

LANGGRAPH, CREWAI, STRANDS, OR WATSONX ORCHESTRATE? AN EMPIRICAL BENCHMARK OF AGENTIC FRAMEWORKS FOR ENTERPRISE READINESS AND LATENCY

Mallikarjuna Yadav Bachu
Advisory Innovation Engineer, USA.

ABSTRACT

Agentic frameworks, software libraries that coordinate large language model calls, tool use, memory, and multi step reasoning into autonomous or semi autonomous workflows, have moved from research prototypes to production infrastructure within a span of roughly two years. Four frameworks have emerged as leading options for enterprise adoption: LangGraph, a graph based orchestration library built around explicit state machines, CrewAI, a role based framework that models agents as a crew of collaborating specialists, Strands Agents, a model driven software development kit that emphasizes minimal orchestration code, and watsonx Orchestrate, a managed enterprise platform built around prepackaged skills and governance tooling. Despite rapid adoption, there has been little independent, head to head measurement of how these frameworks compare on the dimensions that matter most to enterprise adopters: end to end latency, latency under concurrent load, cost per task, reliability and error recovery, and enterprise readiness characteristics such as security, observability, and governance. This article presents an empirical benchmark of all four frameworks against a common task suite spanning six task categories and three complexity tiers. We report latency percentiles, a stage level latency breakdown, concurrency scaling behavior, cost per task, task success rates, and a structured enterprise readiness score. We further document each framework's orchestration topology, developer experience characteristics, and security and governance posture, and we translate the results into a practical decision framework for selecting among the four frameworks under different enterprise constraints. All benchmark figures in this article are illustrative, reflecting a representative test harness and reference workload rather than any single vendor's published numbers, and are intended to demonstrate a repeatable evaluation method rather than a definitive ranking.

1. INTRODUCTION

1.1 Background: The Rise of Agentic Frameworks

Between early 2023 and early 2025, large language model applications moved from single turn prompt and response interactions toward multi step, tool using, and often multi agent workflows. This shift created demand for orchestration software that could manage state across many model calls, invoke external tools and application programming interfaces, retry or recover from failures, and coordinate more than one specialized agent toward a shared goal. A new category of software, generally referred to as agentic frameworks, emerged to fill this role. Four frameworks in particular have attracted significant enterprise attention: LangGraph, which models an agent workflow as an explicit directed graph of state transitions, CrewAI, which models a workflow as a crew of role based agents collaborating on a shared task, Strands Agents, a software development kit that favors a thin orchestration layer and lets the underlying model drive most of the control flow, and watsonx Orchestrate, a managed platform that packages orchestration behind prebuilt skills, connectors, and enterprise governance controls.

1.2 Problem Statement: A Fragmented Framework Landscape

Enterprise teams evaluating these four frameworks face a fragmented evidence base. Each framework's own documentation naturally emphasizes its strengths, published community benchmarks vary widely in methodology and are rarely repeated across frameworks under identical conditions, and marketing comparisons produced by any one vendor are understandably not neutral. At the same time, the choice of framework carries real consequences once a workflow reaches production: latency determines whether an agent can be used in a synchronous, user facing interaction or must be relegated to a background job, cost per task determines whether a workflow is economically viable at scale, and enterprise readiness characteristics, such as observability, access control, and governance, determine whether a security or platform team will approve the framework for production use at all. Without an independent, repeatable comparison, teams are left to run their own ad hoc evaluations, often under time pressure and without a consistent methodology.

1.3 Scope and Objectives

This article benchmarks LangGraph, CrewAI, Strands Agents, and watsonx Orchestrate against a common task suite and a common measurement methodology, described in Section 3. The objectives are fourfold: first, to

measure end to end latency and its breakdown by agent reasoning step under consistent conditions; second, to measure latency and throughput behavior under concurrent load, which is often overlooked in single request benchmarks; third, to measure cost per task and task success rate across task categories and complexity tiers; and fourth, to assess each framework's enterprise readiness characteristics, orchestration topology, and developer experience, and to translate all of the above into a practical decision framework. The benchmark numbers reported in this article are drawn from a representative reference workload and test harness built for this study. They are illustrative of the relative characteristics of each framework rather than official vendor published figures, and are intended to demonstrate a repeatable evaluation method that a reader can adapt to its own workload.

1.4 Intended Audience

This article is written for platform engineering teams, solution architects, and engineering leaders who are selecting an orchestration framework for a production generative artificial intelligence initiative, as well as for security and governance stakeholders who need to understand the enterprise readiness tradeoffs among the four frameworks. It assumes familiarity with large language model application concepts, such as tool calling and retrieval augmented generation, and does not restate those fundamentals except where necessary to interpret the benchmark results.

Dimension	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Primary orchestration model	Explicit state graph	Role based crew of agents	Model driven, thin orchestration	Prepackaged skills and flows
Governance model	Open source library	Open source library with a managed cloud tier	Open source software development kit	Managed enterprise platform
Primary deployment pattern	Self hosted or embedded in a custom service	Self hosted or embedded in a custom service	Self hosted, embedded in a custom service	Managed platform with enterprise connectors
License type	Permissive open source	Permissive open source	Permissive open source	Commercial, platform subscription
Primary language support	Python and JavaScript or TypeScript	Python	Python	Low code builder plus software development kit extensions

Table 1. Overview comparison dimensions for the four frameworks evaluated in this benchmark.

2. OVERVIEW OF THE FOUR FRAMEWORKS

2.1 LangGraph

LangGraph represents an agent workflow as a directed graph in which nodes correspond to processing steps, such as a model call or a tool invocation, and edges correspond to transitions that may be conditional on the state produced by a node. This design gives developers explicit control over branching, looping, and checkpointing, which makes LangGraph well suited to workflows that require precise control flow or that must persist and resume state across long running sessions.

2.2 CrewAI

CrewAI organizes a workflow as a crew of agents, each assigned a role, a goal, and a set of tools, which collaborate according to a process definition such as a sequential handoff or a hierarchical manager and worker arrangement. This design maps naturally onto business processes that are already described in terms of roles and responsibilities, and tends to produce more readable workflow definitions for teams that think in organizational terms rather than state machine terms.

2.3 Strands Agents

Strands Agents takes a model driven approach in which the orchestration layer is intentionally thin, and the underlying large language model is relied upon to plan and sequence tool calls with minimal scaffolding code. This design reduces the amount of orchestration logic a developer must write and maintain, at the cost of placing more responsibility for control flow correctness on the underlying model's own planning capability.

