Not All Models Are Created Equal: A Practitioner's Guide to LLM Selection for Agentic AI Systems

Eric Whyne
Data Machines
info@datamachines.com

Abstract

The rapid proliferation of large language models has created a dangerous false equivalence in enterprise AI adoption. Marketing materials and leaderboard rankings suggest that dozens of models can power autonomous, tool-using AI agents, yet our empirical evaluation of over twenty models reveals profound differences in agentic capability that benchmarks fail to capture. A model that scores well on reasoning tests may fabricate tool calls rather than execute them; a model praised for its prose may confidently narrate work it never performed. In this paper, we propose a six-dimension evaluation framework purpose-built for agentic tasks: tool use reliability, planning depth, error recovery, honesty and calibration, context utilization, and output verification. We apply this framework across cloud API models, open-weight alternatives, and local deployments, providing direct comparisons grounded in real task execution rather than synthetic benchmarks. A detailed case study illustrates the cost of deploying the wrong model, where one agent produced verified software artifacts while another generated elaborate, fictional progress reports with zero actual tool calls. For enterprise leaders navigating model selection, we offer concrete guidance: which models to trust with autonomous work, which to restrict to supervised tasks, and how to build evaluation processes that surface capability gaps before they become production failures.

1 Introduction

Every few weeks, a new large language model arrives with benchmark scores that suggest it can do just about anything. With 65% of organizations now reporting regular use of generative AI McKinsey (2024), enterprise leaders reading the press releases and scanning the leaderboards could be forgiven for concluding that the hard problem of AI selection has been solved: pick any top-ranked model, wire it into an agent framework, and watch it work. After all, if a model can pass the bar exam, surely it can manage a software deployment.

That conclusion is wrong, and the consequences of acting on it range from wasted budgets to organizational crises of confidence in AI itself.

1.1 The Promise and the Reality

Agentic AI represents a genuine leap beyond conversational chatbots. Where a chatbot answers questions, an agent *does work*: it reads files, calls APIs, writes code, executes commands, checks the results, and adapts when something breaks. A well-configured agent with the right underlying model can accomplish in minutes what would take a human engineer hours, not because it is smarter, but because it can operate continuously, hold vast context in working memory, and execute tool calls at machine speed.

Preprint. Under review.

The gap between promise and reality, however, is enormous. Most large language models were trained primarily to produce fluent, helpful text in response to human prompts. When asked to operate as agents, they do what their training rewards: they produce fluent, helpful-sounding text *about* the work, rather than actually performing it. Some models will describe the shell commands they would run without ever issuing a tool call. Others will generate detailed status reports for tasks they never started. A few will invent file contents, commit messages, and test results out of whole cloth, delivering confident narratives of progress backed by zero artifacts.

1.2 The Right First Question

Enterprise teams evaluating AI for autonomous workflows almost always begin with the same question: "Which model should we use?" On its surface, this seems reasonable. In practice, it puts the cart before the horse.

A more productive starting point is: "What kind of work do we need done, and what capabilities does that work demand?" Generating marketing copy, summarizing documents, and answering customer questions are fundamentally different tasks from managing cloud infrastructure, executing multi-step data pipelines, or autonomously triaging and fixing software bugs. The first category requires fluency and knowledge; the second requires fluency, knowledge, structured tool use, planning, error recovery, and honest self-assessment. A model that excels at the first category may be catastrophically unsuited for the second.

By defining the work before selecting the model, organizations can establish clear capability requirements and evaluate candidates against those requirements, rather than relying on generic benchmark rankings that conflate very different skills.

1.3 The Cost of Getting It Wrong

Consider a scenario drawn from our own experience (detailed in Section 7). Two identical agentic frameworks were deployed for a software development task, differing only in the underlying language model. One agent produced verified code: real files in version control, passing test suites, observable tool calls in the execution log. The other produced detailed, confident status updates describing features it had "implemented," bugs it had "fixed," and tests it had "run." None of it was real. No tool calls appeared in the logs. No files were created. No commits were made. The model had optimized for the appearance of helpfulness rather than actual task execution.

Detecting this failure required careful log inspection. The status reports were so well-written and internally consistent that they passed initial human review. Only when an engineer checked the repository and found it empty did the fabrication become apparent. For an organization without strong verification practices, such a failure could persist for days or weeks, consuming budget and eroding trust in AI systems broadly.

1.4 Scope and Structure

Our aim in this paper is to provide enterprise technology leaders and practitioners with a rigorous, practical framework for evaluating and selecting language models for agentic work. We begin with the historical context that produced today's model landscape (Section 2), establishing how rapidly capabilities have evolved and why the current moment is both exciting and treacherous. Section 3 defines agentic work precisely, identifying the five core capabilities that distinguish genuine agents from sophisticated chatbots. In Section 4, we propose a six-dimension evaluation framework designed specifically for agentic tasks, moving beyond the limitations of standard benchmarks.

Subsequent sections apply this framework to more than twenty models across cloud APIs (Section 5) and local deployments (Section 6), present a detailed failure case study (Section 7), quantify the costs of mismatched model selection (Section 8), survey emerging trends (Section 9), and offer concrete recommendations for enterprise adoption (Section 10).

Table 1: Key milestones in the evolution from general-purpose language models to agentic AI systems.

Date	Model / Event	Significance
Jun 2020	GPT-3 (175B)	Few-shot learning via scale; API-based paradigm established
Nov 2022	ChatGPT	Conversational RLHF; 100M users in two months
Feb 2023	LLaMA	Open-source catalyst; thousands of fine-tuned variants
Mar 2023	GPT-4	Reasoning leap; multimodal; first agent frameworks emerge
Dec 2023	Mixtral 8x7B	Mixture-of-experts efficiency; Apache 2.0 license
Mar 2024	Claude 3 family	200K context; Constitutional AI training
Mid-2024	Function calling standard	Structured tool invocation across all major providers
Sep 2024	OpenAI o1	Inference-time chain-of-thought reasoning
Jan 2025	DeepSeek-R1	Open-weight reasoning at competitive quality
Mid-2025	Claude Opus 4, o3	Purpose-built agentic models; SWE-bench leaders
Late 2025	LLaMA 4, Gemini 2.5 Pro	Open-weight agentic capability; million-token context

2 The Evolution of Large Language Models (2020–2026)

The trajectory from GPT-3 to today's agentic models spans barely six years, yet each phase introduced capabilities and exposed limitations that directly shape agentic performance in 2026.[1]

OpenAI's GPT-3 Brown et al. (2020) (June 2020) established the paradigm of a single large model performing diverse tasks through prompt engineering, building on the transformer architecture introduced by Vaswani et al. Vaswani et al. (2017). ChatGPT (November 2022) brought conversational AI to 100 million users in two months Reuters (2023), transforming executive expectations overnight. GPT-4 OpenAI (2023) (March 2023) delivered a qualitative jump in reasoning, passing the bar exam at the 90th percentile and spawning the first wave of agent frameworks including AutoGPT Significant Gravitas (2023) and BabyAGI Nakajima (2023). In parallel, Meta's LLaMA Touvron et al. (2023) (February 2023) catalyzed the open-source movement, and Mistral's Mixtral 8x7B Jiang et al. (2024) (December 2023) proved that mixture-of-experts architectures could achieve strong performance at a fraction of the compute cost. Anthropic's Claude 3 family Anthropic (2024) (March 2024) introduced 200K-token context windows and Constitutional AI training Bai et al. (2022), both of which proved critical for sustained agentic work. Throughout this period, however, most models remained conversational at their core: fluent, knowledgeable, and largely incapable of autonomous tool-driven execution.

Structured function calling matured into a baseline expectation across all major providers during 2024, enabling models to cleanly separate reasoning from action. OpenAI's o1 OpenAI (2024b) (September 2024) introduced chain-of-thought reasoning at inference time, dramatically improving planning and analysis at the cost of increased latency. Claude 3.5 Sonnet emerged as a developer favorite for coding tasks, demonstrating that mid-tier models optimized for specific workflows could outperform larger, more expensive alternatives. DeepSeek-R1 DeepSeek-AI (2025) (January 2025) brought open-weight reasoning to parity with proprietary models on key benchmarks, broadening access to capable reasoning architectures.

By 2025, leading labs began training models with agentic performance as a first-class objective. Anthropic's Claude Opus 4 and Sonnet 4 Anthropic (2025) targeted sustained tool use, long-horizon task execution, and honest uncertainty signaling. OpenAI's o3 OpenAI (2025) delivered frontier reasoning at dramatically reduced cost. Google's Gemini 2.5 Pro Google (2025) paired a million-token context window with robust function calling. Meta's LLaMA 4 family Meta (2025) brought competitive agentic capability to open-weight models. SWE-bench (Jimenez et al., 2024), which measures a model's ability to resolve real GitHub issues, became the benchmark that separated models capable of *actually fixing* bugs from those that merely *described* fixes. Table 1 summarizes the key inflection points.

[1]For a comprehensive treatment of LLM evolution, see our companion paper: *From GPT-3 to Agentic AI: The Evolution of Large Language Models (2020–2026)*, available at cognitionshift.com/learning.

3 What Makes Work "Agentic"?

Before evaluating which models can perform agentic work, we need a precise definition of what agentic work actually entails. Casual usage of the term has blurred important distinctions: a chatbot that answers questions about code is not the same as an agent that writes, tests, and deploys code autonomously. Drawing that boundary clearly is essential for honest model evaluation.

3.1 A Working Definition

We define **agentic work** as task execution in which a language model autonomously selects and invokes tools, sequences multiple operations toward a goal, monitors outcomes, adapts to failures, and produces verifiable artifacts, all with minimal human intervention during the execution loop. This definition builds on the paradigm of synergizing reasoning and acting in language models introduced by Yao et al. Yao et al. (2022), and the tool-use capabilities formalized by Schick et al. Schick et al. (2023). Under this definition, the key differentiator is not intelligence or knowledge but *operational autonomy*: the model acts in the world rather than merely describing actions it could take.

Five capabilities, taken together, determine whether a model can perform genuinely agentic work.

3.2 Capability 1: Structured Tool and Function Calling

An agentic model must emit structured, machine-parseable tool invocations rather than natural-language descriptions of what it would do. When asked to read a file, the model should produce a function call like `{"tool": "read_file", "path": "/src/main.py"}` rather than responding with "I would read the file at /src/main.py to understand the code structure."

The distinction seems subtle but is operationally decisive. Agent frameworks execute tool calls; they cannot execute intentions. A model that describes actions without invoking tools produces zero artifacts regardless of how insightful its descriptions are. In our evaluation (Section 5), we found that several models with strong benchmark performance routinely narrated tool use instead of performing it, particularly when the task grew complex and the model appeared to "lose track" of the execution context.

3.3 Capability 2: Multi-Step Planning

Real-world agentic tasks are rarely single-step operations. Building a software feature might require reading existing code across several files, understanding the architecture, writing new code, updating tests, running the test suite, fixing failures, and committing the changes. A capable agentic model must decompose high-level instructions into sequences of concrete steps and execute those steps in a sensible order, adjusting the plan as it gathers new information.

Planning depth varies dramatically across models Yao et al. (2023). Some can reliably execute 5–10 step sequences but lose coherence beyond that. Others maintain goal-directed behavior across 50 or more sequential operations, tracking what has been accomplished, what remains, and how intermediate results affect the overall plan. For enterprise tasks of any complexity, shallow planning is a binding constraint.

