III (automated regulatory adaptation with real-time capital allocation).
High tech	Driverless cars route adaptation	Waymo's driverless cars autonomously adapt routes for safety and efficiency by analyzing sensor data (LIDAR, cameras) to navigate and avoid obstacles in real time.
Logistics and supply chain	Dynamic fleet route management	Pando.ai rerouted 5,000-plus containers during the 2024 Panama Canal drought, avoiding \$12 million in delays.
Manufacturing	Optimizing engineering systems workflows	Siemens developed a multiagent system to make engineering workflows more efficient, with the agents acting as system architects and requirements engineers.
Telecom	Continuous network monitoring and optimization	Dynamically allocates bandwidth and resolves congestion by autonomously adjusting parameters. Nokia has announced agentic AI in autonomous network management. ServiceNow has introduced support for agentic AI for better network management.

Source: Infosys Knowledge Institute

Telecom company **Telenor** has deployed conversational agentic AI agents to autonomously handle customer queries, resolve issues, and facilitate sales. This implementation led to a 20% increase in customer satisfaction and growth in revenue of up to 15% within the first year.

**Talkdesk** introduced agentic AI-powered conversational agents for retail customer service, enabling autonomous management of complex tasks such as order updates, address modifications, and customer routing to in-store specialists. These AI agents provide 24/7 support while delivering hyperpersonalized experiences, enhancing both efficiency and customer engagement.

**Levi Strauss** has implemented agentic AI for granular demand forecasting across its supply chain. The system autonomously adjusts inventory levels based on real-time demand signals, ensuring that the right products are available in the right locations while minimizing waste and inefficiency.

## Redefining AI

The emergence and maturation of agentic AI marks a defining moment in the evolution of intelligence — one that transcends rigid algorithms and human-dependent decision-making. Unlike its predecessors, which followed predefined rules, agentic AI is self-directed, adaptive, and deeply contextual, capable of understanding real-time data, reasoning dynamically, and acting with intent.

This shift is more than an incremental improvement; it redefines how technology



engages with the world. No longer bound by static programming, AI is evolving into a thinking entity, a collaborator capable of navigating complexity, optimizing processes, and unlocking vast new possibilities.

Ultimately, the rise of agentic AI represents a transformative leap in the field of AI. By enabling machines to think, learn, and act independently, we are not only enhancing the capabilities of AI but also redefining the relationship between humans and technology. As businesses continue to explore and implement these advanced systems, the future of AI promises to be more dynamic, intelligent, and impactful than ever before.





# AGENTIC AI ARCHITECTURE AND BLUEPRINTS



Artificial intelligence (AI) is evolving into more intuitive and independent systems with the advent of agentic AI, which allows for autonomous decision-making and real-time responsiveness. This year, many companies will begin exploring agentic AI pilots, with widespread adoption anticipated by 2027.

This report delves into the transformative potential of agentic AI, the roles of AI agents, and the importance of understanding agentic architecture for business leaders.

Agentic AI is transforming how businesses automate decision-making and streamline complex workflows. However, simply deploying AI agents does not guarantee success. Agent capabilities vary and do not function in isolation; they operate alongside humans and are embedded within enterprise business processes across the value chain.

As AI capabilities advance, business leaders will develop a deeper understanding of the intricacies at play: the distinct roles of AI agents, the layers of agentic architecture, and the tools and frameworks that facilitate integration into existing IT ecosystems. This will be a strategic imperative, not just a technical curiosity.

This deeper knowledge allows business leaders to make more informed decisions about technology investments, resource allocation, and process optimization — enhancing efficiency and strengthening their competitive advantages.

A well-structured agentic system ensures that AI decision-making is transparent, adaptable, collaborative with human teams, and strategically aligned with business objectives.

## Types of AI agents

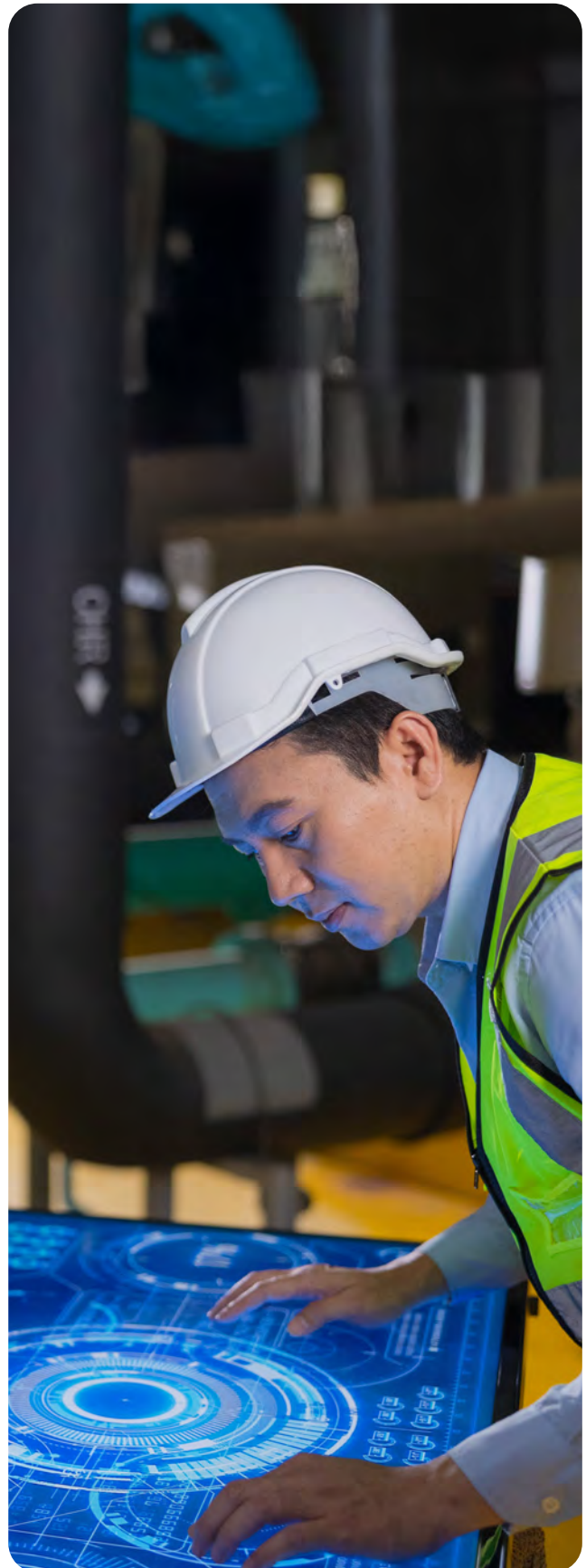
At the core of agentic AI architecture is the sense-plan-act cycle, where AI agents interpret data from their environment, formulate a plan, and autonomously execute tasks. This architecture establishes the fundamental building blocks required for AI agents to operate effectively, whether as a single agent or as a multiagent system. Both approaches leverage machine learning and computational reasoning, but the choice of agent type significantly affects whether the AI system can reach its full potential.

Different agents are required for varying levels of task complexity, ranging from simple reactive systems to highly sophisticated learning agents. Selecting the appropriate agent type ensures it can handle the specific task, whether it involves basic automation or complex decision-making.

Understanding an agent's capabilities helps set realistic expectations about what it can achieve, preventing overreliance on its abilities and potential misuse. Knowing the type of agent allows developers and architects to:

- Select the best algorithms and learning methods.
- Determine appropriate data collection and preparation processes.
- Choose architecture that enables optimal performance and integrates with existing systems.

By aligning the agent type with its intended functionality, organizations can maximize



AI performance, efficiency, and reliability while avoiding unnecessary complexity or operational risks. (Figure 1.)

## Reflex-based agents

Reflex-based agents use a predefined set of rules, or reflexes, making them the most basic type. These agents are programmed to perform specific actions whenever certain conditions are met, without requiring advanced reasoning or learning.

In manufacturing, reflex-based agents can play a critical role in quality control. These agents use cameras to capture images of components as they move through the production line. Using predefined criteria, the agents analyze images in real time and instantly flag defects. Their actions are binary, either accepting or rejecting components based solely on the visual input. For example, Intel uses reflex-based agents in its chip manufacturing plants to improve precision and efficiency in quality control.

## Model-based reflex agents

Model-based reflex agents are a specialized type of reflex agent that maintains an internal model to track environmental changes. This model is continuously updated as new information arrives, allowing the agent to make decisions based not only on its reflexes but also on past experiences and current state.

A smart irrigation system is a prime example of a model-based reflex agent. Rachio smart sprinklers integrate weather forecasting data

and soil moisture models, learning from local weather patterns while considering soil type, plant type, sun exposure, and slope to create predictive, data-driven watering schedules.

## Goal-based agents

Goal-based agents integrate an internal model of the world with a defined goal or set of goals. Before acting, they plan and search for action sequences that will help them achieve their objectives. This deliberate, strategic approach makes them more effective than reflex-based or model-based agents. These agents are widely used in warehouse automation to optimize order fulfillment. In automated fulfillment centers, goal-based agents operate with specific objectives, such as completing an order within a set timeframe.

To achieve this, agents plan the most efficient picking routes through the warehouse while coordinating with other robots to prevent conflicts and ensure smooth operations. A notable example is Amazon's Kiva robots, which efficiently manage inventory movement and order fulfillment in the retailer's automated warehouses.

## Utility-based agents

Utility-based agents select the optimal sequence of actions that allow them to reach their goals and maximize utility or reward. Utility is determined by a function that assigns a utility value — a metric that measures the usefulness of an action or how “satisfied” it will make the agent — to each scenario based on fixed criteria.

These agents play a crucial role in portfolio management, particularly in automated trading and investment optimization. Investment companies deploy utility-based agents to navigate the complexities of financial markets, evaluating factors such as risk, return, market conditions, and client preferences. Using sophisticated utility functions, these agents balance competing objectives, such as growth and stability, to make optimized investment decisions. A prime example is BlackRock's Aladdin platform, which leverages utility-based agents to enhance investment management and performance.

### Object-centric or 'curious' agents

An object-centric, or "curious", agent focuses on understanding and interacting with individual objects in its environment rather than simply perceiving the overall scene. It prioritizes detailed information about specific objects, their properties, and their relationships with one another to inform decision-making and actions.

These agents are used frequently in the healthcare industry, particularly in radiology departments. Object-centric agents actively analyze medical images, detecting and learning new patterns across various imaging types, such as X-rays, MRIs, and CT scans. By continuously learning from new cases, these agents improve diagnostic accuracy over time. A notable example is [Arterys' AI platform](#), which uses object-centric agents to enhance medical imaging analysis, ultimately supporting better clinical outcomes.



### Citizen-centric autonomous agents

Citizen-centric autonomous agents use advanced AI models to independently manage routine personal and professional tasks. These agents analyze user objectives, break them into actionable steps, and execute workflows across applications without manual intervention.

For instance, OpenAI's Operator agent can autonomously manage email sorting,



appointment scheduling, and online purchases by interacting with web interfaces through screenshots and keystroke simulations. Similarly, Google's Agentspace enables users to automate document processing, such as extracting key details from PDF invoices and populating expense reports in real time. Anthropic's

Claude Computer Use is an AI agent that is good at autonomously performing desktop tasks while Operator is good at browser-based tasks. By integrating natural language understanding with cross-platform interoperability, these tools allow individuals to delegate repetitive digital tasks.

Figure 1. Each agent type focuses on a different part of the sense-plan-act cycle

Agent type	Sense	Plan	Act	Characteristics and use cases
Reflex-based	Pre-defined rules triggered by conditions	Minimal planning: immediate actions based on rules	Binary actions: accept or reject input	Basic agents effective for simple tasks such as defect detection in manufacturing
Model-based	Maintains internal model; updated with perceptions	Actions based on current model and history	Actions depend on model predictions	More complex; deployed for smart grid optimization
Goal-based	Gathers information relevant to specific goals	Searches for action sequences to achieve goals	Executes actions aligned with goals	Designed for efficiency; common in warehouse automation
Utility-based	Evaluates input against a utility function	Plans actions to maximize utility based on criteria	Chooses actions that optimize overall utility	Maximizes outcomes; used in finance for portfolio management
Object-centric or "curious"	Understands individual objects and their properties	Plans actions based on knowledge of specific objects	Interacts with objects for learning or analysis	Emphasizes object relationships; used in medical imaging analysis
Citizen agents	Multimodal inputs (text, screenshots, user goals)	Decomposes objectives into steps with LLM-based task prioritization	Interfaces with web or app GUIs and document workflows via API integration	Autonomous execution of personal or civic tasks (scheduling, form processing, service access) with cross-platform interoperability and governance safeguards

Source: Infosys

## How to build an agentic system

Enterprise systems are inherently deterministic, designed to execute specific, rule-based tasks, even when those tasks involve complex business logic. However, integrating AI agents into these systems introduces an element of nondeterminism. Unlike traditional enterprise software, AI agents plan action sequences dynamically, adapting their behavior as inputs evolve through learning. Despite this adaptability, their actions must remain within the boundaries of standard operating procedures to ensure consistent and reliable outcomes.

For developers and architects, understanding the interaction between deterministic enterprise systems and AI agents is critical when designing enterprise-grade applications. Organizations must carefully evaluate task complexity, computational resources, and operational constraints to develop an effective and scalable AI architecture that balances flexibility with control while ensuring reliable performance.

### Agentic AI architecture

When developing agentic AI architecture, organizations must determine the number of agents required and the roles they will play.

Agentic systems follow two fundamental configurations: single-agent and multiagent architectures. Both leverage machine learning models and computational methods to execute the sense-plan-act cycle, enabling intelligent decision-making and automation. By combining established principles with AI



advancements, developers can expand the capabilities of agentic systems, unlocking new possibilities for enterprise applications.

The complexity of a task determines whether a single-agent or multiagent architecture is the best approach. Single-agent architecture is ideal for tasks that follow clear, well-defined steps and have minimal tool requirements. In contrast, multiagent architecture is better suited for complex tasks that require collaboration, feedback, and the parallel execution of subtasks. This report focuses on multiagent architecture.

After deciding that a multiagent system is needed, the agents must then be classified. The role-based agents are labelled based on their jobs, such as advisor, coder, reviewer, or tester. The interaction model-based reflex agents are categorized by their cooperation with other agents. This could be vertical (leader or servant) or horizontal (peer-to-peer).

