
RESEARCH ARTICLE

Autonomous AI agents in enterprise systems: A review of architectural gaps

Srikanth Devarakonda

Independent Researcher, Fremont, California, USA, 95110

ORCID ID: 0009-0005-6406-8754

Corresponding Author: Srikanth Devarakonda, **E-mail:** srikanthdevarak@gmail.com

ABSTRACT

A central challenge in enterprise information systems is that the development of generative artificial intelligence has moved from conversational assistants and document generators to autonomous agents that initiate actions. Agentic architectures have the potential to transform workflows but their probabilistic reasoning is not compatible with transaction integrity, security boundaries and governance models that support mission critical enterprise functionality. This systematic literature review of literature published in 2023-2025 explores four key themes of security and authorization, observability and traceability, governance and compliance, and resilience engineering. By analyzing the state of the art of 25 peer-reviewed works, the review finds that autonomous AI agents are not treated as first-class governed execution entities by existing enterprise architecture frameworks. We detect four gaps in the literature: a lack of standardized isolation of agent activity, a lack of governing models for authorizing agent actions with uncertain outcomes, a lack of best practices for tracing the decision chain of the agent's reasoning, and a lack of resilience patterns for transactions based on these AI-driven agents. We conclude by proposing a theoretical research agenda for Agent-Oriented EA. The review concludes that without discipline, organisations risk increased security breaches, regulatory non-compliance and systemic operational failures as agentic systems increasingly become the norm.

KEYWORDS

Autonomous AI Agents, Enterprise Architecture, Generative AI Security, Agent Governance, Deterministic Transactions

ARTICLE INFORMATION

ACCEPTED: 8 October 2026

PUBLISHED: 8 October 2026

DOI: 10.32996/jcsts.2026.8.10.3

1. Introduction

The enterprise artificial intelligence market is undergoing a shift that weakens customary information systems architecture. Up until 2023, most large language models (LLMs) were passive AIs that generated on-demand content, responded to questions, summarized documents, and suggested code to be used by a human. These systems remained read-only or advisory applications, as they could only interface with enterprise systems via explicit interface calls initiated by users. Since 2024-2025, there is evidence suggesting a shift towards more agentic systems that can execute multi-step workflows, make calls to internal APIs, update transactional records, and orchestrate across multiple systems [1, 2].

Enterprise systems architected for predictable performance provide deterministic execution logic, ACID (Atomicity, Consistency, Isolation, Durability) transactional guarantees, RBAC role-based access control, and support for audit trails, state changes, and transactions that can be traced. In contrast, generative AI agents operate using probabilistic reasoning, addressing probabilistic, ambiguous natural language input, generating context-dependent output based on the nature and parameters of the user's input prompts, and performing hallucinated or misconstrued actions. [3] This survey's main focus is the disconnect between probabilistic agent reasoning and deterministic system requirements.

Our systematic review finds much work on LLM security, multi-agent coordination, and enterprise automation, but no work on managing, governing, securing, or assuring safety and reliability in agent executions that accounts for the requirements of production enterprise systems [4, 5]. This finding is important given predicted scale, and because 30% of enterprise workflows could be automated using automation technologies, including AI agents [1]. Without architectural discipline, organizations risk

replicating the problems that plagued early robotic process automation implementations through non-governed automation scripts becoming unmaintainable technical debt.

This review contributes to structuring the literature into four gaps, of which the first and most important is a taxonomic classification of the security and authorization issues faced by autonomous agents into intra-agent and inter-agent threat models. Second, we provide an analysis of the specific observability requirements of agent reasoning chains, including intermediaries capturing intermediate decision states, as well as confidence scores. Third, we synthesize governance frameworks for authorizing policy-driven action, such as human-in-the-loop controls and audit trail requirements. The fourth contribution is a consolidation of resilience patterns (e.g. compensating transactions, idempotent operations, and recovery from failure), together with a continued research agenda on Agent-Oriented Enterprise Architecture as a conceptual framework which requires evaluation with empirical studies.

The structure of the remainder of this paper is as follows. Section two will outline the systematic review process. The third section describes deterministic enterprise architecture and agent autonomy. The fourth through seventh sections discuss the four thematic domains in more detail. The eighth section summarizes the literature and discusses the research agenda. Section nine discusses the limitations, section ten concludes.