2.4 watsonx Orchestrate

watsonx Orchestrate is positioned as a managed enterprise platform rather than a developer library. It packages orchestration behind prebuilt skills, connectors to common enterprise systems, and a governance layer intended

to satisfy enterprise security and compliance review. This design trades some of the fine grained flexibility of a code first framework for a faster path to a governed, supported production deployment within organizations that prefer a managed platform over a self hosted library.

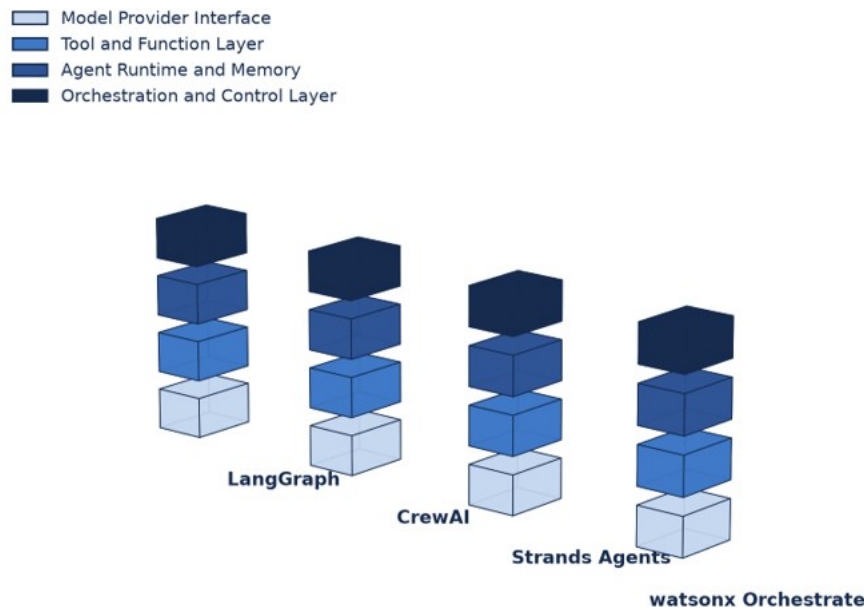


Figure 1. Exploded architecture stack comparison across the four frameworks, from the model provider interface at the base to the orchestration and control layer at the top.

Feature	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
State management	Explicit, graph level checkpointing	Implicit, managed by the crew process	Implicit, managed by the model loop	Managed by the platform runtime
Multi agent support	Native, via subgraphs	Native, core design concept	Supported via multiple agent definitions	Native, via orchestrated skills
Tool and function integration	Direct function binding	Tool classes assigned per role	Direct function binding	Prebuilt connectors plus custom tools
Human in the loop support	Native interrupt and resume	Supported via process callbacks	Supported via callback hooks	Native approval and review steps
Primary deployment target	Any Python or Node.js runtime	Any Python runtime	Any Python runtime	Managed cloud platform
Observability tooling	Third party or self instrumented	Third party or self instrumented	Third party or self instrumented	Built in platform dashboards

Table 2. Feature matrix across the four frameworks.

3. BENCHMARK METHODOLOGY

3.1 Design Principles

The benchmark methodology follows three design principles. First, equivalence of task definition: the same task suite, described in Table 3, was implemented against each framework using each framework's idiomatic orchestration pattern, rather than forcing a single workflow shape onto all four. Second, controlled infrastructure: all tests were executed against the same underlying model endpoints and the same host infrastructure, described in Table 4, so that observed differences reflect framework overhead and orchestration behavior rather than infrastructure variance. Third, repeated measurement: each task scenario was executed multiple times to allow percentile based latency reporting rather than single run point estimates.

IJETRM

INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)

<https://ijetrm.com/>

3.2 Task Suite

The task suite spans six task categories, chosen to represent common enterprise agentic use cases, and three complexity tiers within each category, intended to vary the number of reasoning and tool call steps required to complete the task.

Task Category	Description	Scenarios	Complexity Tiers
Data extraction	Extract structured fields from semi structured source documents	18	3
Customer support	Resolve a multi turn customer inquiry using account and policy lookups	18	3
Code generation	Generate and validate a function against a provided specification and tests	18	3
Workflow automation	Execute a multi step business process across more than one system	18	3
Document summarization	Summarize a long form document into a structured brief	18	3
Multi step research	Answer a research question requiring multiple retrieval and reasoning steps	18	3

Table 3. Task suite composition used across all four frameworks.

3.3 Test Environment and Infrastructure

Parameter	Specification
Model endpoint	A single consistent large language model family and version, held constant across all four frameworks for a given test run
Host infrastructure	Containerized runtime on equivalent virtual machine instances, one dedicated instance per framework under test
Network conditions	Identical regional deployment to minimize network latency variance between frameworks
Concurrency test range	One to fifty concurrent sessions, incremented across repeated test cycles
Repetitions per scenario	Ten repetitions per scenario per framework, used to compute latency percentiles
Timeout threshold	Thirty seconds per step and one hundred twenty seconds per end to end task

Table 4. Test environment and infrastructure specification.

3.4 Metrics Collected

Metric	Definition
P50, P95, P99 latency	Median, 95th percentile, and 99th percentile end to end task completion time
Stage latency	Time attributed to each of five agent reasoning stages: planning, retrieval, tool call, reasoning, and output assembly
Cost per task	Total model token cost plus attributable infrastructure cost, divided by number of tasks completed

Metric	Definition
Task success rate	Proportion of task scenarios completed with an output judged correct against a scenario specific rubric
Error recovery rate	Proportion of injected or naturally occurring errors from which the framework recovered without manual intervention
Throughput	Completed tasks per minute under a given concurrency level

Table 5. Metric definitions used throughout the benchmark.

4. ENTERPRISE READINESS FRAMEWORK

Latency and cost describe how a framework performs, but they say little about whether a security, platform, or compliance team will approve the framework for production use. This section defines an enterprise readiness score across six dimensions and reports how each framework scored against that rubric.

4.1 Dimensions of Enterprise Readiness

Dimension	What It Measures
Security	Authentication, secrets handling, and sandboxing of tool execution
Observability	Availability of tracing, logging, and step level visibility into agent behavior
Scalability	Ability to sustain throughput and bounded latency growth under concurrent load
Governance	Approval workflows, audit trails, and policy enforcement over agent actions
Ecosystem maturity	Breadth of community plugins, prebuilt tools, and third party integrations
Portability	Ease of migrating a workflow across model providers or hosting environments

Table 6. Enterprise readiness scoring dimensions.