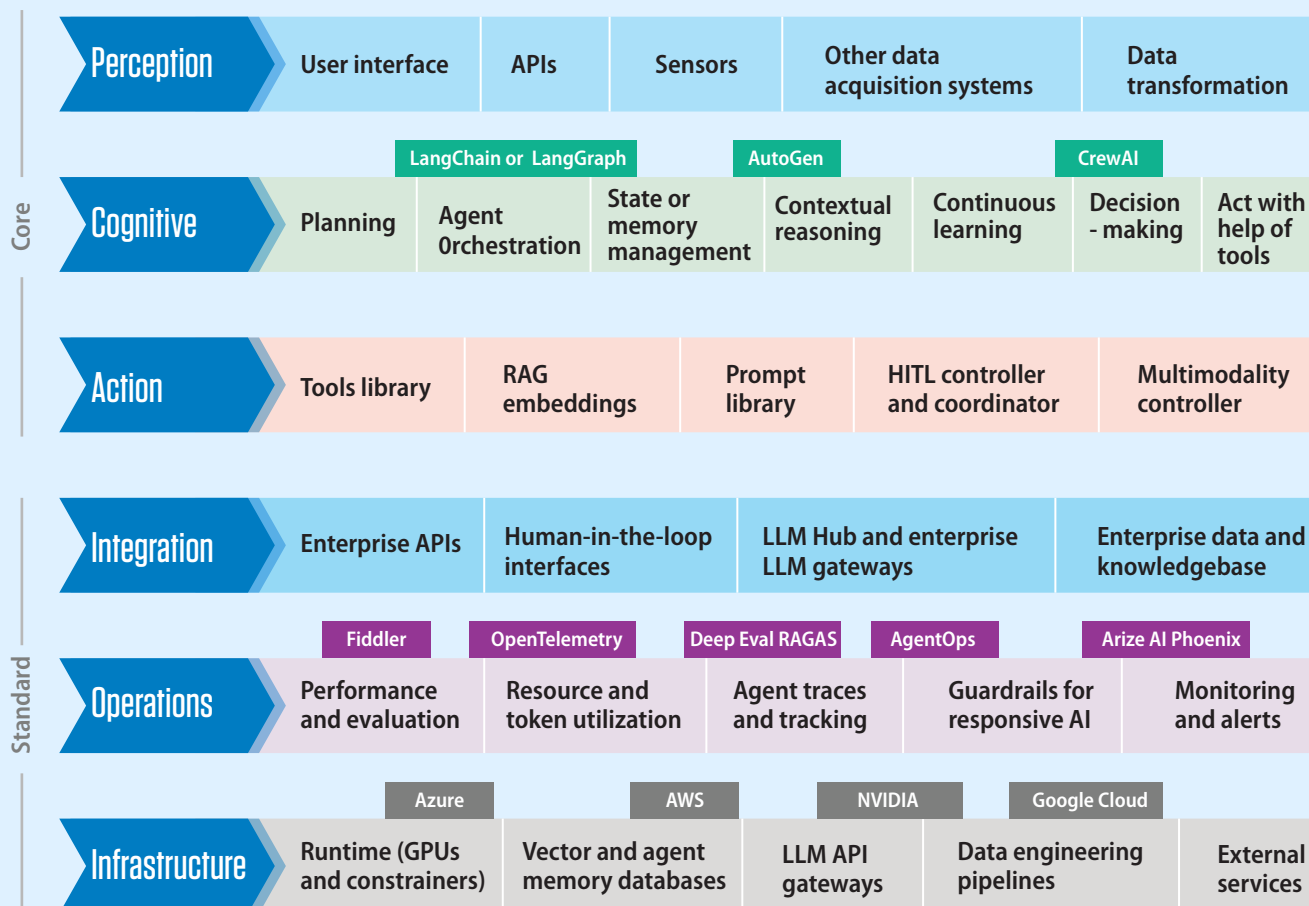
Research indicates that [multiagent systems are more effective](#) when roles are clearly defined. A leader agent coordinates tasks, while servant agents communicate with both the leader and their peers. Teams with an organized leader complete tasks nearly

10% faster than teams operating without a designated leader.

## Agentic architecture layers

The foundation of any agent-based system consists of multiple interdependent layers that work in harmony to enable intelligent behavior. These layers function as a pipeline, transforming raw input into actions while maintaining the agent's internal state and reasoning capabilities. At the core, the sense-plan-act cycle drives decision-making, while the standard architecture layers ensure the system meets critical enterprise architecture requirements (Figure 2).

Figure 2. Agentic architecture layers and supporting technology



Source: Infosys

## Core layers

**Perception:** Functions as the agent's sensory interface with its environment, responsible for processing raw inputs, reducing noise, and extracting relevant features. It converts unstructured data into a structured format, allowing the cognitive layer to process information efficiently. In a robotic system, this layer analyzes camera feeds, sensor data, and environmental readings — transforming them into a structured state representation for further analysis and action.

**Cognitive:** Serves as the agent's brain, handling decision-making, planning, and learning. It manages the agent's internal state, processes information from the perception layer, and determines the most appropriate actions based on its goals and current understanding. Depending on the agent's complexity, this layer can range from a simple rules-based system to an advanced neural network.

**Action:** Translates decisions from the cognitive layer into activity. It is responsible for validating actions, performing safety checks, and monitoring implementation. This layer ensures that planned actions are feasible and safe before implementation, while also monitoring feedback to evaluate outcomes.

## Standard layers

**Integration:** Serves as the connective tissue of the agentic architecture, facilitating seamless communication and data flow between the system's layers and external systems. It enables real-time data access,

manages interoperability, and ensures smooth interactions, enhancing the system's overall functionality and responsiveness.

**Operations:** Oversees and manages the real-time performance of agentic systems. It monitors system activities, provides feedback mechanisms, and facilitates continuous improvement by optimizing processes based on operational data.

**Infrastructure:** Delivers the computational power and storage required for agents to execute complex tasks efficiently while ensuring high availability, scalability, and reliability.



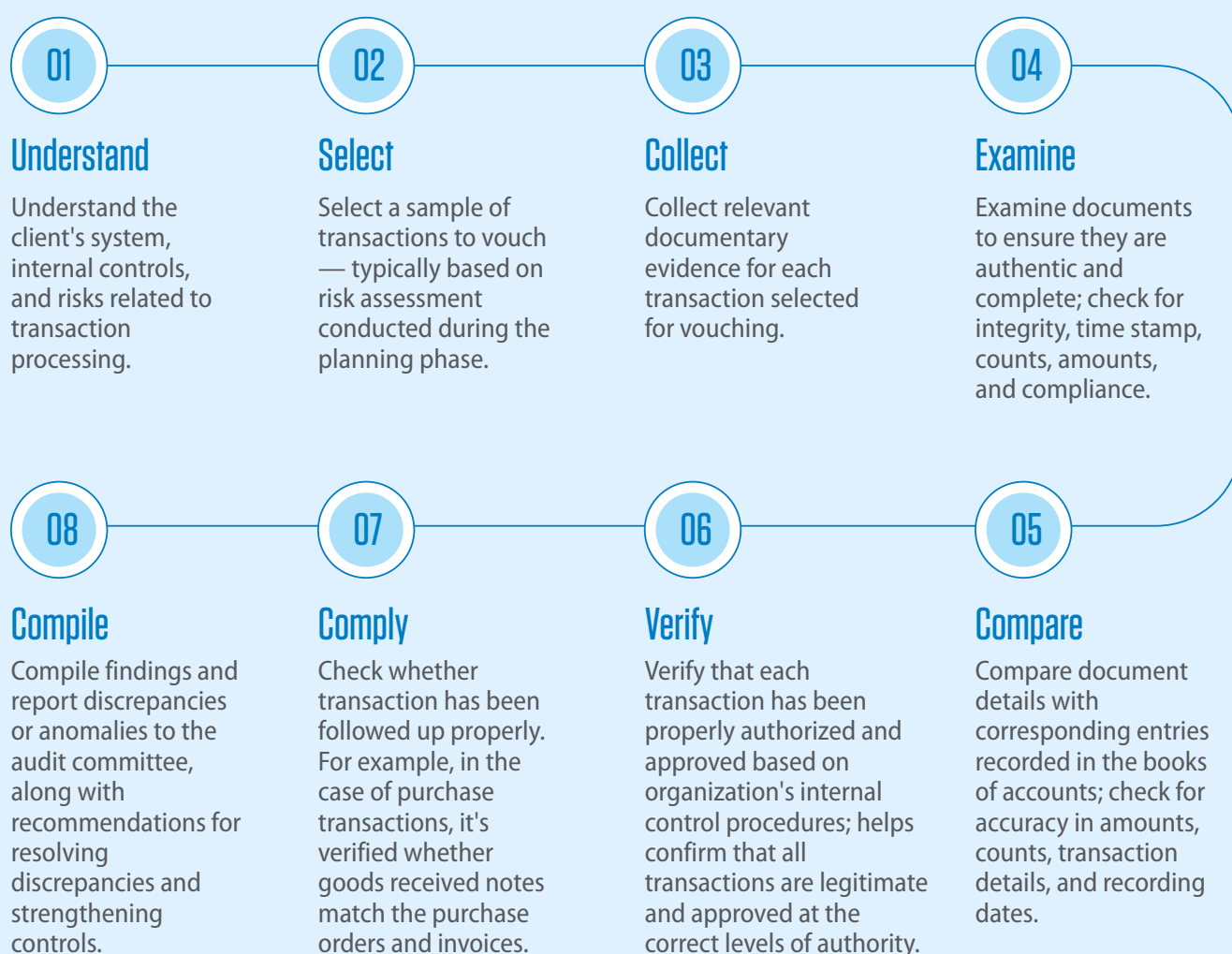


## Agent vs. human: Audit use case

Agentic AI is a versatile and powerful technology capable of handling a wide range of tasks. This use case explores how it can be applied to the standard audit process for expense vouching.

For a human auditor, the vouching process involves verifying recorded transactions against supporting documents to ensure accuracy and authenticity. This method plays a crucial role in detecting and preventing errors and fraud, ultimately ensuring the reliability of financial statements (Figure 3).

Figure 3. Traditional workflow steps for the auditing process



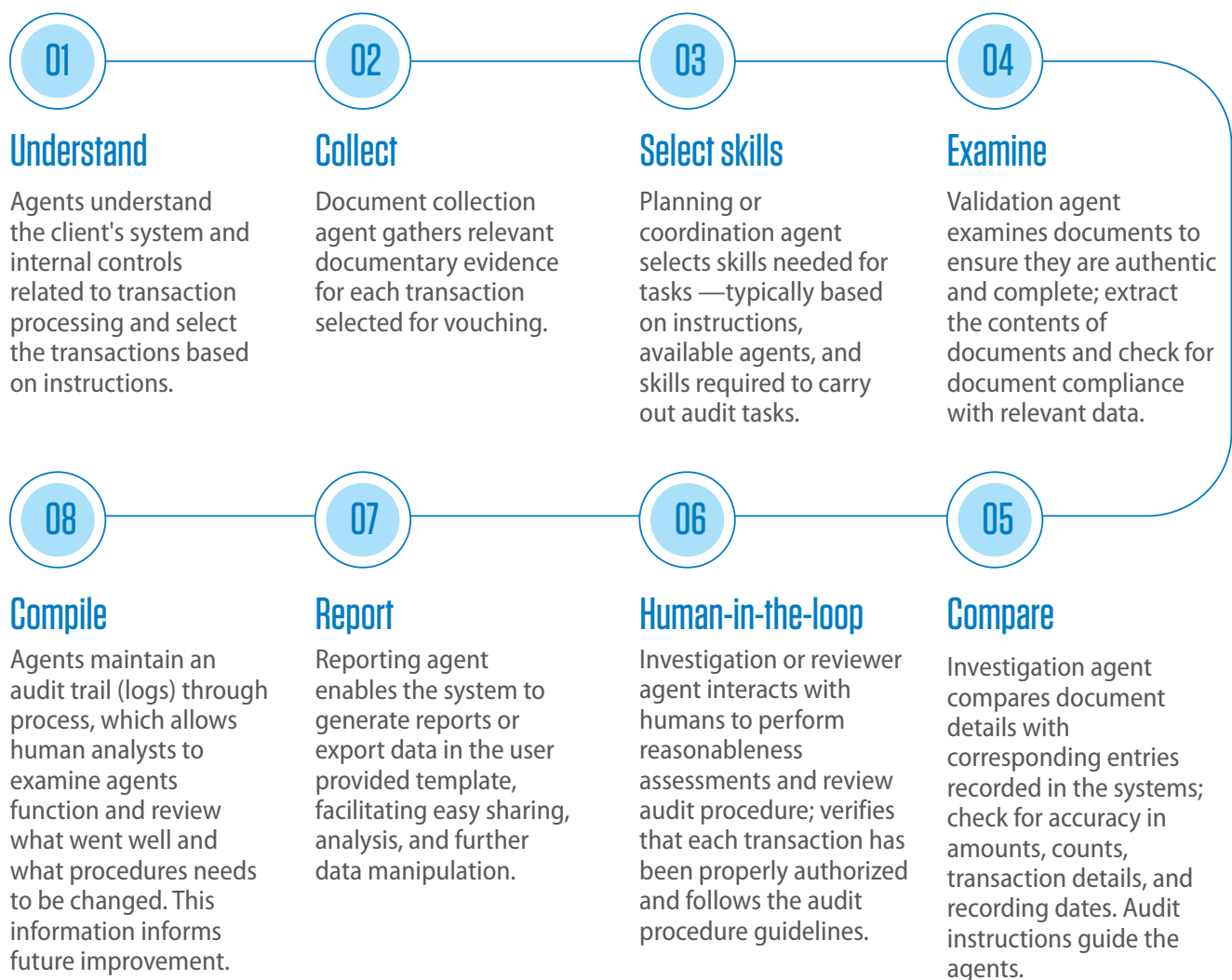
Source: Infosys

To automate the expense vouching process, an agentic AI solution can deploy multiple agents that collaborate within a structured workflow using a human-in-the-loop approach (Figure 4).

AI agents interpret audit instructions, categorize collected files based on type, and extract relevant data elements from

each document. They then populate a tailored vouching template, performing calculations as specified in the audit procedures. Throughout the process, agents meticulously review outputs to ensure compliance with regulatory guidelines, flag discrepancies or exceptions, and facilitate human oversight where necessary.

