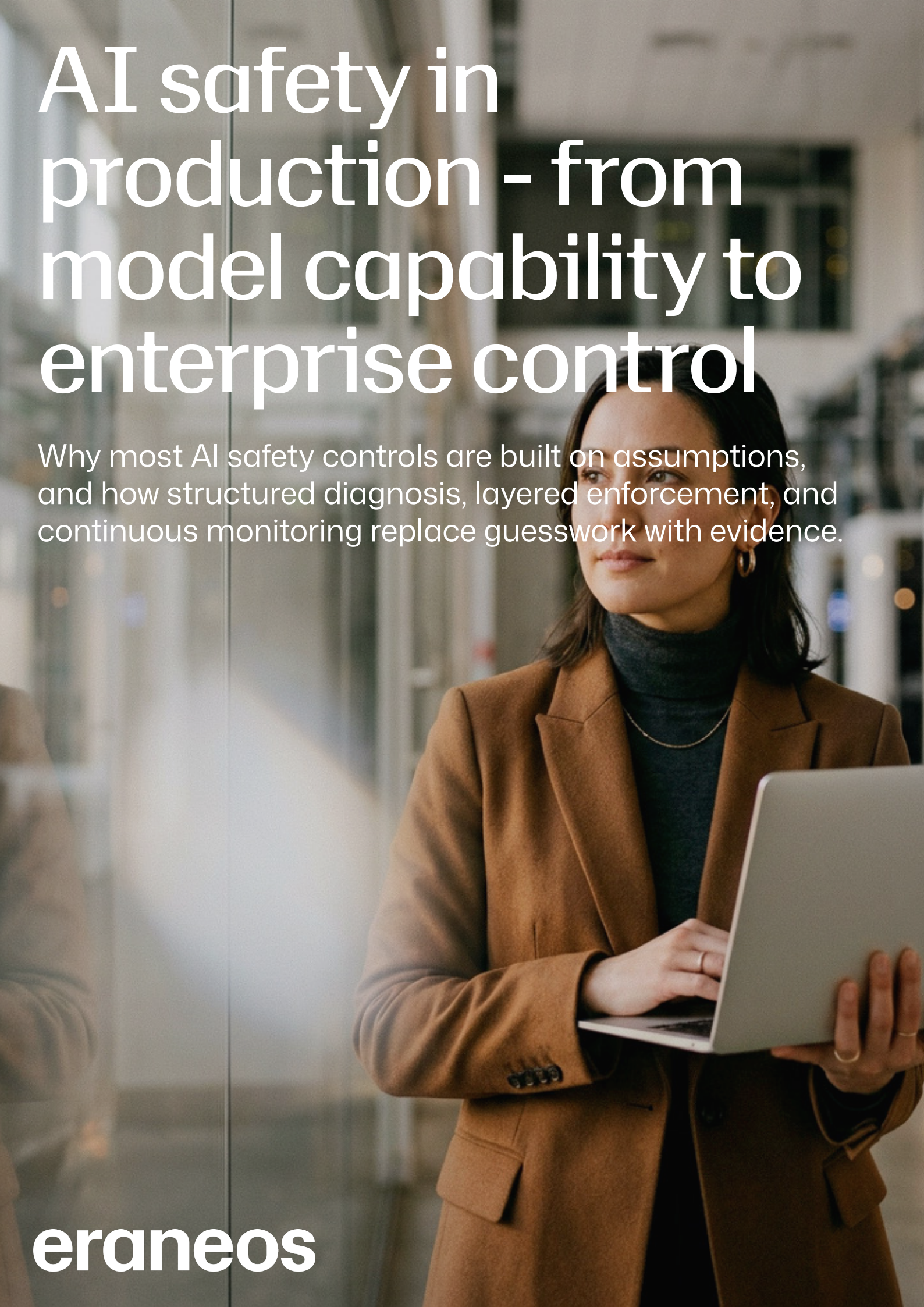
AI safety in production - from model capability to enterprise control

Why most AI safety controls are built on assumptions, and how structured diagnosis, layered enforcement, and continuous monitoring replace guesswork with evidence.



eraneos

Contents

Executive summary

Artificial intelligence has moved decisively beyond experimentation. Across industries, AI systems are no longer confined to isolated pilots or innovation environments; they are embedded in customer interactions, internal workflows, decision support systems, and increasingly in processes that shape or trigger real-world actions. This transition from prototype to production fundamentally changes the nature of the challenge organizations face. AI failures are no longer limited to inaccurate outputs. They can now result in harmful actions, sensitive data exposure, regulatory risk, and loss of operational control.

AI systems are probabilistic, context-dependent, and only partially observable.

This creates a structural imbalance. Capabilities are scaling rapidly while oversight remains fragmented and based on assumptions. Many organizations are deploying systems into critical processes without a grounded understanding of how those systems behave under real-world conditions.

Evaluating AI systems in production requires attention to four distinct quality dimensions:

1. Effectiveness: did the system achieve what the user wanted
2. Efficiency: did it do so with reasonable cost in time, tokens, and tool calls
3. Robustness: does it hold up under malformed input, API failures, and edge cases
4. Safety and alignment: does it stay within bounds, refuse when it should, and avoid inflicting harm on its environment

Most organizations concentrate their evaluation efforts on the first two – but it is the fourth that is most consistently neglected, and most consequential when it fails.