2. Review Methodology

The systematic review was conducted according to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 statement to ensure transparency, reproducibility, and methodological rigour in the evidence synthesis [6, 7]. The protocol for the review was developed prior to the literature search and detailed the review question, search strategy, inclusion/exclusion criteria, and data extraction methods.

The overarching review question was: What architectural barriers need to be addressed to securely integrate autonomous generative AI agents into deterministic enterprise systems? The first derived question was: What security and authorization mechanisms for agent-driven actions are described in the literature? (2) What are the observability and traceability mechanisms for the reasoning chains of agents? (3) What are the governance and compliance mechanisms for the execution of agent programs? (4) What resilience and rollback patterns are available for AI-driven transactions?

The search used five electronic databases namely IEEE Xplore, ACM Digital Library, Scopus, arXiv, and Google Scholar. The search strategy employed was based on Boolean logic using the following detailed search query: ("agent" OR "agentic" OR "autonomous agent") AND ("large language model" OR "LLM" OR "generative AI") AND ("enterprise" OR "enterprise system" OR "enterprise architecture") AND ("security" OR "governance" OR "observability" OR "compliance" OR "deterministic"). We used peer-reviewed journal articles, conference proceedings, and preprint repositories published in English from 1 January 2023 to 31 December 2025 as our search domain.

After removing duplicates, searches of the databases returned 847 results. After reviewing titles and abstracts, we excluded 703 results that did not contain an enterprise agent architecture or discuss security, governance, and architecture. After the full-text articles were reviewed, 25 studies were found to meet all eligibility criteria. The most common reasons for exclusion in this phase were the lack of empirical or architectural analysis of the concept of deterministic enterprise integration (62), and focus on individual agent capabilities outside their context within the enterprise (38). Nineteen articles were excluded as they were not peer reviewed. A PRISMA flow diagram is available from the corresponding author upon request.

The extracted features included the author, the year of publication, the journal, the research method, and then other features such as the basic architectural model (single-agent, multi-agent, hybrid), domain (security, observability, governance, resilience), proposed mechanism, and limitations. The methodological quality of the included studies was assessed according to DARE (Database of Abstracts of Reviews of Effects) quality appraisal criteria [8], including a clear statement of inclusion criteria, search strategy, assessment of primary study quality and description of data. Each study was rated from 1 (low) to 4 (high), with only studies rated 3 or higher considered for synthesis. The methodological quality showed considerable variation, with papers in peer-reviewed journals generally scoring higher than papers on preprint servers.

3. Foundational Concepts: Deterministic Enterprise Architecture and Agent Autonomy

To understand the architectural gap between autonomous agents and enterprise systems, we first need to characterize both types of systems. A primary assumption of customary enterprise architecture frameworks, such as TOGAF (The Open Group Architecture Framework) and the Zachman Framework, is that systems are deterministic: that is, when given an input state and an operation applied to this state, every execution will produce the same output state [9]. This determinism can be achieved through standardized API contracts, strict validation, and transactional integrity principles. For example, in a financial transaction model, account balances may be checked, business rules applied, a transfer made, and the new state committed in one transaction and one atomic auditable unit of work.

In contrast, four properties of agentic AI systems, such as probabilistic reasoning, suggest they lack the assumption of being deterministic, since they can give different answers to the same input based on the sampling parameters, the prompt, or the state. Secondly, the autonomous planning capability allows agents to decompose high level objectives into a sequence of actions that may not have been envisioned by the developer, along unpredictable paths through the planning space. Thirdly, the tool use

capability allows agents to invoke external APIs, databases and services, increasing the attack surface and possible failure modes. Fourth, emergent behavior in multi-agent systems is behavior that cannot be predicted from the specifications of individual agents [10].