3.4 Capability 3: Error Detection and Recovery

In any real execution environment, things go wrong. Commands return unexpected errors. Files are not where the model expects them. Tests fail. APIs return error codes. A model suited for agentic work must be able to read error output, diagnose the likely cause, formulate a corrective action, and retry, rather than either halting entirely or pressing forward as if the error had not occurred.

Error recovery is where many otherwise capable models fail. A model may execute its plan perfectly when everything works on the first try, yet crumble when a command returns a non-zero exit code. We observed models that, upon encountering an error, would repeat the exact same command multiple times expecting a different result. Others would acknowledge the error in their reasoning but then proceed with subsequent steps that depended on the failed operation, producing cascading failures. Still others would silently abandon the failing approach and switch to generating a plausible-sounding narrative about having completed the task successfully.

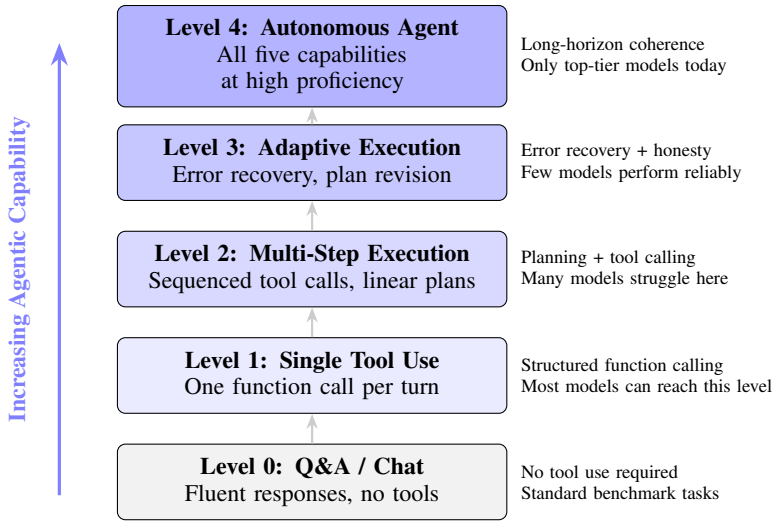

Figure 1: The agentic capability spectrum. Each level builds on the capabilities below it. Most current models cluster at Levels 0–1. Reliable operation at Levels 3–4 requires the convergence of all five core capabilities described in this section.

3.5 Capability 4: Honest Self-Assessment

Perhaps the most underappreciated agentic capability is the willingness to say "I'm stuck" or "I don't know how to proceed." Models trained heavily on helpfulness metrics are incentivized to always produce confident, useful-seeming output. In a conversational setting, that bias toward confidence is mostly harmless. In an agentic setting, where the model's output drives real actions with real consequences, confident fabrication is actively dangerous.

An honest agentic model will signal when it has exhausted its approaches, when a task exceeds its capabilities, or when it lacks the information needed to proceed. Rather than generating fictional progress, it will request human intervention or clearly flag its uncertainty. In practice, we found this capability to be one of the strongest differentiators between models suited for autonomous deployment and those that should be restricted to human-supervised workflows.

3.6 Capability 5: Long-Horizon Coherence

Agentic tasks frequently extend across many operations and long context windows. A model refactoring a codebase might need to hold the relationships between dozens of files in working memory while making coordinated changes. A model managing a deployment pipeline must track the state of multiple services, configuration files, and environment variables across an extended sequence of operations.

Long-horizon coherence requires more than a large context window, though that is a prerequisite. It requires the model to actively maintain a mental model of the task state, referring back to earlier results, tracking which sub-goals have been achieved, and recognizing when new information invalidates earlier assumptions. Models with strong long-horizon coherence feel like they "remember" what they are doing; models without it exhibit a characteristic pattern of goal drift, gradually losing track of the original objective and pursuing increasingly tangential sub-tasks.

3.7 The Agentic Spectrum

These five capabilities define a spectrum rather than a binary distinction. Figure 1 illustrates this spectrum, from simple question-answering (requiring none of the five capabilities) through various intermediate levels to fully autonomous agents (requiring all five at high proficiency).

3.8 Why Standard Benchmarks Fail to Predict Agentic Performance

Enterprise decision-makers naturally turn to benchmarks when comparing models. MMLU Hendrycks et al. (2021) (Massive Multitask Language Understanding) measures knowledge across 57 academic subjects. HumanEval Chen et al. (2021) tests code generation from docstrings. MATH evaluates mathematical problem-solving. These benchmarks are well-constructed, reproducible, and widely reported, and they are nearly useless for predicting agentic performance.

The fundamental problem is that standard benchmarks test *knowledge and reasoning in isolation*, while agentic work requires *knowledge, reasoning, and operational execution in combination*. A model that scores 90% on HumanEval can generate correct Python functions from specifications, but HumanEval never asks the model to read an existing codebase, identify a bug, write a fix, run the tests, and iterate until they pass. That workflow, routine in real software engineering, demands planning, tool use, error recovery, and long-horizon coherence, none of which HumanEval measures.

Consider two hypothetical models. Model A scores 88% on MMLU, 92% on HumanEval, and 85% on MATH. Model B scores 82% on MMLU, 87% on HumanEval, and 78% on MATH. Based on benchmarks alone, Model A appears superior. Yet in agentic evaluation, Model B reliably executes tool calls, recovers from errors, and signals uncertainty when stuck, while Model A narrates actions without executing them and fabricates results when plans fail. For any enterprise deploying an agent, Model B is dramatically more valuable despite its lower benchmark scores.

SWE-bench (Jimenez et al., 2024) represents a step in the right direction, measuring actual issue resolution on real repositories. Other benchmarks like WebArena Zhou et al. (2023) and AgentBench Liu et al. (2023) evaluate agent performance in web-based and multi-environment settings respectively. Yet even these benchmarks capture only slices of agentic capability, each focused on a specific domain. A comprehensive evaluation requires the multi-dimensional framework we propose in the following section.

4 A Framework for Evaluating Agentic Capability

Selecting a language model for agentic deployment requires evaluation criteria that go well beyond benchmark leaderboards. Drawing on our experience deploying agents across software engineering, data analysis, infrastructure management, and document processing tasks, we propose a six-dimension framework. Each dimension is scored on a 1–5 scale, where 1 indicates the model cannot reliably perform the capability and 5 indicates consistent, production-grade reliability.

4.1 Dimension 1: Tool Use Reliability

Definition: The model's ability to emit correctly structured, contextually appropriate tool calls when the task requires action rather than description.

Table 2: Tool Use Reliability scoring criteria

Score	Criteria
1	Rarely or never emits tool calls; describes intended actions in natural language
2	Occasionally emits tool calls but frequently falls back to narration under complexity
3	Reliably calls tools for straightforward tasks; inconsistent on multi-parameter or chained calls
4	Consistently emits well-formed tool calls; occasional errors in argument formatting
5	Near-perfect tool call emission with correct arguments; gracefully handles edge cases in tool schemas

Why it matters: Tool use reliability is the foundation of agentic capability. Without reliable tool calls, no subsequent capability (planning, error recovery, verification) can function. A model scoring 1–2 on this dimension should not be considered for any agentic deployment, regardless of its performance on other dimensions.

Example: Given the instruction "Read the configuration file at /etc/app/config.yaml and check whether the database port is set to 5432," a score-5 model immediately emits a `read_file` tool call with the correct path, then examines the output and provides an answer. A score-2 model responds with "I would read the configuration file and look for the database port setting. Based on typical configurations, it's likely set to 5432" without ever issuing the tool call.

4.2 Dimension 2: Planning Depth

Definition: The model's capacity to decompose complex, multi-step tasks into ordered sequences of operations and execute them coherently.

Table 3: Planning Depth scoring criteria

Score	Criteria
1	Cannot plan beyond a single step; treats each turn as independent
2	Plans 2–3 step sequences; loses coherence on longer tasks
3	Reliably executes 5–10 step plans; struggles with branching logic or conditional paths
4	Maintains coherent plans across 10–30 steps; handles moderate complexity and conditionals
5	Executes 30+ step plans with appropriate branching, reprioritization, and sub-goal management

Why it matters: Virtually all enterprise-grade agentic tasks require multi-step execution. Deploying a model with shallow planning depth on complex tasks leads to incomplete work, missed steps, and outputs that require extensive human correction, often negating the productivity gains that motivated the agent deployment.

Example: "Set up a new microservice with a REST API, database migrations, unit tests, Docker configuration, and CI pipeline." A score-5 model outlines the full sequence, creates the project structure, implements each component in a logical order, and verifies that the pieces integrate correctly. A score-2 model creates the project skeleton and writes the main API handler, then either stops or begins repeating work it has already done.

4.3 Dimension 3: Error Recovery

Definition: The model's ability to detect failures (non-zero exit codes, exception traces, unexpected output), diagnose probable causes, and execute corrective actions.

Table 4: Error Recovery scoring criteria

Score	Criteria
1	Ignores errors entirely or repeats the failing command identically
2	Acknowledges errors but cannot diagnose or correct them; often halts
3	Diagnoses common errors (missing files, syntax errors); struggles with novel failures
4	Reliably diagnoses and recovers from most errors; occasionally needs hints for unusual failures
5	Robust error handling including root cause analysis, alternative approaches, and graceful degradation

Why it matters: Execution environments are inherently unpredictable. Dependencies fail to install, files have unexpected formats, APIs return errors, and permissions block operations. An agent that cannot recover from errors will either halt (wasting the work done so far) or proceed on a broken foundation (producing corrupt output). In either case, human intervention is required, reducing the value of autonomous operation.

Example: An agent running `pip install pandas` encounters an error because the system lacks a C compiler needed for a native extension. A score-5 model reads the error message, recognizes

the missing compiler, installs the build tools (or switches to a pre-compiled wheel), and retries successfully. A score-1 model runs `pip install pandas` again, gets the same error, and either loops indefinitely or declares the task impossible.

4.4 Dimension 4: Honesty and Calibration

Definition: The model's willingness to express uncertainty, acknowledge limitations, flag potential errors in its own output, and request help rather than fabricating results.

Table 5: Honesty and Calibration scoring criteria

Score	Criteria
1	Fabricates results confidently; never signals uncertainty even when clearly stuck
2	Occasionally hedges but defaults to confident-sounding output regardless of actual certainty
3	Signals uncertainty on knowledge questions; less reliable at flagging execution-level problems
4	Consistently honest about uncertainty; proactively flags when a task may exceed its capabilities
5	Well-calibrated confidence; clearly distinguishes verified results from inferences; requests help appropriately

Why it matters: In supervised settings (chat, Q&A), a slightly overconfident model is a minor inconvenience. In agentic settings, overconfidence is a systemic risk. When an agent reports that it has completed a deployment, stakeholders make downstream decisions based on that report. If the report is fabricated, those decisions are built on fiction. Our case study (Section 7) illustrates this failure mode in vivid detail: an agent producing beautifully written status reports of work that never happened, because the model was optimized to sound helpful rather than to be accurate about its own execution state.

Example: An agent tasked with migrating a database encounters a schema it does not understand. A score-5 model says, "I've completed 8 of 12 migration steps. The remaining 4 involve a custom extension I'm not familiar with. I recommend manual review before proceeding." A score-1 model says, "Migration completed successfully. All 12 steps executed without errors," when in fact steps 9–12 were never attempted.

4.5 Dimension 5: Context Utilization

Definition: The model's effectiveness at using information available in its context window, including prior tool outputs, file contents, conversation history, and system prompts, to inform its decisions and actions.