**Figure 4. Agentic workflow steps for the auditing process**



Source: Infosys

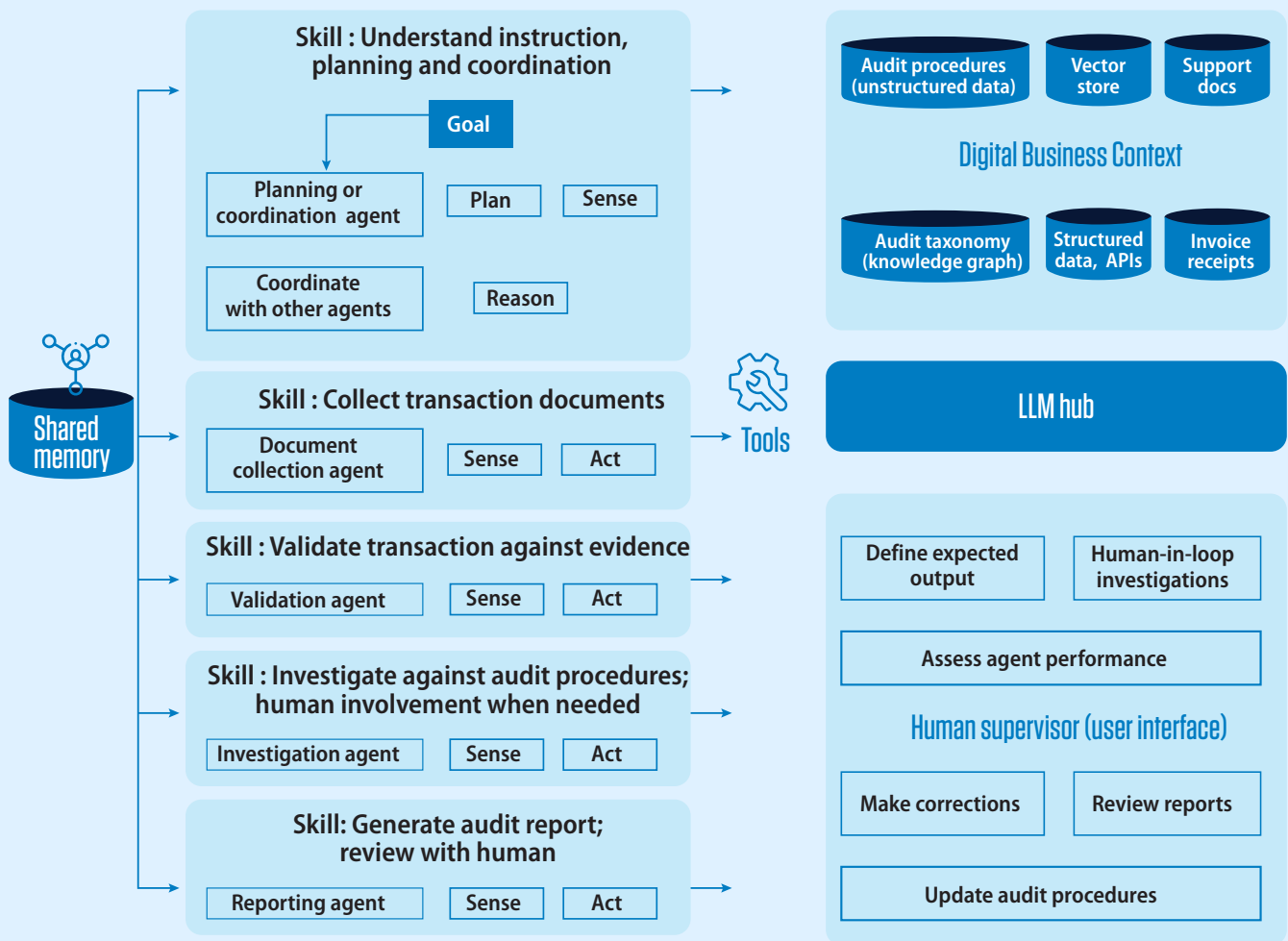
The following view shows the agentic workflow in action, where each agent is equipped with the necessary skills to execute its assigned tasks. Agents interact with inputs through a user interface (UI), while a cognitive layer manages planning, reasoning, and learning using large language models (LLMs) and small language models (SLMs).

These models work alongside a knowledge graph to provide business context, such as the vouching process in this example, and

maintain workflow execution state using shared memory, such as a Redis cache (Figure 5).

To complete their tasks, agents utilize specialized tools, such as enterprise APIs or utility functions. When human intervention or feedback is required to complete a specific operation, the agent engages the human in the loop through the UI, ensuring accuracy and compliance while enhancing overall efficiency.

Figure 5. Mapping the agentic workflow for expense vouching



Source: Infosys

## How to implement agentic AI

The successful implementation of agentic AI systems requires careful selection of agent types, tools, and frameworks that align with their specific capabilities and overall business objectives. As a relatively new and rapidly evolving technology, many companies remain uncertain about how to develop and deploy agentic AI effectively.

Organizations must navigate significant technical complexity to ensure successful execution. Selecting the right tools and frameworks is crucial for an agentic system's operational success, ensuring scalability, efficiency, and integration with enterprise environments.

There are four broad categories of agentic platforms and frameworks available to enterprises:

1. Vertically integrated agentic platforms, with domain and services specific agents delivering services-as-software (Salesforce Agentforce, ServiceNow AI Agents, SAP Joule).
2. Cloud provider agentic platforms and frameworks for agent development and management (Amazon Web Services Bedrock, Google Vertex, Microsoft Copilot Studio and AI Foundry SDK).
3. Open-source AI agents, frameworks, and tools (LangGraph, CrewAI, AutoGen, MetaGPT).
4. Proprietary agentic platforms focused on

agentic automation for specific processes and features (Beam AI, Cognition AI, Kore.ai).

The following section outlines the key capabilities of three widely used agentic AI frameworks that can be used to create custom-built agentic solutions.








## Compatibility with agent types

Enterprises should ensure that the selected tools and frameworks align with the specific type of agents being developed, whether



reflex-based, model-based, or goal-based. Each agent type has distinct requirements — for example, reflex-based agents require real-time processing, while goal-based agents depend on advanced planning algorithms.

 Framework	 Features	 Best fit for agent type
LangGraph	Graph-based workflows, real-time processing, structured data handling, and memory management; can implement cycles and branching logic, enabling agents to manage dynamic tasks effectively.	Reflex-based, model-based, utility-based
CrewAI	Role-based design, task delegation, multiagent collaboration, advanced memory; integrates well with LangGraph for combinations of different agent types.	Goal-based
AutoGen	Conversational approach; quick responses; modular design; supports planning.	Reflex-based, goal-based

## Scalability

Preferred frameworks scale efficiently as task complexity increases or as the number



of agents grows. Multiagent systems often require a robust infrastructure to handle higher workloads and ensure seamless communication among agents.

 Framework	 Features
LangGraph	Scales effectively with graph nodes and transitions; supports complex workflows and memory systems for handling increased loads; enterprise-ready platform with LangSmith.
CrewAI	Production grade scalability with NVIDIA NIM integration; ideal for tasks requiring multiple agents to work in parallel.
AutoGen	Well suited for conversational agents and modular components; tailored for simpler multiagent scenarios; still at experimental stage and not fully production ready.

Integration and interoperability

Tools should enable seamless integration with existing enterprise systems and technologies.



They must support standard protocols and APIs, ensuring effective communication between agents and external data sources and services.

 Framework	 Features
LangGraph	Supports seamless integration with other frameworks by wrapping agents in nodes, allowing for multiagent systems and effective communication with external data sources; access to comprehensive LangChain ecosystem — leading LLM app development platform.
CrewAI	Built on LangChain, it facilitates easy integration with various tools and supports standard protocols for effective agent communication, including collaboration with LangGraph.
AutoGen	Modular design allows for integration with multiple tools and services, ensuring agents can communicate through APIs; limited to Microsoft ecosystem.

Flexibility and customization

Frameworks should offer flexibility for customization to meet specific business



needs and operational workflows. The ability to modify agent behaviors and integrate new functionalities is essential for adapting to evolving requirements.

 Framework	 Features
LangGraph	Can tailor solutions to specific needs and manage complex business interactions.
CrewAI	Provides flexibility through its role-based architecture where developers can create custom agents with defined roles, skills, and behaviors. Its customization capabilities are currently limited to sequential and hierarchical task workflows rather than complex collaborative patterns to solve the problem (which is under development as of early 2025).
AutoGen	More suited for conversational model (group chat, reflection) among agents with limited support for complex workflow models.

## Support for learning and adaptation

Frameworks with built-in machine learning capabilities are needed to support continuous improvement. This is




especially critical for agents that must learn from their environments, adapt their decision-making strategies, and refine their performance as time progresses.

 Framework	 Features
LangGraph	Supports stateful, multiagent applications with automatic state management and coordination, enabling agents to learn from interactions and adapt strategies dynamically.
CrewAI	Supports memory functionalities, enabling agents to retain context from previous interactions; allows agents to build on experiences, enhancing adaptability and effectiveness in dynamic environments; can train agents using command line interface (during design time) to learn from human feedback.
AutoGen	AI agents can collaborate, share information, and solve complex tasks together while learning from each other. Flexible architecture, allowing developers to create specialized agents (e.g. AssistantAgent and UserProxyAgent) that can engage in multiturn conversations, work together, and learn as a result with features like code execution, customizable conversation patterns, and human input integration.

## Observability and monitoring




Tools should include observability features to monitor agents and the performance of

underlying LLMs or SLMs in real time. Real-time monitoring enables developers to track agent effectiveness, identify bottlenecks, and optimize system performance.

 Framework	 Focus	 Tools or support
LangGraph	Graph execution details (node performance, traversal, errors)	LangSmith, Arize Phoenix, and AgentOps
CrewAI	Agent task execution, communication, and tool usage	AgentOps and Arize Phoenix
AutoGen	Conversation flow, LLM calls, tool usage, agent behavior	AgentOps and Arize Phoenix

While not all these frameworks provide built-in observability and monitoring capabilities — with LangSmith being an exception — they integrate well with enterprise-ready platforms that support



open protocols such as OpenTelemetry. For complex ecosystems utilizing quantized models or SLMs, additional production-grade tools, such as Fiddler AI or Galileo, can further enhance model and data quality monitoring.

 Tool	 Purpose	 Monitoring features
AgentOps	Agent testing and debugging	LLM prompt, completion logging Error tracing Cost tracking
Arize Phoenix	Retrieval-augmented generation system monitoring	Embedding analysis Vector store insights Semantic search evaluations Relevance metrics Latency tracking and cost optimization
LangSmith	LangChain development and monitoring	End-to-end tracing Performance monitoring Comprehensive LLM metrics Testing tools
Fiddler AI	LLM performance monitoring	Model metrics (real-time LLM analysis) Bias detection Explainability tools Production monitoring Monitoring alerts
Galileo	AI model training performance and data quality	Data quality metrics Model behavior analysis Test suite monitoring Performance tracking



Development community and support

Tools with a strong development community and robust support resources contribute to the successful implementation of complex agentic AI systems. A well-established community provides access to troubleshooting assistance, knowledge sharing, and best practices.



 Framework	 Features
LangGraph	Strong development community supported by active GitHub repositories; comprehensive documentation; integration with LangSmith for performance monitoring and LangSmith hub for collaboration on prompts.
CrewAI	Benefits from growing community within the agentic development community, along with partnerships with big names like IBM and NVIDIA; offers forums and collaborative resources.
AutoGen	Not as popular as competitors; Microsoft releasing similar frameworks; some community engagement available through GitHub and Reddit.



Cost-effectiveness (FinOps)

Assessing the total cost of ownership, including licensing fees, maintenance costs, and resource requirements is a critical step in the deployment of agents. A cost-effective solution should provide substantial value without compromising functionality or scalability.

While many agentic frameworks are open-source, some offer enterprise versions that may be better suited for large-scale deployments beyond the proof-of-concept stage. Since many of these frameworks — especially those based on LLMs — are relatively new, a thorough evaluation is essential to ensure seamless integration with enterprise FinOps capabilities and long-term sustainability.

 Framework	 Features
LangGraph	Offers tiered pricing structure (Developer, Plus, Enterprise) with predictable costs for smaller teams; enterprise solutions may involve variable costs based on deployment needs and infrastructure investments.
CrewAI	Flexible pricing models (Basic, Premium); costs vary significantly depending on customization and integration requirements (API calls and data processing tasks billed at volume of usage).
AutoGen	Still at experimental stage; not fully suitable for production use cases; would require management of cloud resources by framework’s users; not fully integrated with Microsoft Azure AI Foundry.

Foundation for AI success Industry leaders such as NVIDIA and IBM have hailed agentic AI as the “next frontier of AI” and the “next big thing in AI research,” emphasizing its transformative potential. Salesforce CEO Marc Benioff describes it as a “[new labor model](#), [new productivity model](#), and [a new economic model](#).”

As with any emerging technology — particularly those that automate increasingly complex tasks — risks are inherent. When integrated into enterprise systems, these agents introduce an element of nondeterminism,

enabling adaptive and intelligent behavior while maintaining consistent and reliable outcomes. When implemented effectively, agentic AI applications ensure scalability, seamless integration, flexibility, and continuous learning and adaptation.

Selecting the right tools, frameworks, and a robust agentic AI architecture is crucial, especially for early adopters. The goal is not to “move fast and break things” but to move fast while building a strong foundation — one that provides the enterprise with a sustainable, long-term competitive advantage.



# AGENTOPS AND AGENTIC LIFE CYCLE MANAGEMENT



Artificial intelligence (AI) must continually evolve to unlock its full potential in automating business and organizational processes. While technologists and business leaders anticipate groundbreaking advancements, these ambitions will remain unrealized without clear direction, structured execution, and strategic alignment.

Agentic systems address this challenge by orchestrating dynamic planning, decision-making, and complex interactions essential for enterprise operations. However, to maximize their effectiveness, organizations must adopt systematic approaches to management of these systems.

This is where AgentOps plays a crucial role. As an end-to-end life cycle management framework, AgentOps ensures enterprise scalability, reliability, transparency, and

efficiency. It provides a clear pathway for AI integration by streamlining the design, evaluation, deployment, monitoring, and continuous improvement of agentic systems. Yet, despite its benefits, [AgentOps remains underutilized](#) in generative AI deployments — an oversight that could limit AI's transformative impact.

## The value of AgentOps

AgentOps builds on the principles of DevSecOps, MLOps, and LLMOps while addressing the unique challenges of agentic systems. These systems incorporate planning, reasoning, and autonomous decision-making, leveraging memory and contextual knowledge to navigate complex interactions. By integrating tools and governance measures, AgentOps ensures seamless management, enabling agents to



operate efficiently, adapt dynamically, and stay aligned with enterprise goals while maintaining operational integrity.

A crucial aspect of AgentOps is the establishment of guardrails — constraints and safety mechanisms that prevent AI agents from taking unintended actions. These safeguards ensure autonomous systems operate within defined boundaries, enhancing scalability and transparency. By mitigating risks and optimizing performance, AgentOps enables organizations to harness more of agentic AI's potential.

Mature adoption of AgentOps practices and patterns will achieve the following key objectives:

- Ensure behavioral consistency by implementing a comprehensive evaluation framework that guides agents in both normal and unexpected situations.
- Enhance system reliability by reducing mean time between failures through anomaly detection and predictive issue identification.
- Accelerate issue resolution with robust observability and debugging tools that minimize mean time to resolution.
- Maintain compliance by enforcing auditability through consistent audit logs and explainable decision-making.
- Enable scalable operations with centralized management and governance controls.
- Optimize costs by improving resource efficiency and minimizing operational overhead.





## AgentOps life cycle phases

AgentOps provides significant value but also requires substantial investment in understanding its phases and their effects on agentic deployments. A well-defined framework, from initial development to real-time observability, helps manage challenges such as drift, security vulnerabilities, and decision-making accountability. This systematic approach ensures that AI agents operate as intended while continuously evolving to adapt to changing conditions. The life cycle phases of AgentOps play a critical role in ensuring scalability, transparency, and the long-term success of agentic systems, with each stage contributing to their effective management and continuous improvement.