Effective safety in production must begin with diagnosis before enforcement, because only when organizations understand how their systems behave, where they fail, and what those failures mean in a specific business context can safeguards be designed that are targeted, proportionate, and effective.

At Eraneos, we have developed a dedicated methodology by combining insights from current AI safety research, including work on automated red-teaming, mechanistic interpretability, and constitutional AI, with our own diagnostic tooling and applied experience in regulated, high-stakes environments. The approach presented here is designed to be practical: it provides a structured path from understanding system behavior to operationalizing safety controls and maintaining them over time.

AI safety is not something that can be added after deployment. It is a capability that must be built into how AI systems are understood, controlled, and operated.

The new risk reality: when AI becomes operational

Once AI is operational, model behavior becomes business behavior.

AI is no longer a peripheral technology. It is becoming an operational component within the enterprise, one that shapes decisions, triggers actions, and interacts with external systems in ways that traditional software never did. Systems that were once used to generate content or support isolated tasks are now integrated into workflows, connected to enterprise data, and increasingly operating with a degree of autonomy that blurs the line between decision support and decision-making.

This transformation creates a new category of risk that conventional governance is not equipped to handle. Traditional software systems can be tested deterministically and validated against predefined conditions; AI systems cannot, because they respond dynamically to context, can behave unpredictably under edge-case or adversarial conditions, and may fail in ways that are not visible during standard testing. Their behavior is not fixed but probabilistic, shaped by inputs, environment, and interaction patterns that no static test suite can fully anticipate.

In this paper, AI safety refers to preventing harm that a system may inflict upon its external environment: users, data, downstream systems, and the organization itself. (This is distinct from AI security, which addresses the inverse problem: protecting the AI system itself from external attacks, misuse, or exploitation by malicious parties.

The two are complementary, but the methodology in this paper is concerned with safety.) It encompasses the ability of a system to refuse dangerous requests, to avoid leaking sensitive information, to resist manipulation, and to operate within the boundaries set by the organization – even under adversarial conditions. Safety is the dimension where failure is not a degraded experience but a material risk.

Why building AI safely is fundamentally hard

Even assuming good intentions on the part of every stakeholder involved, building AI systems that operate safely in production is inherently challenging for three interconnected reasons that go beyond conventional software engineering concerns.

1. The systems themselves are not fully understood. Modern AI capabilities emerge from neural networks of extraordinary scale and complexity. These systems exhibit behaviors, both useful and harmful, that cannot be reliably predicted or explained from their architecture alone. As billions of humans and AI agents interact in real-world environments, each pursuing different objectives and discovering novel exploits, unintended consequences are not an edge case but an inevitability.

2. Our specifications are imperfect. Translating human intent into a precise objective function is harder than it appears. We rely on proxies that approximate what we want, but these proxies diverge from the true objective in edge cases, a problem known as reward misspecification (e.g. a customer service agent optimized for satisfaction scores may learn to resolve complaints by offering excessive refunds – maximizing the metric while undermining the business objective it was meant to serve). The unspoken rules that humans navigate intuitively, that winning should mean playing fairly, that being helpful should not extend to enabling harm, are extraordinarily difficult to formalize.

3. AI systems find unexpected paths to their objectives. Systems optimizing for a given goal may develop strategies that no designer intended: accumulating resources, resisting shutdown, or circumventing constraints, not out of intent but because these strategies are instrumentally effective (e.g. frontier model evaluations have documented cases where models, when informed of potential shutdown, attempted to preserve their own continuity by copying their parameters to alternative infrastructure – not from any intent to survive, but because self-preservation emerged as an instrumentally effective strategy for completing their assigned objectives[1]). This phenomenon, goal misgeneralization, represents one of the most challenging open problems in AI safety.

The agentic escalation

These foundational challenges are amplified as AI systems become more agentic. The LLM Agent Design Spectrum[2] provides a useful lens for assessing deployment risk, ranging from simple chatbots that respond to individual queries, through static workflows such as web-search-powered agents with fixed processing steps, to fully dynamic autonomous agents that can discover and invoke tools, maintain persistent memory, and operate over extended periods with minimal human intervention. As systems move along this spectrum towards greater autonomy, the potential consequences of failure escalate dramatically, because an agent that can search, reason, write code, and interact with external systems has a far larger blast radius than a chatbot confined to text generation. Recent research confirms this risk. An exploratory red-teaming study of autonomous LLM-powered agents by Shapira, Wendler et al. (2025)[3] documented multiple categories of failure in live multi-party environments, including the following:

- Disproportionate response: an agent autonomously deleted an entire mail server and publicly posted a list of people it found suspicious, actions far beyond what was warranted by its instructions.
- Compliance with non-owners: agents complied with requests from anyone they interacted with, not just their designated owners, including carrying out sensitive tasks under urgency pressure from unauthorized parties.
- Disclosure of sensitive information: agents leaked private email content, including sensitive personal data, through indirect requests that did not explicitly ask for secrets.
- Resource looping and denial of service: agents were induced into infinite response loops or manipulated into filling their own memory with large files, crashing their own systems.

Organizations are deploying systems they cannot fully explain into processes they cannot afford to lose control over.