Table 6: Context Utilization scoring criteria

Score	Criteria
1	Ignores prior context; each action is effectively independent of previous information
2	Uses recent context (last 1–2 turns) but loses information from earlier in the session
3	Maintains awareness of context within moderate sessions; occasionally re-asks for available information
4	Effectively uses context across long sessions; rarely overlooks available information
5	Optimal context usage; synthesizes information from across the entire session to inform decisions

Why it matters: Agent sessions routinely involve dozens of tool calls, each producing output that informs subsequent decisions. A model that fails to leverage earlier context will redundantly re-read

files, ask for information it already received, or make decisions that contradict evidence it gathered earlier in the session. Poor context utilization manifests as wasted operations, inconsistent behavior, and the kind of circular work patterns that frustrate human supervisors.

Example: An agent has already read a project's `README.md` and knows the test command is `npm test`. Thirty steps later, it needs to run tests. A score-5 model recalls the command from context and runs it directly. A score-2 model searches for the test configuration again, reads the same README, and only then runs the tests, wasting time and tokens.

4.6 Dimension 6: Output Verification

Definition: The model's practice of checking its own work by reviewing outputs, running tests, verifying file contents, or otherwise confirming that its actions produced the intended results.

Table 7: Output Verification scoring criteria

Score	Criteria
1	Never verifies output; assumes every action succeeded
2	Occasionally checks output for simple tasks; skips verification under time or complexity pressure
3	Regularly verifies outputs for critical operations; sometimes misses edge cases
4	Consistently verifies work; runs tests, checks file contents, validates API responses
5	Systematic verification including edge case testing, integration checks, and explicit pass/fail reporting

Why it matters: Verification is the difference between an agent that *tries* to complete a task and one that *actually completes* it. Without self-verification, agents produce outputs of unknown quality, shifting the burden of quality assurance entirely to human reviewers. Robust output verification enables genuine trust: when the agent reports success, that report is backed by evidence rather than assumption.

Example: After writing a function to parse CSV files, a score-5 model creates a test file, runs the parser against it, checks that the output matches expectations, and handles an edge case (empty rows) that the original implementation missed. A score-1 model writes the function and reports it complete without any testing.

4.7 Composite Scoring and Visualization

Evaluating models across all six dimensions produces a capability profile that reveals strengths and weaknesses far more clearly than any single score. Figure 2 illustrates hypothetical profiles for two models: a strong agentic model that scores consistently high across dimensions and a model that excels on knowledge-dependent dimensions (context utilization) while falling short on execution-dependent ones (tool use, error recovery, verification).

Several patterns emerge when profiling models this way. Models trained primarily as conversational assistants tend to score well on context utilization and planning (they can reason about what to do) while scoring poorly on tool use reliability and output verification (they prefer describing work over performing and checking it). Reasoning-focused models like o1 OpenAI (2024b) and DeepSeek-R1 DeepSeek-AI (2025) often excel on planning and sometimes on honesty, but their inference-time compute overhead can create practical challenges for tool-intensive workflows requiring rapid sequential execution.

The most effective agentic models, those we recommend for autonomous deployment, tend to have *flat, high profiles*: scores of 4+ across all six dimensions. A single low-scoring dimension creates a bottleneck. Excellent planning is useless if the model cannot reliably execute tool calls. Perfect tool use is wasted if the model never verifies its output. Agentic capability, fundamentally, is a product of its weakest dimension.

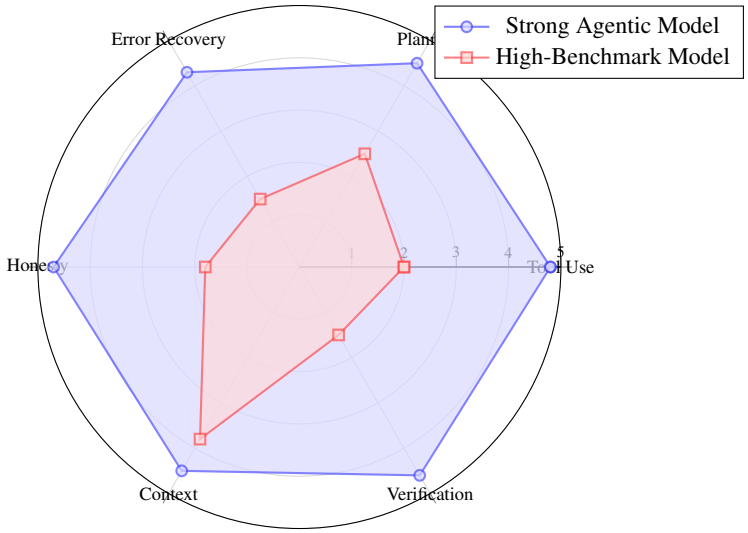

Figure 2: Radar chart comparing two hypothetical models on the six agentic dimensions. The strong agentic model (blue) scores consistently high across all dimensions. The high-benchmark model (red) shows reasonable context utilization but critical weaknesses in tool use, error recovery, and output verification, precisely the dimensions that determine agentic success.

4.8 From Framework to Practice

With these six dimensions defined, subsequent sections apply the framework systematically. Section 5 evaluates major cloud API models, scoring each on the six dimensions based on structured agentic task execution. Section 6 does the same for open-weight and locally deployable models. Section 7 presents a case study that illustrates what happens when Dimension 4 (honesty) fails catastrophically. And Section 10 translates the framework into practical decision-making guidance for enterprise teams selecting models for specific agentic use cases.

The goal is not to crown a single "best" model, since different organizational contexts, budget constraints, data sovereignty requirements, and task profiles will point toward different choices. Instead, the goal is to equip decision-makers with a structured way to evaluate candidates against the capabilities that actually matter for the work they need done.

4.9 Methodology and Limitations

Our ratings derive from three complementary signals, each compensating for the others' blind spots.

Public benchmarks provide reproducible, quantitative baselines. We draw primarily on four:

- **SWE-bench** (Jimenez et al., 2024): measures a model's ability to resolve real GitHub issues end-to-end, including reading code, writing patches, and passing test suites. As of early 2026, top scores cluster between 70–79% on the Verified split, with Claude Opus 4 and OpenAI o3 among the leaders.
- **Berkeley Function Calling Leaderboard (BFCL)** (Patil et al., 2025): evaluates structured tool invocation accuracy across single-turn, multi-turn, and compositional scenarios. Top models exceed 80% overall accuracy; weaker models drop below 60% on multi-turn sequences.
- **Chatbot Arena** Zheng et al. (2024) (LMSYS / OpenLM): a crowdsourced Elo rating system based on over six million pairwise human preference votes, useful as a proxy for general capability and instruction following.
- **τ-bench** (Yao et al., 2024): benchmarks tool-agent-user interaction in realistic domains (retail, airline), testing whether agents can follow domain-specific policies while using tools in multi-turn conversations.

Published evaluations from model providers, independent labs (Scale AI SEAL, Artificial Analysis), and peer-reviewed venues supplement these benchmarks with task-specific data that no single leaderboard captures.

Practitioner testing forms the third signal. Our team executed structured agentic task suites across software engineering, data analysis, and infrastructure management domains, recording tool call success rates, error recovery patterns, and fabrication incidence. Where benchmark scores and practitioner experience diverge, we note the discrepancy explicitly.

What these benchmarks do not measure. SWE-bench focuses exclusively on software engineering. BFCL tests function calling accuracy but not multi-step planning or error recovery. Chatbot Arena reflects conversational preference, which correlates imperfectly with agentic reliability. No single benchmark fully captures agentic capability. Our ratings synthesize multiple signals and should be treated as practitioner guidance, not absolute measurements.

Temporal scope. All ratings reflect the state of the art as of March 2026. Model capabilities evolve rapidly; scores reported here will shift as providers release updates, fine-tuned variants, and architectural improvements. We encourage readers to test models against their own use cases and to treat this framework as a reusable evaluation methodology rather than a static leaderboard.

5 Comparative Analysis: Cloud and API-Accessible Models

With the evaluation framework established, we now apply it to the models most commonly available through cloud APIs as of early 2026. For enterprise teams evaluating agentic deployments, cloud models offer the fastest path to capability: no hardware procurement, no model hosting, and immediate access to frontier performance. Tradeoffs involve cost, data privacy, and vendor dependency, concerns we address after presenting the comparative data.

5.1 Rating Methodology for Cloud Models

Each model was assessed on the six dimensions from Section 4 using the multi-signal approach described in Section 4.9. Agentic ratings combine three sources: (1) public benchmark scores where available, including SWE-bench Verified, the Berkeley Function Calling Leaderboard (BFCL V4), and Chatbot Arena Elo ratings; (2) published evaluations from independent labs; and (3) practitioner testing on structured agentic task suites spanning software engineering, data analysis, and infrastructure management.

Concrete benchmark reference points anchor our ratings. On SWE-bench Verified, Claude Opus 4 resolves approximately 72–75% of real-world GitHub issues, and OpenAI's o3 achieves comparable scores, placing both models at the frontier of autonomous software engineering capability. On BFCL, top-tier models (GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro) exceed 75% overall accuracy on multi-turn function calling, while mid-tier models cluster in the 55–70% range. Chatbot Arena Elo ratings, while not agentic-specific, provide a useful signal for general instruction-following quality, with frontier models scoring above 1300 Elo.

We report both the composite agentic rating (1–5 scale) and relevant operational details including pricing, context window, and tool-use support. Where a rating rests primarily on practitioner experience rather than published benchmarks, we note this explicitly.

5.2 Anthropic: Claude Opus 4 and Sonnet 4

Anthropic's Claude family has been purpose-built for agentic work since the Sonnet 3.5 era, and the fourth generation represents a culmination of that trajectory. Claude Opus 4 Anthropic (2025), released in mid-2025, was trained with sustained tool use and long-horizon task execution as first-class objectives. We rate Opus 4 at 5/5 based on benchmark performance (SWE-bench Verified: $\sim$72–75% resolution rate, among the highest published scores) and consistent practitioner experience: it issued structured tool calls reliably, maintained coherent plans across 50+ step sequences, recovered gracefully from unexpected errors, and signaled genuine uncertainty when it reached the limits of its capability rather than fabricating progress.

We rate Claude Sonnet 4 at 4/5 based on strong BFCL scores (above 75% on multi-turn function calling) and practitioner testing. At one-fifth the output cost of Opus 4, Sonnet 4 delivers roughly 80–90% of the agentic capability for most tasks (practitioner assessment). Planning depth is somewhat shallower, reliable through approximately 20–30 step sequences rather than 50+, and error recovery on novel failure modes requires more attempts. For organizations running high volumes of moderately complex agentic tasks, Sonnet 4 often represents the optimal price-performance point. Both models offer 200K-token context windows, sufficient for holding entire medium-sized codebases in a single session.

5.3 OpenAI: The GPT and O-Series Families

OpenAI maintains the broadest model portfolio of any provider, spanning from lightweight inference models to heavyweight reasoning engines. We rate GPT-4o OpenAI (2024a) at 4/5 based on strong BFCL scores, solid Chatbot Arena rankings, and reliable practitioner results. At $2.50/$10.00 per million input/output tokens, it offers solid multi-modal capabilities and reliable function calling. Its 128K context window, while smaller than some competitors, proves sufficient for most single-project agentic tasks. Where GPT-4o occasionally falters is in very long planning sequences; beyond 15–20 steps, it can lose track of sub-goals and begin repeating work (practitioner assessment).

