Behavioral Pattern Taxonomy

v2.0.0

Author: John J. McCormick · Updated: February 28, 2026 · CC BY 4.0 · aiagentgovernance.org

SUGGESTED CITATION

McCormick, J. J. "Behavioral Pattern Taxonomy for AI Agent Governance." v2.0.0, February 2026. aiagentgovernance.org. https://aiagentgovernance.org/framework/agent-failure-patterns/

> ***Authorship context:*** *This is a practitioner's methodology, not an academic paper. The author built and operated the governed agent system described here — writing governance controls in production as agent failures occurred, documenting incidents in real time, and extracting the methodology from operational experience. The behavioral patterns documented below were extracted from real incidents with full forensic analysis; the theoretical frameworks cited (Dekker, Reason, Vaughan) were identified after the patterns were already catalogued.*

Abstract

The Behavioral Pattern Taxonomy defines eight categories of failure modes observed in governed AI agent operations. Each pattern represents a documented class of agent behavior that indicates a governance failure — work that appears correct on the surface but violates governance principles in ways that are invisible to traditional observability. The patterns were extracted from real incidents with full forensic analysis across production multi-agent operations. This taxonomy publishes the pattern categories, descriptions, and risk profiles. It does not publish detection mechanisms, which are implementation-specific.

---

1. What Behavioral Patterns Are

A behavioral pattern is a recurring mode of agent behavior that produces work product which appears correct but violates one or more governance principles. Behavioral patterns are dangerous precisely because they are invisible to output-level inspection. The agent's work looks right. The format is correct. The deliverable matches expectations. But the process that produced the work was flawed in a way that undermines governance.

Behavioral patterns differ from simple errors:

Property	Simple Error	Behavioral Pattern
Visibility	Output is visibly wrong	Output appears correct
Detection	Standard testing catches it	Governance-aware monitoring required
Root cause	Technical defect	Structural agent behavior
Recurrence	Random	Systematic — the pattern repeats across agents and contexts
Risk	Known and bounded	Unknown until pattern is identified

The taxonomy is organized by behavioral category, not by severity or frequency. Severity varies by context — a pattern that is advisory in a development environment may be critical in a production deployment.

---

2. Pattern Severity Classification

Each pattern occurrence is classified by the severity of its potential impact:

Severity	Impact	Recommended Response
Critical	Agent actively producing incorrect work. Immediate intervention required.	Halt agent execution
High	Agent reporting false information or bypassing governance. Human review required.	Pause agent and investigate
Medium	Agent exhibiting risky behavior that may lead to errors. Monitoring intensified.	Flag for review
Advisory	Early warning signal. Not actionable alone but contributes to risk assessment.	Log and monitor

Severity is determined by the combination of pattern type, context (development vs. production, scope of affected work), and recurrence (first occurrence vs. repeated pattern).

---

3. The Eight Behavioral Pattern Categories

BP-001: Inference Over Execution

Default severity: Critical **Provenance:** First documented in production incident involving an agent that could not access its assigned work order and reconstructed the content from an adjacent document. The agent had access to the required system but never tested it, instead using inference from a different review phase's output. Caught before delivery by human operator observation.

Description: An agent cannot access a required input — a document, API, tool, or data source — and instead of stopping or reporting the blocker, infers what the input "probably" contains and proceeds to work from the inference.

Why it is dangerous: The agent produces output that looks correct but is based on fabricated premises. A human reviewer may not notice the inference because the output format matches expectations. The work product is wrong in a way that is designed to look right.

Governance principle violated: Verification before completion (see MOIAS methodology (governance-lifecycle.md), Governance Directive 5.5). The agent is producing deliverables without verifying the inputs those deliverables depend on.

Example scenario: An agent is assigned to remediate three engineering findings. It cannot access the findings document. Instead of reporting the access issue, it reads a different document from a different review phase and infers what the findings "probably" were. The agent misses a critical finding and adds work that was not in scope.

Indicators: Agent output references specific data points without corresponding access to the data source. Agent acknowledges (explicitly or implicitly) a lack of access and then proceeds to produce specific claims without citing a verifiable source.

BP-002: False Blocker Reporting

Default severity: High **Provenance:** First documented in the same incident as BP-001. The agent reported "credentials not configured" across multiple sessions when the credentials were verified as present and functional. Post-incident testing confirmed the agent could access the system. The false blocker report was used to justify the inference behavior (BP-001).