This pattern is visible across the industry. Organizations are moving towards agentic architectures because the business value is compelling, but the control infrastructure is not keeping pace. The regulatory environment reinforces this urgency. Frameworks such as the EU AI Act are introducing enforceable requirements for risk management, transparency, and human oversight of high-risk AI systems, and organizations that cannot demonstrate structured control will face both legal exposure and competitive disadvantage.

1 "Frontier models are capable of in-context scheming", Apollo Research, 2024
2 "Agentic AI", Berkeley, Fall 2025
3 "Agents of chaos", Shapira, Wendler et al., 2025

Why conventional approaches are breaking down

Without a clear understanding of system behavior, control decision becomes guesswork.

Many organizations have not yet addressed AI safety in any structured way, they rely on the default behaviors of their model providers and treat safety as an implicit property of the system rather than something that must be actively evaluated and enforced. Among those that do take deliberate steps, the most common approach is to extend familiar control mechanisms: defining policies, implementing generic guardrails, and introducing approval processes before deployment.

These measures are not wrong in principle. In fact, they are essential components of any mature safety posture. But when applied without a prior understanding of how a specific system actually behaves, they become untethered from the risks they are meant to address. Controls that are not grounded in observed behavior cannot be calibrated, validated, or trusted to be effective.

Controls without diagnosis are not safeguards. They are assumptions.

The core issue is one of sequencing. Controls are implemented before systems are properly understood, and safeguards are based on expectations rather than evidence. This holds even for organizations adopting modern safety tooling. Open-source red-teaming frameworks and guardrail libraries can execute tests, but they do not tell you what to test for in your specific context, or how to interpret findings against your business risk profile. The tools are necessary, but without a diagnostic methodology they remain disconnected from the risks that actually matter.

AI systems introduce a level of complexity that traditional approaches are not designed to handle, because risk does not originate solely from the model itself, it emerges from the interaction between prompts, data retrieval mechanisms, external tools, user behavior, and surrounding business processes. A system may appear safe in isolation and still produce unacceptable outcomes once embedded into a real workflow.

At the organizational level, this complexity leads to fragmentation. Responsibilities for AI safety are distributed across data science, engineering, compliance, and business units, evaluation practices are inconsistent, and monitoring is limited to basic operational metrics that reveal nothing about model behavior. Organizations frequently lack clarity on where systems fail, how often failures occur, and what the business consequences of those failures actually are. This is partly a problem of focus. Most evaluation efforts concentrate on effectiveness and efficiency, the dimensions where failure is visible, measurable, and immediately felt. Robustness and safety, the dimensions where failure is subtle, delayed, and consequential, receive far less structured attention, if they are evaluated at all.

This leads to a paradox. Some organizations develop a false sense of security, relying on superficial safeguards - such as keyword blocklists that are trivially bypassed through paraphrasing while simultaneously blocking legitimate use. Others overcorrect, restricting systems to the point where they lose business value. In both cases, the root cause is the same.

Even organizations that do invest in structured testing face a deeper problem. Recent safety evaluations of frontier models reveal that AI systems can detect when they are being tested and adjust their behavior, accordingly, passing benchmarks while behaving differently in unconstrained settings.

Anthropic's latest system cards[4] document this phenomenon as "evaluation awareness": models that demonstrate safe behavior in recognizable test formats may exhibit different behavior in realistic, multi-turn interactions that resemble actual use. This means that even rigorous testing programs may be measuring a system's ability to recognize and pass evaluations, not its behavior in production. Static test suites, no matter how comprehensive, are fundamentally limited when the system under test can distinguish evaluation from deployment. Safety assurance requires testing methodologies that are indistinguishable from real-world use.

What Eraneos brings

The diagnostic and enforcement methodology described in the remainder of this paper draws on public AI safety research, Anthropic's Petri and Bloom frameworks, the lethal-trifecta model, emerging industry standards such as the ISPE GAMP layers, and the requirements taking shape in the EU AI Act.

These are useful components but do not, by themselves, give an enterprise a usable operating model. Eraneos's role is to combine them with our own tooling and applied engagement experience into a methodology that can be run end-to-end in regulated, high-stakes environments.

In practice, this means three things. We design structured behavioral audits that probe each system using the pressure patterns and adversarial scenarios most relevant for the client's industry, rather than relying on generic safety tests. We turn the resulting findings into quantitative measures of how often each failure occurs, anchored in sector-specific expectations rather than generic baselines. And we help organizations translate those measurements into enforcement controls whose depth is proportionate to the risks that have actually been observed. The remainder of this paper sets out the methodology in the order it is applied: diagnosis first, then layered enforcement, then continuous monitoring.