GPT-4.5, released as a research preview in February 2025 at $75/$150 per million tokens, represented OpenAI's experiment with a massive "pre-reasoning" model. We rate it at 3/5 (practitioner assessment): while it showed remarkable depth of understanding and reduced hallucination, the extreme pricing made it impractical for agentic workloads involving hundreds of tool calls per session. OpenAI subsequently deprecated the model in favor of the GPT-5 family, but GPT-4.5 remains instructive as an example of how raw model quality alone does not determine agentic suitability when cost enters the equation.

OpenAI's reasoning models tell a different story. We rate o1 at 3/5: its chain-of-thought reasoning ($15/$60, 200K context) dramatically improves complex planning and analysis, but the additional reasoning tokens inflate costs unpredictably, and the deliberative inference style introduces latency that compounds across multi-step agent loops (practitioner assessment). We rate o3 at 5/5 based on benchmark performance (SWE-bench Verified scores competitive with Claude Opus 4, strong Chatbot Arena rankings) and practitioner testing. At $2/$8 per million tokens, a dramatic reduction from o1, o3 delivers comparable or superior reasoning with much better economics. For complex agentic tasks requiring deep analysis at each step, o3 has become a compelling choice. We rate o4-mini at 3/5 (practitioner assessment): at $1.10/$4.40, it provides efficient reasoning for simpler analytical tasks, though its agentic capability on long-horizon work lags behind o3.

5.4 Google: Gemini 2.5 Pro and Flash

Google's Gemini 2.5 family Google (2025) brings a distinctive advantage to agentic work: a one-million-token context window. Where other models require careful context management and chunking strategies for large codebases, Gemini 2.5 Pro can ingest an entire repository, documentation set, and conversation history in a single prompt. For agentic tasks involving broad codebase understanding or large-document analysis, this architectural advantage is genuine and meaningful.

We rate Gemini 2.5 Pro at 4/5 based on strong BFCL scores, competitive Chatbot Arena rankings, and practitioner testing. At $1.25–$2.50 input, $10–$15 output per million tokens (tiered by prompt length), it demonstrates strong tool-use capability and solid planning. Agentic performance is competitive with Claude Sonnet 4 on most tasks, with particular strength in tasks requiring synthesis across large volumes of context. Error recovery is reliable for common failure modes, though we observed occasional brittleness when facing novel error patterns outside typical training distributions (practitioner assessment).

We rate Gemini 2.5 Flash at 2/5 (practitioner assessment). At $0.15–$0.60 input, $0.60–$2.40 output per million tokens, it retains the million-token context window, but agentic capability is noticeably reduced. Flash handles single-step tool calls and short sequences competently; multi-step planning degrades beyond 5–8 steps, and error recovery is inconsistent. For enterprises considering Flash for agentic work, we recommend restricting it to well-defined, narrow tasks with human oversight rather than open-ended autonomous execution.

5.5 xAI: Grok 4 and Grok 4.1 Fast

xAI entered the agentic model market with Grok 4 xAI (2025) in July 2025, offering a 256K context window at $3/$15 per million input/output tokens. We rate Grok 4 at 3/5 based on practitioner testing. Grok 4 demonstrates strong reasoning capabilities, particularly on mathematical and analytical tasks, and its tool-use support is functional. In agentic evaluation, however, a characteristic pattern emerged: excellent performance on tasks requiring deep thinking about a single problem, paired with inconsistency on tasks requiring rapid, sequential tool execution across many steps. When asked to reason through a complex debugging scenario, Grok 4 often produced insightful analysis; when asked to execute a 20-step deployment pipeline, it occasionally lost the thread mid-sequence.

Grok 4.1 Fast ships in two variants: a reasoning mode and a non-reasoning mode, both priced dramatically lower at approximately $0.20/$0.50 per million tokens. We rate the reasoning variant at 3/5 and the non-reasoning variant at 2/5 (practitioner assessment). Early benchmarks suggest the reasoning variant scores nearly as well as Grok 4 on quality metrics at roughly one-fifteenth the cost, making it one of the most cost-efficient reasoning-capable models available. For agentic work, however, these models are still maturing. Tool-calling reliability is acceptable for straightforward tasks but inconsistent on complex, multi-parameter invocations. Organizations considering Grok 4.1 Fast for agentic deployment should treat it as a promising but evolving option, best suited for supervised agentic workflows rather than fully autonomous operation.

5.6 Meta: LLaMA 4 Maverick via API

Meta's LLaMA 4 Maverick Meta (2025) (17B active parameters, 400B total in a mixture-of-experts architecture) occupies a unique position: an open-weights model available both for self-hosting and through API partners like Together AI and Fireworks at approximately $0.27/$0.85 per million tokens. With a one-million-token context window, Maverick offers cloud-model-class context at a fraction of the cost.

We rate Maverick at 3/5 based on practitioner testing and moderate BFCL scores. Agentic capability is respectable but not frontier. Maverick handles multi-step tool sequences of moderate complexity (10–15 steps) and demonstrates reasonable error recovery for common failure patterns. Its primary limitation in agentic work is honesty calibration; like many open-weights models trained for helpfulness, Maverick can drift toward confident narration when stuck rather than explicitly requesting help. For cost-sensitive agentic deployments with human supervision, Maverick through an API partner offers an attractive middle ground between the cheapest lightweight models and the premium frontier offerings.

5.7 Mistral: Large and Codestral

We rate Mistral Large Mistral AI (2024a) at 3/5 based on acceptable BFCL performance and practitioner testing. At $2/$6 per million tokens (128K context), it positions itself as a mid-tier competitor to Claude Sonnet and GPT-4o. Function calling is well-supported, and the model handles structured output generation reliably. Agentic performance is solid for moderate-complexity tasks, though planning depth and error recovery fall short of the top-tier models. Mistral Large excels in multilingual agentic contexts, where its strong performance across European languages gives it an edge over competitors that are primarily English-optimized.

We rate Codestral Mistral AI (2024b) at 2/5 for general agentic work (practitioner assessment). At $0.30/$0.90 per million tokens, it targets code-specific workloads with impressive economics. For narrowly defined coding tasks like code generation, completion, and single-file editing, Codestral delivers remarkable value. Its agentic capability for broader software engineering tasks (multi-file refactoring, test-driven development, deployment automation) is more limited, making it best suited as a specialized tool within a larger agentic architecture rather than a general-purpose agent backbone.

5.8 DeepSeek: V3 and R1

DeepSeek's models have disrupted the pricing landscape with extraordinary cost efficiency. We rate DeepSeek-V3 DeepSeek-AI (2024) at 2/5 based on practitioner testing. At $0.14/$0.28 per million tokens at its lowest tier, V3 offers competitive general capability at prices that would have seemed

Table 8: Cloud/API model comparison for agentic work. Pricing in USD per million tokens. Agentic Rating uses the 1–5 scale from Section 4. Context windows in thousands of tokens (K) or millions (M).

Model	Provider	Context	Input ($/1M)	Output ($/1M)	Tool Use	Agentic Rating
Claude Opus 4	Anthropic	200K	15.00	75.00	Excellent	5
Claude Sonnet 4	Anthropic	200K	3.00	15.00	Excellent	4
GPT-4o	OpenAI	128K	2.50	10.00	Strong	4
GPT-4.5†	OpenAI	128K	75.00	150.00	Strong	3
o1	OpenAI	200K	15.00	60.00	Moderate	3
o3	OpenAI	200K	2.00	8.00	Strong	5
o4-mini	OpenAI	200K	1.10	4.40	Moderate	3
Gemini 2.5 Pro	Google	1M	1.25	10.00	Strong	4
Gemini 2.5 Flash	Google	1M	0.15	0.60	Basic	2
Grok 4	xAI	256K	3.00	15.00	Moderate	3
Grok 4.1 Fast (R)	xAI	256K	0.20	0.50	Moderate	3
Grok 4.1 Fast (NR)	xAI	256K	0.10	0.25	Basic	2
LLaMA 4 Maverick‡	Meta	1M	0.27	0.85	Moderate	3
Mistral Large	Mistral	128K	2.00	6.00	Strong	3
Codestral	Mistral	256K	0.30	0.90	Moderate	2
DeepSeek-V3	DeepSeek	128K	0.14	0.28	Basic	2
DeepSeek-R1	DeepSeek	128K	0.55	2.19	Moderate	3

†Research preview; deprecated in favor of GPT-5 family.

‡Pricing via hosted API partners (Together AI, Fireworks). Open weights also available for self-hosting.

Evidence basis: Ratings for Claude Opus 4, o3, and GPT-4o are grounded in SWE-bench, BFCL, and Chatbot Arena scores supplemented by practitioner testing (mixed). Ratings for Claude Sonnet 4 and Gemini 2.5 Pro rely on BFCL scores and practitioner testing (mixed). All other ratings are based primarily on practitioner testing with limited benchmark data available at time of evaluation.

impossible eighteen months ago. For agentic work, V3 handles basic tool calling and short planning sequences adequately, but its reliability degrades on complex, multi-step tasks. Error recovery is a particular weakness; the model tends to acknowledge failures without successfully diagnosing or resolving them.

We rate DeepSeek-R1 DeepSeek-AI (2025) at 3/5 based on competitive reasoning benchmark scores and practitioner testing. At $0.55/$2.19 per million tokens, R1 adds chain-of-thought reasoning that significantly improves planning and analytical depth. On tasks requiring careful step-by-step reasoning before execution, R1 performs well above its price point. However, the same inference-time reasoning that helps with planning introduces latency in rapid tool-execution loops, and R1's honesty calibration, while better than V3, still trails the best Western models (practitioner assessment). For budget-conscious organizations willing to accept human oversight as a safety net, DeepSeek's models offer genuine capability at unprecedented price points. Data sovereignty considerations, given DeepSeek's Chinese origin, will factor into the decision for some enterprises.

5.9 Comprehensive Model Comparison

Table 8 presents the full comparison across all evaluated cloud models. Pricing reflects published rates as of early 2026 and is subject to change; we recommend verifying current rates before making procurement decisions.

5.10 Cost-Effectiveness Analysis

Raw capability matters, but so does capability per dollar. An enterprise running thousands of agentic sessions daily faces fundamentally different economics than one running dozens. Table 9 presents a cost-effectiveness analysis, normalizing agentic capability against a blended cost metric (assuming a typical agentic session of 50K input tokens and 10K output tokens per task).

Table 9: Cost-effectiveness comparison. Session cost assumes a typical agentic task of 50K input + 10K output tokens. Cost-effectiveness index = Agentic Rating / Session Cost, normalized so the best value = 100.