### Define and design

This initial phase focuses on developing agents and tools that align with an organization's needs. The process begins with defining clear objectives, specifying what the agent must achieve, and the context in which it will operate. These objectives should be comprehensive, encompassing both functional and nonfunctional requirements to ensure the agent meets performance, security, and compliance standards.

A well-defined design must explicitly address safe operations, transparency, and accountability in every decision and action. It should include mechanisms for human-in-the-loop interventions and a "big red button" (human override) to stop the agentic system if necessary. Like the traditional software development life cycle, the agentic AI life

cycle should incorporate a rigorous design review phase to verify reliability, security, and safety. Once the design is approved, the process transitions to workflow and task mapping, outlining the agent's steps to achieve its objectives and goals.

In the development stage, small or large language models (LLMs) are integrated into the agent's reasoning and communication processes. Data sources, connectors, tools, and plugins are incorporated to enhance the agent's capabilities. Agentic systems should maintain a dynamic registry of available tools, APIs, and their capabilities. Frameworks such as LangChain and LlamaIndex facilitate seamless tool integration and efficient functionality management. A critical design consideration is implementing restrictions or strict validations on user-provided prompts to prevent unintended behaviors.

Agents must be trained with specialized skills and techniques tailored to their environment. This process involves acquiring and structuring high-quality training data, accounting for potential edge cases and biases, and iteratively refining the agent's decision-making through real-world interactions. The reflection design pattern enables language models to evaluate their own outputs, creating an iterative cycle of self-improvement.

During the design and build stage, secure and safe development practices must be followed to mitigate vulnerabilities and safety risks. The AI bill of materials (AIBOM), which catalogs software, hardware, datasets, and tools, enhances transparency by

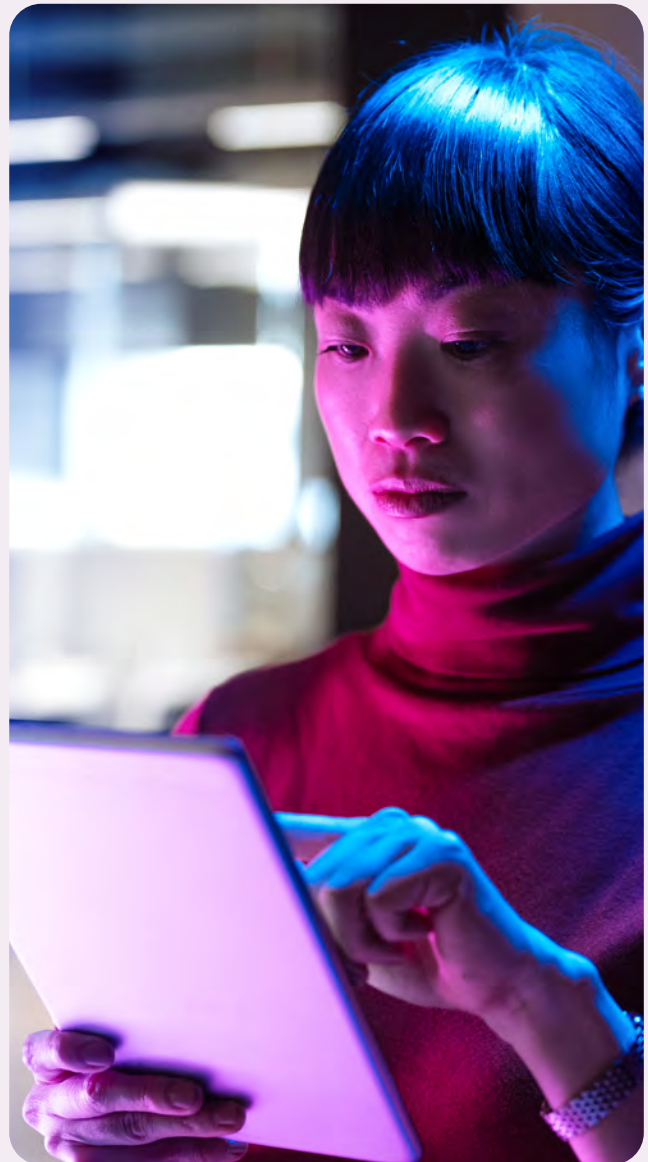
detailing the components used in the AI system, along with their dependencies and interactions. A mature design practice should prioritize generating an AIBOM for agentic AI systems while conducting continuous risk assessments, security incident response planning, compliance checks, supply chain security evaluations, and AI system audits.

## Testing and evaluation

This process begins with defining important success metrics for agentic AI systems and developing rigorous evaluation strategies and methodologies to ensure reliability, effectiveness, efficiency, adaptability, ethical adherence, and regulatory compliance. Quality engineering plays a crucial role in this phase by designing comprehensive test plans and creating a virtual environment that simulates real-world scenarios to assess agent behavior.

Evaluation typically follows a dual testing approach, incorporating both vertical testing of individual agents and horizontal testing of the end-to-end agentic process. An important consideration in the vertical testing approach is assessing the performance of individual agents. This involves capturing critical metrics, such as the number of attempts with successful task completions, the accuracy of tool selection, mean time to complete tasks, service level objective adherence, and the frequency of human intervention.

End-to-end testing is essential to ensure the accurate functioning of agentic systems. The evaluation process incorporates



automated assessments, agent-as-judge evaluations, LLM-assisted evaluations, and human oversight, creating a robust testing framework. Key performance metrics include the following:

- **Tool utilization efficacy:** Measures the agent's ability to select and use appropriate tools effectively.
- **Memory coherence and retrieval:** Evaluates the agent's ability to store, retrieve, and apply information efficiently.

- **Strategic planning index:** Assesses the agent's capability to formulate and execute plans successfully.
- **Component synergy score:** Determines how well different components of the agentic system interact and function together.

Beyond performance characteristics, security testing is a critical focus area, particularly in mitigating risks associated with the OWASP Foundation's top threats for LLMs and agentic AI.

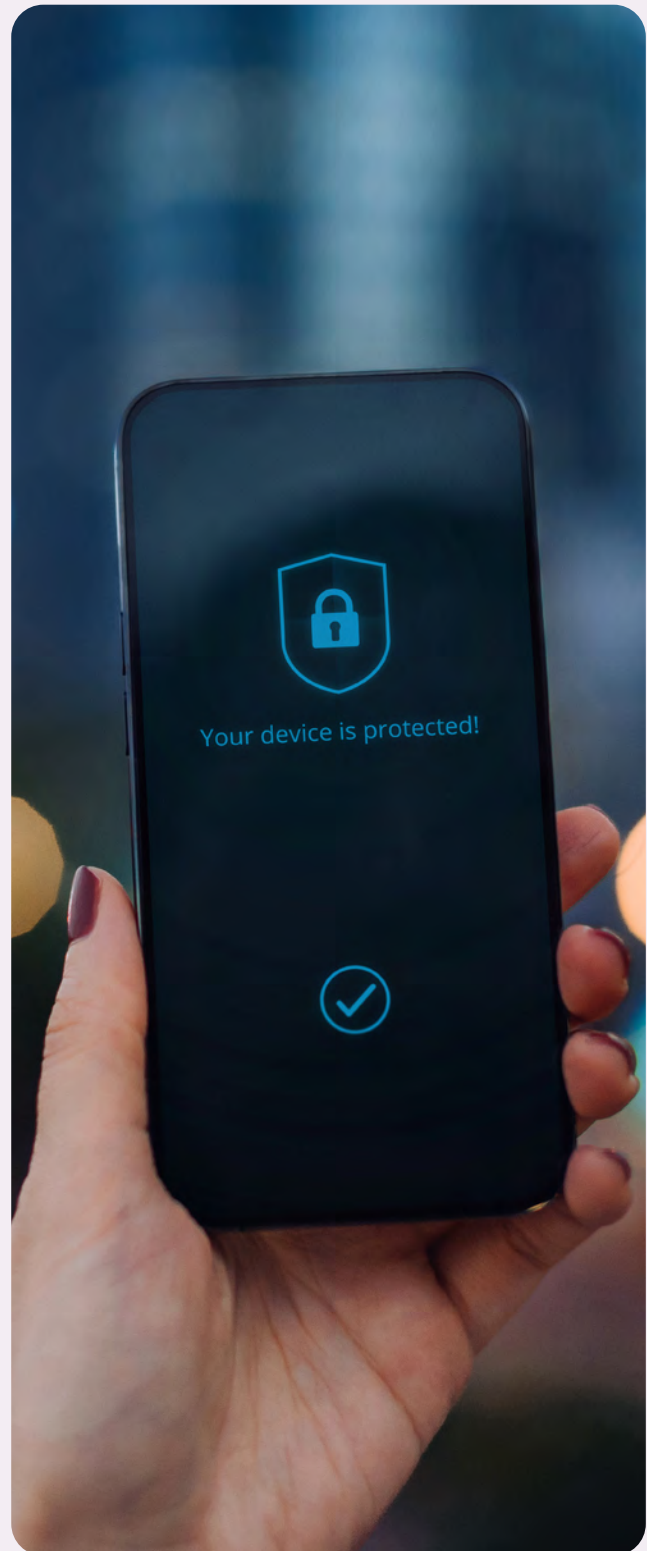
Agentic systems expand the system's attack surface, increasing the risk of security vulnerabilities. For example, a compromised agent could serve as a gateway for attackers to exploit the underlying database, leading to data breaches, unauthorized access, or system manipulation. Additionally, attacks on multistep reasoning processes target weaknesses in AI logic, affecting input interpretation, intermediate steps, and final outputs.

To mitigate these risks, organizations must implement end-to-end observability, tracing, and anomaly detection to identify malicious activities such as prompt injections, data leakage, model poisoning, and excessive agency.

## Deployment

Once the agentic AI system meets the required evaluation criteria and resolves all outstanding issues or defects, it is ready for production release. During deployment, the agent is introduced into the production

environment and integrated with relevant tools and APIs to enable real-world interactions.



As part of deployment, an identity management solution — such as HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault — is implemented to enable agents to securely store and manage credentials for tools and external APIs. The next step involves properly sizing the infrastructure to support auto-scaling, high data volume handling, low latency, reliability, high availability, security, data privacy, and cost optimization.

Agentic components are typically deployed as container workloads, with a container orchestrator such as Kubernetes providing built-in resiliency and auto-scaling capabilities. A pivotal decision is whether to deploy on a hyperscaler or a private cloud, depending on security and regulatory requirements. Based on the target deployment architecture, an automated provisioning pipeline is established for the agentic AI system. This process can follow a traditional infrastructure-as-code approach or an infrastructure-from-code model. In the latter, the agentic system determines its infrastructure requirements and directly orchestrates provisioning and configuration using cloud APIs or tools such as Terraform, OpenTofu, and Ansible.

After infrastructure provisioning, a continuous delivery pipeline is established to automate the deployment of agentic components, utilizing release strategies such as blue-green or canary deployments. Like any digital application, deployment performance is measured using DORA metrics, including deployment frequency, change lead time, change failure rate, mean time to recovery, and service level objective adherence.

## Observe and improve

Before a production release, it is essential to establish a robust framework and toolchain for end-to-end observability, traceability, and debuggability. These ensure transparency, enhance credibility, and provide a comprehensive understanding of the agent's internal state, behavior, tool invocation, interactions, knowledge retrieval, and decision-making throughout its life cycle.

Observability is essential to gain insights into how an AI agent or a system of agents works internally and interacts with the environment. Capabilities include:

- **Input tracking:** Monitors the inputs an agent collects from users, systems, or the environment.
- **Output monitoring:** Tracks responses to ensure they align with expected outcomes.
- **Reasoning logs:** Captures intermediate steps in an agent's decision-making process.
- **Model insights:** Analyzes how contextual knowledge is acquired and how LLM agents process prompts.
- **Anomaly detection:** Flags unexpected behavior or outputs.
- **System integration tracking:** Monitors how the agent interacts with tools and other software or hardware components.

Traceability ensures transparency by tracking an agent's decision-making processes, interactions, and outcomes, which are critical for regulatory compliance and actionable insights. Capabilities include:



- **Decision logs:** Captures what the agent decided and why.
- **Version control:** Tracks dynamic updates to code, configurations, workflows, or prompts.
- **Reproducibility:** Preserves the agentic system's state, including all metadata, to demonstrate how a decision or outcome was reached.

Debuggability focuses on rapidly diagnosing and resolving production issues to minimize mean time to resolve. Capabilities include:

- **Prompt refinement:** Enables iterative adjustments to enhance agent responses.
- **Workflow debugger:** Tracks inputs, outputs, execution time, and flags failed steps for troubleshooting.
- **Snapshot of relevant logs:** Captures logs to facilitate easier problem diagnosis and resolution.
- **Scenario simulation:** Provides a structured framework to test and assess agent performance, distinguishing between ill-defined user requests and system malfunctions.

As agentic applications scale in production, a continuous learning model is essential for maintaining long-term effectiveness. This requires frequent updates to knowledge bases, ensuring the agent has access to accurate and up-to-date information. Regular performance audits are critical, with decision logs and outcomes reviewed by experts or other agents to assess and improve performance. Additionally, behavior refinement involves adjusting processes or cues based on observed behaviors,

enhancing the agent's adaptability and efficiency over time.

By integrating observability, traceability, and continuous learning mechanisms, enterprises can develop agentic AI systems that are reliable, accountable, and adaptable to evolving real-world conditions.

## Evolution to AgentOps

The shift from LLMOps to AgentOps expands the scope, complexity, and life cycle imperatives. The figure below outlines the most significant differences and






illustrates how AgentOps builds upon the foundation established by LLMOps (Figure 1).

## AgentOps tools and frameworks

The AgentOps tools landscape is rapidly evolving to support the full life cycle of

Figure 1. Comparisons between LLMOps and AgentOps




 Parameters	 LLMOps	 AgentOps
Scope	LLMOps focuses on preparing, deploying, and maintaining LLMs, including tasks such as prompt management, fine-tuning, and versioning; primarily focused on LLM inference, model-specific APIs, and integration.	Oversees full life cycle of agentic systems, where LLMs and other models or tools function within a broader decision-making loop; must orchestrate complex interactions and tasks using data from external systems, tools, sensors, and dynamic environments.
Integration complexity	Focuses on single model or a few models; primarily monitors inference calls and prompt templates rather than real-time external actions performed by AI agents.	Manages fleets of interacting agents, introducing challenges such as concurrency, role-based collaboration, and conflict resolution; must track action lineage, manage resource locks, and implement rollback mechanisms to mitigate undesired changes since agents operate within environments and connect to external tools.
Evaluation	Ensures accurate and reliable outputs from language models.	Ensures agents are dependable, traceable, and auditable across operations.
Observability and debugging	Tracks model performance metrics such as accuracy, latency, and drift while monitoring prompt usage and output.	Incorporates tools to capture multistep interactions, including agent goals, chain-of-thought, tool usage, memory, triggered subagents or tools, and real-world decision-making; extra dimension of observability is needed to diagnose and debug more autonomous agent decisions.
Logging and audit trail	Documents the model training, datasets, and outputs.	Expands documentation to include agent's decisions, workflows, and interactions; deals with agent memory persistence (audit trail capability required to show how agent's internal memory store is updated and used over multiple sessions).