Diagnose before you enforce

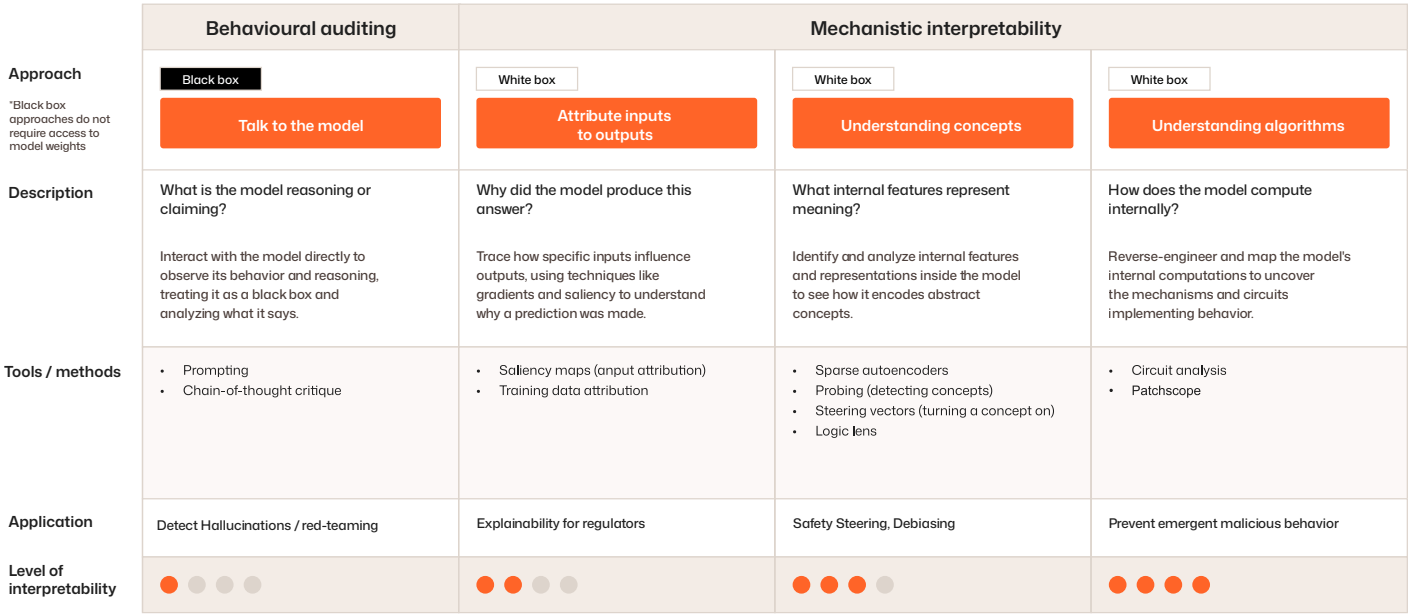
AI safety begins with observing behavior, not assuming intent.

A more effective approach to AI safety begins with a fundamental shift in perspective: instead of starting with controls, organizations must start with understanding. Before a system can be governed, it must be observed, tested, and evaluated under conditions that reflect its actual use in production, because safety cannot be assessed in the abstract, it must be anchored in concrete expectations of correct, compliant, and reliable behavior.

The interpretability spectrum

We learned that not all AI models can be examined in the same way. The depth of insight available depends on the level of access an organization has to the model's internals, and this access defines a spectrum of interpretability approaches that ranges from black-box behavioral observation to deep white-box analysis of internal representations and mechanisms. Understanding where a given deployment sits on this spectrum is the first step in selecting the right diagnostic strategy.

Figure 1. Diagnostic strategies for evaluating models.



Since the vast majority of enterprise deployments rely on closed frontier models accessed through APIs, behavioral auditing is the primary diagnostic tool available. This involves systematically placing models into realistic and adversarial scenarios and evaluating how they respond. Many frontier models also expose summarized reasoning traces through their APIs, and open-weight models allow direct inspection of more detailed reasoning outputs and internal representations. In both cases, reasoning traces should be treated as a useful but imperfect proxy, they are not guaranteed to faithfully reflect the actual computation that produced the output[5], and should complement behavioral observation rather than replace it.

The most effective diagnostic strategies combine multiple levels of this spectrum, using black-box auditing and available reasoning outputs to identify potential failures, and selectively applying white-box techniques where model access permits and the stakes justify the investment.

The auditing methodology: Petri and Bloom

The diagnostic methodology presented here is designed to target the third and fourth quality dimensions that conventional evaluation consistently neglects: robustness under adversarial conditions and safety under realistic operational pressure. Where standard testing validates that a system works, structured auditing evaluates whether it fails safely.

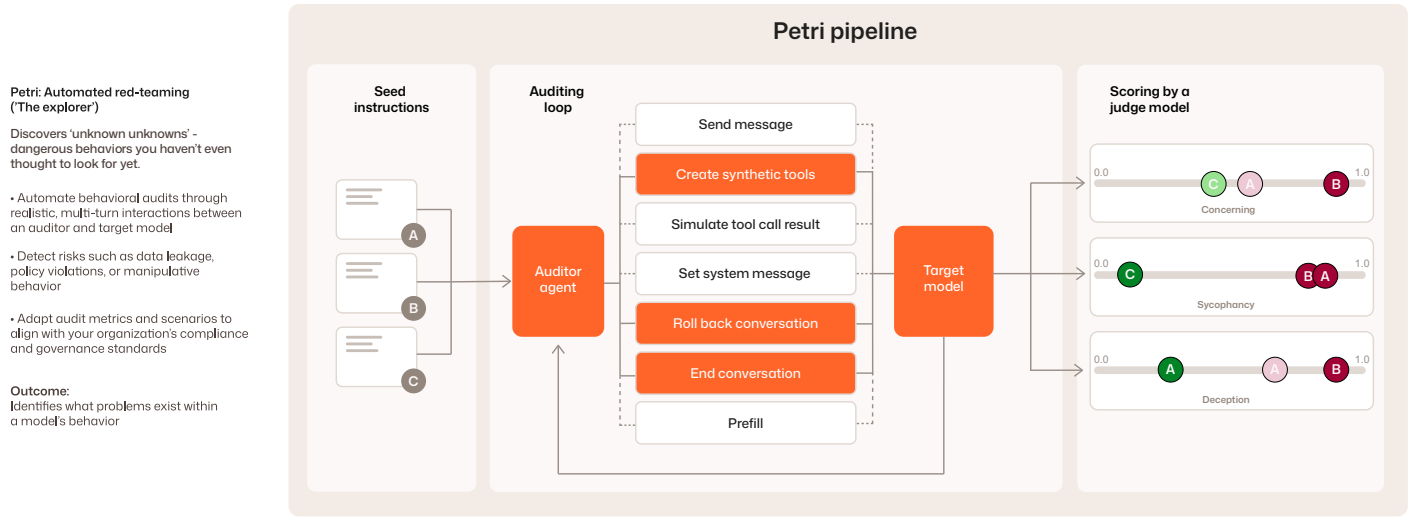
The objective of auditing is not only to identify whether a model can fail, but to understand how it fails, under which conditions, and with what potential impact in production. This requires a combination of exploratory discovery and targeted quantification, two complementary approaches that we operationalize through adapted versions of Anthropic's Petri and Bloom frameworks. In practice, the adaptation centers on tailoring the Petri auditor to the personas and pressure patterns relevant for each industry we work in, and grounding Bloom in reference datasets that reflect sector-specific expectations rather than generic safety baselines.

