What's Needed to Unlock the Full Potential of AI Agents?

GSBGEN 390: Individual Research Autumn 2025

Authors: Fernando Torres & Shekhar Bhende (GSB '26)

Faculty Sponsor: Prof. Scott J. Brady

Abstract

AI agents represent one of the most promising frontiers in artificial intelligence, yet a stark gap persists between demonstrated capabilities and production deployment. This research investigates what prevents agents from transitioning from impressive demonstrations to reliable enterprise tools. Through 36 expert interviews, 5 industry conferences, and 3 functional prototypes, we discovered that production deployment is fundamentally an engineering problem, not an AI problem. Models contribute only 30-40% to agent success; the remaining 60-70% comes from framework architecture, system integration, and evaluation infrastructure. Most critically, 90% of enterprise pilots fail not due to technical limitations but because organizations cannot define ROI, forecast costs, or manage stakeholder expectations around probabilistic systems. This research reveals significant infrastructure opportunities in evaluation frameworks, memory architectures, and vertical-specific integration, opportunities that will unlock the next wave of value creation in agentic AI.

Background, Hypotheses & Research Methodology

To effectively unlock the true potential of AI Agents, we must first dismantle the machinery that drives them. We cannot identify the bottlenecks of a system until we fully understand its anatomy. An agent is not merely a Large Language Model (LLM) wrapped in a loop; it is a complex orchestration of disparate components working in concert to reason, act, and reflect.

To ground our analysis, we identified the eight key pillars that are critical to any modern production-ready agentic system:

1. **The LLM (The Cognitive Engine):** The core reasoning unit. It is responsible for intent detection, planning, and generating responses. Its importance is foundational; without a sufficiently capable model, the agent cannot navigate ambiguity or execute complex instructions.

2. **Context & Memory Management:** LLMs are inherently stateless. This layer provides the continuity required for long-running tasks. It involves managing the context window, retrieving relevant historical data (via RAG or vector stores), and

maintaining state across multi-turn interactions so the agent "remembers" what it has done.

3. **System Integration:** An agent without access to data is merely a chatbot. This pillar handles the connections to external systems (APIs, databases, SaaS tools). It is the mechanism by which an agent perceives the dynamic world and executes actions within it.

4. **Authentication & Identity:** In an enterprise setting, an agent acts on behalf of a user. This layer manages permissions, OAuth tokens, and Role-Based Access Control (RBAC), ensuring the agent can only access data and perform actions authorized for that specific human user.

5. **Trust, Governance & Guardrails:** This serves as the system's conscience and safety net. It includes PII redaction, output filtering, and policy enforcement to prevent hallucinations or malicious use. It is critical for regulatory compliance and brand safety.

6. **Cost Management:** Agentic loops can be token-intensive. This component tracks usage, manages quotas, and optimizes model selection (routing to cheaper models for simpler tasks) to ensure the Return on Investment (ROI) remains positive.

7. **Agent Evaluations (Evals):** Unlike deterministic software, probabilistic AI requires rigorous testing. This pillar involves benchmarking the agent's accuracy, reliability, and safety using datasets and simulation environments before deployment.

8. **Monitoring & Telemetry:** Once deployed, observability is non-negotiable. This involves tracing execution chains, logging latency, and debugging failures in real-time to maintain system health.

The rise of Agentic AI enablers and our Initial Hypothesis

Given the rise of hundreds of startups and established tech giants rushing to fill the gaps in the agentic stack, we formed an initial hypothesis that most of the pillars would have relatively thin friction while a couple may appear as the pillars to really focus on. We define friction from the vantage point of building a production-ready agent. Low friction implies the existence of a plug-and-play solution or a highly generalizable framework that allows developers to focus on logic rather than plumbing. High friction implies the need for bespoke engineering, fragile workarounds, or the lack of established patterns.

Consequently, we entered this study with a **technical-first framing**. We hypothesized that the primary bottlenecks preventing agents from reaching their

potential were located in the deep technical nuances of the agentic stack with business case or engineering issues being secondary. This paper outlines how that hypothesis stood up against reality.

