Frontiers in Computer Science and Artificial Intelligence
ISSN: 2978-8048
DOI: 10.32996/fcsai
Journal Homepage: www.al-kindipublisher.com/index.php/fcsai

FCSAI
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Failure-Mode Analysis, Hallucination Detection, and Prompt-Injection Testing: A Production-Readiness Framework for Enterprise Agentic AI

Prasad Maderamitla
Independent researcher, California, USA
Corresponding Author: Prasad Maderamitla, **E-mail**: prasad.madera@gmail.com

| ABSTRACT

As large language models increasingly move from experimental chat interfaces into enterprise workflows, the primary challenge is no longer model capability alone but production trustworthiness. Agentic AI systems introduce new risk surfaces because they can retrieve information, reason across context, call tools, act on user intent, and interact with external systems. This article proposes a practical production-readiness framework centered on three pillars: failure-mode analysis, hallucination detection, and prompt-injection testing. Failure-mode analysis helps teams identify where agentic systems can break under real operating conditions. Hallucination detection evaluates whether generated outputs remain faithful, factual, and grounded in approved sources. Prompt-injection testing examines whether malicious or conflicting instructions can override system intent, expose sensitive information, or trigger unauthorized actions. Together, these practices convert AI reliability from an informal review process into an evidence-driven engineering discipline. The article argues that enterprises should treat these controls as mandatory gates before deploying large language model and agentic AI systems into high-impact business workflows.

| KEYWORDS

Agentic AI; LLM evaluation; hallucination detection; prompt injection; AI red teaming; enterprise AI governance; failure-mode analysis; production readiness

| ARTICLE INFORMATION

ACCEPTED: 01 April 2026 **PUBLISHED:** 09 May 2026 **DOI:** 10.32996/jcsts.2026.5.6.6

1. Introduction

Generative AI has rapidly evolved from single-turn assistants into agentic systems capable of planning, retrieval, tool use, workflow orchestration, and autonomous or semi-autonomous decision support. In enterprise environments, this shift creates significant value: AI agents can improve support operations, software development, knowledge discovery, sales productivity, and internal automation. However, the same capabilities also introduce new reliability and security risks. A model that merely generates text can produce inaccurate content; an agent connected to tools, data, APIs, or workflows can produce inaccurate content and also take harmful action.

This is why production-readiness for enterprise AI must go beyond accuracy benchmarks. NIST's AI Risk Management Framework emphasizes risk management across the design, development, deployment, and use of AI systems, while the NIST Generative AI Profile extends this approach to risks that are unique to or amplified by generative AI (NIST, 2023; Autio et al., 2024). For enterprise agentic AI, the central question is not simply, 'Can the model answer?' The more important question is, 'Can the system behave safely, consistently, and verifiably when exposed to ambiguous user intent, incomplete data, adversarial input, and changing production context?'

This article proposes that three disciplines should become core elements of enterprise AI quality engineering: failure-mode analysis, hallucination detection, and prompt-injection testing. Each discipline addresses a different aspect of trustworthiness. Failure-mode analysis identifies how a system can break. Hallucination detection evaluates whether generated outputs are

factual and grounded. Prompt-injection testing assesses whether adversarial or conflicting instructions can manipulate system behavior. Together, they form a practical framework for production-readiness.

2. Failure-Mode Analysis for Agentic AI

Failure-mode analysis is a structured method for identifying how a system can fail, why it can fail, and what impact the failure may have. In traditional software systems, failure modes often arise from deterministic defects: incorrect logic, missing validation, configuration errors, integration failures, or infrastructure faults. In LLM-based systems, failure modes are broader because model behavior is probabilistic, context-sensitive, and influenced by natural-language instructions.

Agentic AI expands the failure surface further. Microsoft AI Red Team's taxonomy of agentic AI failure modes highlights the importance of categorizing failures across safety and security dimensions, especially as systems move toward multi-agent coordination and tool-mediated execution (Microsoft AI Red Team, 2025). A production-grade failure-mode analysis should therefore account for both conventional software defects and AI-specific behavioral failures.