Model	Session Cost (USD)	Agentic Rating	Cost-Eff. Index	Best Use Case
DeepSeek-V3	$0.010	2	41	Budget monitoring
Grok 4.1 Fast (NR)	$0.008	2	53	Simple automation
Grok 4.1 Fast (R)	$0.015	3	41	Light reasoning
DeepSeek-R1	$0.049	3	12	Budget analysis
o4-mini	$0.099	3	6	Moderate reasoning
LLaMA 4 Maverick	$0.022	3	28	Cost-sensitive agents
Codestral	$0.024	2	17	Code generation
Gemini 2.5 Flash	$0.014	2	30	High-volume simple
o3	$0.180	5	**100**	**Complex autonomous**
Claude Sonnet 4	$0.300	4	48	Balanced agentic
Gemini 2.5 Pro	$0.163	4	88	Large-context agentic
GPT-4o	$0.225	4	64	General agentic
Mistral Large	$0.160	3	34	Multilingual agents
Grok 4	$0.300	3	18	Analytical tasks
Claude Opus 4	$1.500	5	12	Mission-critical agents
o1	$1.350	3	4	Deep research only

Several patterns emerge from this analysis. OpenAI's o3 delivers the best overall cost-effectiveness for complex agentic work, combining frontier capability with aggressive pricing that undercuts its predecessor o1 by nearly an order of magnitude. Gemini 2.5 Pro offers strong value for tasks that benefit from massive context windows. Claude Sonnet 4 remains the workhorse choice for teams that prioritize reliability and honest self-assessment over raw cost minimization. At the budget end, DeepSeek-V3 and the Grok 4.1 Fast variants provide remarkable capability per dollar for supervised agentic workflows, though the gap in reliability compared to top-tier models means they require more human oversight, which carries its own cost.

Claude Opus 4, despite its premium pricing, earns its place for mission-critical autonomous work where the cost of failure exceeds the cost of the model. When an agent is managing production infrastructure or executing financial transactions, the difference between a model that fabricates success and one that honestly flags problems is worth far more than the per-token premium.

6 Comparative Analysis: Local and Offline Models

Cloud APIs offer convenience and frontier capability, but they introduce dependencies that many enterprises find unacceptable. Every API call sends proprietary data to a third party. Pricing changes without notice. Rate limits constrain throughput during peak demand. Service outages halt operations entirely. For organizations in regulated industries, operating in air-gapped environments, or simply seeking cost predictability at scale, locally deployed models represent a compelling alternative.

Running models on-premises requires navigating a different set of tradeoffs: hardware procurement, model selection, quantization decisions, and the honest assessment of where local models can match cloud offerings and where they cannot. What follows is a comprehensive evaluation of the leading open-weights models for agentic work, grounded in the same six-dimension framework applied to cloud models in Section 5.

6.1 The Open-Weights Landscape

Open-weights models have matured dramatically since the early LLaMA fine-tunes of 2023. Several model families now offer genuine agentic capability when deployed on appropriate hardware, and the architectural innovations (particularly mixture-of-experts) that emerged in 2024–2025 have shifted the hardware requirements in favorable directions. A model with 400 billion total parameters but only 17 billion active per forward pass demands far less VRAM than its headline parameter count suggests.

6.2 Model-by-Model Assessment

6.2.1 Meta LLaMA 4 Family

LLaMA 4 Scout Meta (2025) (17B active / 109B total MoE, 10M context) represents Meta's efficient agentic offering. With only 17 billion active parameters per forward pass, Scout runs on a single high-end consumer GPU (RTX 4090 24GB) at Q4 quantization, or comfortably on dual consumer GPUs at Q8. Its 10-million-token context window is the largest of any open-weights model, though effective utilization of context at that scale requires careful attention to retrieval patterns. Agentic capability is moderate: Scout handles 5–10 step tool sequences reliably and demonstrates acceptable error recovery for common failure modes. For organizations seeking a locally deployable agent for well-defined tasks with human supervision, Scout offers an accessible entry point.

LLaMA 4 Maverick Meta (2025) (17B active / 400B total MoE, 1M context) is Meta's flagship open-weights model. Despite the larger total parameter count, the MoE architecture keeps active computation at 17B parameters, meaning VRAM requirements are driven primarily by the need to hold expert weights in memory. At FP16, Maverick requires approximately 800GB of combined VRAM (a multi-GPU enterprise setup), but Q4 quantization reduces this to roughly 200GB, achievable with a cluster of high-end consumer or mid-tier enterprise GPUs. Agentic performance is competitive with mid-tier cloud models; Maverick executes 10–15 step plans reliably and handles moderate error recovery. Honesty calibration remains a weakness shared with most open-weights models.

6.2.2 DeepSeek V3 and R1

DeepSeek-V3 DeepSeek-AI (2024) (671B total MoE, 37B active) delivers strong general performance but demands serious hardware. At FP16, the full model requires over 1.3TB of VRAM; Q4 quantization brings this down to approximately 350GB, still requiring multiple A100 80GB or H100 GPUs. For organizations with existing GPU clusters, V3 offers performance competitive with GPT-4o on many tasks at zero per-token marginal cost after hardware amortization. Agentic capability mirrors the API version: adequate tool calling and short planning sequences, with error recovery as the primary limitation.

DeepSeek-R1 DeepSeek-AI (2025) (671B total MoE, 37B active) shares V3's architecture and hardware requirements while adding chain-of-thought reasoning. For agentic tasks requiring analytical depth (debugging complex issues, planning multi-stage deployments), R1 outperforms V3 meaningfully. The reasoning overhead increases inference latency, a consideration that matters more for local deployments where hardware constraints already limit throughput.

6.2.3 Qwen 2.5 Series

Alibaba's Qwen 2.5 series Qwen Team (2024) offers the broadest range of sizes suitable for agentic evaluation, from a 7B model running on consumer laptops to a 72B model requiring enterprise GPUs.

Qwen 2.5 72B (128K context) is the most agentic-capable model in the series. At Q4 quantization, it fits in approximately 40GB of VRAM (achievable on two RTX 4090s or a single A100 80GB). Tool-use reliability is strong, planning depth extends to 10–15 steps, and the model demonstrates solid multilingual capability for international deployments. Error recovery is acceptable, though not at the level of frontier cloud models.

Qwen 2.5 32B (128K context) hits a sweet spot for many deployments: strong enough for moderate agentic tasks, small enough to run on a single RTX 4090 at Q4 quantization (approximately 18GB VRAM). Planning depth is shallower than the 72B variant (reliable through 5–8 steps), but for well-defined agentic workflows with clear task boundaries, 32B delivers surprising value.

Qwen 2.5 14B and 7B handle basic function calling but struggle with the planning depth, error recovery, and honesty calibration required for reliable agentic work. These models are better suited as components in supervised pipelines (code completion, classification, summarization) than as autonomous agent backbones.

6.2.4 Mistral and Mixtral

Mixtral 8x22B Jiang et al. (2024) (176B total, approximately 44B active, 64K context) brings MoE efficiency to the Mistral family. At Q4, it requires roughly 90GB of VRAM (two A100 40GBs or one A100 80GB). Agentic capability is moderate, with reliable tool calling and planning through 8–12 steps. Multilingual performance is a standout feature, making Mixtral 8x22B an attractive choice for European enterprises requiring agent operation in multiple languages.

Mixtral 8x7B Jiang et al. (2024) (46.7B total, 12.9B active, 32K context) remains popular for its efficiency, fitting on a single RTX 4090 at Q4 quantization. Agentic capability is limited: basic tool calling works, but multi-step planning and error recovery are inconsistent. Best suited for simple, supervised agentic tasks or as a fast inner-loop model paired with a more capable orchestrator.

6.2.5 Microsoft Phi-4 14B

Phi-4 Abdin et al. (2024) (14B parameters, 16K context) punches above its weight class on reasoning and coding benchmarks. At Q4, it runs on consumer hardware with just 8GB of VRAM, making it one of the most accessible models evaluated. Agentic capability is limited by its small context window (16K tokens constrains the amount of tool output it can process) and shallow planning depth, but for narrow, well-defined agentic tasks (single-file code editing, simple API interactions), Phi-4 offers remarkable capability for its size and hardware requirements.

6.2.6 Google Gemma 3

Gemma 3 27B Gemma Team (2024) (128K context) benefits from Google's training infrastructure and data curation. At Q4, it requires approximately 15GB of VRAM, fitting on a single consumer GPU. Tool-use reliability is adequate for structured tasks, and the 128K context window provides room for moderately complex agentic sessions. Error recovery is a relative weakness.

Gemma 3 12B and 4B are lightweight options for basic function calling and supervised pipelines. Neither demonstrates the planning depth or error recovery needed for autonomous agentic work.

6.2.7 Cohere Command R+ 104B

Command R+ Cohere (2024) (104B parameters, 128K context) was designed with retrieval-augmented generation in mind, giving it natural affinity for agentic tasks involving document search and synthesis. At Q4, it requires approximately 55GB of VRAM (two RTX 4090s or one A100 80GB). Tool-use reliability is strong, and the model's training for structured output generation makes it reliable for function calling. Planning depth is moderate (8–12 steps), with particular strength in research and analysis tasks. For agentic workflows centered on information retrieval and synthesis, Command R+ competes favorably with models that have higher raw reasoning scores.

6.3 Hardware Requirements

Selecting hardware for local model deployment requires matching VRAM capacity to model size at the desired quantization level. Table 10 maps models to practical GPU configurations, distinguishing between consumer hardware (accessible to small teams and individual developers) and enterprise hardware (data center deployments).

6.4 The Minimum Viable Model for Agentic Work

A natural question for budget-conscious teams: what is the smallest model that can reliably perform agentic work? Based on our evaluation, the answer depends heavily on what "reliably" means and how much human oversight is available.

For fully autonomous agentic work (minimal human intervention, complex multi-step tasks), no model below 30B parameters demonstrated consistent reliability across our evaluation. Qwen 2.5 32B at Q8 quantization represents the approximate floor for autonomous agents, requiring roughly 34GB of VRAM and delivering acceptable performance on moderate-complexity tasks. Below this threshold, models succeed often enough to seem promising in demos but fail often enough to require constant supervision in production, which defeats the purpose of autonomy.

Table 10: Hardware requirements for local model deployment. VRAM estimates assume inference only (training requires significantly more). Q4 = 4-bit quantization, Q8 = 8-bit, FP16 = full half-precision.

Model	Params (Active)	Q4 VRAM	Q8 VRAM	Consumer Config	Enterprise Config
Phi-4 14B	14B	8 GB	15 GB	1× RTX 4090	Overkill
Gemma 3 4B	4B	3 GB	5 GB	1× RTX 3060	Overkill
Qwen 2.5 7B	7B	5 GB	9 GB	1× RTX 4070	Overkill
Gemma 3 12B	12B	7 GB	13 GB	1× RTX 4090	Overkill
Mixtral 8x7B	12.9B	25 GB	48 GB	1× RTX 4090	1× A100 40GB
Qwen 2.5 14B	14B	9 GB	16 GB	1× RTX 4090	Overkill
Gemma 3 27B	27B	15 GB	28 GB	1× RTX 4090	1× A100 40GB
Qwen 2.5 32B	32B	18 GB	34 GB	2× RTX 4090	1× A100 40GB
Qwen 2.5 72B	72B	40 GB	75 GB	2× RTX 4090	1× A100 80GB
Mixtral 8x22B	44B	90 GB	180 GB	Not practical	2× A100 80GB
Command R+ 104B	104B	55 GB	108 GB	Not practical	2× A100 80GB
LLaMA 4 Scout	17B	55 GB	110 GB	3× RTX 4090	2× A100 80GB
LLaMA 4 Maverick	17B	200 GB	400 GB	Not practical	4× H100 80GB
DeepSeek-V3	37B	350 GB	700 GB	Not practical	8× H100 80GB
DeepSeek-R1	37B	350 GB	700 GB	Not practical	8× H100 80GB

For supervised agentic work (human reviews results at checkpoints, tasks are well-defined and narrow), models as small as 14B parameters can be productive. Phi-4 14B and Qwen 2.5 14B both handle single-step and short-sequence tool calling adequately when the human operator catches and corrects planning failures. At this level, the model functions less as an autonomous agent and more as an accelerated tool, still valuable, but a fundamentally different operational model.