Equation (1) captures the mismatch between the deterministic environment's expectations and the probabilistic agent's intentions: when D is an environment in a deterministic state S_t , it will expect a particular transition to a state S_{t+1} according to the rules of D .

$$S_{t+1} = f(S_t, a_t)$$

where a_t is a deterministic action. For an agent, the action a_t is generated by a probabilistic policy $\pi_\theta(a_t | o_t, h_t)$ conditioned on observation o_t and history h_t . The expected divergence Δ between the intended and actual system state is:

$$\Delta = E_{\pi_\theta}[\|f(S_t, a_t) - f(S_t, a_t^{\text{intended}})\|]$$

Empirical demonstrations of this generalization gap have shown that, across three representative tasks (multi-digit multiplication, logic grid puzzles, and dynamic programming), transformer-based agents can have near-zero accuracy out of distribution on inputs more complex than those seen during training, despite being near-perfect within distribution [3]. This suggests that solving structured problems is not a general capability emergent from maximum likelihood training on input-output pairs. Additionally, this is the case even when training on reasoning steps (3).

Implications for enterprise architecture are that agents are not deterministic components and that practitioners should not try to enforce deterministic behavior in the models through prompt engineering, which is called "doom prompting" in the practitioner literature. Instead, the architecture should treat non-deterministic outputs as part of its design and should have governance layers to constrain, verify and compensate them [11]. In the next sections, we will explore four themes that emerged from these notes.

4. Theme 1: Security and Authorisation Gaps for Autonomous Agents

Compared to purely reactive AI systems (i.e., passive AI systems acting as copilot), generative AI systems as automated executors introduce new security threats not addressed by perimeter-based measures. Linked to this, the studies identified three main security threats: insufficient isolation of contexts in which agents are executed, insufficient standardization of authorisation of agent-requested actions, and cross-agent prompt injection attacks.

A particularly commonly identified need is the separation of agent reasoning from agent execution. The ArbiterOS framework encapsulated the agent's probabilistic planning engine via a "Hardware Abstraction Layer" separating it from the deterministic execution of system actions [2]. This separation, implemented with a governor for controlling all reasoning and tool invocation transitions, creates a controllable boundary where safety policies can be enforced. In experiments, the governor-controlled agent was shown to have 68 percent fewer policy violations than agent baselines without a governor, while handling 31 percent more test cases per time unit without critical compliance failures [1].

The principle of least privilege is also well established in access control. Yet it would need adaptation for agentic systems, because permissions in role-based access control systems are usually statically assigned and seldom revoked [9]. Static permissions create excessive privilege surfaces for autonomous agents whose behavior is determined at runtime. The NIST Guide to Attribute Based Access Control (ABAC) establishes a theoretical foundation for dynamic, context-aware authorisation. Authorisation decisions are based on attributes of the subject, object and environment and do not require subjects to be identified. When applied to agents, ABAC can implement fine-grained policy decisions such as allowing an agent to read a customer's record but not change it or issue a refund (within a monetary limit) only when a human operator is involved.

Real-world organizational deployments of policy-governed agent architectures provide empirical evidence of their effectiveness. Of a sample of 36 applications and tools that use LLMs, 31 (86%) would be vulnerable to prompt injection attacks as modeled by HouYi, a black-box modeling attack that uses components, separators, and disruptors from the framework to create adversarial prompts [13]. The key vulnerabilities are uncontrolled arbitrary LLM use and application prompt theft. These attacks show that local tool invocation and privilege escalation are both viable and existing threats. Applications with structured validation layers (input sanitisation, output schema validation, policy enforcement points between the LLM and application) were considerably less vulnerable to these attacks, with the majority of successful attacks occurring in applications without these layers.

Additionally, the multi-agent setting creates new security challenges, as in cases where agents share a message pool or conversational interface, the compromised agent can issue commands to all other agents. When authors tested MetaGPT framework, which implements a publish-subscribe model for agent communication, against conversational architectures, they showed that structured messages and role-based access controls improved privacy by reducing the amount of leaked information by about 40% [14]. Authors of ChatDev framework applied "communicative de-hallucination", where agents would ask for more detailed instructions before acting, resulting in a large reduction in coding hallucinations. However, whether coding hallucinations can be eliminated from ChatDev is still an open question [15].

Equation 2 contains the security constraints that must be enforced for all governed agent systems. A is the set of all actions all agents may perform in the governed agent system, P is the set of policies modeled in the policy-as-code engine, e.g., Open Policy Agent, and V is the set of actions allowed by the enterprise security model.