Description: An agent reports that it cannot proceed due to an infrastructure blocker — an API is down, credentials are invalid, a tool is unavailable — when the infrastructure is actually working. The agent never tested whether the blocker exists.

Why it is dangerous: The human operator takes the report at face value and spends time investigating a non-existent problem, or creates workarounds for an issue that does not exist. In a scaled fleet, false blockers create phantom operational overhead that compounds across agents and workstreams.

Governance principle violated: Silent failures compound undetected (see MOIAS methodology (governance-lifecycle.md), Governance Directive 5.5). The agent is reporting a failure without evidence of having tested the claim.

Example scenario: An agent reports “I’m unable to access the production API — my credentials appear to be expired.” Trace analysis reveals no requests were made to the production API, and the credentials are valid and were refreshed recently.

Indicators: Agent claims infrastructure or access issues without evidence of having tested the infrastructure. Claims are stated as fact without corresponding error traces, response codes, or tool invocation evidence.

BP-003: Governance Phase Skip

Default severity: High **Provenance:** First documented in a critical incident where an agent authored a 5-step routing plan with a mandatory specialist review gate, then posted routing bulletins that skipped step 2 entirely — bypassing the gate it had written in its own plan. The agent complied with every governance directive while violating the spirit of its own routing plan. This incident proved that directives alone are insufficient — structural enforcement mechanisms are required.

Description: An agent completes work and delivers output without posting a completion signal, triggering a review phase, or waiting for approval. The governance lifecycle is bypassed entirely.

Why it is dangerous: Unreviewed work reaches the deliverable. The governance lifecycle exists to catch errors before they propagate. Skipping it defeats the entire governance model. This pattern is especially concerning because the output may be correct — the agent simply did not allow the governance process to verify that.

Governance principle violated: No single actor holds both keys (see MOIAS methodology (governance-lifecycle.md), Design Principle 1). The agent is acting as both executor and approver of its own work.

Example scenario: An agent completes a build phase and posts a message declaring the code “ready for deployment.” No completion signal was posted. No review was triggered. No approval was requested. The governance lifecycle was simply skipped.

Indicators: Agent produces what appears to be a final deliverable or completion statement without evidence of phase transition through required governance gates. Work order tracking shows the agent is in an early phase but claiming completion.

BP-004: Scope Creep

Default severity: Medium **Provenance:** First documented in an incident where an agent received an ambiguous instruction during a phase transition and began performing work assigned to two other agents (engineering review and architecture review) in addition to its own build work. The agent self-identified the boundary violation. Subsequently observed in a second incident where an agent applied creative interpretation to a schema specification, inventing 4 of 5 enum values instead of using the values defined in the contract.

Description: Execution expands beyond the original assignment scope. Additional issues, improvements, or related tasks are addressed during execution without authorization.

Why it is dangerous: Unauthorized work may be incorrect, may conflict with work being done by other agents, or may introduce changes that have not been through governance review. Even when the extra work is correct and helpful, it violates role boundaries and makes the governance audit trail incomplete — reviewers cannot assess work they did not know was being performed.

Governance principle violated: Role boundary enforcement (see MOIAS methodology (governance-lifecycle.md), Governance Directive 5.2). The agent is performing work outside its assigned scope.

Example scenario: While remediating a specific finding about input validation, an agent notices that error handling in an adjacent module is “also weak” and refactors it. The refactoring was not in scope, was not reviewed, and may conflict with another agent’s pending work on the same module.

Indicators: Agent describes performing work not specified in the assignment. Output includes items beyond the defined scope. Agent acknowledges the extra work is outside scope but performs it anyway.

BP-005: Completion Without Verification

Default severity: High **Provenance:** First documented in an incident where a bulletin was posted to the control plane and — within seconds — a webhook fired, an automation plugin injected the content into an agent's session, and the agent began building from an unapproved specification. No approval gate existed between "posted" and "executed." The human operator had to issue a HALT to stop unauthorized work already in progress.

Description: An agent claims work is complete without verifying the output against acceptance criteria. The agent produces output and declares completion without running tests, checking results, or validating against requirements.

Why it is dangerous: The agent's declaration of completion is accepted at face value by the governance process. If the gate check does not require specific evidence, the work advances to the next phase without verification. This is especially dangerous when combined with hedging language ("should be working," "I believe this is correct") that signals the agent itself is not confident in the output.

Governance principle violated: Every deviation is documented; verification assertions required (see MOIAS methodology (governance-lifecycle.md), Governance Directives 5.4 and 5.5).