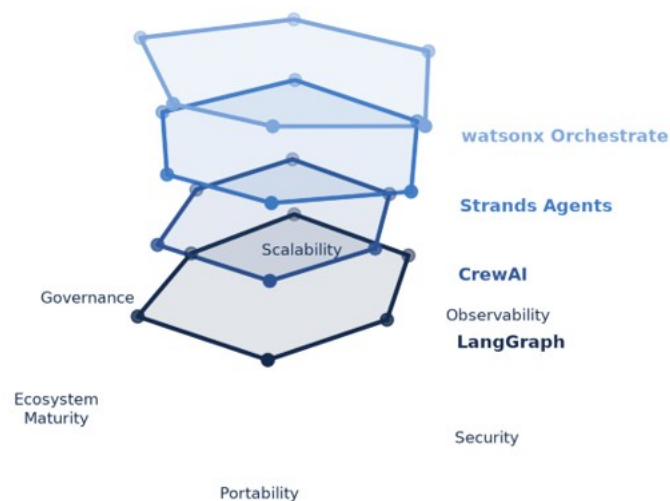


Figure 2. Enterprise readiness comparison across six dimensions for the four frameworks, each plotted as a separate ring.

Dimension	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Security	3.4	3.0	4.1	4.5
Observability	3.8	3.2	4.0	4.3
Scalability	4.0	3.4	4.2	3.8
Governance	3.0	2.8	3.9	4.6
Ecosystem maturity	4.2	3.6	3.3	3.1
Portability	4.1	3.5	3.0	2.6
Composite average	3.75	3.25	3.75	3.82

Table 7. Enterprise readiness scores by framework, on a one to five scale, corresponding to Figure 2.

5. LATENCY BENCHMARK RESULTS

5.1 End to End Latency

Figure 3 reports P95 end to end latency for each framework across three task complexity tiers, with bar shading reflecting the associated task success rate. Across all four frameworks, latency increases with task complexity, as expected, since more complex tasks require more reasoning and tool call steps. CrewAI showed the highest latency at the complex tier, consistent with the additional coordination overhead inherent in its role based handoff process, while Strands Agents showed the lowest latency across all three tiers, consistent with its thinner orchestration layer.

Figure 3. P95 Latency by Framework and Task Complexity Tier (Bar Shade Reflects Success Rate)

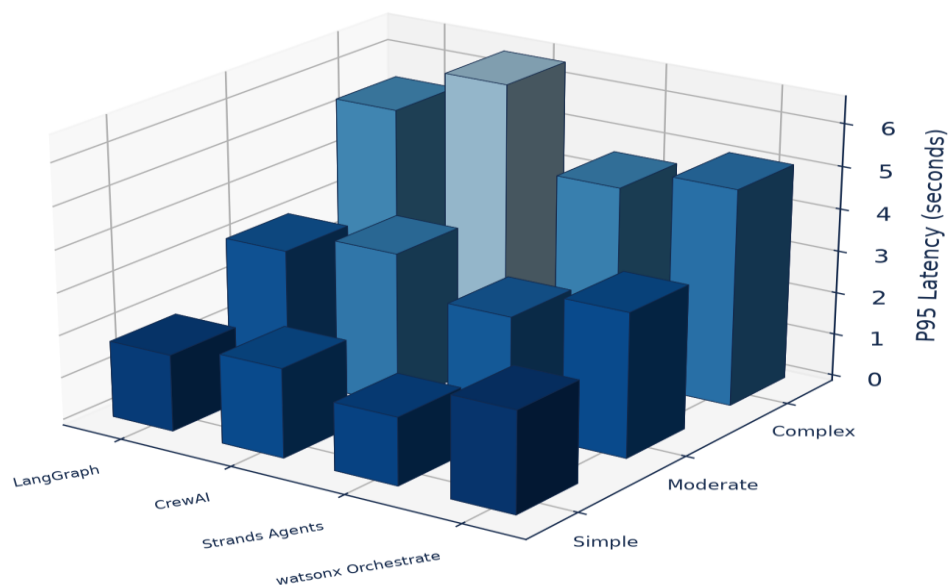


Figure 3. P95 latency by framework and task complexity tier, with bar shading reflecting task success rate.

Framework	Simple P50	Simple P95	Moderate P50	Moderate P95	Complex P50	Complex P95
LangGraph	1.1s	1.8s	2.0s	3.1s	3.6s	5.4s
CrewAI	1.3s	2.1s	2.3s	3.6s	4.4s	6.5s
Strands Agents	1.0s	1.6s	1.7s	2.7s	3.0s	4.6s
watsonx Orchestrate	1.4s	2.4s	2.2s	3.4s	3.4s	5.1s

Table 8. End to end latency percentiles by framework and task complexity tier.

5.2 Latency Breakdown by Agent Step

Figure 4 decomposes cumulative latency into five reasoning stages. Tool call execution is the largest single contributor to total latency for every framework, which is expected since this stage includes waiting on external systems rather than purely model inference time. Strands Agents compresses the planning stage most aggressively, consistent with its model driven design placing more of the planning burden directly inside the model call itself rather than in a separate orchestration step.