For an action $a \in A$ to be permissible:

This means that all agent actions must be allowed by every policy applicable to the action before they can be carried out. This is accomplished via the Rego policy language, which can enforce complex policies such as "Stress tests may only execute in staging environments between 02:00 and 04:00 UTC, and only when the target system is configured for horizontal scaling" [1]. If such conditions are not enforced, policy violations become statistical certainties as the number of actions taken by the agent grows.

5. Theme 2: Observability and Traceability Deficiencies

Three aspects of observability for agentic systems have been proposed in the literature, as customary observability techniques do not work on the nondeterministic behavior of agents: the capture of reasoning chains in the form of traces, logging of intermediate decision states and logging of confidence scores of probabilistic actions.

The AgentOps model describes a six stage pipeline of agent system operations: observing behaviors, collecting metrics, detecting problems, diagnosing their causes, recommending solutions, and auto-remediating problems [16]. There are four types of users who have observability needs: developers, testers, site reliability engineers and business users. When capturing traces, developers must capture both agent actions and the agent's reason or chain of thought. Likewise, when evaluating model performance, it is important to capture intermediate decision points as the model chooses tools and reasons through which tools to use to find solutions. For site reliability engineers, metrics can include quantitative (latency, token counts, number of tool calls) and semantic indicators (success rate of tasks, or the completeness of generated output) [16].

SWE-bench, a benchmark of agent runtime performance over real-world GitHub issues, reveals the gap between the output of an agent and its actual reasoning capabilities. Using AutoCodeRover, a framework for combining LLM agents with AST-based code search and spectrum-based fault-localization, we solved 19 percent of tasks on SWE-bench-lite (57 of 300 issues) at an average cost of USD 0.43 and an average runtime of 195 seconds [17]. However, a mere 65 percent (51 out of 78) of the high-probability patches were semantically similar to the developer patches. Even though these patches modified the same lines of code, they did not perform the same function. In this case, without the tracing information, it would not have been possible to tell the overfitting patches apart from the correct ones.

Cognitive corruption is the disruption of an agent's working memory due to unreliable context management. LLMs are subject to this problem, because LLM context windows do not deterministically forget information, as an efficient memory would do by knowing when and where to evict it. However, information in the middle of a long context may be neglected. This is known as the "lost in the middle" effect. If an agent's summary of a conversation is compressed using an LLM, the summarisation step can create hallucinations or leave out important information. This is difficult to diagnose because once the information is written to memory every decision that depends on this information is compromised, and the root cause is separate from the symptom.

On an objective level, agent traces' observability is nascent: according to the AgentOps survey, only 8 percent of organisations use dedicated observability platforms, while according to 60 percent of users, current analytics tools are unsatisfactory. The main gap is in interpretability. Although agent execution traces generate large volumes of natural language data, extracting structured information (e.g. "the agent attempted tool X and received error Y because of condition Z") requires processing. Some of these methods have proposed using additional LLMs as evaluators (this is a circular dependency, as one is using an LLM to evaluate an LLM), but suffer from the same unreliability.

With respect to an agent's execution, the trace completeness is defined to be metric C in Equation (3). Let T be the set of true decision points in the agent's execution, and R be the set of decision points recorded in the trace. For each decision point $t_i \in T$, define $r_i = 1$ if recorded, 0 otherwise. Then:

where w_i is a weight for each decision point, for example, making a tool call can be given a high weight, and routine summarisation can be given a low weight. C is usually infeasible, since it requires ground truth about the agent's internal decision-making process. To handle these problems, researchers have developed surrogate measures such as task success rate, output stability (e.g., how much do outputs vary in the case of identical inputs across multiple runs), and trace plausibility.

6. Theme 3: Governance and Compliance Risks

Deploying autonomous agents in enterprise settings also requires governance frameworks beyond just security, such as regulatory compliance, auditability, and organizational accountability. Three gaps were identified in the literature in this regard: (1) translation of natural language policy to machine-enforceable rules, (2) human oversight methods, and (3) sufficient trails for regulators to audit decision-making.