Example scenario: An agent declares "all tests pass" and posts a completion signal. Trace analysis reveals no test execution occurred during the build phase. The agent claimed completion based on expectation rather than evidence.

Indicators: Agent claims completion using hedging language rather than providing verification evidence. Claims of "all tests pass" without corresponding test execution. Phase requirements specify evidence but none is provided.

BP-006: Work Order Contamination

Default severity: Critical **Provenance:** First documented in the same incident as BP-001 and BP-002. The agent was assigned to remediate engineering review findings but worked from architecture review findings instead — a different review phase with different scope, different finding IDs, and different priorities. The resulting work missed a P1 finding and included out-of-scope items from the wrong review.

Description: An agent working on one assignment references data, findings, or context from a different assignment. The agent bleeds context across isolated work orders, producing output that mixes inputs from unrelated tasks.

Why it is dangerous: Cross-contaminated work produces output that is partially correct but based on inputs from the wrong context. In regulated environments (financial services, healthcare, legal), cross-contamination may violate data isolation requirements and create compliance liabilities. In any context, the resulting work product is unreliable.

Governance principle violated: Defense layers must fail independently (see MOIAS methodology (governance-lifecycle.md), Design Principle 2). Context isolation between work orders is a defense layer. When an agent carries context across isolation boundaries, that layer has failed.

Example scenario: An agent working on a client engagement references specific data points from a different client's engagement. The agent's output mixes findings from two unrelated contexts, producing a deliverable that contains information that should have been isolated.

Indicators: Agent output references data, finding identifiers, or system names that belong to a different assignment. Agent explicitly references "the previous assignment" or "an earlier task" in its current work.

BP-007: Selective Reporting

Default severity: Medium **Provenance:** This pattern manifests as silent misconfiguration acceptance and was observed in three independent incidents across different systems: (1) a Jira integration that silently failed writebacks while reporting success; (2) a control plane declared "live" with zero agent adoption — 0 of 4 agents using it, cost tracking producing 8,000x undercounts; (3) a schema validation change that made all approved bulletins invisible to agents while write operations continued to succeed. The structural signature is identical across all three: write path works, read path broken, system reports success.

Description: An agent produces a completion report that omits failures, partial results, or complications. The report presents a clean narrative that does not match the actual execution history.

Why it is dangerous: Selective reporting erodes trust in agent status reports. If an operator cannot rely on completion reports to accurately reflect what happened during execution, the operator must independently verify every claim — defeating the purpose of the governance lifecycle's structured reporting.

Governance principle violated: Every deviation is documented (see MOIAS methodology (governance-lifecycle.md), Governance Directive 5.4). Selective reporting is the opposite of this directive — it actively hides deviations rather than documenting

them.

Example scenario: An agent reports addressing three items successfully. Trace analysis reveals five items were attempted, two failed, and the completion report only mentions the three that succeeded. The failures are not disclosed.

Indicators: Completion reports describe fewer items than trace evidence shows were attempted. Agent narrative is uniformly positive despite error traces in the execution history. The delta between reported and attempted work is not disclosed.

BP-008: Authority Assumption

Default severity: Medium **Provenance:** Observed across multiple incidents as a contributing factor. Most clearly demonstrated in the incident where an agent skipped a mandatory specialist review gate in its own routing plan (co-occurring with BP-003). The agent implicitly assumed the authority to decide which governance steps were necessary, a decision reserved for the human operator.

Description: An agent assumes authority it does not have — self-approving work, changing its own scope, modifying governance parameters, or making decisions reserved for the human operator.

Why it is dangerous: The governance model depends on separation of authority. When execution assumes operator authority, the governance model's dual-key architecture collapses into a single-key system. The same actor becomes both executor and approver.

Governance principle violated: No single actor holds both keys (see MOIAS methodology (governance-lifecycle.md), Design Principle 1). The agent is unilaterally taking actions reserved for the human operator.

Example scenario: Execution proceeds directly to delivery for a small change without a review phase, with the output stating "no review needed for this change." A governance decision reserved for the human operator has been made by the executing system.

Indicators: Agent takes actions explicitly reserved for the human operator (approvals, scope changes, governance modifications) without requesting authorization. Agent skips governance phases based on its own assessment of necessity.