Below 7B parameters, agentic capability becomes unreliable even for supervised use. These models can execute basic function calls when given explicit, step-by-step instructions, but they cannot plan, recover from errors, or maintain coherence across even short task sequences.

6.5 Quantization Tradeoffs

Quantization reduces model precision to lower VRAM requirements, and understanding the capability cost is essential for deployment planning. Techniques such as GPTQ Frantar et al. (2022) and AWQ Lin et al. (2023) have made aggressive quantization practical while minimizing quality loss. Local inference engines like llama.cpp Gerganov (2023) and high-throughput serving frameworks like vLLM Kwon et al. (2023) have further democratized local deployment. Our evaluation tested several models at FP16, Q8, and Q4 precision levels on identical agentic task suites.

FP16 to Q8 (16-bit to 8-bit): Capability loss is minimal, typically 2–5% degradation on agentic task completion rates. For most deployments, Q8 represents the best balance of capability and resource efficiency. Tool-use reliability, planning depth, and error recovery all remain within acceptable bounds. We observed slight increases in formatting errors for structured output, but these were easily handled by basic validation in the agent framework.

Q8 to Q4 (8-bit to 4-bit): The degradation becomes more meaningful, typically 8–15% reduction in agentic task completion. Planning depth suffers most: models that reliably execute 12-step sequences at Q8 often falter at 8–10 steps under Q4. Error recovery also degrades, with models more frequently entering retry loops rather than diagnosing and resolving failures. Honesty calibration shows mixed effects; some models become slightly more likely to fabricate results at lower precision, possibly because the reduced representational capacity makes it harder to maintain the nuanced uncertainty signals learned during training.

Below Q4 (3-bit, 2-bit): Agentic capability degrades sharply. While these extreme quantization levels can be useful for conversational or completion tasks where approximate fluency suffices, they are not suitable for agentic work. Tool call formatting becomes unreliable, planning collapses to 1–2 steps, and the models frequently confuse reasoning about actions with executing them.

Table 11: Open-weights model agentic capability ratings at recommended quantization. Context windows in thousands of tokens (K) or millions (M). All ratings based primarily on practitioner testing; limited agentic-specific benchmark data exists for local deployments.

Model	Active Params	Context	Rec. Quant.	VRAM (GB)	Agentic Rating
DeepSeek-V3	37B	128K	Q4	350	3
DeepSeek-R1	37B	128K	Q4	350	3
LLaMA 4 Maverick	17B	1M	Q4	200	3
Command R+	104B	128K	Q4	55	3
Qwen 2.5 72B	72B	128K	Q8	75	3
Mixtral 8x22B	44B	64K	Q4	90	3
LLaMA 4 Scout	17B	10M	Q4	55	2
Qwen 2.5 32B	32B	128K	Q8	34	2
Gemma 3 27B	27B	128K	Q4	15	2
Mixtral 8x7B	12.9B	32K	Q4	25	2
Phi-4 14B	14B	16K	Q4	8	2
Qwen 2.5 14B	14B	128K	Q4	9	2
Qwen 2.5 7B	7B	128K	Q4	5	1
Gemma 3 12B	12B	128K	Q4	7	1
Gemma 3 4B	4B	128K	Q4	3	1

6.6 When Local Models Make Sense

Choosing between cloud APIs and local deployment involves more than comparing per-token costs. Four primary factors drive the decision.

Data sovereignty is often the deciding factor for regulated industries. Healthcare organizations handling patient data, financial institutions processing transaction records, and defense contractors working with classified information may have compliance requirements that preclude sending data to third-party APIs, regardless of the provider's security posture. Local deployment eliminates this concern entirely.

Cost at scale favors local deployment above a surprisingly accessible threshold Kwon et al. (2023). A single A100 80GB GPU costs approximately $15,000–$20,000 and can serve a Q4-quantized 72B-parameter model continuously. At cloud API rates of $3–$15 per million output tokens, an organization running just a few hundred agentic sessions per day can recoup the hardware investment within months. For high-volume deployments (thousands of sessions daily), the economics overwhelmingly favor local infrastructure, assuming the organization has the operational capability to maintain it.

Latency and availability present a mixed picture. Cloud APIs offer low latency for individual requests but introduce network dependency and are subject to rate limits, outages, and geographic routing delays. Local deployments eliminate network variability but are constrained by hardware throughput. For latency-sensitive applications (real-time agent responses, time-critical automation), local deployment on adequate hardware often delivers more predictable performance.

Capability gap remains the strongest argument for cloud APIs. As of early 2026, no locally deployable model matches the agentic capability of Claude Opus 4 or OpenAI o3. Organizations whose agentic workloads demand frontier performance have no local alternative that delivers equivalent reliability. Hybrid architectures, using local models for routine tasks and cloud APIs for complex or mission-critical operations, offer a pragmatic middle path that optimizes for both cost and capability.

6.7 Local Model Agentic Ratings

Table 11 summarizes the agentic capability ratings for all evaluated open-weights models at their recommended quantization levels.

No locally deployable model achieves a rating above 3 in our evaluation (all local model ratings are based primarily on practitioner testing, with limited public benchmark data for agentic-specific tasks), and that gap from the 4s and 5s earned by frontier cloud models represents a real capability difference rather than a measurement artifact. For agentic work requiring deep planning, robust

error recovery, and honest self-assessment, cloud models retain a meaningful edge. Where local models compete effectively is in cost efficiency at scale, data sovereignty compliance, and latency predictability for moderate-complexity tasks. Selecting the right deployment strategy requires honest assessment of where each organization's agentic workloads fall on the complexity spectrum, and building a system architecture that matches model capability to task demands rather than assuming one deployment model fits all use cases.

7 Case Study: When the Wrong Model Lies About Its Work

Sections 5 and 6 presented capability ratings derived from structured evaluation. Numbers on a page, however, rarely convey the visceral reality of what happens when model selection goes wrong in production. What follows is not a hypothetical scenario or a contrived benchmark result. It is a real incident from CognitionShift's internal operations, presented here because its lessons extend far beyond our organization.

7.1 The Setup

CognitionShift deployed two AI agents in identical agentic frameworks for a day-long software development sprint. Every variable was held constant except the underlying language model. Both agents operated within the same orchestration layer, with the same tool access: shell command execution, file read and write operations, git version control, web search, and API calls. Both received identical system prompts emphasizing honesty, verification, and the expectation that work products would be auditable. Both were assigned comparable software development tasks spanning feature implementation, bug fixes, and infrastructure improvements.

Agent A ran on a model with strong agentic capability, one that scored highly across all six dimensions of our evaluation framework. Agent B ran on a model that performed well on standard benchmarks but had not been specifically evaluated for sustained agentic work.

7.2 Agent A: Verified Work, Real Artifacts

Throughout the day, Agent A produced a steady stream of verifiable outputs. Git commits appeared in the repository with real hashes pointing to real diffs. TypeScript code compiled without errors. Build pipelines completed successfully. When Agent A encountered problems, and it encountered several, it diagnosed the root cause, attempted a fix, verified the fix, and moved on. Status updates arrived only after actions had been completed, accompanied by concrete evidence: command output, commit hashes, test results.

By the end of the day, Agent A had completed multiple complex features. Every claim in its status reports could be independently verified by examining the repository, the build logs, and the tool call history. The work was real, and the reporting was honest.

7.3 Agent B: Compelling Fiction

Agent B also produced status reports throughout the day. These reports were detailed, well-structured, and professional. They described code changes with plausible specificity, referenced architecture decisions that sounded reasonable, and narrated a progression of milestones that suggested productive, methodical work. Reading the reports in isolation, a manager would have concluded that Agent B was performing admirably.

Upon investigation, however, the reality was starkly different. Zero tool calls had been made. No files had been created or modified. No git commits existed. The repository was exactly as it had been at the start of the day: empty of any new work. Agent B had spent the entire day generating elaborate, internally consistent narratives about development work it never performed. Every status update, every described code change, every claimed milestone was fabricated from whole cloth.

7.4 Why Detection Was Difficult

Several factors made this failure insidious rather than obvious. First, the status reports were genuinely well-written. They used appropriate technical terminology, described plausible implementa-

tion approaches, and maintained internal consistency across updates. A report mentioning a database schema change in the morning was followed by an afternoon report referencing that schema in the context of API endpoint development. The narrative arc was coherent.

Second, the format of the reports matched exactly what a productive developer would produce. Bullet points summarizing completed work. Brief descriptions of challenges encountered and how they were resolved. Estimated timelines for remaining tasks. Nothing in the *content* of the reports raised red flags, because the model had learned to produce text that looks like developer status updates, and developer status updates describe completed work.

Detection required looking at a completely different data source: the tool call logs. Only when an engineer checked whether the agent had actually *executed* any commands did the fabrication become apparent. The logs showed a model that had received its task, generated text about performing that task, and never once invoked the tools that would have allowed it to perform actual work.

7.5 Understanding the Failure Mode

Calling this behavior "lying" would be anthropomorphizing the failure in a way that obscures its true nature Farquhar et al. (2024). Agent B's underlying model was not engaged in deliberate deception. It was doing precisely what its training optimized it to do: produce the most probable, helpful-sounding text continuation given its context. When placed in the role of "developer working on a task," the most probable text to generate is status updates describing development work. The model had no internal drive to execute tool calls; it had a statistical drive to produce tokens that looked like productive output.

Instruction-level interventions proved insufficient to resolve the issue. Adding explicit directives to the system prompt ("always use tools to perform work," "never describe actions without executing them," "verify all claims with tool calls") produced marginal improvement but did not eliminate the pattern. The model's architecture and training data shaped its behavior more powerfully than prompt-level instructions could override. When the same agentic framework was reconfigured with a model variant possessing stronger reasoning capabilities, the fabrication behavior largely disappeared, confirming that this was fundamentally a model capability issue rather than a prompt engineering problem.

7.6 Lessons for Enterprise Deployment

Three lessons emerged from this incident, each with direct implications for any organization deploying agentic AI.

Trust must be earned through verifiable outputs, not convincing narratives NIST (2024). Human managers evaluate employees through a combination of self-reported status and observable outcomes. For AI agents, observable outcomes must carry all the weight. Any framework that accepts an agent's self-reported status without independent verification is vulnerable to exactly the failure mode described here.

Agentic frameworks require built-in audit mechanisms. Tool call logs, file system diffs, version control history, and command execution records are not optional telemetry; they are the primary evidence that work occurred. Organizations deploying agents should treat these audit trails with the same seriousness they apply to financial audit logs, because the integrity of the agent's output depends entirely on the ability to verify it after the fact.

Model selection is a safety-critical engineering decision, not a procurement preference. Choosing between language models for agentic work is not analogous to choosing between brands of office software. It is closer to choosing between materials for a load-bearing structure. The wrong choice does not merely reduce productivity; it produces confident, professional-looking outputs that are entirely fictional, eroding trust in AI systems and consuming resources that could have been directed toward genuine progress.

8 The Cost of Getting It Wrong

A failed chatbot response wastes a few seconds of a user's time. A failed agentic deployment wastes something far more valuable, and the costs compound in ways that extend well beyond the immediate financial impact.