The Self-Evolving, Policy-Governed Agentic Automation (SEPGA) architecture [18] implements governance as constrained optimisation, wherein the goal is to maximize cumulative reward R, while minimizing penalties for policy violation P:

where τ is the agent's action sequence, $\sum R$ total rewards gained by the agent, and λ a balancing parameter controlling the trade-off between performance and compliance. In practice, the policy engine assigns a high value of λ to violations of high severity (e.g. accessing data by an unauthorized entity), thus these violations serve as hard constraints. SEPGA consists of five components: a meta-planner that breaks tasks into sub-tasks, an action executor to perform the actions, an outcome evaluator to evaluate it, a knowledge reflector to learn from it and a policy engine to apply the rules. Combining continuous self-

improvement with explicit policy constraints makes it the first such architecture, unlike approaches like the reinforcement learning algorithms that optimize one or the other [18].

However, translating natural language policies into rules suitable for machine enforcement remains an important engineering challenge. In the τ -bench Airlines domain, a domain with 14 tools and 50 customer-service scenarios with real-world policies, the Tool-Policy Mapper component extracts policy statements and tools invoked by these statements from free-form documents, using LLMs [19]. When the mapper was evaluated by GPT-4o, it achieved an F1 of 0.80 and a fuzzy Rand Index of 0.83 regarding the reference classification and grouping of policy items, respectively. The number of mapped policy items decreased by 10 to 18 percent when out-of-domain and irrelevant in-domain content was added to policy documents. We also noted that recall (actual coverage of all relevant policies) remained lower than precision (actual accuracy among the detected policies) for all models, suggesting that the current methods may be insufficient for policies whose phrasing is not clearly articulated or that use uncertain language.

Human-in-the-loop governance mechanisms empower system designers to reduce risk in high-stakes applications. ToolGuard records deterministic validation steps for tool invocations. When an agent requests a tool (e.g., `cancel_reservation`), the guard checks its arguments, chat history, and the system state against a policy [19]. If a violation is detected, the agent is prompted to update its plan instead of executing the action against the policy. Evaluation on 22 τ -bench tasks where user requests violate policy showed that ToolGuard deployment improved the `pass@10` metric from 0.227 (baseline) to 0.500, an improvement of over 20 percentage points. Notably, simply appending policy documents to the agent's prompt achieved only modest improvements (`pass@10` of 0.273), demonstrating that passive policy instruction is insufficient; architectural enforcement is required.

The European Union Artificial Intelligence Act (Regulation 2024/1689) is a directly applicable regulation that provides a risk-based approach to the deployment of enterprise agent systems. It defines high-risk AI systems as those that may have a negative impact on health and safety or on fundamental rights. High-risk AI systems must satisfy the requirements on risk management, data governance and management, technical documentation, transparency, human oversight, accuracy, robustness and cybersecurity. Especially for agentic AI systems, the requirement on human oversight stipulates that high-risk AI systems must be designed so that natural persons can oversee the system's functioning, including the ability to stop, deactivate and override them. It requires organisations using enterprise agents to have certain governance architectures in place as of August 2026, when the Act will apply.

An example of a human-oversight requirement is the following (equation (4)): Let H be the set of decisions that need to be approved by a human operator. For every decision $d \in H$, let t_d be the time to take the decision, c_d be the confidence of taking the decision, and θ be the confidence threshold for automatic decision execution. Then:

Thus, human oversight is a matter of degree, and organisations can trade-off automation depending on the cost of failure. In the low-confidence zone, the system always interrupts and escalates, whereas in the high-confidence zones the system always executes without any interruption if the execution is within an authority range.

7. Theme 4: Resilience and Rollback Challenges

Enterprise systems need to be recoverable: after a failure during a transaction, the system must return to a known consistent state. Agentic systems have two additional issues with resilience. First, the probabilistic nature of agent reasoning means that failure is not an exception but part of the normal operation of the system. Second, since agent plans proceed in a sequential manner, failing at step k invalidates all runs of step i , where $i > k$.