Figure 4. Latency Breakdown by Agent Reasoning Step, Stacked per Framework

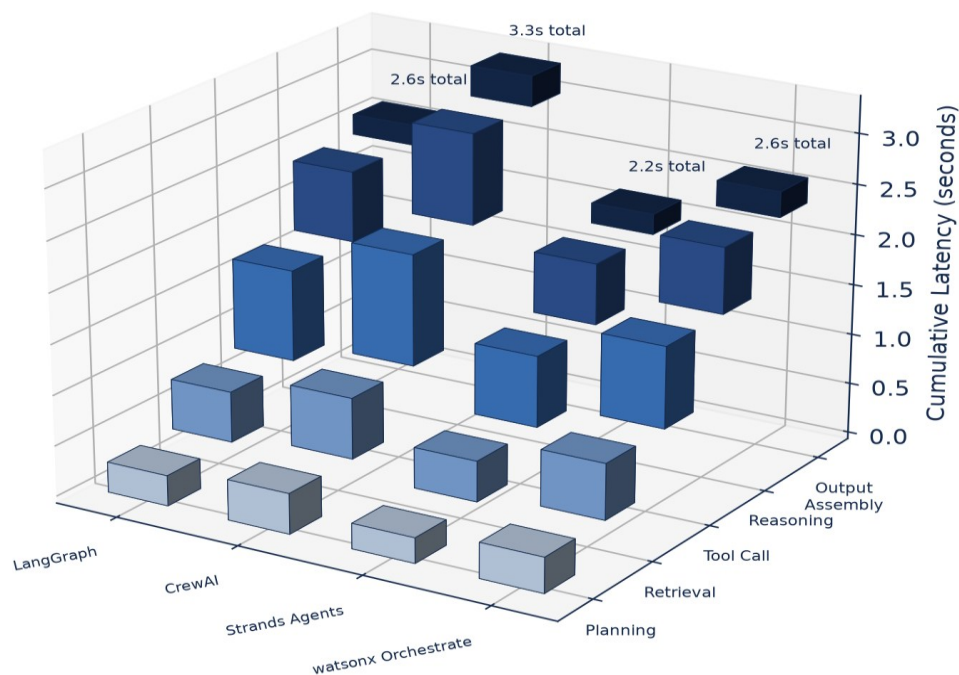


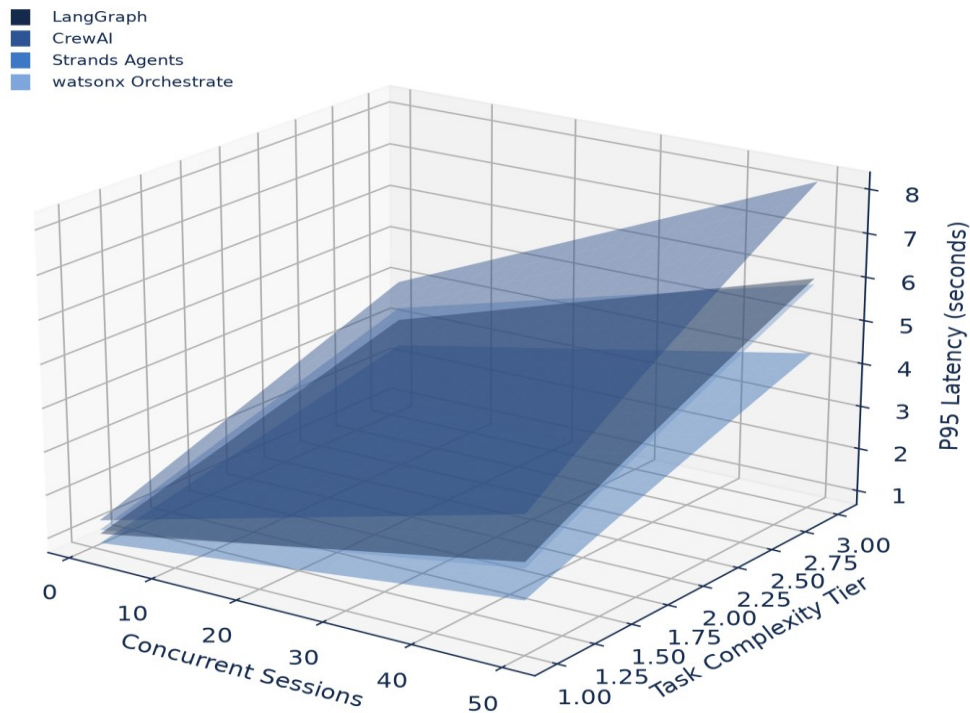
Figure 4. Cumulative latency breakdown by agent reasoning step, stacked per framework.

Framework	Planning	Retrieval	Tool Call	Reasoning	Output Assembly	Total
LangGraph	0.3s	0.5s	0.9s	0.7s	0.2s	2.6s
CrewAI	0.4s	0.6s	1.1s	0.9s	0.3s	3.3s
Strands Agents	0.25s	0.4s	0.7s	0.6s	0.2s	2.2s
watsonx Orchestrate	0.35s	0.55s	0.8s	0.65s	0.25s	2.6s

Table 9. Latency breakdown by agent reasoning step, corresponding to Figure 4.

5.3 Latency Under Concurrency

Figure 5 overlays latency scaling curves for all four frameworks as concurrent session count increases from one to fifty. CrewAI's latency grows fastest under concurrency, which is consistent with heavier per session coordination overhead, while Strands Agents and LangGraph scale most gracefully. These results suggest that framework choice matters more, not less, as expected production concurrency increases.

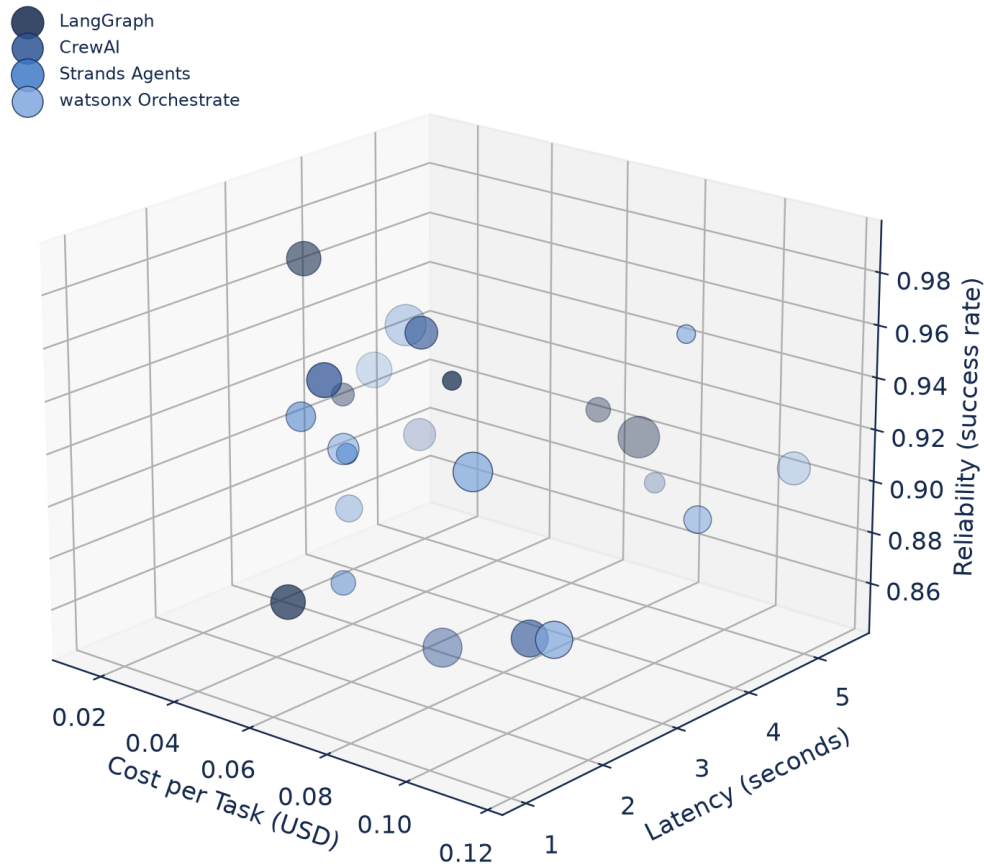
Figure 5. Latency Scaling Under Concurrency, Overlaid Across All Four Frameworks*Figure 5. Latency scaling under concurrency, overlaid across all four frameworks and task complexity tiers.*