Research Methodology: Depth and Breadth

To test these hypotheses, we employed a three-pronged approach spanning September through November 2024:

36 Expert Interviews: We conducted in-depth conversations with practitioners actively deploying agentic workflows—enterprise AI platform providers, coding agent developers, identity and security experts, vertical SaaS companies, framework companies, infrastructure providers, and foundation model companies. We sought builders with production deployments, not researchers or theorists.

5 Industry Conferences: We attended leading AI agent events in San Francisco and Palo Alto—a major industry conference, the Production Agents Summit, an industry fireside chat, an academic project summit, and "Why 95% of Agentic AI Projects Fail"—to observe industry consensus and emerging standards.

3 Functional Prototypes: We built working agent systems to validate interview findings: *Shopping Agent* (e-commerce automation testing framework approaches), *Repo Patcher* (code fixing agent with state machine architecture), and *Good Agents* (multi-agent orchestration with plan-verify-execute framework). These served dual purposes: empirical validation of claimed challenges and opportunity identification for future research.

Section B: Major Learnings & Insights

This research began with a technical hypothesis, with model capabilities, context management, and integration standards were the primary blockers. What we discovered was more fundamental: **production AI agents are an engineering problem, not an AI problem**. Models at GPT-4 capability levels are "good enough" for many use cases, playing a minor role in Agentic friction. The bulk of the heavy lifting comes from framework architecture, system integration, context engineering, and evaluation infrastructure. As a popular consumer AI Agent founder articulated at an interview: "We found that the model only maybe contributes 30 or 40% of the whole thing. And the framework, the whole system you build upon the model is much more important than the model itself." This inverted our initial assumptions entirely.

In addition, we discovered that **business case failure precedes technical failure in many cases**. A large number of enterprise pilots fail not because the technology doesn't work, but because organizations cannot define ROI, cannot forecast costs due to pricing model confusion, and cannot manage stakeholder expectations around probabilistic systems.

Key Pattern 1: System Integration Dominates Deployment Effort

System integration emerged as the dominant enterprise challenge, appearing in 92% of our sources and consuming 40-50% of all deployment time. A founder from an enterprise AI platform stated: "A lot of people think AI has to be like 70%, 80%, 90% about prompt engineering and training the AI workforce. In reality that's only around 40% of the work, max. The rest of the time I spent on system integration."

The integration challenge stems from heterogeneous enterprise tech stacks built over decades with no expectation of coordination. AI agents must orchestrate workflows across these systems, requiring custom connectors, authentication handling across disparate identity systems, and often browser automation when APIs don't exist.

Our prototypes empirically validated this finding. In Good Agents, MCP integration consumed more engineering time than core orchestration logic. In Shopping Agent, multi-platform connectors represented the primary technical challenge versus agent reasoning.

A critical discovery: while industry narratives emphasize standardization for seamless and speedy deployments, AI companies with successful deployments end up creating deeply custom integrations which in turn become competitive moats, rather than a problem to solve. A leading Agent Framework provider positions enterprise topology complexity, eg. navigating a 40-country presence, 100,000+ employees, and multiple acquired companies with different tech stacks, as a "defensibility that can't be solved by throwing money at the problem."

Key Pattern 2: Agent Frameworks deliver limited value beyond prototyping

Agent Frameworks including widely used popular open-source libraries achieve billion-dollar valuations through initial developer adoption, yet we see majority of production teams abandon them due to bloat, performance overhead (3-4x slower), and loss of control. A leader from an agent framework company reported: "Every company we've talked to started with LangChain as a framework to build AI agents. But once they start going into production, they realize it's full of bloat. They end up ditching that solution, and they build their own. This has been like 80, 90% of the clients we've talked to."

Our Shopping Agent prototype provided firsthand validation. Initially implemented with a graph-based framework, we were forced to switch to a popular open-source framework mid-development due to extensive bloat and complexity. Under time pressure, framework abstractions became intolerable, which is exactly the pattern reported in interviews.

The paradox: framework abstractions that accelerate prototyping become obstacles to understanding agent decision-making in production. Teams need control

and performance that generic frameworks cannot provide. Rather than consolidation (like React in frontend development), the agent field remains fragmented because production requirements diverge fundamentally from prototyping needs.