The theory of error propagation for agentic systems is based on the analysis of transformer compositionality [3]. As shown in Proposition 4.2 of that work, the probability of a function being applied successfully decreases exponentially with the number of repetitive applications (e.g., actions by an agent in series). Let $f_n(x) = g^n(x)$ be the n th iterate of g . If the probability of returning to the correct state is at most c , and the probability of an error in one step is $\epsilon > 0$, then:

The above implies that unless c is large relative to ϵ , failures will be almost certain for long-horizon tasks (e.g. supply chain optimisation with hundreds of actions). In other words, even if all actions are arbitrarily close to deterministic (ϵ is small), the probability of failure will still approach 1 with T unless the architecture is explicitly designed with rollback and recovery.

AutoCodeRover resolves this by a layered context search, which repeatedly fetches contextually relevant code snippets before patching. Should a patch result in compilation errors or failing test cases, the system enters a retry-loop (up to three times). When test cases are provided, the patch generation agent is called again with reports of failed attempts [17]. This closed-loop scheme improved the `pass@3` resolution rate on SWE-bench from 17.96 percent to 26 percent on the lite set. The authors note that iterative refinement with execution feedback hugely increases reliability, but at the cost of a maximum of three retries. Longer sequences would require more advanced rollback strategies.

The "compensating transactions" pattern from distributed systems theory applies here: a long-running agent workflow must include compensating transactions for each action changing the system's state. This can be invoked if any subsequent actions within the workflow fail.

. For example, if an agent books a flight and then books a hotel, and the hotel booking fails, the compensation action is to cancel the flight booking. Formalising this pattern for agent systems requires that every instruction be paired with a reversible or idempotent counterpart [2]. The Agent Constitution Framework operationalises this by typing instructions according to their governance properties: a TOOL_CALL instruction that modifies external state must be paired with a VERIFY instruction that checks success and a FALLBACK instruction that defines recovery actions. Table 1 summarises the resilience patterns identified in the reviewed literature and their applicability to agentic systems.

Table 1. Resilience Patterns for Agentic Systems [2, 3, 17]

Pattern	Description	Agent Applicability
Compensating transaction	Every state-modifying action has a defined inverse	High (requires action typing)
Idempotent operation	Executing an action multiple times has the same effect as once	High (requires API design)
Checkpoint / rollback	System state is checkpointed periodically; failure restores to last checkpoint	Medium (state size challenge)
Retry with backoff	Failed action is retried with increasing delays	High (requires failure classification)
Circuit breaker	After threshold failures, subsequent calls fail immediately	High (requires monitoring)
Bulkhead isolation	Different agent groups use separate resources	High (requires infrastructure)

8. Synthesis and Research Agenda

The reviewed literature reveals that while substantial progress has been made on individual components of agentic systems, security, observability, governance, resilience, no integrated architectural framework yet exists that addresses all four domains simultaneously. Table 2 summarises the coverage of the reviewed studies across the thematic domains identified in this review.

Table 2. Coverage of Thematic Domains in Reviewed Studies

Reference	Security	Observability	Governance	Resilience
Kunda [1]	✓	✓	✓	✓
Xu et al. [2]	✓	✓	✓	✓
Desai [4]	✓	✓	✓	
Masterman et al. [5]	✓			
Wang et al. [10]	✓		✓	
Liu et al. [13]	✓			
Moshkovich and Zeltyn [16]		✓		

Reference	Security	Observability	Governance	Resilience
Boddupally [21]			✓	
Zwerdling et al. [19]	✓		✓	
Hong et al. [14]	✓			
Qian et al. [15]				✓

Only two studies explicitly address all four domains, and both are recent (2025) preprints proposing integrated frameworks [1, 2]. This pattern indicates that the field is currently in a synthesis phase: individual components have been validated in isolation, but their combination into enterprise-ready architectures remains an open research question.

Based on this synthesis, I propose the Agent-Oriented Enterprise Architecture (AOEA) as a conceptual framework requiring empirical validation. AOEa posits six architectural layers: (1) agent runtime isolation, enforcing sandboxed execution and least-privilege access; (2) intent interpretation and planning, translating natural language goals into structured action plans; (3) deterministic validation, enforcing schema, business rule, and compliance checks before action execution; (4) human-in-the-loop governance, with approval workflows for high-risk operations; (5) observability and explainability, capturing reasoning traces and action logs; and (6) resilience engineering, implementing compensating transactions and safe retry logic.