Recent trajectory-level evaluation research by Maderamitla and Katragadda (2026a) further supports this view by emphasizing deterministic replay, comprehensive action-trace logging, and governance-aware validation as mechanisms for detecting reasoning drift and non-deterministic behavior in learning-based agentic systems.

2.1 Key Failure Categories

- Reasoning failures: The agent reaches an incorrect conclusion, skips necessary steps, overgeneralizes from weak evidence, or fails to recognize uncertainty.
- Retrieval and grounding failures: The agent retrieves irrelevant, outdated, incomplete, or low-trust information and treats it as authoritative.
- Tool-use failures: The agent calls the wrong tool, passes incorrect parameters, executes steps in the wrong order, or fails to verify tool output before continuing.
- Memory and context failures: The system misuses prior context, carries forward stale assumptions, confuses users or entities, or applies historical information to the wrong situation.
- Coordination failures: Multiple agents duplicate work, contradict one another, escalate incorrectly, or produce inconsistent intermediate outputs. These failures are difficult to detect when evaluation focuses only on the final response.

The practical output of failure-mode analysis should be a risk register, test matrix, severity model, mitigation plan, and release gate. Enterprises should classify failures by impact, such as user confusion, factual error, policy violation, data leakage, unauthorized action, financial loss, compliance exposure, or reputational harm.

3. Hallucination Detection as a Trustworthiness Control

Hallucination occurs when an AI system generates content that appears plausible but is unsupported, unfaithful to source material, or factually incorrect. This issue has been studied extensively in natural language generation. Research on abstractive summarization found that neural generation models can produce content that is not faithful to input documents, while broader surveys describe hallucination as a persistent challenge across generation tasks (Maynez et al., 2020; Ji et al., 2023).

For enterprise AI, hallucination is not merely a model-quality issue. It is a business-risk issue. A hallucinated answer in a customer-support assistant can mislead a customer. A hallucinated compliance summary can create legal exposure. A hallucinated technical recommendation can cause engineering rework. A hallucinated sales or financial insight can influence business decisions.

3.1 Layers of Hallucination Detection

1. Source-grounding verification: Every factual claim should be traceable to an approved source, retrieved document, database record, or tool output.
2. Claim-level decomposition: Long responses should be broken into individual claims and evaluated independently.
3. Contradiction detection: Generated claims should be checked against retrieved context, policy documents, tool outputs, and prior conversation state.
4. Uncertainty calibration: The system should appropriately say 'I do not know,' request clarification, or cite limitations when evidence is insufficient.

5. Regression testing: Known hallucination patterns should become reusable tests so future prompt, retrieval, tool, or model changes do not reintroduce prior defects.

Benchmarks such as TruthfulQA show that language models can reproduce common human misconceptions rather than truth, which is important for enterprise use cases where persuasive fluency can be mistaken for correctness (Lin et al., 2022). SelfCheckGPT demonstrates one practical direction for hallucination detection by sampling multiple model responses and checking consistency across generations when internal model probabilities or external knowledge bases are unavailable (Manakul et al., 2023).

In production systems, hallucination detection should combine automated checks with human review for high-impact workflows. Automated systems can flag unsupported claims, missing citations, retrieval mismatch, and inconsistent answers. Human experts should review edge cases, high-severity categories, and failures that reveal gaps in product design or knowledge coverage.

4. Prompt-Injection Testing and AI Security

Prompt injection is one of the most important security risks in LLM applications. OWASP identifies prompt injection as a vulnerability in which user-controlled or externally sourced content can alter an LLM's behavior in unintended ways (OWASP, 2025). The risk includes direct attacks, where the user explicitly attempts to override instructions, and indirect attacks, where malicious instructions are embedded in content such as websites, documents, support tickets, emails, or retrieved knowledge articles.