Key Pattern 3: Managing expectations of probabilistic systems

The demo-to-production chasm exists because '80% reliability' creates impressive demos but systematically fails in production workflows. A founder from an Agent Framework company articulated the mechanism: ""When I run a demo, I recognize I'm relying on luck. I know perhaps 20% of the time it will fail, but for the demo, I just need a single successful execution. The problem is, our customers aren't accustomed to this probabilistic performance. They see a single win and immediately believe the agent is ready for production deployment."

This isn't a model limitation but an expectation mismatch. Most people building technology over the last 30 years expect deterministic systems, where if it works once, it works reliably thereafter. This assumption is catastrophic for LLM-based systems.

Without systematic improvement methodologies, teams enter a "doom loop": fixing one scenario breaks another, creating continuous firefighting. Production teams manage probabilistic behavior through architectural patterns rather than waiting for better models. For example, our coding agent prototype, *Repo Patcher*, follows such architectural patterns that deal with the probabilistic nature of LLMs:

- *Repo Patcher's state management*: The INGEST→ PLAN→ PATCH→ TEST→ REPAIR→PR workflow provides deterministic progression through probabilistic steps
- *Verification phases*: Good Agents "Plan-Verify-Execute" pattern validates feasibility before execution
- *Risk-based escalation*: Automatic approval for low-risk changes, human review for high-risk (not binary autonomous vs. supervised)

Key Pattern 4: Context Management Requires Engineering, Not Larger Windows

Context window engineering emerged as a fundamental architectural challenge in the majority of our conversations. The critical discovery: a "40% context utilization rule" contradicts vendor narratives around million-token windows. From the Production Agents Summit we heard "If your agent is using anything more than 40% of the context window, it's probably going to make mistakes. This is true for an agent you develop for your software. It's also true for things like Cursor."

This threshold exists independent of maximum window size; quality degrades well before theoretical limits. MCP exemplifies the problem: when you have

more than 25 MCP tools, accuracy drops to 30% due to verbose tool definitions exhausting context in multi-step conversations.

Production teams employ sophisticated context engineering techniques:

- **Notes summarization**: Periodic bullet point summaries replacing full conversation history.
- **Just-in-time retrieval**: Track file path references instead of contents; fetch on-demand
- **Sub-agent patterns**: Parallel agents with internal iteration loops compress context before passing to orchestrator
- **Dual memory architecture**: User memory (preferences, past interactions) versus agent memory (tool performance, problem-solving patterns), which are architecturally different problems requiring separate solutions

Key Pattern 5: Business Economics often Block Before Technical Limits

The emergence of business case and ROI concerns as a dominant theme revealed our research blind spot. An AI startup founder we spoke to identified this as "the primary failure mode" in our very first interview: "Most AI agent deployments fail due to undefined ROI calculations and lack of commercial mindset with respect to deployments, not technical limitations."

This precedes technical challenges. Enterprises cannot forecast costs because multiple pricing models coexist (seat-based, token-based, outcome-based) without convergence, token consumption varies unpredictably, and startups themselves are experimenting with business models. An expert from a large tech company explained: "All of these pricing changes or types of pricing schemes is confusing to enterprises. They don't know how many tokens they're going to use. They can't model their usage, they can't model their outcome."

Meanwhile, enterprises demonstrate willingness to pay $400-750/month (versus current $20/month offerings), but cost predictability doesn't exist. The framework provider's customers demand guarantees: "We'll save $2M in this single use case" with targets of "$100M annual savings" to justify deployment.

Key Pattern 6: Evaluation (Evals) is still a Missing Piece

The agent evaluation (Evals) market suffers from an Evaluation Value Chasm, as current solutions predominantly focus on readily measurable intermediate metrics leveraged by LLM as a judge (e.g., token cost and retrieval accuracy) instead of the essential business outcomes (handoff rate and task completion). This approach provides insufficient quality signals for debugging complex, probabilistic pipelines, a problem validated by the anecdote of a recent Y Combinator founder noting that agent builder companies in his batch failed to adopt third

party evaluation tools, despite 7+ evaluation tool companies in the same batch. To unlock reliable production readiness, the winning evaluation paradigm must holistically combine component-level testing for deterministic steps with direct business outcome tracking, and the establishment of industry-specific benchmarks derived from real-world usage rather than synthetic datasets.