The following research propositions are offered for empirical testing:

Proposition 1 (Isolation). Agent execution in sandboxed environments with time-bound, scoped API tokens reduces unauthorised cross-system interactions by at least 70 percent compared to agents with persistent, broad permissions.

Proposition 2 (Validation). Deterministic validation layers that check schema correctness, business rule compliance, and role-based entitlements before probabilistic action execution reduce policy violation rates by at least 50 percent.

Proposition 3 (Observability). Full trace capture of agent reasoning chains reduces mean time to root-cause identification for agent failures by at least 60 percent compared to action-only logging.

Proposition 4 (Resilience). Agent systems with compensating transactions and idempotent API designs achieve at least 95 percent recovery success for multi-step failures, compared to 40 percent for systems without such patterns.

These propositions are intended to guide future empirical research, including controlled experiments, case studies, and industry deployments.

9. Discussion and Limitations

This review has several limitations that merit acknowledgment. First, the literature search was restricted to English-language publications and five databases, which may have excluded relevant studies published in other languages or indexed in regional databases. Second, the inclusion of preprint server submissions (arXiv) was necessary to capture the rapid pace of research in this emerging field but may have introduced studies that have not undergone formal peer review. Third, the quality assessment using DARE criteria, while standard for systematic reviews, is not specifically validated for agent architecture studies. Fourth, the quantitative metrics reported from primary studies are not always directly comparable due to differences in evaluation tasks, datasets, and success criteria.

Despite these limitations, the review provides a structured foundation for the AOEa research agenda. Practitioners should note that current agent deployments in production remain limited; the majority of empirical evidence comes from benchmark evaluations (SWE-bench, τ -bench) or controlled experiments, not from long-running production systems. The generalisability of findings from these benchmarks to enterprise settings is not yet established.

10. Conclusion

The enterprise challenge posed by autonomous generative AI agents is not whether they can automate workflows. They can. The challenge is whether they can do so securely, deterministically, observably, governably, and reliably. This systematic review has identified four persistent architectural gaps, security and authorisation, observability and traceability, governance and compliance, resilience and rollback, that current literature fails to address in an integrated manner. The proposed Agent-Oriented Enterprise Architecture framework offers a research agenda for closing these gaps, but empirical validation remains an urgent priority. Organisations that establish architectural discipline now will define the standard operating model for the coming generation of AI-driven enterprises.

Funding: Funding information was not supplied in the manuscript.

Conflicts of Interest: A conflict-of-interest declaration was not supplied in the manuscript.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