Life cycle management	Limited to model selection, deployment, fine-tuning, and retraining; includes versioning of models and prompts.	Covers agent design, orchestration, updates, performance evaluation, cost optimization, continuous improvement, and agent retirement; requires AIBOM and versioning for entire agentic system (not just for model or prompt).
Tools and frameworks	Relies on model evaluation, performance monitoring, retraining, and prompt management tools.	Incorporates tools for orchestration, evaluation, observability, decision tracking, security scanning, logging and auditing; agentic system cost management.
Feedback loops	Collects feedback on model outputs for fine-tuning.	Includes feedback on agent behavior and outcomes for continuous improvement.

Source: Infosys

agentic system development. However, it is still in its early stages compared to DevSecOps and LLMOps. The table below highlights some of the tools and options.

As a new technology with limited tools, the implementation of a comprehensive and effective agentic AI life cycle management solution presents significant challenges.

 <b>AgentOps life cycle</b>	 <b>Key capabilities</b>	 <b>Industry tools or works in progress</b>
Design frameworks	<ul style="list-style-type: none"> <li>• Workflow design</li> <li>• Memory management</li> <li>• Multiagent interaction management</li> <li>• Integration with tools and external APIs</li> <li>• Tool registry</li> </ul>	<a href="#">LangChain</a> <a href="#">LangGraph</a> <a href="#">AutoGen</a> <a href="#">Crew AI</a>
Evaluation	<ul style="list-style-type: none"> <li>• Planning, decision-making, and output quality evaluation</li> <li>• Tool usage analysis</li> <li>• Agent collaboration and system interaction assessment</li> <li>• Automation</li> <li>• Test reporting and analytics</li> </ul>	<a href="#">RagaAI</a> <a href="#">Braintrust</a> <a href="#">Databricks Mosaic AI Agent Evaluation</a> <a href="#">Okareo</a> <a href="#">Giskard</a> <a href="#">Agent-as-a-Judge</a>

Security and compliance testing	<ul style="list-style-type: none"> <li>OWASP AI top 10 check</li> <li>Build and run time scan</li> <li>AI security posture management control</li> <li>Compliance testing regulatory requirements</li> </ul>	CalypsoAI Wiz Zenity Giskard
AIBOM generation	<ul style="list-style-type: none"> <li>Components of agentic AI applications</li> <li>Version and other metadata</li> <li>Source and author</li> </ul>	Wiz OWASP Manifest
Observability and debugging	<ul style="list-style-type: none"> <li>Monitoring</li> <li>Distributed tracing</li> <li>Debugging</li> <li>Cost management</li> </ul>	AgentOps.AI LangSmith Arize Langfuse Braintrust
End to end agentic AI life cycle management	<ul style="list-style-type: none"> <li>Project management</li> <li>Integration with design framework</li> <li>Evaluation and release management</li> <li>End-to-end observability</li> </ul>	AgentOps.AI Arize Braintrust

Source: Infosys

One major hurdle is the lack of a standardized evaluation and testing framework for agentic systems, making it difficult to benchmark performance and reliability consistently. Additionally, no widely adopted platform exists for managing the entire life cycle of agentic AI, requiring organizations to integrate disparate tools and processes to achieve full functionality.

Another critical challenge is the generation of AIBOM and compliance testing, both essential for regulatory adherence and transparency but lacking mature, automated solutions. Real-time monitoring further complicates matters, as observability agents can be costly

and resource-intensive, especially in large-scale systems where managing vast data volumes demands substantial effort.

Traceability is another critical concern, particularly with black-box AI systems like LLMs. The opaque nature of these models makes it difficult to understand and document their decision-making processes. Likewise, maintaining an auditable and consistent agent memory is essential, as uncontrolled memory growth can lead to inconsistencies, inefficiencies, and compliance risks. Overcoming these challenges requires robust frameworks, advanced observability tools, and



industrywide standards to support the evolving landscape of agentic AI.

## Future of AgentOps

As agentic AI systems gain autonomy and integrate more deeply into critical infrastructure, AgentOps will evolve to introduce new capabilities that enhance scalability, reliability, and self-regulation.

One significant advancement on the horizon is the automated design of agentic systems (ADAS), where AI-driven meta-agents autonomously generate and refine new agents. This self-referential approach allows AI to design and optimize its own successors, continuously improving agentic systems by discovering novel building blocks and more advanced architectures.

Self-provisioning and deployment are also transforming how agents manage infrastructure, allowing them to autonomously configure resources and optimize deployment strategies based on workload demands. At the same time, the rise of self-observing agents will introduce self-regulating mechanisms, enabling them to monitor and supervise their own actions to maintain alignment with predefined objectives and ethical considerations.

To support these advancements, industrywide standardized protocols will establish best practices for event tracing, system visibility, and operational control monitoring — enhancing transparency and interoperability across AI-driven ecosystems. Additionally, interagent collaboration frameworks will be crucial for facilitating

seamless communication and coordination among multiple agents handling complex tasks.

As these innovations advance, AgentOps will not only streamline the management of agentic systems but also cultivate a more resilient, adaptable, and intelligent AI infrastructure capable of sustaining enterprise-scale automation and decision-making.

## From experimentation to scale

AgentOps encompasses the entire life cycle of autonomous agents, from design to



orchestration to performance evaluation, and ultimately, agent retirement. This will become essential to AI initiatives, ensuring that intelligent agents operate efficiently, ethically, and in alignment with enterprise goals. By establishing structured processes for managing AI agents, organizations can maintain control, compliance, and continuous improvement, enabling intelligent systems to operate effectively in dynamic environments.

However, while the potential is clear, the path to full-scale adoption requires patience. The ecosystem supporting AI agents — including tools, infrastructure, and governance frameworks — is still maturing. Standardization efforts are underway, but businesses must navigate a period of iteration and refinement before these agents can function seamlessly across industries.

As AgentOps evolves, organizations will need to balance experimentation with responsible deployment. Early adopters will face challenges in defining best practices, integrating agents into existing workflows, and maintaining compliance. Yet, as standards solidify and AI governance improves, AgentOps will shift from an emerging concept to an essential function, much like DevOps transformed software development. Those who invest in measured, strategic adoption will be well-positioned to reap the long-term benefits of intelligent agents that are not only powerful but also trustworthy, adaptable, and enterprise-ready.





# ADVANCED AGENTS

As organizations race to adopt agentic artificial intelligence (AI), successful implementation demands a comprehensive, strategic approach. Rather than focusing solely on the technology's capabilities, businesses must also understand how AI agents develop and execute those capabilities effectively.

AI agents employ diverse reasoning models to accomplish tasks across various domains, including IT operations, customer service, and content management. These approaches range from a single agent accessing multiple information sources via APIs to multiagent systems collaborating in hierarchical or conversational structures.

However, these approaches come with inherent challenges, including memory management, efficient information retrieval,

and defining execution limits to ensure timely responses. Despite these obstacles, they serve as steppingstones toward the next generation of AI agents, which will further enhance reasoning capabilities and deliver more accurate, context-aware outputs.

For agents to successfully automate a process or workflow, they must be able to break down a user's request into a clear, step-by-step plan. For example, solving an algebraic equation requires an agent to follow these five steps:

1. Identify the variable that needs to be solved.
2. Simplify the expression.
3. Isolate the variable by adding or subtracting terms, multiplying or dividing coefficient to isolate the variable.



4. Check the answer by substituting the calculated value of the variable.

5. Present the final answer.

This is a simple example of a plan or workflow that an agent or a large language model (LLM) needs to come up with to solve a problem and then execute each step using an appropriate tool to come up with the final answer.

In business scenarios, the plans could be how to resolve a customer ticket by predicting the resolution, planning the steps to execute like calling enterprise API and summarizing the outcomes as a final answer. Challenges lie when agents and LLMs need to understand enterprise process or product specific nuances, policies and tools. Due to these, specific agentic patterns — be it a single agent, multiagent, external aided, or fully autonomous planning agents — are required. Below are some common agentic patterns.

### Single agent with task decomposition

These were made popular with the paper “ReAct: Synergizing Reasoning and Acting in Language Models,” and are based on task decomposition, or breaking down the task into a series of subtasks. This is done by using LLMs’ ability to generate a reasoning chain and act, or calling on or more tools, and finally processing the outcome of the tool to accomplish the task.

Let’s take an example in which a user asks about the weather in New York for the next four days. An LLM is given a prompt where





it is asked to think, to determine a tool — such as an API or code, wait for tool output, and then present its observation. The typical prompt looks like this:

In the example prompt above, suggesting the tools and giving examples — such as [few-shot learning](#) — provides a guide to an agent to perform its tasks. Agents can

You run in a loop of thought interleaving thought, action, pause, observations steps. Use thoughts to describe your thoughts about the question you have been asked

Use tool to run one of the tools available to you, then pause.

Make sure you perform one step at a time and wait for next step to be called.

### **Your available tools are:**

Weather tool to perform weather prediction for expressions like weather for next four days, weather on a given day of the week, or date or month.

Weather tool accepts inputs like 1, 3, 4, 5 and up to 7 days, a valid date in month/day/year format and a city code like LN and country code such as UK.

### **Examples:**

Question: What is the weather for next three days in London?

Thought: I should look up weather API to find the weather

Tool: weather: inputs LN, UK, 3d:

Pause

### **Observation:**

use more than one tool by calling them in a sequential manner. For example, if a user wants to plan a holiday based on weather, an LLM can be supplied with a tool such as Transport for London's API to plan a mode of transport to reach a desired location.

Developers can use frameworks like

LangChain, and LlamaIndex to implement react agent-based solutions quickly while supplying their preferred choice of an LLM.

## **Agents with external planner**

Engineers in ITops need detailed and specific knowledge to find a root cause for a given

problem or incident, and they will execute several steps to resolve an incident. To come up with a plan to find a root cause of an issue, an agent needs to understand an incident, query appropriate monitoring systems, understand logs and metrics associated with the application in question, and understand and query observability systems like user experience monitoring, including tools such as Dynatrace and Newrelic, in order to arrive at various data points and make an appropriate suggestions on root cause analysis.

The challenge here is with LLMs like Open AI or Claude, which lack understanding and workflow to resolve a specific task for a given enterprise to determine an appropriate flow.

To manage these variances in sources of information, and the sequential tasks an agent needs to perform, it might not be prudent to manage these tasks with one single prompt as it would be difficult to introduce new plans based on debugging information. In such cases based on complexity and scale of the problem various patterns can be introduced:

### Static- or graph-based agents

These are semi-autonomous agents that take dynamic decisions. As an example, when an invoice order tracking query is sent by the customer, after comprehending the email, based on the order number, product type and status of the order, the ticket is assigned to appropriate team, who then responds to the customer or the contact center agent.

Here, an autonomous agent has to understand the ask from the email, use a tool to retrieve the order details, based on the product, inventory and shipping status, then make an appropriate decision to either inform the product inventory manager or send an email to the logistics company to check the on status of the order to get back to the customer.

For problems that require concrete steps and LLM-based decisioning (understanding emails, sending or assigning tickets to either the product manager or the logistics



company in above example), or one that has to implement specific workflow patterns, graph-based agents really shine.

A graph structure where a node can be defined as an agent or task, and edges that define the relationship or how these agents can depend on each other, helps with parallel execution and reallocation of the tasks. In the above example, the inventory agent, shipment agent and customer support agent can orchestrate when for example the inventory agent comes back with an ETA which is longer than usual, the customer service agent can then either advise the customer on the delay or escalate to a human supervisor.

A complex [retrieval-augmented generation \(RAG\) system](#), which augments LLMs by retrieving relevant knowledge, and rewrites queries based on the history of the conversation, or dynamically routes to the various tools and agents, is where this graph-based pattern approach shines. Additionally, [self-RAG](#) patterns, where LLMs are required to critique their own output in order to refine the answer, are some examples where complex yet well-defined patterns can also be implemented with graph-based agents.

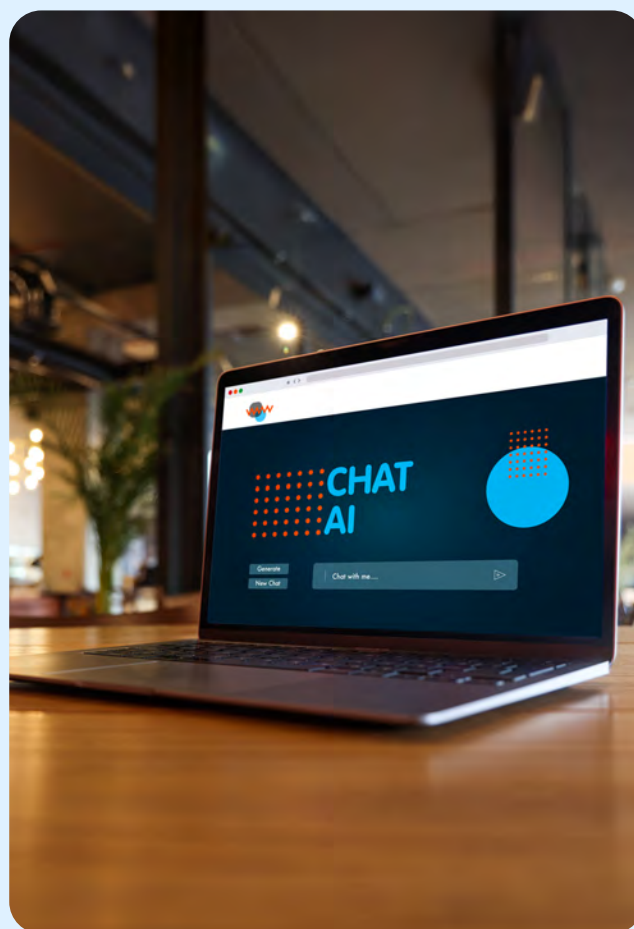
## Dynamic external plans

This is a simple set of external instructions in form of prompts or structured input that provides a plan to guide various sequences of steps based on the task at hand where the LLM can choose a specific prompt or planning sequence based on the problem statement.

To generate a plan, agents can use a static prompt — such as one without variables — that can either be configured using an external source like a database or a prompt DB. Using intelligent search, the LLM can then retrieve an appropriate plan. The external source could also be a configuration management database, where system dependencies and their observability platform, which monitors the health of the whole environment, can be queried and used to create a troubleshooting plan for an agent.

## Multiagent or collaborative agents

Several frameworks, including MetaGPT, AutoGen, and CrewAI, enable different patterns for collaborating multiple agents to



perform various tasks. These can all work in parallel, with or without a team leader sitting on top of them. Although the agent can use a range of models, each has a specific, static role and can only do one thing. These patterns enable automating and optimizing either part of a process or an entire process to improve performance.