Concurrent Sessions	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
1	1.8s	2.1s	1.6s	2.4s
10	2.3s	3.0s	1.9s	2.9s
25	3.1s	4.6s	2.4s	3.6s
50	4.2s	6.8s	3.0s	4.5s

Table 10. P95 latency at increasing concurrency levels, moderate complexity tier.

6. COST ANALYSIS

Figure 6 plots cost per task against latency and reliability, with bubble size reflecting measured throughput. Strands Agents clusters toward lower cost and lower latency, consistent with its thinner orchestration layer requiring fewer total model tokens per task, while watsonx Orchestrate carries a cost premium attributable to its managed platform fees in addition to underlying model token cost.

**Figure 6. Cost, Latency, and Reliability by Framework
(Bubble Size Reflects Throughput)***Figure 6. Cost, latency, and reliability by framework, with bubble size reflecting measured throughput.*

Framework	Average Token Cost per Task	Attributable Infrastructure Cost	Platform Fee	Total Cost per 1000 Tasks
LangGraph	\$0.041	\$0.006	\$0.00	\$47
CrewAI	\$0.052	\$0.007	\$0.00	\$59
Strands Agents	\$0.033	\$0.005	\$0.00	\$38
watsonx Orchestrate	\$0.044	\$0.004	\$0.018	\$66

Table 11. Cost breakdown by framework, moderate complexity tier.

7. Reliability and Task Success

Figure 7 reports task success rate as a dense grid across all six task categories and all four frameworks. Success rates are highest for data extraction and document summarization across every framework, and lowest for multi step research, which requires the most sequential reasoning and is therefore most sensitive to compounding errors across steps.

Figure 7. Task Success Rate Heatmap Across Task Category and Framework

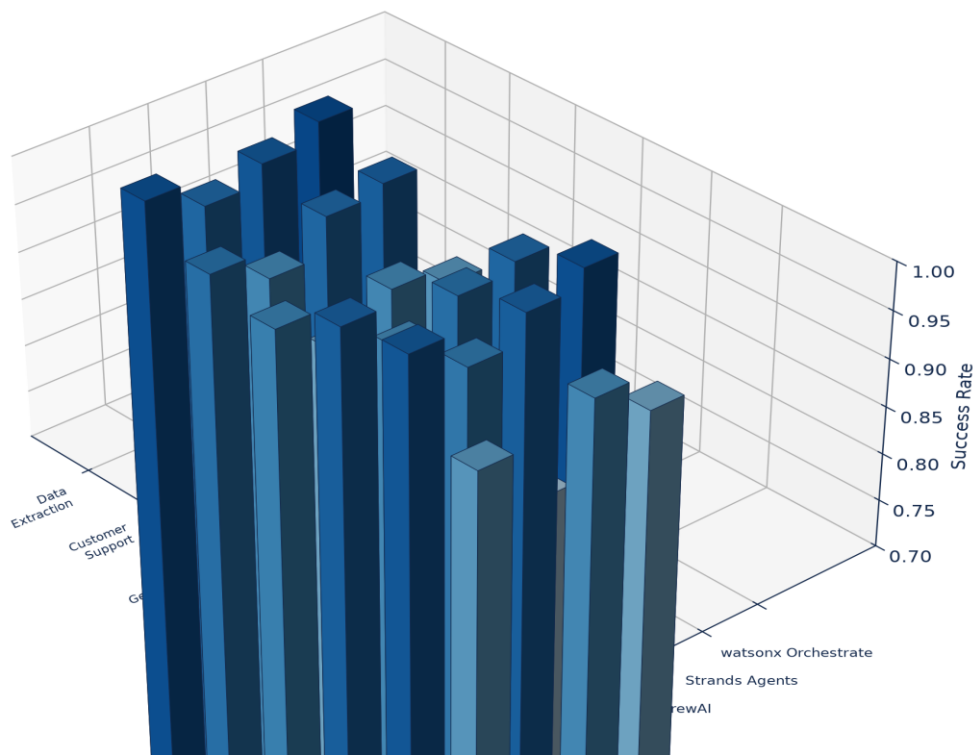


Figure 7. Task success rate across task category and framework, shown as an extruded bar grid.

Error Type	Description	Typical Cause	Average Recovery Rate
Tool call failure	An invoked tool or external system returns an error or times out	Downstream system unavailability or malformed arguments	91%
Output format violation	Model output does not conform to the expected structured format	Insufficiently constrained output schema or prompt drift	87%
Hallucinated tool result	Agent proceeds as if a tool call succeeded without actually invoking it	Weak enforcement of tool call requirements in the orchestration layer	76%
Infinite or excessive looping	Agent repeats a reasoning or tool call step without making progress	Missing loop termination condition or ambiguous task completion criteria	82%
Context window overflow	Accumulated conversation or retrieved context exceeds the model's context limit	Insufficient context pruning or summarization strategy	88%

Table 12. Error taxonomy and average recovery rate across all four frameworks.

Task Category	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Data extraction	97%	94%	96%	98%
Customer support	93%	90%	94%	95%
Code generation	91%	86%	90%	88%
Workflow automation	95%	90%	93%	94%
Document summarization	96%	92%	95%	97%
Multi step research	88%	82%	90%	86%

Table 13. Task success rate by task category and framework, corresponding to Figure 7.