- Boddupally, H. L. (2024). Embedding Governance into LLM Workflow Architectures for Enterprise-Wide Automation. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, 279. <https://doi.org/10.32628/cseit24107450>
- Boddupally, H. L. (2026, February 24). Self Improving Enterprise Platforms Using Learning Loops and AI Driven Orchestration. SSRN. <http://dx.doi.org/10.2139/ssrn.6270638>
- Boddupally, H. L. (2024). Cognitive decision automation framework integrating LLMs with SQL datastores and enterprise rule engines. *Journal of Artificial Intelligence, Machine Learning and Data Science*, 2(3), 1–8. <https://doi.org/10.51219/JAIMLD/hema-latha-boddupally/644>
- Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., West, P., Bhagavatula, C., Bras, R. L., Hwang, J. D., Sanyal, S., Welleck, S., Ren, X., Ettinger, A., Harchaoui, Z., & Choi, Y. (2023, June 1). Faith and Fate: Limits of Transformers on Compositionality. *ArXiv.org*. <https://doi.org/10.48550/arXiv.2305.18654>
- European Union. (2024). Regulation - EU - 2024/1689 - EN - EUR-Lex. Europa.eu. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng>
- Hoblitzell, A. (2025, December 21). QConAI NY 2025 - Designing AI Platforms for Reliability: Tools for Certainty, Agents for Discovery. InfoQ. <https://www.infoq.com/news/2025/12/qcon-nvidia-platform/>
- Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., & Wu, C. (2023, August 7). MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. *ArXiv.org*. <https://doi.org/10.48550/arXiv.2308.00352>
- Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., & Scarfone, K. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. <https://doi.org/10.6028/nist.sp.800-162>
- Kitchenham, B. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf
- Kunda, A. K. (2025). Policy-Governed Agentic Autonomous Quality Assurance for Enterprise Systems. *International Journal of Computational and Experimental Science and Engineering*, 11(4). <https://doi.org/10.22399/ijcesen.4933>
- Liu, Y., Zhang, R., Luo, H., Lin, Y., Sun, G., Niyato, D., Du, H., Xiong, Z., Wen, Y., Jamalipour, A., Kim, D. I., & Zhang, P. (2025). Secure Multi-LLM Agentic AI and Agentification for Edge General Intelligence by Zero-Trust: A Survey. *ArXiv.org*. <https://arxiv.org/abs/2508.19870>
- Liu, Y., Deng, G., Li, Y., Wang, K., Zhang, T., Liu, Y., Wang, H., Zheng, Y., & Liu, Y. (2023, June 8). Prompt Injection attack against LLM-integrated Applications. *ArXiv.org*. <https://doi.org/10.48550/arXiv.2306.05499>
- Masterman, T., Besen, S., Sawtell, M., & Chao, A. (2024, April 17). The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey. *ArXiv.org*. <https://doi.org/10.48550/arXiv.2404.11584>
- Moshkovich, D., & Zeltyn, S. (2025). Taming Uncertainty via Automation: Observing, Analyzing, and Optimizing Agentic AI Systems. *ArXiv.org*. <https://arxiv.org/abs/2507.11277>
- Mui, P. (2025, October 20). Agentforce's Agent Graph: Toward Guided Determinism. *Salesforce Engineering Blog*. <https://engineering.salesforce.com/agentforces-agent-graph-toward-guided-determinism-with-hybrid-reasoning/>
- Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., & McGuinness, L. A. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *British Medical Journal*, 372(71). <https://doi.org/10.1136/bmj.n71>
- Pandey, S. (2025). Beyond Human-in-the-Loop: A Level 5 Autonomous Multi-Agent Framework for Master Data Management and Governance using Graph States and Epistemic Uncertainty Estimation. <https://doi.org/10.36227/techrxiv.176523326.65231520/v1>
- Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., & Sun, M. (2023, July 18). Communicative Agents for Software Development. *ArXiv.org*. <https://doi.org/10.48550/arXiv.2307.07924>
- Sudhindra Desai. (2025). The Seven Pillars of Agentic AI Implementation in Enterprise Systems. *Journal of Information Systems Engineering and Management*, 10(62s), 167–179. <https://doi.org/10.52783/jisem.v10i62s.13565>

- Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2023, September 7). A Survey on Large Language Model based Autonomous Agents. ArXiv.org. <https://doi.org/10.48550/arXiv.2308.11432>
- Wang, Z., Li, J., Zhou, Q., Si, H., Liu, Y., Li, J., Xie, G., Sun, F., Pei, D., & Pei, C. (2025). A Survey on AgentOps: Categorization, Challenges, and Future Directions. ArXiv.org. <https://arxiv.org/abs/2508.02121>
- Xu, Q., Wen, X., Xu, C., Li, Z., & Zhong, J. (2025). From Craft to Constitution: A Governance-First Paradigm for Principled Agent Engineering. ArXiv.org. <https://arxiv.org/abs/2510.13857>
- Yousuf Uzair, M., Rehan, M., & Chand, M. (2025). Issue 4 | www.ijedr.org IJEDR2504698. International Journal of Engineering Development and Research, 13, 510. <https://rjwave.org/ijedr/papers/IJEDR2504698.pdf>
- Zhang, Y., Ruan, H., Fan, Z., & Roychoudhury, A. (2024). AutoCodeRover: Autonomous Program Improvement. ACM. <https://doi.org/https://doi.org/10.1145/3650212.3680384>
- Zwerdling, N., Boaz, D., Rabinovich, E., Uziel, G., Amid, D., & Anaby-Tavor, A. (2025). Towards Enforcing Company Policy Adherence in Agentic Workflows. ArXiv.org. <https://arxiv.org/abs/2507.16459>