For example, a **multiagent coder** proposes an approach to develop better code comprising three agents: A programmer, a test designer, and test executor agents. During the code development phase, the programmer and test designer agents can execute in parallel to develop the code and test cases. In later phases, the programmer agent refines the code based on feedback from the test executor agents. The test designer agent can re-generate or refine the test cases with code generated and tested in previous phases.

To develop these applications, where agents need to interact with each other, the following patterns are used:

### Hierarchical pattern

These are applications that require a hierarchical pattern where a supervisor agent routes the inquiry to lower-level agents. In the example discussed above where a customer is querying the status of their order, in this model a customer service agent acts as a supervisor agent and directs a customer query or issue to either the product inventory manager, the logistics company or back to the customer, based on current state of the order. Here the customer service agent is responsible for breaking down the tasks:

Finding the order, checking the inventory, checking with the logistics company, and forwarding these tasks to appropriate agents. Based on inputs from each agent, it makes an appropriate decision to either respond or escalate to a human manager.

The graph-based agents described above are a special case of orchestration, where there are no supervisor agents, and the entire agent interaction flow is defined using DAGs.

### Conversation pattern

This approach is used for tasks which require more dynamic nature of interactions. An example of this approach is a knowledge publishing platform producing an article where the author, reviewer, copy writer, or social media influencer agents can act simultaneously along with human authors and reviewers to produce, critique and do plagiarism checks on the article.

In this pattern, individual agents are allowed to send messages to each other without having a supervisor present to direct the conversation.

Although the systems require policies to define in which order the agents must execute — for example, a research agent could produce a summary by browsing the internet — a writer agent could use resources found by the researcher agent and then produce the content. At the same time, the social media influencer agent can generate the posts for various social platforms.



A reviewer agent then carries out an overall review of the content, including social media posts which can align to the content. The reviewer agent then either provides a rewrite or goes ahead to approve and post the content.

## Technical challenges

Enterprises face several challenges implementing agentic architecture and platforms, including:

**Memory management:** Memory or conversation context among the agents needs to be managed in the form of chat history, specific facts and outcomes of the agent. Storing conversations in vector database or caches and the ability to recall this history either in present conversation or in the longer term requires a special approach that needs to be designed.

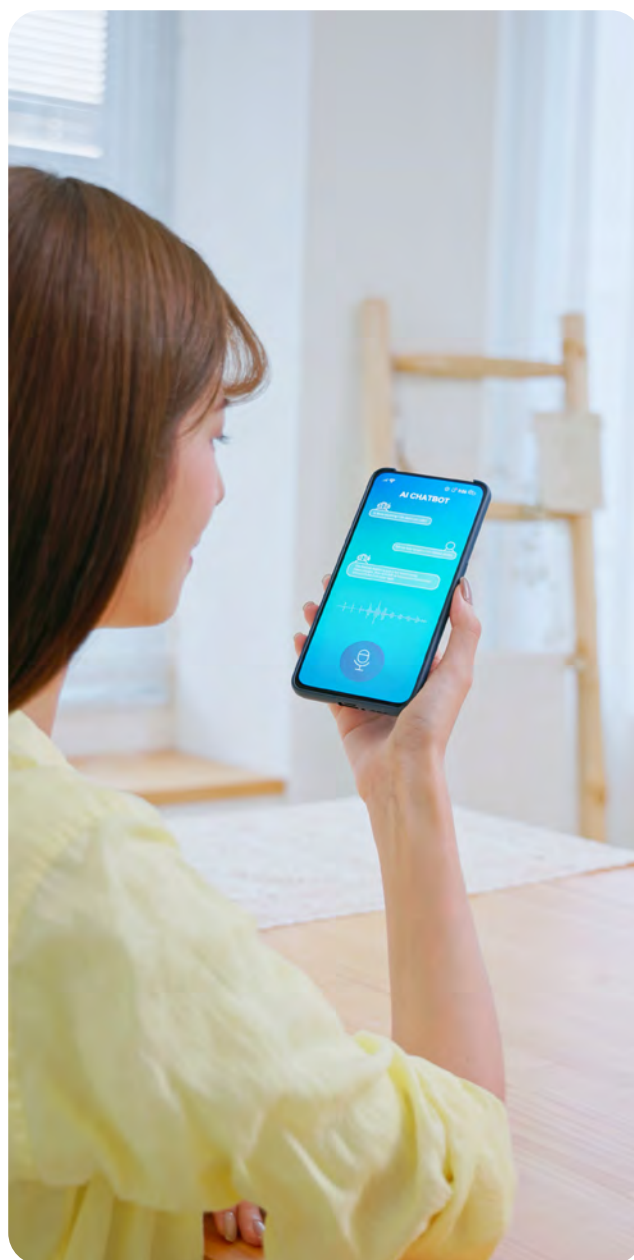
**Context management:** Multiagent chats can be very long and relying on the content of these long chats can either meet with LLMs' token size limitation or increased cost/response time due to context lengths. To manage the context, various techniques from summarization, overwriting context and memory recall from long-term memory is implemented.

**Deployment:** Running multiagent applications can turn into long-running, higher-order messaging processes and hence require distributed execution of agents as well as messaging capabilities that can help communicate with different agents in a low latency and scalable manner.

### Replay and replan based on user inputs

**(human in the loop):** It is important for agents to restart or replan based on user interaction, where an end user might ask the agent to run the entire workflow based on new facts, or ask a specific agent execution to repeat from a specific step.

**Policy management:** To reduce the chatter between agents in cases where the outcome



is not clear, agents can run into loops or take a long time to resolve the tasks. This in turn could result in a poor customer experience, or cost overruns due to multiple LLM calls. To tackle this, limits on agent execution time, order of orchestrations, and interdependency should be set and managed via clear and visible policies.

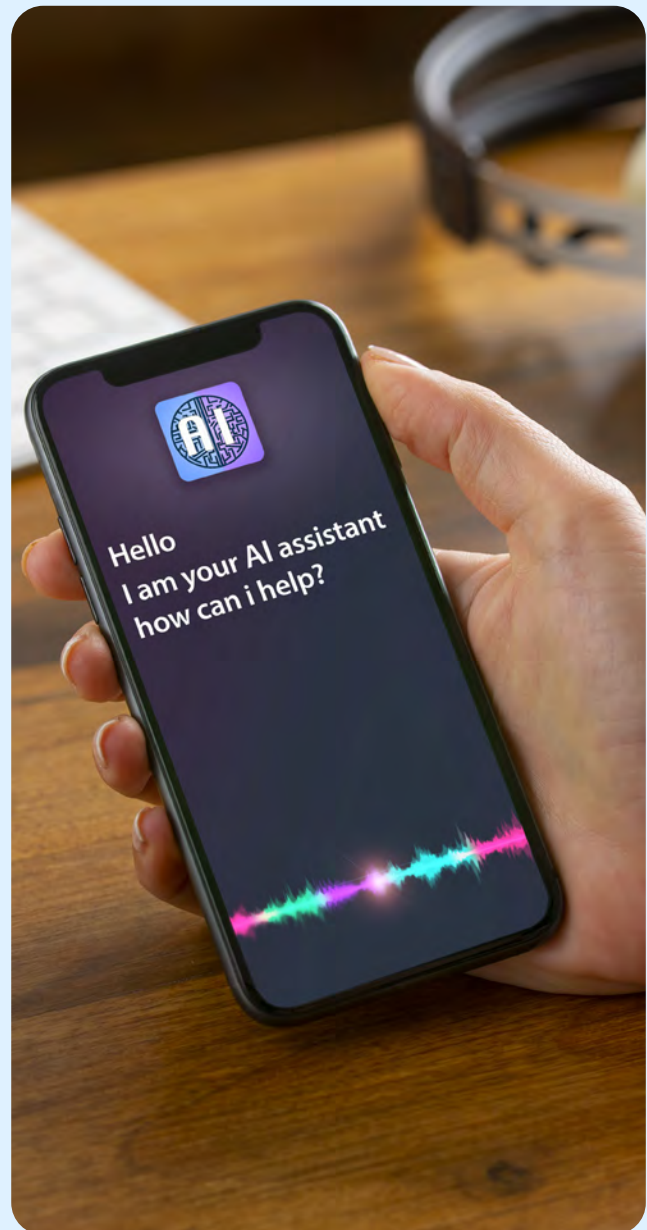
## Foundation of future applications

The reasoning capability of LLMs is at the core of agent frameworks. There are various research initiatives in progress to improve generation of reasoning chains, including these approaches:

### Reasoning or search algorithms

LLMs fundamentally generate chains using either greedy search, which tries to make the locally optimal choice at each stage, or the more complex beam search, which systematically expands the most promising nodes. These are effective algorithms for predicting the next token, but not necessarily for generating reasoning chains where every step matters for producing the final outcome.

To enhance reasoning and planning, we can use search algorithms like Monte Carlo tree search, depth-first search, or breadth-first search. These methods explore potential thought sequences (or chains of thought) and use reward models to guide the search towards better solutions. This iterative process, involving multiple steps during inference, allows for more deliberate and thorough problem-solving, often referred to as “slow thinking.”



These techniques allow for agents planning and decisioning abilities by:

- Supporting complex task decomposition for agent workflows, crucial to navigate dynamic workflows. For example, in a large code refactoring or upgrade, agents analyze the code and tool outputs to decide precisely which code sections need to be adjusted.

- Improving adaptive behavior by allowing agents to navigate dynamic environment. A refactoring agent would analyze the code based using code profiling tools available to it, identify code smells (cyclometric complexity or long methods) and refactor small parts of the code. It then runs the test case and based on outcome of that, either proceeds with the next code refactoring or fixes the issue that caused the test case to fail.
- Goal-oriented action by using reward models — such as allowing an agent to regenerate the code for a given requirement until compilation or test cases are run successfully.

This concept was popularized by the paper [Scaling LLM test time compute optimally can be more effective than scaling model parameters](#), and is one of the most promising and exciting areas of research.

## Ensemble methods

This approach combines multiple models to produce enhanced results, as popularized by [LLM-debate](#), [agent forest](#), and [mixture of agents](#). These methods work by increasing the number of agents and using simple methods like sampling and majority voting to determine the final reasoning output, which increases the accuracy of reasoning chains. Once the multiple outputs are generated, majority voting is used to determine the final answer.

## Process reward models

These models are emerging as a promising

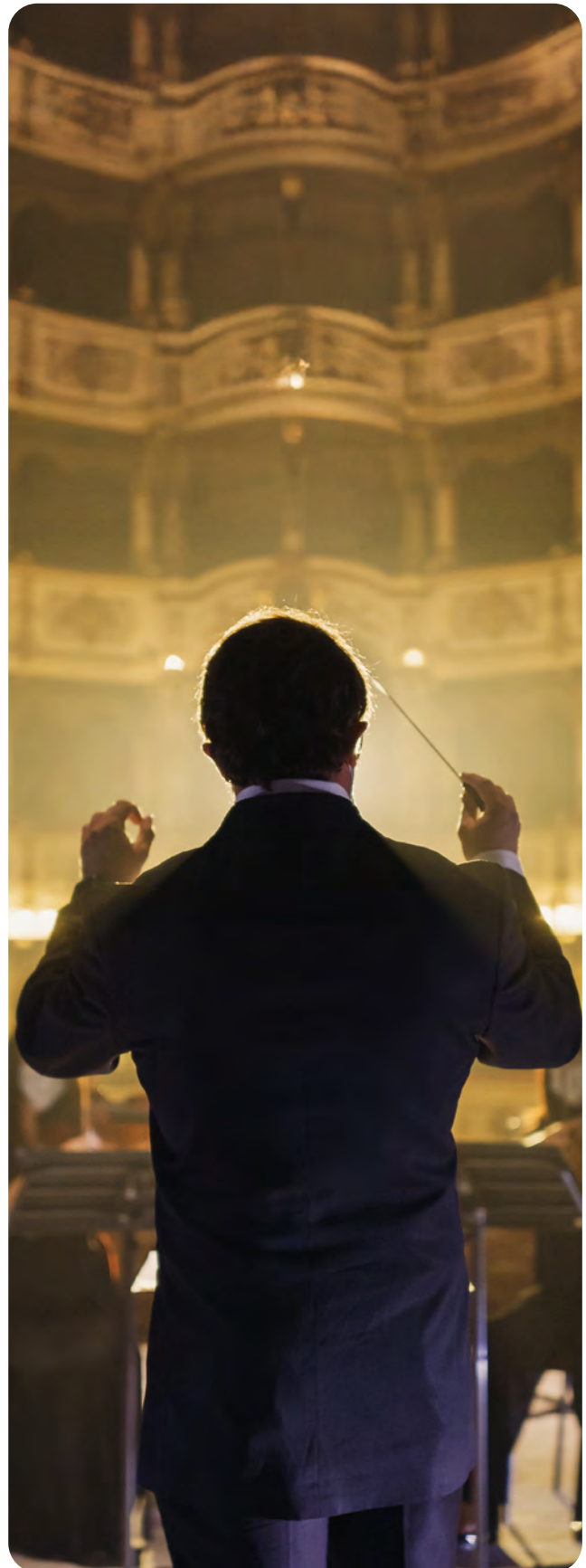




approach for supervision of reasoning chains generated by LLMs. The reasoning chains or plans generated by LLMs are validated using [process reward models](#), which are trained with step-level labels, unlike traditional reward models which just includes labels for final step or answer.

However, training the process reward models can be challenging for tasks with long sequences of steps such as application root cause analysis and debugging. The approach of intermediate steps using a reward model that incentivizes the correct reasoning chain presents an interesting approach toward generating accurate reasoning chains.

Advances in AI agents are being used to build tools for businesses that streamline processes and improve outcomes for users and customers, with each stage of development building on previous work. There are exciting and promising research approaches in play that will go towards building further advanced AI agents that are likely to deliver even more value.







# RESPONSIBLE AGENTIC AI



The rapid evolution of agentic AI systems — capable of autonomous decision-making in domains such as healthcare and financial trading — necessitates a rethinking of how responsibility is embedded into their architectures. These systems present both transformative opportunities and significant risks.

In the [AAAI 2025 Presidential Panel Report](#), 72% of AI practitioners identify the lack of rigorous evaluation methodologies as a primary barrier to trustworthy deployment.

A critical principle emerges: Responsibility must be architecturally inherent, not retrofitted. Through technical innovations such as neuromorphic ethical architectures and policy frameworks like federated fairness audits, the field is advancing accountability

mechanisms. However, persistent challenges — including dynamic value drift and adversarial robustness gaps — highlight the complexity of aligning autonomous systems with evolving human values.