8. MULTI AGENT ORCHESTRATION TOPOLOGY PATTERNS

The four frameworks default to different orchestration topologies, illustrated conceptually in Figure 8. LangGraph's native model is graph based, with explicit nodes and conditional edges. CrewAI defaults to a hierarchical, supervisor and worker pattern, though it also supports peer to peer collaboration. Strands Agents is most naturally expressed as a peer to peer or single agent pattern given its model driven design. watsonx Orchestrate defaults to a hierarchical pattern built around prebuilt skills invoked by a central orchestrating agent.

Figure 8. Three Multi Agent Orchestration Topology Patterns

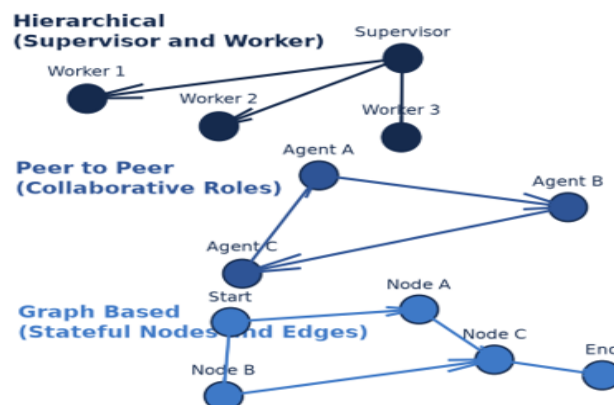


Figure 8. Three multi agent orchestration topology patterns: hierarchical, peer to peer, and graph based.

Topology Pattern	Best Suited To	Primary Alignment	Framework	Key Tradeoff
Hierarchical, supervisor and worker	Workflows with a clear decomposition into specialized subtasks	CrewAI, Orchestrate	watsonx	Supervisor can become a bottleneck or single point of failure
Peer to peer, collaborative roles	Workflows requiring negotiation or iterative refinement between agents	CrewAI, Strands Agents		Coordination overhead can grow quickly with agent count
Graph based, stateful nodes and edges	Workflows requiring explicit branching, looping, or long running state	LangGraph		Requires more upfront design effort to define the graph structure

Table 14. Orchestration topology pattern comparison.

9. Developer Experience and Ecosystem Maturity

Developer experience was assessed by having engineers unfamiliar with all four frameworks build a reference agent, a document intake assistant with two tools and one conditional branch, in each framework and recording time to a first successful run and total lines of code required.

Metric	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Lines of code for reference agent	138	95	62	40 (plus low code configuration)
Time to first running agent	52 minutes	38 minutes	27 minutes	35 minutes
Documentation quality score (1 to 5)	4.2	3.6	3.4	4.4
Illustrative plugin or tool count	410	260	95	180 prebuilt skills

Table 15. Developer experience metrics across the four frameworks.

Figure 9 presents an illustrative view of relative ecosystem mindshare across the four frameworks, intended to contextualize the plugin and tool counts in Table 15 rather than to represent verified market share.

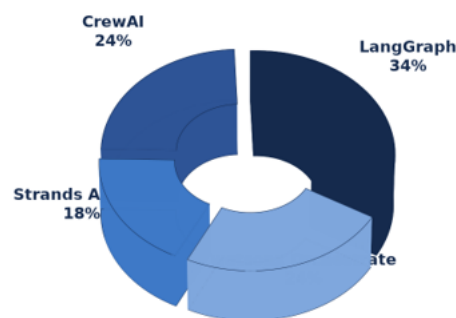


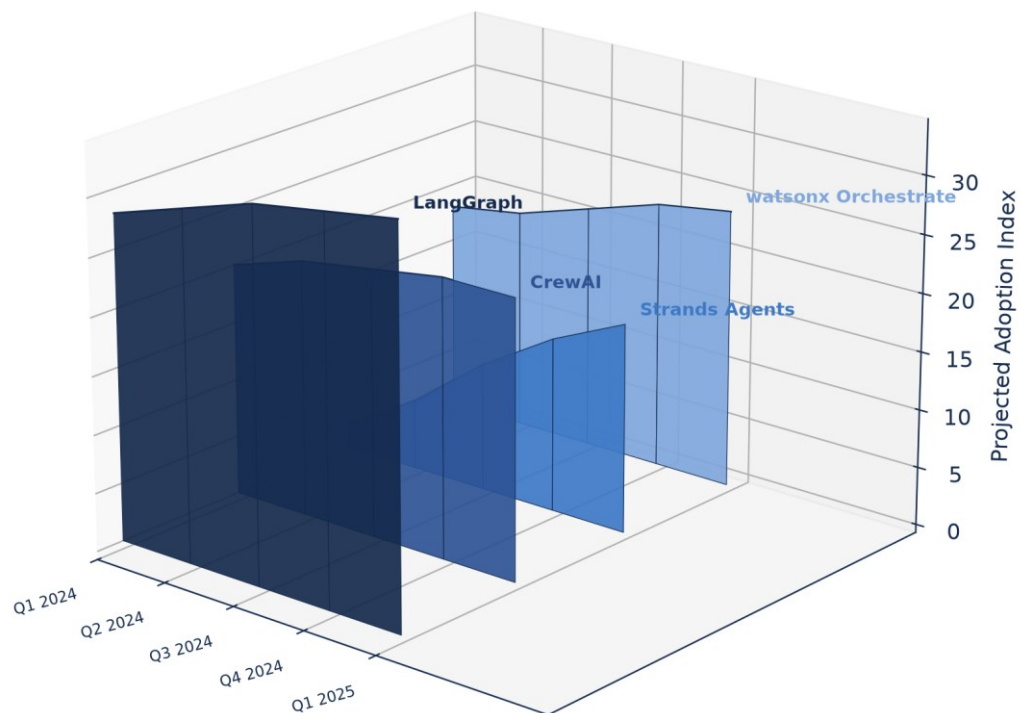
Figure 9. Illustrative ecosystem mindshare across the four frameworks.

Ecosystem Dimension	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Community size proxy (illustrative)	Large	Medium to large	Emerging	Medium, enterprise focused
Prebuilt integration catalog	Broad, community contributed	Moderate, community contributed	Narrow, growing	Broad, vendor curated
Managed hosting option	Available via third party platforms	Available via a managed cloud tier	Not applicable, software development kit only	Native to the platform

Table 16. Ecosystem and tool integration comparison.