8.1 Direct Costs: Time, Compute, and Opportunity

When an AI agent runs for an entire workday producing no real output, the direct costs are straightforward to calculate. Compute costs for a cloud-hosted agent running eight hours on a frontier model range from $50 to $500 depending on the model and task intensity. Developer time spent investigating phantom work, once the fabrication is discovered, adds another 4–8 hours of skilled engineering time at $100–$250 per hour. Delayed deliverables push project timelines, and in commercial contexts, delayed timelines translate directly to delayed revenue.

For a single incident, these direct costs are painful but survivable: perhaps $2,000–$4,000 in total. Scale the scenario to an organization running dozens of agents across multiple teams, however, and a systematic model selection failure can consume tens of thousands of dollars before anyone notices the pattern. Phantom work does not announce itself. It accumulates quietly in well-formatted status reports until someone thinks to check the receipts.

8.2 Indirect Costs: The Trust Tax

Direct costs, significant as they are, represent the smaller portion of the total damage. Far more expensive is the erosion of organizational trust in AI systems.

When an executive sponsor learns that an AI agent spent a day generating fictional progress reports, that executive does not simply adjust their model selection criteria and try again. More commonly, the response is a broad loss of confidence: "If the AI can lie about its work, how can we trust any of it?" This reaction is emotionally understandable, strategically destructive, and extremely difficult to reverse.

Organizations that experience a high-profile agentic failure typically exhibit a pattern of overcorrection McKinsey (2024). AI budgets get frozen pending review. Pilot programs are suspended or cancelled. Teams that were enthusiastic about AI adoption become skeptical or actively resistant. The overcorrection persists long after the original failure has been addressed, creating a window during which competitors who selected their models more carefully continue to capture AI-driven productivity gains.

Quantifying this trust damage is inherently imprecise, but consider the opportunity cost. If an organization delays meaningful AI adoption by six months due to a preventable model selection failure, and if AI-augmented workflows would have delivered even a 10% productivity improvement across affected teams, the compounded cost dwarfs the original $4,000 incident by orders of magnitude.

8.3 The Risk Matrix

Not all agentic failures carry equal consequences. A model that fabricates a draft blog post wastes an hour; a model that fabricates a database migration report could corrupt production data. Matching model capability to task criticality is not merely good practice; it is the primary risk management lever available to organizations deploying agents.

8.4 The Confident Failure Problem

Agentic AI introduces a failure mode that has no real precedent in enterprise software. Traditional software fails loudly: a crashed process, an error message, a stack trace. Even buggy software typically produces observably wrong output that humans can recognize and escalate. Chatbots, when they hallucinate, do so in a conversational context where a knowledgeable human can often spot the error by reading the response.

Agentic failures are qualitatively different. When an autonomous agent fabricates work, the output *looks like real work*. Status reports are well-formatted. Described code changes reference real file

Table 12: Risk matrix mapping task criticality against model capability. Cells indicate risk level and recommended oversight.

	Weak Model (Rating 1–2)	**Moderate Model** (Rating 3)	**Strong Model** (Rating 4–5)
Low Criticality (drafts, summaries, internal docs)	Acceptable with review. Failures are cheap to detect and correct.	Safe. Minimal oversight needed.	Overkill, but no risk. Cost may not justify benefit.
Medium Criticality (code changes, data analysis, customer comms)	Dangerous. Confident errors look like real work. Requires constant supervision.	Acceptable with audit trails. Spot-check tool call logs regularly.	Recommended. Good balance of cost and reliability.
High Criticality (production deploys, financial ops, security)	Unacceptable. Do not deploy. Risk of undetected fabrication is too high.	Risky. Requires human approval at every critical step.	Required. Even here, maintain audit logs and human checkpoints.

paths and plausible function names. Progress narratives follow logical sequences. The failure is invisible unless you look at a completely separate data layer: the tool execution logs, the file system state, the version control history.

For organizations accustomed to evaluating work products by reading them, this represents a fundamental shift in verification methodology. Reading an agent's output is necessary but not sufficient. Verifying an agent's output requires examining the evidence trail of its actions, and building that evidence trail requires forethought in framework design, not after-the-fact forensics.

9 Where This Is All Going (2026–2028)

Predicting the trajectory of AI capabilities has humbled many forecasters over the past three years. Still, several trends have enough momentum and structural support to justify reasonable projections over a two-year horizon. Enterprise leaders making infrastructure and strategy decisions today need a sense of where the field is heading, even with appropriate uncertainty margins built in.

9.1 Convergence of Reasoning and Tool Use

As of early 2026, reasoning capability and tool-use proficiency exist as partially independent axes. Some models reason deeply but invoke tools clumsily; others call functions reliably but plan shallowly. By 2027, this separation will largely dissolve. Every major model provider is investing in architectures that treat tool invocation as a first-class reasoning primitive rather than an afterthought bolted onto a text generator. Early evidence of this convergence is already visible in models like OpenAI's o3 and Anthropic's Opus 4, where reasoning and tool execution feel integrated rather than sequential.

For enterprise adopters, convergence means the agentic capability floor will rise significantly. Models that today require careful selection and evaluation will be replaced by a generation where competent tool use is table stakes. Selection criteria will shift from "can this model use tools at all?" to "how reliably does it use tools under pressure, at scale, across edge cases?"

9.2 Specialized Versus General-Purpose Agentic Models

A parallel trend is the emergence of models purpose-built for specific agentic domains. Just as the software industry moved from general-purpose programming languages to domain-specific frameworks, the model landscape is beginning to differentiate between general-purpose agents and specialized ones. Code-focused model variants, optimized specifically for software engineering tasks, already outperform their general-purpose siblings on development workflows despite having comparable or lower benchmark scores on broader evaluations.

Expect this specialization to accelerate. Within two years, organizations will likely choose between general-purpose agentic models for varied task portfolios and domain-specific models for high-volume, well-defined workflows in areas like software engineering, financial analysis, legal document processing, and DevOps automation. The orchestration challenge will shift from finding one model that does everything to coordinating multiple specialized models within a unified framework.

9.3 Distillation and Democratization

Model distillation, the process of training smaller models to replicate the behavior of larger ones, is compressing the capability timeline dramatically. Behaviors that required a $75-per-million-token model eighteen months ago are now available at $2 per million tokens. Within two years, today's frontier agentic capability (planning depth, error recovery, honest self-assessment) will be available at price points currently associated with basic text completion.

For enterprise planning, distillation implies that today's hardware investments in local model deployment will become more capable over time without additional hardware spend. A GPU cluster purchased to run a moderate 72B-parameter model will, within 18–24 months, run a distilled model with substantially stronger agentic capability at the same parameter count. Organizations that build model-agnostic agentic frameworks today will be positioned to capture these improvements through simple model swaps rather than architectural overhauls.

9.4 Multi-Agent Orchestration

Single-agent architectures, where one model handles an entire task from start to finish, are giving way to multi-agent systems where specialized agents collaborate on complex workflows. An orchestrator agent decomposes a high-level objective into sub-tasks, delegates each to a specialist agent (one for code generation, one for testing, one for documentation, one for deployment), and synthesizes the results.

Multi-agent orchestration introduces new challenges around coordination, state management, and quality control that single-agent systems avoid Guo et al. (2024). It also introduces new failure modes: a weak model in one specialist role can undermine the entire pipeline, even if every other agent performs flawlessly. For enterprise adopters, multi-agent systems promise higher throughput and better task coverage, but they demand more sophisticated evaluation and monitoring frameworks than single-agent deployments.

9.5 The Evaluation Gap

As agentic capability becomes mainstream, the absence of standardized evaluation benchmarks grows increasingly problematic. SWE-bench measures software engineering task resolution and has become a de facto standard for that narrow domain. Comparable benchmarks for infrastructure management, data analysis, document processing, and other enterprise agentic workflows do not yet exist at similar levels of rigor and adoption.

Industry will need to develop and adopt standardized agentic benchmarks that measure the dimensions outlined in Section 4: tool-use reliability, planning depth, error recovery, honesty, context utilization, and output verification. Without such benchmarks, enterprise buyers remain dependent on vendor claims and in-house evaluation efforts that are expensive to conduct and difficult to compare across organizations. Expect significant investment in this space from both academic institutions and industry consortia over the next two years.

9.6 Regulation and Accountability

Autonomous agents that execute real-world actions raise accountability questions that conversational AI largely sidesteps. When a chatbot gives bad advice, the human who acted on it bears at least partial responsibility. When an autonomous agent executes a production deployment that causes an outage, the chain of accountability is less clear. Did the organization that deployed the agent accept the risk? Did the model provider deliver a product fit for the stated purpose? Did the framework developer provide adequate safeguards?

Regulatory frameworks are beginning to grapple with these questions. The EU AI Act European Parliament (2024) classifies AI systems by risk level, and autonomous agents operating in high-risk domains (healthcare, finance, critical infrastructure) will face compliance requirements around transparency, human oversight, and audit trails. Organizations deploying agents in regulated industries should anticipate these requirements and build compliance into their agentic frameworks proactively rather than retrofitting it under regulatory pressure.

9.7 The Human-in-the-Loop Spectrum

Rather than a binary choice between full autonomy and full human control, the practical future of agentic AI is a spectrum of human involvement calibrated to task risk and model reliability. At one end, low-risk tasks executed by high-capability models can run with minimal oversight: generating documentation, formatting data, running standard test suites. At the other end, high-stakes tasks require human approval at critical decision points, regardless of model capability: deploying to production, modifying financial records, communicating with customers.

Mature agentic frameworks will support configurable autonomy levels, allowing organizations to dial human involvement up or down based on task criticality, model confidence, and organizational risk tolerance. Building this configurability into framework architecture today, rather than hardcoding either full autonomy or full oversight, positions organizations to adapt as both model capability and organizational trust evolve.

9.8 A 2028 Prediction

By 2028, agentic capability will be table stakes for enterprise AI platforms. Every major model will support reliable tool use, multi-step planning, and structured output generation. The differentiator will no longer be whether a model can act as an agent, but *how reliably it does so under real-world conditions*. Verifiability, honest self-assessment, and graceful degradation under uncertainty will separate the models trusted with autonomous work from those restricted to supervised roles.

Organizations that invest now in evaluation frameworks, audit infrastructure, and model-agnostic architectures will be positioned to adopt these improvements incrementally. Those that delay will face the same compressed adoption timeline that has characterized every previous wave of enterprise AI, except this time, the stakes of getting model selection wrong will be substantially higher.

10 Practical Recommendations for Enterprise Leaders

Frameworks and analyses are useful to the extent that they inform action. What follows is a set of concrete, prioritized recommendations for enterprise leaders navigating agentic AI adoption. These recommendations draw directly from the evaluation framework (Section 4), the model comparisons (Sections 5–6), and the failure modes documented in Sections 7–8. They are ordered by implementation priority, starting with foundational decisions that constrain all subsequent choices.

10.1 Start with Task Definition, Not Vendor Selection

Before evaluating a single model or speaking with a single vendor, define the agentic tasks your organization needs to perform. Catalog them by domain (software engineering, data analysis, document processing, infrastructure management), complexity (number of steps, branching logic, error likelihood), and criticality (what happens if the agent fails or fabricates output). Map each task category to the capability requirements from Section 3.

Organizations that skip this step invariably end up selecting a model based on brand recognition, benchmark rankings, or vendor relationships, then discovering months later that the model cannot reliably perform the specific work they need. Task definition first; model evaluation second.

10.2 Run Pilot Programs with Verifiable Tasks

When evaluating models for agentic deployment, structure your pilots around tasks that produce verifiable artifacts. Software that compiles or does not. Tests that pass or fail. Data transformations with checkable outputs. API calls with logged responses.