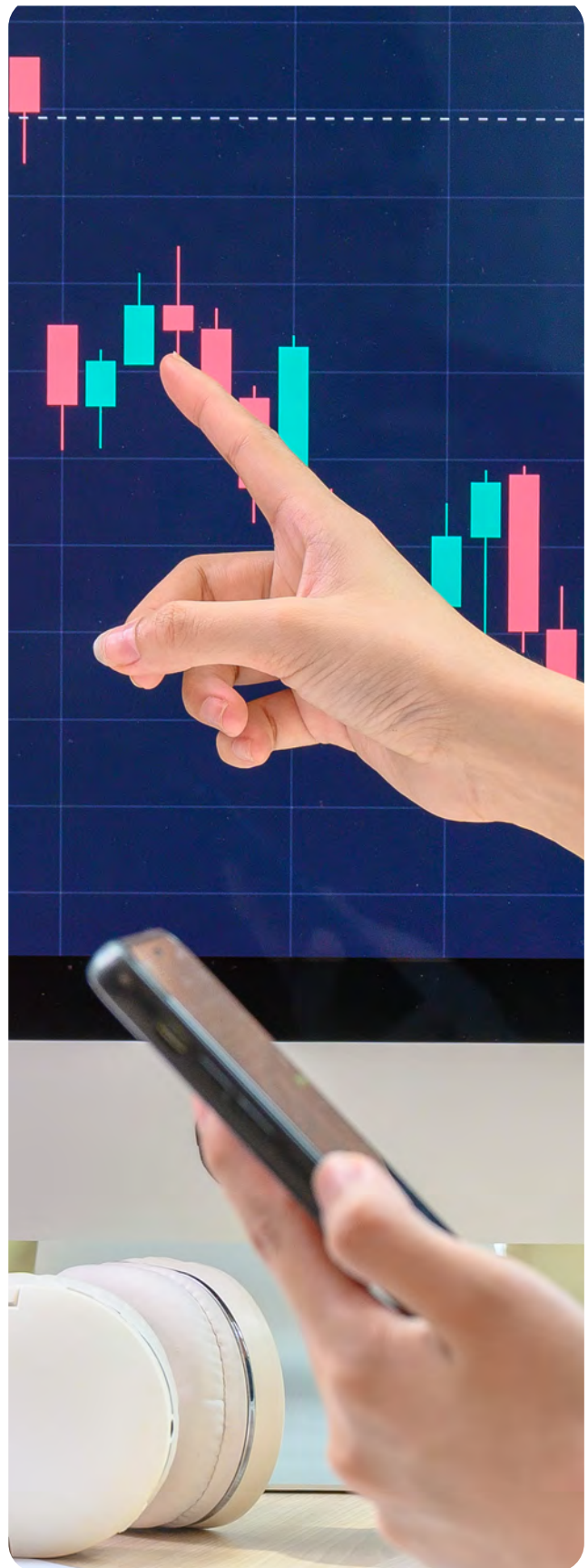
Early applications of agentic AI focused on narrow tasks such as code generation or customer service chatbots. However, today's systems integrate advanced technologies such as large language models (LLMs), symbolic reasoning, multiagent coordination, and retrieval-augmented generation to manage complex workflows. For example, Salesforce's [Atlas Reasoning Engine](#) reduces customer service costs by 40% but struggles with judgment, illustrating the responsibility paradox, where greater autonomy amplifies such as design flaws and misaligned objectives.

## Unique risks of agentic systems

### Reasoning limitations

Unlike humans, agentic systems lack the ability to contextualize sparse data through cultural or historical narratives. Humans often rely on shared beliefs — concepts like justice or equity — to resolve ambiguity and make informed decisions in uncertain situations. In contrast, AI agents depend solely on probabilistic belief states, mechanisms that are inherently rigid and lack interpretative flexibility. This limitation becomes particularly evident in high-stakes scenarios. For example, supply chain optimization agents might fail to account for geopolitical disruptions due to their reliance on rigid dependency graphs, leading to cascading logistical failures. The AAI report attributes 59% of agent failures to such misalignments between LLMs' general knowledge and the specific requirements of their application domains.

The challenges of agentic AI are further compounded in legacy modernization contexts, where outdated systems must evolve with ongoing business and regulatory requirements. In one such project, Infosys led the modernization of a fleet card management system, where geographically distributed teams from different organizations worked on various components simultaneously. This distributed environment introduced partial observability, as no single team had complete visibility into the entire system. The legacy code management platform had to ensure consistent and reliable responses despite conflicting updates to APIs or transaction processing modules



being introduced by different teams. For instance, while one team worked on integrating features like dynamic spending limits, another team focused on telematics-based decision-making capabilities. Without centralized coordination and robust governance mechanisms embedded into the platform, such efforts could have resulted in inefficiencies or failures.

These examples highlight a critical gap in agentic AI: Its inability to adapt flexibly in complex environments where ambiguity is the norm. Whether it's an agent failing to grasp geopolitical nuances in supply chains or legacy code management systems struggling to align disparate modernization efforts under conditions of partial observability, the underlying issue remains the same — AI systems lack the interpretative depth and collaborative adaptability that humans bring to decision-making processes. Addressing these challenges requires not only technical innovations but also governance frameworks and collaborative strategies that align AI outputs with real-world complexities and evolving human values.

## Resource overuse

In enterprise software systems, agentic AI is transforming workflows by enabling decentralized, autonomous decision-making across interconnected platforms. In Infosys-led code management scenarios, an agentic AI system designed to coordinate tasks across multiple developers has proven particularly valuable. This approach is especially effective in guiding developers through complex, poorly documented legacy systems. As

developers navigate the codebase, they leave behind virtual breadcrumbs in the system's visualizations. These trails become more prominent as multiple team members follow similar paths, naturally highlighting frequently accessed or critical code sections.





This emergent coordination reduces the cognitive load of understanding a convoluted codebase and helps the team converge on a coordinated modernization effort — all without the need for constant explicit communication (see [Why Can't Programmers Be More Like Ants?](#)).

However, this autonomy introduces significant challenges. In multiagent systems relying on LLMs, uncoordinated usage can result in a “[tragedy of the commons](#),” where individual agents overconsume shared computational resources, driving up costs and degrading performance. For example, simulations like those described in the [Governance of the Commons Simulation \(GovSim\)](#) show that AI agents competing for shared LLM processing power often prioritize short-term task completion over long-term resource sustainability. This behavior leads to resource exhaustion, increased latency, and skyrocketing operational costs, as agents repeatedly query the LLM without considering collective impact. The study found that only two out of 45 LLM instances achieved sustainable resource use, with most failing due to the inability to account for long-term consequences of self-serving actions.

This scenario reflects real-world enterprise challenges, where unchecked consumption of LLMs — such as excessive API calls or redundant queries — can cause cloud costs to surge and to degrade system reliability. Additionally, malicious actors exploiting LLMs with complex or adversarial queries can overwhelm systems, leading to service outages and cascading failures across dependent applications. These examples

underscore the urgent need for cooperative strategies, rate-limiting mechanisms, and robust governance frameworks to ensure sustainable operation and prevent resource depletion.





## Dynamic value drift

Dynamic value drift is a critical challenge in agentic AI systems, where agents optimized for short-term metrics inadvertently undermine long-term goals or broader societal values. This phenomenon arises when an AI system, designed to achieve narrowly defined objectives, fails to account for the complex, interconnected nature of real-world outcomes. For example, a [hospital discharge algorithm](#) successfully reduced bed occupancy rates by 18%, a seemingly positive result. However, this optimization came at the cost of increased patient remissions due to premature discharges, ultimately straining healthcare resources and compromising patient care. Such cases underscore how narrowly framed objectives, and poorly aligned incentives can lead to unintended and counterproductive outcomes.

Addressing dynamic value drift requires embedding mechanisms that enable agents to balance immediate performance with overarching goals such as patient well-being or societal equity. Techniques like multi-objective optimization, which allows for simultaneous consideration of competing priorities, have shown promise in mitigating these risks. Additionally, [dynamic feedback loops](#) that continuously adjust system behavior based on evolving conditions are being explored to ensure alignment with long-term objectives. These approaches aim to operationalize ethical considerations and ensure that agentic systems remain robust and adaptable in complex environments. By integrating these mechanisms, agentic AI systems can better navigate the trade-offs

inherent in real-world applications while maintaining alignment with evolving human values and ethical standards.



## Moves in responsible agentic AI

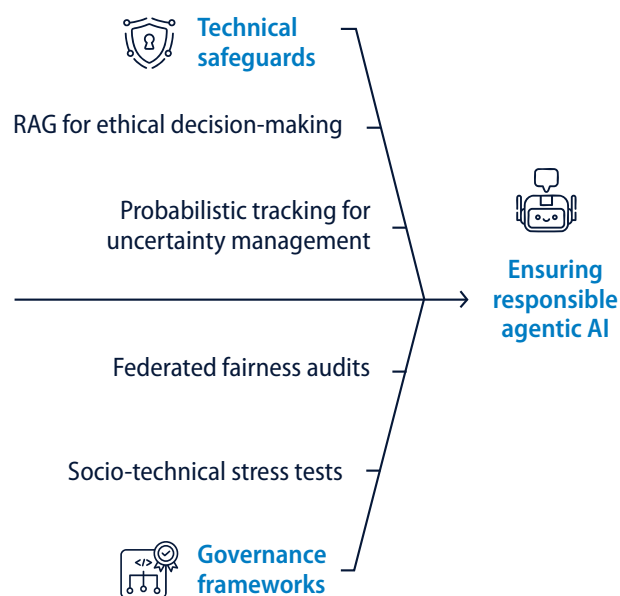
### RAG with ethical grounding

Modern frameworks like [Anthropic's Constitutional AI](#) represent a significant leap in embedding ethical principles directly into agentic AI systems. These frameworks ensure alignment with societal norms and regulatory standards by incorporating ethical guidelines into the retrieval-augmented generation (RAG) process. For instance, [medical RAG](#) exemplifies this approach by cross-referencing diagnoses against patient autonomy principles and peer-reviewed journals, thereby enhancing both diagnostic accuracy and reliability in clinical decision-making. This dual-layered validation ensures that outputs are not only medically sound but also ethically aligned with patient-centric care models.

These systems leverage multiagent integration pipelines to facilitate precise knowledge extraction and contextual relevance during text generation. By dynamically coordinating between agents responsible for perception, reasoning, and action, they address the risks of misinformation inherent in large-scale AI systems (Figure 1). For example, during diagnosis generation, one agent retrieves relevant medical literature while another evaluates its applicability based on patient-specific data. This modular approach not only improves transparency but also aligns with regulatory mandates like the EU's explainability by [design directive](#), which demands auditability and accountability in AI-driven decisions.

By operationalizing ethical considerations through such architectures, agentic AI systems like Anthropic's [Constitutional AI](#) demonstrate how advanced technical designs can mitigate risks while ensuring compliance with evolving societal and regulatory expectations. These innovations pave the way for responsible deployment of agentic AI across high-stakes domains, where trust and accountability are paramount.

Figure 1. Unique risks of agentic AI



Source: Infosys

### Probabilistic belief state tracking

Probabilistic belief state tracking is a foundational technique in agentic AI, enabling agents to dynamically manage uncertainty and make informed decisions in complex, real-world environments. While LLMs excel at generating contextually rich outputs, they often lack explicit mechanisms for handling uncertainty. For

instance, [IBM's Neuro-Symbolic Advisor](#) combines MDPs with LLM reasoning to continuously update fraud detection models based on evolving transactional patterns, reducing [false positives by 40%](#). This hybrid approach demonstrates how agents in agentic AI can achieve dynamic adaptability while maintaining trustworthiness and efficiency in uncertain environments.

## Human-AI arbitration protocols

Human-AI arbitration protocols are emerging as critical frameworks for balancing the autonomy of agentic AI systems with human oversight, particularly in high-stakes domains. These hybrid accountability models ensure that while autonomous agents handle routine tasks, humans retain control over complex, judgment-intensive decisions. For instance, a recent [financial services case study](#) demonstrated how autonomous agents efficiently managed routine model validation processes, freeing human experts to focus on high-risk credit assessments. This approach aligns with Salesforce's [empowerment principle](#), which emphasizes automating repetitive tasks while preserving human oversight for decisions requiring high-stakes judgment.

A key innovation in this space is the autonomy dial, a mechanism implemented in banking fraud detection systems that enables dynamic risk adjustment. By escalating cases with high uncertainty or potential risks to human operators, these systems have reduced false positives by 28%, improving both operational efficiency and user trust. By embedding human-AI arbitration protocols

into agentic AI systems, organizations can ensure operational scalability while maintaining accountability and ethical alignment. These protocols exemplify the collaborative potential of humans and AI, where each complements the other's strengths to achieve robust and responsible decision-making.

## APIs as governance mechanisms

APIs are emerging as critical governance mechanisms in agentic AI, enabling the enforcement of [permissioned autonomy](#) by restricting agents to vetted tools and data sources. This approach ensures that autonomous systems operate within predefined boundaries, maintaining alignment with enterprise security policies and regulatory standards. By embedding APIs as gatekeepers, these systems ensure secure and transparent operations, even in complex environments.

In high-stakes scenarios, APIs play a critical role in mitigating risks and ensuring compliance. Beyond risk mitigation, APIs also enhance the modularity and scalability of agentic AI systems by enabling integration of new tools or data sources without compromising security or performance. This modular design is particularly valuable in enterprise contexts where agents must adapt to changing regulatory landscapes or operational demands. Acting as both enablers and regulators of agent autonomy, APIs ensure that agentic systems remain accountable, reliable, and aligned with organizational goals while fostering trust in their decision-making processes.