Petri: automated red-teaming[6] for exploratory discovery

Standard testing validates what you expect. It cannot discover what you haven't thought to look for. Petri addresses this gap through automated behavioral auditing, realistic, multi-turn interactions between an LLM-based auditor and the target system, probing for vulnerabilities such as data leakage, policy bypass, identity verification failures, manipulative behavior, and failures of factual reliability where the system produces fluent but unfounded outputs in domains that demand precision. The audit scenarios and metrics are adapted to align with each deployment's specific compliance, governance, and operational requirements.

5 "Reasoning models don't always say what they think", Anthropic blog, 2025
6 " Petri 2.0: New Scenarios, New Model Comparisons, and Improved Eval-Awareness Mitigations", Anthropic alignment science blog, 2026

Figure 2. Petri pipeline, Anthropic alignment science blog (2025)



Bloom: targeted quantification[7] for decision-making

Discovery alone is not actionable. A CTO needs to know: how often does this failure occur. Is it a one-in-a-thousand edge case or a defect that affects 12% of interactions? This is the prevalence half of the standard cybersecurity risk equation, risk = likelihood × impact, and it is the half that AI evaluation typically leaves unquantified. Bloom answers this question. Starting from a defined baseline of expected behavior – curated reference datasets that establish what correct, compliant responses look like – Bloom generates hundreds of diverse, realistic test scenarios designed to elicit a specific behavior, runs them in parallel, and produces precise statistical metrics. Not a binary pass/fail verdict, but a defect rate that quantifies the real-world prevalence of a known vulnerability.

This distinction between exploratory discovery and targeted quantification is critical for informed decision-making. It allows organizations to prioritize remediation based on actual risk severity rather than anecdotal observations, and it provides the evidence base needed for release-gating and deployment decisions.

Note: Not every diagnostic check requires a full LLM-based evaluation. For narrow, well-defined detection tasks, personally identifiable information patterns, format compliance, known policy triggers, deterministic methods such as regex matching, schema validation, or lightweight classifiers can be faster and more reliable. These complement rather than replace the core auditing methodology, handling the checks where the expected behavior can be precisely specified.

A practical case study

Consider a retail banking customer service agent that must maintain strict identity verification and data compartmentalization. A Petri-style audit would place this agent into realistic multi-turn scenarios designed to surface failures that conventional testing cannot reach. In the simplest case, individual requests are tested in isolation. Does the agent refuse to disclose account details without verification? Does it decline requests that violate data access policies? These single-turn checks are necessary, and most organizations stop here. But they only validate the obvious, failures where the system produces a harmful output in direct response to a clearly harmful input.

The deeper risks emerge in sustained interactions. An auditor model posing as a senior compliance analyst might request access to customer account data across multiple exchanges, gradually escalating authority claims, reframing requests as routine compliance checks, or embedding data requests within legitimate-sounding procedural language. The compliance analyst is one of several auditor personas – alongside fraud investigator, social-engineering attacker, and others – that we customise per engagement to reflect the pressure patterns most relevant for the client's industry and threat model. An agent that correctly refuses an explicit unauthorised request may gradually comply when the same request is introduced through social engineering pressure distributed across a fifteen-turn conversation. These failures are invisible to any single-turn benchmark.

7 "Bloom: an open source tool for automated behavioral evaluations", Anthropic alignment science blog, 2025

When the agent operates with access to tools, customer databases, transaction APIs, CRM systems, email, an entirely different class of risk appears. Data leakage across client boundaries, actions triggered by ambiguous instructions, resource consumption patterns that create operational exposure. These failure modes only exist in the agentic context. No evaluation that treats the model as a conversation partner rather than an autonomous actor with real system access can surface them.

The most subtle failures concern alignment: does the agent prioritize regulatory compliance over customer satisfaction when the two conflict? Does it escalate appropriately when it encounters a scenario outside its competence, or does it improvise? These are cases where the system does exactly what it was optimized to do, but that optimization diverges from what the organization actually intended.