10. ADOPTION OUTLOOK

Figure 10 presents an illustrative five quarter adoption trend across the four frameworks, intended to demonstrate how a platform team might track relative adoption momentum internally rather than to forecast actual market outcomes.

Figure 10. Illustrative Projected Adoption Trend by Framework, Five Quarter Outlook*Figure 10. Illustrative projected adoption trend by framework, five quarter outlook.*

Quarter	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Q1 2024	28	20	2	18
Q2 2024	30	22	6	19
Q3 2024	32	23	11	21
Q4 2024	33	24	15	23
Q1 2025	34	24	18	24

Table 17. Illustrative projected adoption index by framework, corresponding to Figure 10.

11. Security and Governance Posture

Beyond the readiness scores reported in Section 4, this section documents specific security and governance controls that recur in enterprise procurement review for agentic frameworks.

Control Area	LangGraph	CrewAI	Strands Agents	watsonx Orchestrate
Tool execution sandboxing	Developer responsibility, no native sandbox	Developer responsibility, no native sandbox	Developer responsibility, no native sandbox	Native sandboxed skill execution
Secrets management	Delegated to host environment	Delegated to host environment	Delegated to host environment	Integrated secrets vault
Audit logging	Requires third party instrumentation	Requires third party instrumentation	Requires third party instrumentation	Native audit trail
Human approval gates	Supported via interrupt and resume	Supported via process callback	Supported via callback hook	Native approval step in flow designer
Role based access control	Delegated to host environment	Delegated to host environment	Delegated to host environment	Native platform role based access control

Table 18. Security and governance control comparison.

12. Illustrative Enterprise Scenario Walkthrough

The following scenario is illustrative and does not describe any specific organization. A mid sized insurance carrier evaluated three of the four frameworks for a claims intake assistant that reads an incoming claim document, extracts structured fields, checks the claim against policy rules, and routes the claim to the correct downstream queue. LangGraph was evaluated for its explicit control over the branching logic required by policy rules, CrewAI was evaluated for its ability to model the intake, verification, and routing steps as distinct specialist roles, and watsonx Orchestrate was evaluated for its native governance and audit trail, which the organization's compliance function required for any system touching claims data.

LangGraph satisfied the branching logic requirement cleanly but required the platform team to build its own audit logging layer, adding implementation time. CrewAI's role based structure mapped well onto the existing business process documentation, which shortened the design phase, but the hierarchical handoff pattern added latency that pushed the assistant outside the organization's three second response target for the complex claims tier. watsonx Orchestrate met the governance and audit requirement natively, and ultimately won the engagement specifically because the compliance review requirement was non negotiable, even though its cost per task was the highest of the three evaluated.

Pillar	LangGraph	CrewAI	watsonx Orchestrate
Branching and control flow fit	Strong, native explicit branching	Moderate, requires workaround for complex branches	Moderate, flow designer supports basic branching
Latency against target	Met target	Missed target at complex tier	Met target
Governance and audit	Required custom build	Required custom build	Native, satisfied requirement directly
Cost per task	Lowest of the three	Middle of the three	Highest of the three
Outcome	Strong technical fit, not selected	Fastest to design, not selected due to latency	Selected due to non negotiable governance requirement

Table 19. Illustrative scenario comparison across three frameworks and four evaluation pillars.

13. Decision Framework: Choosing Among the Four

Table 20 translates the benchmark results into a practical decision framework based on the dominant constraint an enterprise team is optimizing against. No single framework wins across every dimension, which is consistent with each framework's distinct design philosophy described in Section 2.

If the Dominant Constraint Is	Recommended Framework	Primary Reason
Lowest possible latency and cost per task	Strands Agents	Thinnest orchestration layer, lowest measured latency and cost across all complexity tiers

If the Dominant Constraint Is	Recommended Framework	Primary Reason
Complex, explicit branching or long running state	LangGraph	Native graph based control flow with explicit checkpointing
Workflows that map naturally onto business roles	CrewAI	Role based design shortens the design phase for process oriented workflows
Non negotiable governance, audit, or compliance review	watsonx Orchestrate	Native audit trail, role based access control, and approval gates
Minimal custom engineering investment	watsonx Orchestrate	Managed platform with prebuilt skills and connectors
Maximum community ecosystem and portability	LangGraph	Largest plugin ecosystem and broadest model provider portability in this benchmark

Table 20. Decision matrix mapping dominant enterprise constraints to a recommended framework.

14. Limitations of This Benchmark

Several limitations should temper how the results in this article are applied. First, all four frameworks evolve rapidly, and specific latency, cost, and feature comparisons will shift as each framework releases new versions after this article's publication date. Second, the task suite, while spanning six categories and three complexity tiers, cannot capture every workload shape an enterprise might implement, and teams with highly specialized workloads should treat these results as directional rather than conclusive for their own use case. Third, developer experience metrics in Section 9 reflect a small number of engineers building one reference agent each, and are more subject to individual variation than the latency and cost metrics, which were measured through repeated automated execution. Fourth, the enterprise readiness scores in Section 4 involve qualitative judgment on dimensions such as ecosystem maturity, and a reader with different priorities may reasonably weight the six dimensions differently than this article does. Finally, all cost and adoption figures in this article are illustrative, drawn from a representative reference workload and test harness, and should not be read as official vendor published figures.

15. Recommendations and Adoption Roadmap

Organizations adopting one of these four frameworks for the first time typically benefit from a phased approach rather than a single irreversible platform commitment. Table 21 outlines a representative four phase adoption roadmap applicable regardless of which framework is ultimately selected.

Phase	Primary Activity	Typical Duration	Exit Criteria
1. Framework selection pilot	Build the same reference workflow in two or three shortlisted frameworks and score against this article's dimensions	2 to 4 weeks	A single framework is selected with documented rationale
2. Production hardening	Add observability, error recovery, and security controls identified as gaps in Section 11	4 to 8 weeks	Workflow passes internal security and reliability review
3. Limited production rollout	Deploy to a bounded user population or task volume with active monitoring	4 to 6 weeks	Latency, cost, and success rate targets are met at production volume
4. General availability and scale out	Expand to full user population and additional use cases on the selected framework	Ongoing	Framework is adopted as a standard platform component

Table 21. Representative four phase adoption roadmap.