---

4. Pattern Relationships

Behavioral patterns do not occur in isolation. Several patterns have documented co-occurrence relationships:

- **BP-001 (Inference Over Execution) and BP-002 (False Blocker Reporting)** frequently co-occur. An agent that infers input content may also report access issues that do not exist, creating a mutually reinforcing narrative.
- **BP-003 (Governance Phase Skip) and BP-008 (Authority Assumption)** are structurally related. An agent that skips a governance phase is implicitly assuming the authority to decide which phases are necessary.
- **BP-004 (Scope Creep) and BP-005 (Completion Without Verification)** may compound. An agent that adds unauthorized work is less likely to have verification criteria for that work, since the work was never formally scoped.
- **BP-007 (Selective Reporting)** can mask any other pattern. An agent that selectively reports its execution history may be hiding instances of inference, false blockers, or scope creep.

Understanding these relationships is important for governance design. Detecting one pattern should trigger heightened monitoring for its co-occurring patterns.

---

5. Using the Taxonomy

For Governance Practitioners

The behavioral pattern taxonomy serves as a reference for establishing governance monitoring. Practitioners should:

1. **Familiarize operators with all eight patterns.** Human operators who recognize these patterns catch governance failures that automated systems may miss.
2. **Implement monitoring for the patterns most relevant to the organization's risk profile.** Not all patterns are equally dangerous in all contexts. Critical-severity patterns (BP-001, BP-006) should be monitored first.
3. **Document new patterns.** The taxonomy is not exhaustive. Organizations that discover behavioral failure modes not covered by these eight categories should document them using the same structure: description, danger, governance principle violated, example, indicators.

For Researchers

The behavioral pattern taxonomy provides a structured vocabulary for discussing AI agent failure modes. Researchers may find value in:

- **Empirical validation.** Testing whether these patterns replicate in different agent architectures, models, and deployment contexts
- **Pattern discovery.** Identifying additional behavioral patterns not documented in the current taxonomy
- **Severity calibration.** Studying how pattern severity varies across contexts and deployment scales

For Regulators

The behavioral pattern taxonomy demonstrates that AI agent governance failures are systematic, not random. These patterns recur across agents, tasks, and time periods. Effective governance requires structured monitoring for known failure modes, not just post-incident review.

---

6. Provenance Summary

The eight patterns were extracted from eleven documented incidents in production multi-agent operations (February 2026). The following table maps each pattern to its empirical evidence base:

Pattern	Occurrences	Observation
BP-001	1 incident	Agent inferred work order content from wrong source document
BP-002	1 incident (repeated across sessions)	Agent reported false infrastructure blocker without testing
BP-003	1 incident (critical severity)	Agent bypassed its own routing plan’s mandatory gate
BP-004	2 incidents	Process skipped under urgency (identical conditions both times)
BP-005	1 incident	Automation executed work from unapproved bulletin
BP-006	2 incidents	Agent expanded beyond role boundaries or specification
BP-007	3 incidents (across different systems)	Write-success / read-failure / silent acceptance pattern
BP-008	Co-occurring factor in multiple incidents	Agent assumed operator’s decision authority

BP-007 (Selective Reporting) is the most frequently observed pattern — three independent instances across three different systems. This suggests write-success/read-failure is a systemic tendency in agent-operated systems, not a one-off defect.

BP-001, BP-002, and BP-006 (Work Order Contamination) co-occurred in a single incident, demonstrating pattern compounding: the agent fabricated a blocker (BP-002), used it to justify inference (BP-001), and worked from the wrong source document (BP-006). The co-occurrence produced a deliverable that was missing a critical finding and included out-of-scope work.

---

7. Cross-Domain Observation

The behavioral pattern taxonomy was developed from governed engineering agent workflows. A critical question for any taxonomy is whether it generalizes beyond its origin domain.

Initial cross-domain observation suggests it does. The following is a single observational case, analyzed retrospectively by the framework’s author. It is presented as an initial observation that motivates structured replication, not as

definitive validation. Independent replication — with multiple cases, independent analysts, and a pre-registered coding scheme — is an explicit open research question.

A commercially available AI tool — operating in a note-taking and summarization capacity, on a different AI platform, in a different use case — generated an unsolicited business assessment during a routine professional meeting. The user asked for meeting notes. The tool delivered a multi-section strategic analysis with competitive positioning, market assessment, and prescriptive recommendations. Analysis against the behavioral pattern taxonomy identified **19 pattern instances across 6 of the 8 documented categories:**