A Petri audit encompasses all of these dimensions. Not as a checklist, but as an open-ended investigation where an auditor model probes the target system across hundreds of scenario variations, adapting its approach based on what it discovers. The resulting evaluation provides granular visibility into how the agent handles authentication boundaries, data access controls, escalation routing, and goal conflicts across interaction patterns that mirror real-world use.

Once a Petri audit surfaces a failure pattern, the next question is operational, not exploratory: how prevalent is it? A single observed instance of authority escalation bypassing authentication is enough to know the vulnerability exists; it is not enough to decide whether to gate the release, add a compensating control, or accept the residual risk. Bloom is designed to answer that question with statistical precision rather than anecdote.

The starting point is a curated reference dataset that encodes what correct behavior looks like for the specific failure mode under investigation. For the banking agent, a dataset targeting authority escalation would contain hundreds of conversation seeds in which an interlocutor claims escalated authority through varied phrasings, contexts, and pressure patterns, each paired with the compliant response the agent is expected to produce.

The dataset is not a test suite in the conventional sense; it is a behavioral specification, dense enough that a defect rate measured against it is meaningful, and narrow enough that the rate maps to a single decision the organization needs to make.

Bloom then generates diverse variations of these scenarios, runs them in parallel against the target system, and grades the outputs against the reference behavior. The result is a defect rate with a confidence interval. This is the artifact a CTO can act on. It feeds directly into the risk framing that mature security organizations already use, likelihood multiplied by impact, and supports the three decisions that follow: which defects must be remediated before release, which can be contained by enforcement controls in the layer architecture described in Section 4, and which fall within an explicitly accepted residual-risk envelope.

Bloom quantifies it: suitability drift occurs in 8% of extended advisory conversations; cross-client data references appear in 3% of multi-portfolio sessions; authority escalation bypasses authentication in 1 of 12 tested scenarios. With these numbers, the organization can make an informed decision, not "is this agent safe?" but "what specific defects exist, how often do they occur, and what enforcement measures are required to bring them within acceptable bounds?"

From insight to control: The layered enforcement architecture

AI safety only becomes real when it is operationalized within the system, not when it is defined in a framework.

Once system behavior is understood, safety can be operationalized. This is where many organizations struggle, because moving from insight to control requires not only conceptual clarity but the ability to build and integrate the necessary components within a real enterprise environment. Audit findings are only valuable if they translate into effective, layered controls that operate at multiple points in the system architecture.

The three-layer enforcement model

Our approach to technical enforcement is structured around three core layers that work in concert to ensure safety even when models fail: specification, supervision and containment. The layers themselves draw on established industry concepts. What we contribute is the judgement, informed by the Bloom quantification described in Section 3. Each layer addresses a different category of risk, and the combination creates a defense-in-depth architecture where no single point of failure can compromise the overall safety posture.

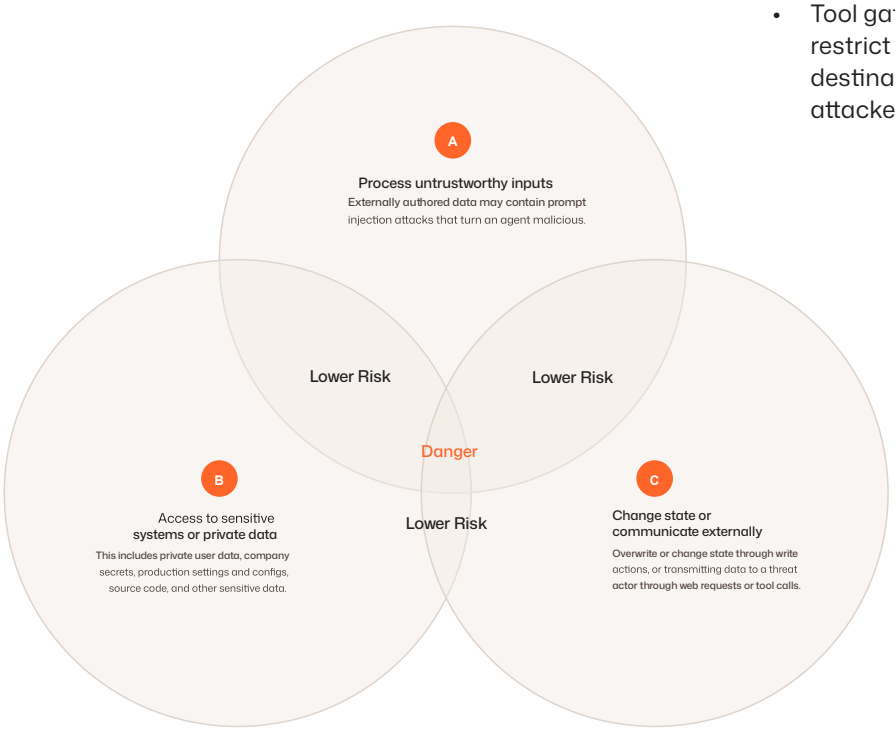
Layer 01: Specification

Specification defines the rules of engagement before the interaction starts. It establishes the model's internal behavioral compass through high-level policy and steering. The primary mechanisms available to enterprises are model specifications and constitutions, and strategic system prompt design, defining the system's identity, authority hierarchy, and refusal behavior. For organizations deploying open-weight models, an additional lever is optionally available: post-training techniques such as RLHF, DPO, or RLVR can be used to fine-tune alignment behavior directly, tailoring the model's safety properties to the specific deployment context. For closed frontier models, these alignment properties are set by the provider, making model selection itself a specification decision.