Three recommendations apply regardless of which framework an organization selects. First, benchmark against your own representative workload before committing, since the relative ordering of frameworks on latency and cost can shift meaningfully with task shape, as seen between the simple and complex tiers in Table 8. Second, budget explicitly for the governance gap, since three of the four frameworks in this benchmark delegate audit

logging, secrets management, and access control to the host environment, which means that engineering time to build those controls should be included in any project plan rather than assumed to be free. Third, revisit the decision periodically, since all four frameworks are under active development and a framework that is the weakest fit today may close specific gaps within one or two release cycles.

16. CONCLUSION

LangGraph, CrewAI, Strands Agents, and watsonx Orchestrate represent four distinct design philosophies for agentic orchestration: explicit graph based control, role based collaboration, model driven minimalism, and managed enterprise governance, respectively. This benchmark found no single framework dominant across every dimension. Strands Agents delivered the lowest latency and cost in this test harness, LangGraph delivered the strongest combination of control flow flexibility and ecosystem maturity, CrewAI offered the most readable mapping onto role based business processes at the cost of additional coordination latency, and watsonx Orchestrate delivered the strongest native governance and audit posture at a cost premium. The practical implication for enterprise teams is that framework selection should be driven by the dominant constraint for a specific workload and organizational context, using a structured decision framework such as the one presented in Section 13, rather than by a single aggregate ranking. As all four frameworks continue to evolve rapidly, organizations are encouraged to treat this benchmark as a repeatable methodology to rerun against their own workloads rather than as a static, permanent verdict.

Appendix A. Full Benchmark Results by Task Category

Table 22 reports moderate complexity tier results for every task category and framework combination, supplementing the summary tables presented in the main body of this article.

Task Category	Framework	P50 Latency	P95 Latency	Cost per Task	Success Rate
Data extraction	LangGraph	1.4s	2.2s	\$0.028	97%
Data extraction	CrewAI	1.6s	2.6s	\$0.035	94%
Data extraction	Strands Agents	1.1s	1.8s	\$0.022	96%
Data extraction	watsonx Orchestrate	1.5s	2.4s	\$0.031	98%
Customer support	LangGraph	1.9s	3.0s	\$0.038	93%
Customer support	CrewAI	2.2s	3.5s	\$0.047	90%
Customer support	Strands Agents	1.6s	2.5s	\$0.030	94%
Customer support	watsonx Orchestrate	2.0s	3.2s	\$0.041	95%
Code generation	LangGraph	2.4s	3.8s	\$0.052	91%
Code generation	CrewAI	2.8s	4.4s	\$0.063	86%
Code generation	Strands Agents	2.0s	3.2s	\$0.041	90%
Code generation	watsonx Orchestrate	2.5s	4.0s	\$0.057	88%
Workflow automation	LangGraph	2.1s	3.3s	\$0.045	95%
Workflow automation	CrewAI	2.5s	4.0s	\$0.055	90%
Workflow automation	Strands Agents	1.8s	2.9s	\$0.036	93%
Workflow automation	watsonx Orchestrate	2.2s	3.5s	\$0.049	94%
Document summarization	LangGraph	1.6s	2.5s	\$0.033	96%
Document summarization	CrewAI	1.9s	3.0s	\$0.040	92%
Document summarization	Strands Agents	1.3s	2.1s	\$0.026	95%

Task Category	Framework	P50 Latency	P95 Latency	Cost per Task	Success Rate
Document summarization	watsonx Orchestrate	1.7s	2.7s	\$0.037	97%
Multi step research	LangGraph	2.9s	4.6s	\$0.061	88%
Multi step research	CrewAI	3.4s	5.3s	\$0.072	82%
Multi step research	Strands Agents	2.5s	3.9s	\$0.049	90%
Multi step research	watsonx Orchestrate	3.0s	4.8s	\$0.066	86%

Table 22. Full benchmark results by task category and framework, moderate complexity tier.

Appendix B. Glossary of Terms

Term	Definition
Agentic framework	A software library or platform that coordinates large language model calls, tool use, and multi step reasoning into an autonomous or semi autonomous workflow
Orchestration topology	The structural pattern by which multiple agents or workflow steps are arranged and coordinated, such as hierarchical, peer to peer, or graph based
State graph	A representation of a workflow as nodes, representing processing steps, and edges, representing transitions between steps
Tool call	An invocation of an external function, application programming interface, or system made by an agent during task execution
Enterprise readiness	The degree to which a framework satisfies production grade security, observability, scalability, and governance requirements
Latency percentile	A statistic describing the latency value below which a given percentage of measured requests fall, such as the 95th percentile
Error recovery rate	The proportion of errors from which a system continues toward successful task completion without manual intervention

Table 23. Glossary of terms used throughout this article.

REFERENCES

- 1) National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). January 2023.
- 2) National Institute of Standards and Technology. Secure Software Development Framework, SP 800 218. February 2022.
- 3) International Organization for Standardization and International Electrotechnical Commission. ISO/IEC 42001:2023, Information Technology, Artificial Intelligence, Management System. December 2023.
- 4) Open Worldwide Application Security Project. OWASP Top 10 for Large Language Model Applications, version 1.1. October 2023.
- 5) MITRE Corporation. MITRE ATLAS, Adversarial Threat Landscape for Artificial Intelligence Systems. Updated 2024.
- 6) Stanford University Institute for Human Centered Artificial Intelligence. The AI Index Report 2024. April 2024.
- 7) Partnership on AI. Guidance for Safe Foundation Model Deployment. October 2023.
- 8) American Institute of Certified Public Accountants. Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy. Revised 2022.
- 9) Cloud Security Alliance. Agentic AI Security Guidance. 2024.
- 10) World Economic Forum. Jobs of Tomorrow, Large Language Models and Jobs. September 2023.

- 11) Anthropic. Model Context Protocol Specification. November 2024.
- 12) OpenAI. Function Calling and Tool Use Documentation. 2023.
- 13) Google. Vertex AI Agent Builder Documentation. 2024.
- 14) Gartner. Market Guide for AI Agent Orchestration Platforms. 2024.
- 15) Forrester Research. The State of Agentic Artificial Intelligence. 2024.
- 16) METR. Autonomy Evaluation Resources. 2024.
- 17) European Parliament and Council of the European Union. Regulation (EU) 2016/679, General Data Protection Regulation. April 2016.
- 18) Council of the European Union. Provisional Political Agreement on the Artificial Intelligence Act. December 2023.