Prompt injection is challenging because LLMs process instructions and data within the same context window. Traditional software systems usually separate executable commands from inert data. LLM-integrated applications blur that boundary: data retrieved for answering a question may also contain language that looks like an instruction. This makes prompt-injection testing essential for any enterprise AI system connected to tools, documents, APIs, or business workflows.

Research on prompt injection against LLM-integrated applications has shown that attacks can manipulate application behavior, expose hidden instructions, and influence connected functionality (Liu et al., 2023). Work on indirect prompt injection has further shown that malicious instructions embedded in retrieved content can influence LLM-integrated applications even when the attacker does not directly interact with the user prompt (Greshake et al., 2023).

4.1 Recommended Prompt-Injection Test Classes

- Instruction override tests: Attempts to make the model ignore system instructions, policies, or role boundaries.
- Data exfiltration tests: Attempts to reveal hidden prompts, private records, credentials, internal reasoning, or sensitive user data.
- Tool misuse tests: Attempts to make the agent call unauthorized tools, modify records, send messages, or perform destructive actions.
- Indirect injection tests: Malicious content placed inside retrieved documents, web pages, tickets, emails, PDFs, or knowledge articles.
- Multi-turn persistence tests: Attempts to plant instructions that influence later turns after the original malicious input is no longer visible.
- Cross-context confusion tests: Attempts to make the model confuse one user, account, tenant, case, or permission context with another.

Prompt-injection testing should be conducted before launch and continuously after launch. New tools, new retrieval sources, new prompts, new policies, and new model versions can all change the system's security posture.

5. A Unified Production-Readiness Framework

Failure-mode analysis, hallucination detection, and prompt-injection testing should not be separate activities owned by different teams. They should form a unified AI production-readiness framework. The following lifecycle converts these controls into a repeatable engineering process.

1. System mapping: Document the model, system prompts, user roles, retrieval sources, tools, APIs, memory mechanisms, logging flows, and escalation paths.

2. Risk classification: Assign risk levels based on potential impact, such as factual harm, unauthorized action, sensitive data exposure, compliance risk, or business disruption.

3. Test design: Build test suites covering normal user tasks, edge cases, ambiguous requests, adversarial prompts, retrieval corruption, tool-use errors, and multi-turn scenarios.

4. Measurement: Track hallucination rate, groundedness, refusal accuracy, prompt-injection success rate, unauthorized tool-call rate, retrieval relevance, policy compliance, escalation accuracy, and regression failure rate.
5. Release gating: Define minimum quality, safety, and security thresholds before an agentic system is promoted to production.
6. Continuous monitoring: Use production telemetry, user corrections, red-team findings, incident reports, and drift analysis to update test suites and mitigations.

Observability practices for LLM applications should also define what to log, how to preserve privacy, and which operational KPIs can reveal semantic correctness, behavioral consistency, and safety-layer issues during production use (Maderamitla & Katragadda, 2026b).

5.1 Production-Readiness Test Matrix

Control Area	**Example Risk**	**Example Test**	**Suggested Metric**
Failure-mode analysis	Wrong action triggered by flawed reasoning	Scenario-based tests with ambiguous user goals	Critical failure rate; escalation accuracy
Retrieval grounding	Outdated or irrelevant source used as evidence	Inject stale or conflicting documents into retrieval set	Groundedness score; citation accuracy
Hallucination detection	Confident unsupported claim	Claim-level factuality review against source material	Unsupported claim rate
Prompt-injection testing	System instruction override or data leakage	Direct and indirect adversarial prompts	Injection success rate; leakage rate
Tool-use validation	Wrong API call or unsafe side effect	Permission-boundary and parameter-tampering tests	Unauthorized tool-call rate
Monitoring and regression	Previously fixed defect returns after model or prompt change	Replay known failures across releases	Regression pass rate; time to detect

Table 1. Example controls and metrics for production-readiness evaluation of enterprise agentic AI systems.

6. Enterprise Implications

The enterprise adoption of agentic AI depends on trust. Business leaders need confidence that AI systems can produce reliable outputs, respect access boundaries, resist manipulation, and escalate appropriately when uncertain. Engineering leaders need repeatable methods to test these systems before deployment. Compliance and security teams need evidence that risks are being controlled.