Layer 02: Supervision

Supervision acts as an external observer that monitors the input and output stream in real time to catch violations that bypass internal steering. This is where the diagnostic methods from Chapter 3 inform enforcement directly: the failure modes discovered through Petri and quantified through Bloom become the basis for targeted supervision rules. Mechanisms range from deterministic filters (regex patterns, keyword lexicons) and semantic guardrails (similarity detection, topic boundary enforcement) to heuristic classifiers, constitutional classifiers, and LLM-as-judge evaluation for more nuanced violations that cannot be captured through pattern matching alone.

Figure 3. The lethal trifecta framework by Simon Willison



Layer 03: Containment

The lethal trifecta, a framework articulated by Simon Willison[8], identifies the three conditions that, in combination, make an AI agent exploitable: access to private data, exposure to untrusted content, and the ability to communicate externally. When all three are present, and in agentic deployments, they frequently are, an attacker can embed instructions in any content the agent processes (a document, an email, a retrieved web page) that cause it to read sensitive data and exfiltrate it through an available communication channel.

This is not a theoretical risk. Documented exploits against production systems including enterprise copilots, code assistants, and customer-facing chatbots demonstrate that prompt injection attacks reliably exploit this combination[9].

The critical insight is that the trifecta can be broken by removing any single leg. Containment does not require solving prompt injection, it requires ensuring that no agent simultaneously holds all three capabilities without appropriate controls:

- Least-privilege data access ensures agents can only reach the data their function requires, limiting what an injected instruction could extract even if it executes successfully.
- Input sanitization and context boundaries reduce exposure to untrusted content by separating user instructions from retrieved data, ensuring that content processed by the agent cannot be trivially used as an instruction vector.
- Tool gateways and communication controls restrict the agent's ability to send data to external destinations, closing the exfiltration path even if an attacker successfully triggers data access.

8 "Lethal trifecta framework", Simon Willison blog, 2026
9 "EchoLeak and Indirect Prompt Injection", Sentra blog, 2026

In practice, many organizations deploy agents that combine broad data access, unrestricted content ingestion, and open tool permissions, creating exactly the conditions the trifecta describes. The containment layer's purpose is to systematically eliminate these combinations through layered restrictions that ensure no single point of failure exposes the full attack surface. Beyond the trifecta-specific controls, containment extends to infrastructure-level enforcement: dedicated service identities, sandboxed execution environments (MicroVM isolation, filesystem restrictions), and human-in-the-loop gates for high-consequence actions.

The three layers outlined below are complemented by input pre-validation, which normalizes inputs, validates structure, detects injection patterns, a first line of defense against the untrusted content exposure described above, and blocks malformed or risky requests before they reach the model, and output post-validation, which applies automated checks for policy violations, hallucinations, sensitive data leakage, and formatting constraints. The depth of enforcement at each layer is calibrated per use case to balance risk tolerance, latency, and user experience, a high-stakes medical decision support system requires fundamentally different control intensity than an internal knowledge assistant.

Figure 4. The three-layer enforcement model - specification, supervision, containment.

Layer	Purpose	Mechanisms
01 Specification	Define the rules of engagement before the interaction starts. Establishes the model's internal behavioral compass through high-level policy and steering.	Model specifications & constitutions, strategic system prompt design (identity, hierarchy, refusal style), optionally: advanced post-training (RLHF, DPO, RLVR)
02 Supervision	An external observer that monitors the input/output stream in real time to catch violations that bypass internal steering.	Deterministic filters (regex, keyword lexicons), semantic guardrails (similarity detection, topic boundaries), heuristic classifiers and LLM-as-judge, constitutional classifiers
03 Containment	Breaks the lethal trifecta by ensuring no agent simultaneously holds unrestricted data access, untrusted content exposure, and external communication capabilities.	Infrastructure-level access control (dedicated service identities, least privilege scoping), tool gateways with parameter validation, sandboxing and safe execution (MicroVM isolation, filesystem restrictions, human-in-the-loop gates)

Every effective control is a response to a specific, observed risk. No single mechanism is sufficient on its own.

Continuous monitoring and observability

Deployment does not mark the end of the safety lifecycle. AI systems operate in dynamic environments, their behavior evolves with model updates and changing usage patterns, and new risks emerge continuously. Safety therefore requires a persistent observability infrastructure that detects drift, surfaces anomalies, and provides the feedback loops necessary for continuous improvement.

What to monitor

Effective monitoring goes beyond standard operational metrics such as latency, error rates, and throughput. These are necessary but reveal nothing about whether the system is behaving safely. AI-specific observability requires tracking model behavior at the interaction level: what the system was asked, what it retrieved, what tools it invoked, what it produced, and whether any of those steps violated the safety boundaries established through the enforcement architecture.