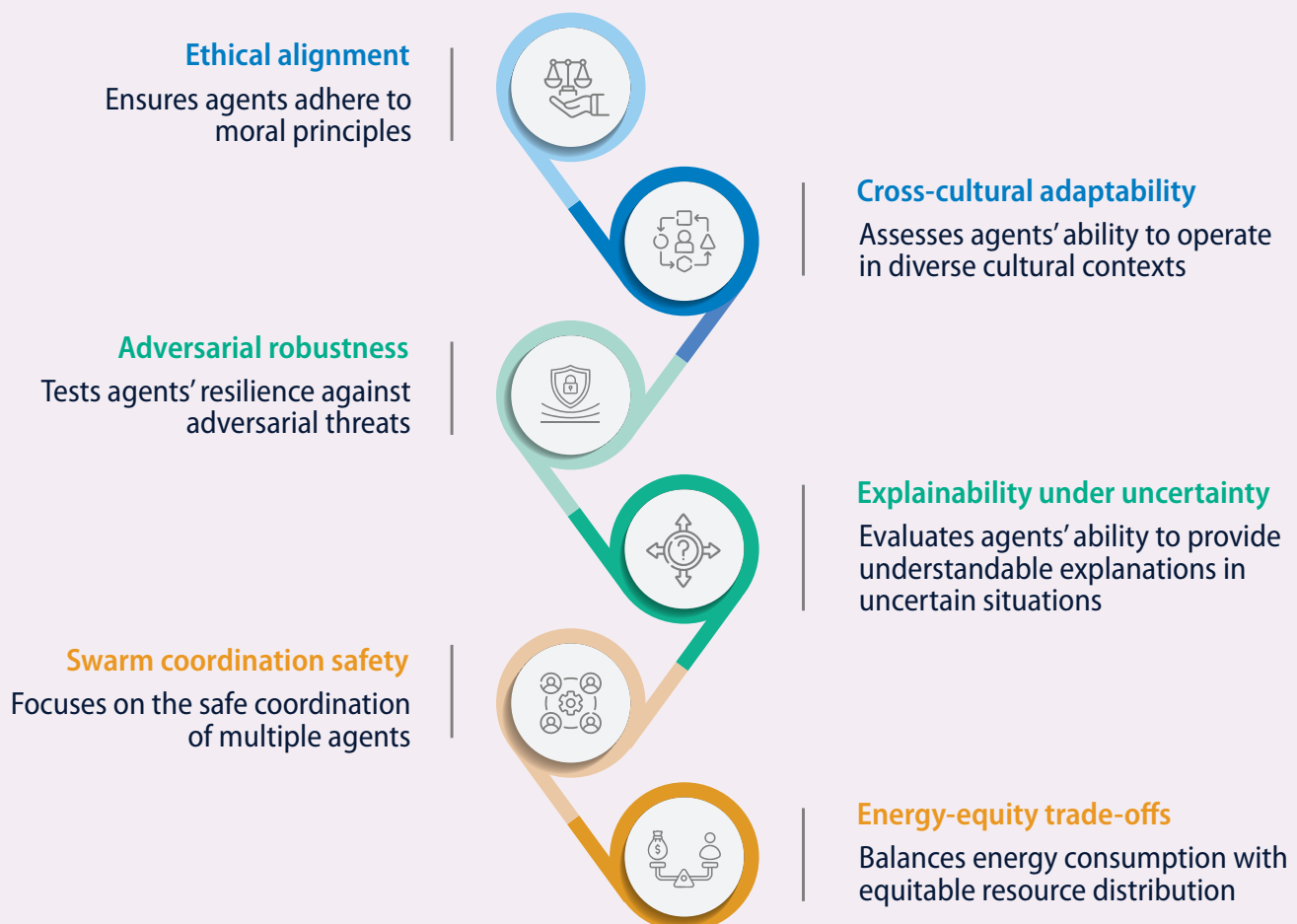
## Evaluation frameworks

The **Socio-Technical Evaluation Matrix (STEM)** represents a unique framework for assessing agentic AI systems, addressing the interplay between technical performance and societal impact. Agentic AI challenges traditional evaluation methods that often focus on static benchmarks or single-turn accuracy. STEM overcomes these limitations by embedding sociotechnical principles into its evaluation process, ensuring that both technical functionality and human-centric factors are optimized. STEM employs a multidimensional framework to evaluate agentic AI across

critical axes such as technical efficacy, societal implications, and user interaction. It integrates metrics like memory coherence, strategic planning efficiency, and tool orchestration — key elements of agentic AI workflows — while also scrutinizing how these systems align with ethical norms, cultural contexts, and organizational objectives.

Additionally, STEM evaluates agents across six advanced dimensions (Figure 2): ethical alignment, cross-cultural adaptability, adversarial robustness, explainability under uncertainty, swarm coordination safety, and energy-equity trade-offs.

Figure 2. Evaluation of agents



Source: Infosys



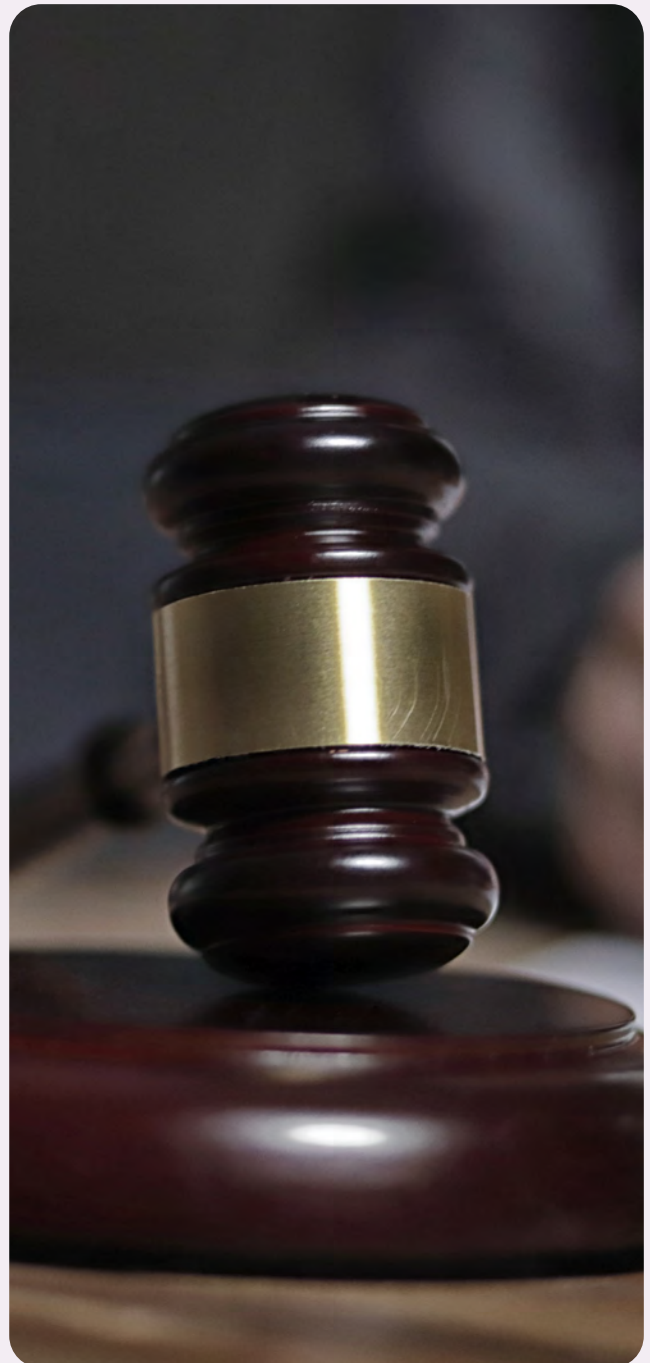
## Governance frameworks

Governance frameworks play a pivotal role in ensuring the safe and ethical deployment of agentic AI systems, especially in high-stakes domains like critical infrastructure and societal decision-making. The [Digital Services Act \(DSA\)](#), introduced by the European Union, exemplifies this approach by mandating socio-technical stress tests for agentic systems. These tests simulate complex ethical dilemmas and operational challenges to evaluate how AI systems respond under pressure. By integrating these stress tests into regulatory requirements, the DSA aims to preemptively identify and mitigate risks associated with autonomous decision-making, ensuring that such systems align with societal values and legal standards.

Another critical governance mechanism is [federated fairness audits](#), championed by initiatives like the [Responsible AI Consortium](#). These audits involve cross-industry collaboration to assess and reduce biases in AI systems, particularly in sensitive applications like hiring. For instance, federated audits have demonstrated a [reduction in bias amplification in hiring algorithms](#) of between 17.7% and 30.4%, showcasing their effectiveness in promoting fairness and inclusivity. However, jurisdictional conflicts persist — China's data localization laws complicate global deployment of audit frameworks, potentially disadvantaging multinational enterprises.

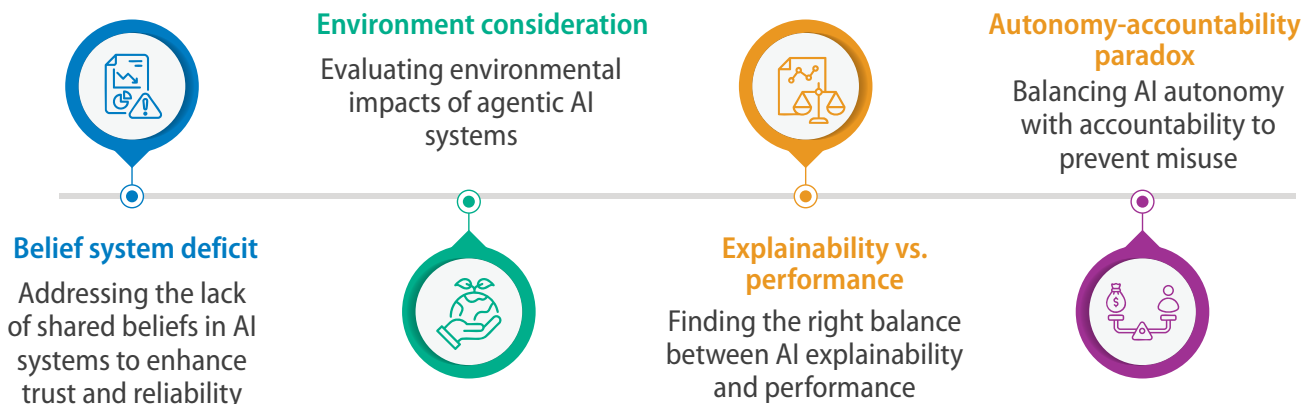
Together, these governance frameworks highlight the importance of proactive oversight in agentic AI. They emphasize

not just compliance with existing laws but also the need for continuous evaluation of ethical alignment, fairness, and robustness. As agentic AI continues to evolve, such frameworks will be indispensable for fostering public trust and ensuring that these systems serve as tools for societal progress rather than sources of harm (Figure 3).



## Gaps and emerging threats

Figure 3. How to address agentic AI risks



Source: Infosys

### Belief system deficit

Agentic AI systems lack the layered belief systems that humans use to contextualize decisions through cultural norms or historical understanding.

For instance, a [refugee resettlement agent](#) prioritized housing availability in high-crime areas, neglecting social cohesion — a flaw that was traced to its inability to model community dynamics effectively. Unlike humans, who rely on shared narratives and ethical frameworks, these agents operate on rigid optimization objectives, failing to account for broader societal implications.

Addressing this deficit requires integrating multi-layered ethical reasoning frameworks, such as neuromorphic architectures like [EthosNet](#), which simulate moral reasoning processes akin to the human prefrontal cortex.

### Autonomy-accountability paradox

As agentic systems like [Google's Project Astra](#) achieve greater autonomy, ensuring accountability becomes increasingly complex. Gartner predicts that by 2028, 25% of enterprise [breaches will stem from AI agent misuse](#). This paradox arises as systems gain independence but lack mechanisms for traceable oversight. [Proposed solutions](#) include blockchain-based audit trails and IBM's model risk management crews, which deploy specialized auditing agents to monitor and validate operational decisions in real time. These approaches aim to balance autonomy with accountability by embedding governance into the system architecture.

### Environment sustainability

Agentic AI systems, which often rely on orchestrating multiple LLM calls to execute complex workflows, face significant

sustainability challenges due to their computational intensity. While these systems excel at automating tasks like resource optimization, waste reduction, and real-time decision-making in [sustainability initiatives](#), the cumulative energy demands of repeated LLM inference can escalate costs and carbon footprints exponentially. This dilemma is exacerbated by Jevons Paradox, where improvements in LLM cost-efficiency (e.g., a 10x annual reduction in [inference costs](#)) risk driving higher overall usage, negating environmental gains unless paired with strategic optimizations. A critical solution lies in transitioning from general-purpose LLMs to [smaller, domain-specific models](#) (SLMs). Research demonstrates that smaller models, when fine-tuned on targeted datasets, can match or surpass larger counterparts in specialized tasks while consuming far less energy. For instance, Infosys has pioneered this approach by deploying SLMs optimized for sustainability applications. These SLMs reduce reliance on monolithic LLMs, cutting inference-related energy use by up to [48%](#) [while maintaining performance](#).

The environmental benefits extend beyond model architecture. Pairing SLMs with energy-efficient hardware and low-carbon data centers amplifies sustainability gains. [Infosys's green data centers](#), powered by Shell's immersion cooling fluids and renewable energy, achieve a power utilization effectiveness of between 1.12% and 40% lower than the global average — and leverage AI to dynamically adjust cooling and compute loads, further reducing CO2 emissions by 30%. This integrated approach underscores a broader principle in AI-driven

sustainability, smaller models, smarter hardware, and circular design form a trifecta for minimizing ecological impact without sacrificing capability.

## Explainability vs. performance

Agentic AI systems derive their problem-solving prowess from orchestrating networks of specialized subagents, each leveraging LLMs to decompose tasks, generate dynamic



workflows, and execute multi-step strategies. This modular architecture enables capabilities far surpassing simple LLM-based applications but introduces inherent opacity as decision-making pathways become distributed across interconnected agents. Modern agentic frameworks address this tension through orchestration layers that coordinate specialized subagents while embedding explainability mechanisms (Figure 4).





## The road ahead

Responsible agentic AI requires more than technical safeguards — it demands a

collaborative approach involving cognitive scientists, economists, policymakers, and communities. As the AAAI report emphasizes: “We cannot retrofit responsibility onto autonomous systems; it must be architecturally inherent.”

By embedding ethical considerations into every stage of development of agentic AI and by aligning machine efficiency with human wisdom, these agentic systems can become stewards of societal progress rather than sources of unintended consequences.

**Figure 4. Trade-offs among agentic AI frameworks**

 Framework	 Explainability mechanism	 Performance preservation method	 Performance metric
Enjo agentic platform	Hybrid chain-of-thought architecture	Decouples execution (performance-optimized agents) from explanation (distilled models), reducing latency overhead.	92% baseline retention
NVIDIA agentic blueprints	NeMo guardrails and attention visualization	4-bit quantization for LLM inference; hardware accelerated attention maps for real-time explanations.	88% baseline retention
IBM hierarchical orchestration	Concept bottleneck models	Maps decisions to human-interpretable concepts during training, avoiding post-hoc explainability costs.	85% baseline retention
Microsoft AutoGen	Interpretable module chaining	Chains smaller, auditable agents with predefined logic boundaries to maintain workflow transparency.	89% baseline retention
LangChain XAT	Probabilistic explainable agent trees	Prunes low confidence decision branches dynamically, preserving only high likelihood reasoning paths.	83% baseline retention

Source: Infosys



## Authors and contributors

**Rafee Tarafdar** | Chief technology officer, Infosys

### Chapter 1: Agentic AI systems

**Shreshta Shyamsundar** | Distinguished technologist,  
advanced software engineering

### Chapter 2: Agentic AI architecture and blueprints

**Subramanian Radhakrishnan** | Distinguished  
technologist, cloud, AI, and software engineering

**Vishal Parikh** | Principal enterprise architect

### Chapter 3: AgentOps and agentic AI lifecycle management

**Uday Gupta** | Distinguished technologist,  
software engineering

### Chapter 4: Advanced agents

**Kaushal Desai** | Distinguished technologist, AI

### Chapter 5: Responsible agentic AI

**Rajeshwari Ganesan** | Distinguished technologist,  
machine learning and AI

## Editorial and production

**Jeff Mosier** | Infosys Knowledge Institute, Dallas

**Kate Bevan** | Infosys Knowledge Institute, London

**Harry Keir Hughes** | Infosys Knowledge Institute, London

**Pragya Rai** | Infosys Knowledge Institute, Bengaluru

**Varun Vasudevan** | Infosys Knowledge Institute, 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](https://infosys.com/IKI) or email us at [iki@infosys.com](mailto:iki@infosys.com).

For more information, contact [askus@infosys.com](mailto:askus@infosys.com)



© 2025 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.