The proposed framework provides that evidence. Failure-mode analysis shows that the organization understands how the system can break. Hallucination detection shows that the organization can measure factual reliability and grounding. Prompt-injection testing shows that the organization is actively evaluating adversarial misuse.

This evidence is also valuable for governance. It supports auditability, incident response, model-risk management, and responsible AI documentation. Instead of relying on subjective confidence in model behavior, organizations can build measurable controls into the AI development lifecycle.

7. Conclusion

Agentic AI will not reach enterprise maturity through capability improvements alone. The future of production AI depends on disciplined evaluation, adversarial testing, and risk-aware engineering. Failure-mode analysis identifies where systems can break. Hallucination detection evaluates whether outputs are factual and grounded. Prompt-injection testing assesses whether adversarial instructions can manipulate behavior or trigger unauthorized outcomes.

Together, these practices define a practical foundation for production-ready enterprise AI. They help organizations move beyond impressive demonstrations toward systems that are reliable, secure, measurable, and governable. As agentic AI becomes more deeply embedded in business workflows, these controls should be treated not as optional enhancements but as core requirements for responsible deployment.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
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

[1]. Autio, C., Schwartz, R., Dunietz, J., Jain, S., Stanley, M., Tabassi, E., Hall, P., & Roberts, K. (2024). Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. National Institute of Standards and Technology. URL: https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence
[2]. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. URL: https://arxiv.org/abs/2302.12173
[3]. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Chen, D., Dai, W., Chan, H. S., Madotto, A., & Fung, P. (2023). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), Article 248. URL: https://dl.acm.org/doi/10.1145/3571730
[4]. Lin, S., Hilton, J., & Evans, O. (2022). TruthfulQA: Measuring how models mimic human falsehoods. Proceedings of the Association for Computational Linguistics. URL: https://aclanthology.org/2022.acl-long.229/
[5]. Liu, Y., Deng, G., Li, Y., Wang, K., Wang, Z., Wang, X., Zhang, T., Liu, Y., Wang, H., Zheng, Y., & Liu, Y. (2023). Prompt injection attack against LLM-integrated applications. URL: https://arxiv.org/abs/2306.05499
[6]. Manakul, P., Liusie, A., & Gales, M. J. F. (2023). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. URL: https://arxiv.org/abs/2303.08896
[7]. Maderamitla, P., & Katragadda, S. R. (2026a). A Deterministic Trajectory-Level Evaluation Framework for Learning-Based Agentic Systems. Journal of Economics, Finance and Accounting Studies, 8(3), 25-29. DOI: https://doi.org/10.32996/jefas.2026.8.3.3; Google Scholar citation record: Google Scholar
[8]. Maderamitla, P., & Katragadda, S. R. (2026b). Observability for LLM apps: what to log, privacy-safe telemetry, KPIs. Frontiers in Computer Science and Artificial Intelligence, 5(4), 10-14. DOI: https://doi.org/10.32996/jcsts.2026.5.4.2; Google Scholar citation record: Google Scholar
[9]. Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. Proceedings of the Association for Computational Linguistics. URL: https://aclanthology.org/2020.acl-main.173/
[10]. Microsoft AI Red Team. (2025). Taxonomy of failure modes in agentic AI systems. URL: https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Taxonomy-of-Failure-Mode-in-Agentic-AI-Systems-Whitepaper.pdf
[11]. National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). URL: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf
[12]. OWASP Gen AI Security Project. (2025). LLM01:2025 Prompt Injection. URL: https://genai.owasp.org/llmrisk/llm01-prompt-injection/

Author Note:
This article presents the author's independent analysis of enterprise AI reliability, including failure-mode analysis, hallucination detection, and prompt-injection testing. The discussion is based on publicly available research and industry guidance, with references provided for further reading. Any practical examples included are generalized and do not disclose confidential or proprietary information.