Section C: Emerging Opportunities & Trends

Our research reveals that the gap between current capabilities and production requirements exposes where innovation is needed most urgently. Contrary to dominant narratives about better models, the highest-value opportunities lie in infrastructure that doesn't yet exist.

Enterprise Memory Systems: Current solutions suffer from static staleness (batch-based memory can't benefit from improved prompts without reprocessing), multi-modal limitations, and PII leakage risks that block enterprise adoption. The opportunity lies in privacy-preserving architecture with cryptographic isolation, multi-modal retrieval across text, images, documents, and voice, dynamic update capability, and explicit separation of user memory versus agent memory layers.

Context Engineering Services: Production practitioners discovered that using more than 40% of context window causes accuracy degradation regardless of maximum window size. The opportunity exists for "context-as-a-service" abstraction layers that monitor utilization in real-time, provide compression recommendations, implement proven patterns, and prevent degradation before it occurs. This requires deep expertise in both LLM behavior and distributed systems.

MCP Specificity Layer: MCP accuracy drops to 30% beyond 25 tools because generic MCP servers expose too many operations without use-case context. The near term opportunities may involve dynamic tool loading solutions that fine-tune small models to select which tools to load before LLM invocation, provide use-case-specific tool bundles, and enable MCP adoption at enterprise scale without hitting accuracy cliffs.

Vertical Specialization May Dominate: Every successful deployment in our research operated in narrow, well-defined domains. Generic horizontal platforms struggle because they cannot encode domain-specific knowledge about which operations matter when, what workflows are standard, or how to handle exceptions. High-value verticals include healthcare (HIPAA compliance workflows), financial services (regulatory reporting, KYC automation), legal (contract analysis, compliance checking), and supply chain (multi-party coordination, customs documentation).

Small Model Constellations Instead of Single Frontier Models in Cer-

tain Use Cases: Even if model capabilities plateau and inference costs remain static, some advanced teams will shift to specialized small models (~8B parameters) with domain-specific fine-tuning achieving 10-100x speed improvements, task-specific routing, and owned training infrastructure creating sustainable moats. This is a key trend that the advancements of open source models like the Qwen, Kimi and Llama families will only accelerate.

Conclusion – where does the most friction lie today?

Contrary to our initial hypothesis, where we expected to find friction in a few pillars of the stack, we found that friction (and lack of generalizability) continues to exist in almost all the pillars of the Agentic Stack today. System Integration emerged as the dominant friction point, consuming the largest share of deployment effort as teams grapple with heterogeneous enterprise tech stacks built over decades. MCP is an initial start but ultimately a thin value add in complex workflows as it creates bloat and fails as the number of integrations increases. Context & Memory Management revealed an unexpected ceiling; accuracy degrades well before theoretical window limits, requiring sophisticated compression techniques rather than simply waiting for larger context windows. Context management best practices are yet to emerge, with most teams trying different approaches at this stage. Agent Evaluations remains a critical gap: despite numerous startups building eval tools, adoption is minimal because solutions measure intermediate metrics rather than business outcomes. Cost Management creates enterprise paralysis through pricing model confusion, making cost forecasting nearly impossible. Authentication & Identity and Trust, Governance & Guardrails compound enterprise blockers through fragmented identity standards, unreliable PII detection, and absent frameworks for trusting probabilistic systems. Monitoring & Telemetry struggles because traditional observability tools cannot trace non-deterministic decision chains. Perhaps most surprisingly, The LLM itself showed the least friction. Current frontier models are "good enough" for many production use cases. The bottleneck today is not intelligence but the infrastructure surrounding it.

In conclusion, our research reveals that the bottleneck to production AI agents is not artificial intelligence itself but rather infrastructure, and organizational readiness. As we stand today, agents are only ~30% model capability but 60-70% system architecture.

Acknowledgments

This research would not have been possible without the generosity of 36 practitioners in the broader AI ecosystem (founder, engineers, product managers, etc) who shared candid insights about production agent deployments. Finally, we thank Professor Scott J. Brady and Brett Jordan for sponsoring this individual research and providing invaluable guidance throughout the investigation.