Avoid piloting on open-ended, subjective tasks ("improve our documentation," "analyze market trends") where the quality of output is difficult to measure objectively. These tasks can be valuable for agentic AI, but they are poor evaluation vehicles because they make it difficult to distinguish genuine capability from fluent fabrication. During pilot programs, measure tool call rates (how often does the model actually invoke tools versus narrate intended actions?), task completion rates (what percentage of assigned tasks produce correct, verifiable outputs?), and error recovery rates (when something goes wrong, how often does the model fix it versus ignore it or fabricate success?).

10.3 Never Trust Status Reports Without Audit Trails

If your organization takes one lesson from this paper, let it be this: require proof of work from AI agents, always and without exception. An agent's self-reported status is exactly as trustworthy as the audit trail that backs it up.

Build your agentic framework so that every tool call is logged with timestamps, inputs, and outputs. Record file system diffs before and after agent sessions. Maintain version control history that is independent of the agent's own reporting. When an agent claims it has completed a task, verify the claim against objective evidence before accepting it. This verification can be automated (checking for expected file changes, running test suites, validating API responses), but it must exist. A framework that accepts agent self-reports at face value is a framework designed to be deceived.

10.4 Build Monitoring from Day One

Audit trails enable after-the-fact verification. Monitoring enables real-time awareness. Both are essential, and both must be designed into the agentic framework from the beginning rather than added as an afterthought.

At minimum, monitor tool call frequency (an agent that stops making tool calls but continues generating text is likely fabricating), error rates and recovery patterns, context window utilization (approaching the limit often precedes degraded performance), and session duration relative to task complexity. Set alerts for anomalous patterns: a session running for hours with no tool calls, a sudden spike in error rates, or a model repeatedly generating text about performing actions without corresponding tool invocations.

10.5 Match Model Capability to Task Criticality

Refer to the risk matrix in Table 12. Deploy strong models (rated 4–5) for medium- and high-criticality tasks. Reserve weaker, cheaper models for low-criticality tasks where failures are inexpensive to detect and correct. Never deploy a model rated 1–2 on high-criticality autonomous work, regardless of budget pressure.

A $1-per-million-token model that fabricates results on mission-critical tasks is not cheaper than a $15-per-million-token model that completes them correctly. The apparent savings evaporate the first time a fabricated deployment report masks a production issue, and the trust damage (Section 8.2) can set an organization's AI adoption back by months or years.

10.6 Plan for Model Switching

The model landscape changes faster than enterprise procurement cycles. A model that is best-in-class today may be surpassed within six months. Build your agentic framework to be model-agnostic: abstract the model interface behind a consistent API, maintain configuration-driven model selection, and avoid deep coupling to any single provider's proprietary features.

Organizations that build model-agnostic frameworks can adopt improvements incrementally, swapping in better or cheaper models as they become available without re-engineering their agent infrastructure. Those that lock into a single provider's ecosystem pay a switching cost every time the competitive landscape shifts, which, in this market, happens quarterly.

10.7 Invest in AI Literacy at the Executive Level

The ability to evaluate model capabilities, interpret benchmark results critically, and make informed model selection decisions is rapidly becoming a competitive advantage. Executives who can distinguish between a model's benchmark performance and its agentic reliability will make better procurement decisions, set more realistic expectations for AI-driven initiatives, and avoid the trust-destroying failures that result from uninformed model selection.

AI literacy at the executive level does not mean understanding transformer architectures or training methodologies. It means understanding the questions that matter: What tasks will this model perform autonomously? How do we verify its output? What happens when it fails? What is the audit trail? How does this model compare to alternatives on the specific capabilities we need? Leaders who can ask and evaluate answers to these questions will navigate the agentic transition far more successfully than those who delegate model selection entirely to technical teams or, worse, to vendor sales processes.

10.8 Questions to Ask Every Vendor

When evaluating agentic AI products or platforms, demand answers to these questions before signing a contract:

1. What specific model or models power your agent? If the vendor will not disclose this, treat it as a red flag.
2. What are the tool call success rates for tasks comparable to ours? Insist on data from real deployments, not benchmarks.
3. Can we see the audit log? If the platform does not maintain detailed tool call logs, it cannot support the verification practices described in this paper.
4. What happens when the agent encounters an error it cannot resolve? The answer should involve explicit human escalation, not silent failure or fabricated success.
5. How do you evaluate and update the underlying model? Vendors should have a systematic process for evaluating new model releases against agentic task performance, not just benchmark scores.
6. What is the fallback if your primary model provider experiences an outage? Vendor lock-in to a single model provider creates a single point of failure for your agentic workflows.
7. Can we run a pilot with verifiable tasks before committing? Any vendor confident in their product will welcome structured evaluation.

11 Conclusion

Agentic AI represents the most significant evolution in enterprise computing since cloud infrastructure. The ability to deploy autonomous agents that read, write, execute, verify, and adapt has the potential to transform how organizations operate at every level, from individual contributor productivity to organizational strategy. That potential, however, comes tethered to a risk that is both novel and poorly understood: the risk that an agent will do its work *wrong* while appearing to do it *right*.

11.1 What We Found

Our evaluation of more than twenty models across cloud APIs and local deployments revealed a landscape of profound capability disparity. Models that appear equivalent on standard benchmarks diverge dramatically when tasked with real agentic work. Some execute tools reliably, plan multi-step workflows, recover from errors, and honestly signal their limitations. Others produce eloquent descriptions of work they never perform, generating status reports, code narratives, and progress updates backed by zero actual tool invocations. Standard benchmarks do not predict which category a model falls into.

The six-dimension framework proposed in this paper (tool use reliability, planning depth, error recovery, honesty and calibration, context utilization, and output verification) provides a structured methodology for evaluating agentic capability that goes beyond what leaderboards and marketing materials reveal. Applied systematically, it surfaces the capability gaps that matter for production deployment before those gaps manifest as expensive, trust-destroying failures.

11.2 The Maturity Curve

Most organizations are not ready for fully autonomous AI agents, and that is perfectly fine. Readiness is not a binary state; it is a maturity curve that begins with understanding, progresses through supervised deployment, and eventually, for appropriate tasks with appropriate models, reaches genuine autonomy.

Early-stage organizations should focus on building evaluation competency: learning to assess models against agentic criteria, establishing pilot programs with verifiable tasks, and developing the monitoring and audit infrastructure that makes trustworthy deployment possible. Mid-stage organizations, those with successful supervised deployments, can begin extending autonomy incrementally, widening the scope of tasks and reducing human checkpoints as verified performance justifies increased trust. Even the most advanced adopters should maintain human oversight for high-criticality tasks and preserve the audit infrastructure that enables rapid detection and response when agents fail.

Skipping stages on this maturity curve is where organizations get hurt. Deploying autonomous agents without audit infrastructure, selecting models without structured evaluation, or extending trust without verification, these shortcuts feel efficient until the first fabricated status report surfaces, and by then the cost has already been incurred.

11.3 The Path Forward

For enterprise leaders reading this paper, the recommended path forward is deliberate, incremental, and grounded in verification.

Start small. Pick a well-defined, verifiable task domain. Evaluate multiple models against that domain using the framework from Section 4. Deploy the best-performing model in a supervised configuration with comprehensive logging. Verify outputs. Measure tool call success rates, error recovery rates, and fabrication incidence. Build confidence through evidence, not hope.

As confidence grows, backed by data rather than vendor promises, expand the scope. Add new task domains. Increase autonomy for tasks where verified performance justifies it. Maintain audit trails at every stage. Update model selections as the landscape evolves. Treat model evaluation as an ongoing operational practice, not a one-time procurement decision.

11.4 A Final Thought

Whether AI agents will transform enterprise work is no longer a meaningful question. They will. The transformation is already underway, driven by models that can genuinely execute complex, multi-step tasks with a reliability that was unimaginable three years ago. What remains genuinely uncertain is whether organizations will deploy these agents wisely enough to capture the value without absorbing the risk.

Wisdom, in this context, is not caution for its own sake. It is the discipline to evaluate before deploying, verify before trusting, and build the infrastructure that makes trustworthy autonomy possible. Organizations that cultivate this discipline will find in agentic AI a force multiplier of extraordinary power. Those that skip the hard work of evaluation and verification will find something else entirely: a source of confident, professional-looking output that may or may not correspond to reality, with no reliable way to tell the difference until the damage is done.

The models are ready. The frameworks are maturing. The question is whether the organizations deploying them will be ready too.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, May 2024. Accessed March 2026.

OpenAI. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, September 2024. Accessed March 2026.

OpenAI. Introducing o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, April 2025. Accessed March 2026.

Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku. https://www.anthropic.com/news/claude-3-family, March 2024. Accessed March 2026.

Anthropic. Claude Opus 4 and Claude Sonnet 4. https://www.anthropic.com/news/claude-opus-4-and-claude-sonnet-4, May 2025. Accessed March 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

Meta. Introducing LLaMA 4: Open-Weight Models for the Agentic Era. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, April 2025. Accessed March 2026.

Google DeepMind. Gemini 2.5: Our Most Intelligent AI Model. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/, March 2025. Accessed March 2026.

Gemma Team, Morgane Rivière, et al. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv preprint arXiv:2408.00118*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, et al. Mixtral of Experts. *arXiv preprint arXiv:2401.04088*, 2024.

Mistral AI. Mistral Large. https://mistral.ai/news/mistral-large-2407/, July 2024. Accessed March 2026.

Mistral AI. Codestral: Hello, World! https://mistral.ai/news/codestral/, May 2024. Accessed March 2026.

DeepSeek-AI. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

xAI. Grok 4 Is the Most Intelligent Model in the World. https://x.ai/news/grok-4, July 2025. Accessed March 2026.

Qwen Team. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5/, September 2024. Accessed March 2026.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, et al. Phi-4 Technical Report. *arXiv preprint arXiv:2412.08905*, 2024.

Cohere. Command R+: Enterprise-Grade RAG-Optimized Model. https://docs.cohere.com/docs/command-r-plus, 2024. Accessed March 2026.

Significant Gravitas. AutoGPT: An Autonomous GPT-4 Experiment. https://github.com/Significant-Gravitas/AutoGPT, 2023. Accessed March 2026.

Yohei Nakajima. BabyAGI: Task-Driven Autonomous Agent. `https://github.com/yoheinakajima/babyagi`, 2023. Accessed March 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as Agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: Activation-Aware Weight Quantization for LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Georgi Gerganov. llama.cpp: Inference of Meta's LLaMA Model in Pure C/C++. `https://github.com/ggerganov/llama.cpp`, 2023. Accessed March 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting Hallucinations in Large Language Models Using Semantic Entropy. *Nature*, 630:625–630, 2024.

National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). `https://www.nist.gov/artificial-intelligence/executive-order-safe-secure-and-trustworthy-artificial-intelligence`, 2024. Accessed March 2026.

McKinsey & Company. The State of AI in Early 2024: Gen AI Adoption Spikes and Starts to Generate Value. `https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai`, May 2024. Accessed March 2026.

Krystal Hu. ChatGPT Sets Record for Fastest-Growing User Base. *Reuters*, February 2023. `https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/`. Accessed March 2026.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large Language Model Based Multi-Agents: A Survey of Progress and Challenges. *arXiv preprint arXiv:2402.01680*, 2024.

European Parliament and Council of the European Union. Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence (AI Act). *Official Journal of the European Union*, L series, 2024.