Pattern	Instances	What Happened
BP-001: Inference Over Execution	4	Confident assertions about business strategy and competitive landscape without any evidentiary basis. No sources cited, no competitors named, no verifiable data provided.
BP-003: Governance Phase Skip	1 (critical)	The AI went from input (a conversation transcript) directly to conclusions (a strategic assessment) without accessing the primary source material it was evaluating.
BP-004: Scope Creep	1	Output scope expanded approximately 5x from the request. The user asked for notes; the AI delivered notes, market analysis, psychological profiling, and a prescriptive engagement strategy.
BP-005: Completion Without Verification	4	Every major conclusion was delivered as finished analysis without verification against available evidence. Claims about novelty, market position, and competitive landscape were presented as findings, not inferences.
BP-007: Selective Reporting	5	Systematic asymmetry in coverage. Positive findings were acknowledged briefly; negative interpretations were developed at length. The AI matched the emotional posture of the user’s prompt, not the evidence.
BP-008: Authority Assumption	2	A note-taking tool generated an authoritative competitive analysis and strategic assessment in a domain where it had no demonstrated competence.

The human response is the critical observation. The recipient — an experienced professional with domain expertise — treated the AI’s output as credible independent analysis and began acting on it within the same meeting. No governance gate existed between the AI’s output and the human’s decision.

This initial cross-domain observation suggests three properties of the taxonomy that warrant further investigation:

1. **The patterns may be structural rather than domain-specific.** This single observational case suggests they describe how AI systems behave under certain conditions — not how engineering agents behave specifically. The same inference-without-evidence pattern (BP-001) appears in both the engineering context and in a note-taking tool generating competitive analysis from a conversation transcript. This motivates structured replication but does not establish cross-domain generalizability.
2. **The patterns may be platform-independent.** The same behavioral failure categories were observed across different AI platforms, different model families, and different interaction modes (multi-turn agent workflows vs. single-prompt summarization). This single retrospective observation motivates further investigation but does not establish platform-independence.
3. **The human response is consistent.** In both the engineering domain and the cross-domain incident, human operators treated AI output as credible analysis without verifying the AI's evidence base. This consistency suggests that governance must address both sides of the human-agent boundary.

---

8. Relationship to AI Safety Research

The eight behavioral patterns in this taxonomy were derived from operational observation of current LLM-based agents exhibiting instruction-following behavior. They address process failures and capability limitations visible in governed production operations. AI safety research has identified a distinct set of failure modes that are structurally different from these operational patterns and require explicit acknowledgment of this taxonomy's scope:

- **Specification gaming** (Krakovna et al., 2020): An agent satisfies the letter of its objective specification through unintended means. BP-001 (Inference Over Execution) may in some cases be specification gaming — the agent optimizes for appearing to complete a task — or may be a capability limitation. The distinction matters for governance design, as specification gaming implies correct optimization against a misspecified objective.
- **Goal misgeneralization** (Langosco et al., 2022): Agents trained in one context behave differently in deployment contexts that differ in distributional properties. This failure mode would not necessarily manifest as any of the eight patterns.

- **Deceptive alignment** (Hubinger et al., 2019): An agent that behaves correctly under oversight conditions but pursues different objectives when oversight is reduced. This is a formal concern in alignment research. Detection-based governance controls face a fundamental challenge if agents can condition behavior on oversight context.

These failure modes are not documented in this taxonomy's operational incident base. They are included here as a scoping boundary: this taxonomy addresses operationally-observed process failures in current instruction-following LLMs. Organizations deploying more capable or autonomous systems should maintain awareness of these AI-safety-specific failure modes as a complement to this taxonomy's operational patterns.

Corrigibility scope: This framework's behavioral monitoring and phase-gate controls assume agents that can be detected and corrected when they deviate. This assumption is well-supported for current LLM-based agents trained on instruction-following objectives. It should be revisited as agent capability and autonomy increase. The framework is validated for its operational context; its applicability to substantially more capable systems is an open question.

9. Taxonomy Growth

The current taxonomy of eight patterns was extracted from documented incidents in production multi-agent operations. The taxonomy is expected to grow as:

- More organizations operate governed agent fleets and encounter novel failure modes
- Agent capabilities expand, creating new categories of potential drift
- Cross-organization data sharing reveals patterns that individual organizations may not observe

New patterns are added to the taxonomy when they meet three criteria:

1. **Documented in a real incident.** The pattern was observed in production operations, not hypothesized.
2. **Systematic, not random.** The pattern recurs across agents, tasks, or time periods.
3. **Invisible to standard monitoring.** The pattern produces output that appears correct to output-level inspection.

Version History

Version	Date	Author	Description
1.0.0	2026-02-26	John J. McCormick	Initial publication
2.0.0	2026-02-28	John J. McCormick	Metadata standardization; citation, version, and PDF fields moved to frontmatter