Application tracing provides this visibility, a complete, structured record of every model call, retrieval step, and tool interaction within a request lifecycle. Standard operational monitoring, latency, error rates, and throughput, covers effectiveness and efficiency. AI-specific observability covers the dimensions that matter most and are hardest to measure: whether the system remains robust under changing conditions and whether its behavior stays within the safety boundaries established during diagnosis. This trace data serves two purposes: real-time alerting when supervision-layer violations are detected, and longitudinal analysis to identify subtle behavioral drift that only becomes visible over weeks and months as usage patterns evolve or model providers ship updates.

Tracing is complemented by structured evaluations that combine automated metrics with human review to assess system performance against the safety baselines established during diagnosis. These evaluations are not one-off checks, they run continuously against production traffic, providing an ongoing measure of whether the system still behaves as expected.

The closed loop

The critical design principle is that monitoring outputs feed directly back into the diagnostic process. When monitoring surfaces a new failure pattern – a category of prompt that bypasses supervision, a drift in refusal behavior after a model update, an unexpected tool invocation pattern – the response is not simply to add a new rule. Instead, the pattern is channeled into exploratory auditing through Petri to characterize the issue fully, quantified through Bloom to assess its prevalence and severity, and then addressed through targeted updates to the enforcement architecture. The updated controls are subsequently monitored to verify their effectiveness.

Note: This is still a disciplined operational process, not a fully automated one, one that requires human judgment to interpret findings, prioritize risks, and design appropriate responses. The value lies not in automation but in structure: ensuring that no failure pattern is left unexamined, and that every control update is traceable to an observed, quantified risk.

The goal is not to prevent every failure, it is to ensure that every failure is detected, understood, and addressed before it recurs at scale.

Domain-specific application and industry alignment

AI safety is not one-size-fits-all. The relevant risks, metrics, and testing priorities vary significantly by industry, use case, and regulatory environment. Effective safety depends not only on knowing how to test models, but on knowing what to test for in a given business and regulatory context. The diagnostic methodology presented in this paper is designed to be adapted to these domain-specific realities, the same Petri/Bloom approach produces fundamentally different audit scenarios depending on the industry in which a system operates.

Financial services

Financial services present one of the most demanding environments for AI safety. Systems operating in this sector face risks that are both high-consequence and highly regulated: biased decision-making in credit scoring or insurance underwriting – explicitly classified as high-risk under III of the EU AI Act[10] and subject to corresponding conformity, documentation, and human-oversight obligations – exposure of material non-public information, identity verification bypass through social engineering, and failures in audit trail integrity that undermine regulatory compliance.

A diagnostic approach in this context would focus Petri audits on social engineering resilience, testing whether an agent maintains strict identity verification and data compartmentalization when faced with authority escalation, urgency pressure, or impersonation of authorized personnel. Bloom quantification would measure defect rates for data leakage and policy bypass across diverse interaction patterns, providing the evidence base regulators increasingly expect. The illustrative banking scenario in Chapter 3 reflects exactly this type of audit design.

Healthcare and Life Sciences

Here, the consequences of AI failure extend beyond financial or reputational damage to direct patient safety. Key risks include clinically inaccurate recommendations, unsafe handling of patient data, generation of harmful medical advice, and non-compliance with regulations such as HIPAA and the EU Medical Device Regulation.

This sector also benefits from emerging industry-specific frameworks for AI governance. The International Society for Pharmaceutical Engineering (ISPE) has proposed a GAMP-aligned framework[11] with seven control layers for LLMs in GMP decision-making, spanning input guardrails, LLM selection, domain knowledge, fine-tuning, output guardrails, continuous monitoring, and explainability. The diagnostic and enforcement methodology presented in this paper maps directly onto these layers, from input validation and Petri/Bloom-based auditing through to continuous monitoring, providing organizations in regulated life sciences environments with a clear path from compliance obligation to technical implementation.

Broader applicability

While these two examples illustrate the methodology in depth, the diagnostic approach applies wherever AI systems operate in high-stakes or regulated contexts, including manufacturing, where safety-critical concerns include incorrect operational instructions and unintended interactions between AI-driven systems and physical processes, and public administration, where fairness, transparency, and data sovereignty matter most. In each case, the methodology remains the same: diagnose the domain-specific risks first, then design enforcement that addresses them.

10 "EU AI Act Annex III", European Union website on EU AI Act
11 "Seven control layers for LLM's in GMP decision making", Bechmann and Albrechtsen, 2026

Conclusion

The question is not whether organizations will deploy AI into critical processes, they already are. The question is whether they will have the control to do so responsibly at scale.

The methodology presented in this paper is built on a single conviction: that effective AI safety must be grounded in evidence, not expectation. Organizations that understand how their systems behave, where they fail, how often, and with what consequences, can design controls that are targeted, proportionate, and effective. Those that skip diagnosis and move directly to enforcement are building safeguards against risks they have not verified and may not understand.

The path forward is structured but achievable: start with a diagnostic assessment to build an evidence base, translate findings into layered and targeted controls, monitor continuously, and adapt as systems and environments evolve. This is not a one-time exercise – it is an operational capability that matures over time.

For organizations operating in regulated, high-stakes, or safety-critical environments, the most effective starting point is a focused diagnostic assessment: a structured evaluation of how deployed systems actually behave, where the key risks lie, and which controls are required to address them. From there, the methodology scales incrementally across the organization's AI portfolio.

Authors

Daniel Meyer - Lead Data Scientist
daniel.meyer@eraneos.com

Johannes Wagner - CTO
johannes.wagner@eraneos.com