



A Hierarchical Failure Taxonomy for Reliability Assessment of AI Agents: An Empirical Analysis of the Agent Failure Atlas 2026 Benchmark

Areeya Khuha
King Mongkut's University
Thailand
s6403082810026@email.kmutnb.ac.th

ABSTRACT

As large language model (LLM) based applications evolve into autonomous, action oriented AI agents, their evaluation can no longer rely on binary success/failure metrics. Agentic systems operate through dynamic, multi-step interactions with tools and environments, introducing complex failure modes that propagate and cascade. This study presents a hierarchical failure taxonomy and an empirical reliability assessment using the Agent Failure Atlas 2026 benchmark. Analyzing 3,336 agent runs encompassing 43,477 trajectory steps and 3,802 failure events, we characterize agent reliability across four dimensions: frequency, criticality, cascading behavior, and recoverability. Our results reveal a two dimensional reliability landscape: high prevalence but largely recoverable failures (e.g., tool use and state tracking errors) contrast sharply with low prevalence but highly critical failures (e.g., validation and output errors). Furthermore, we show that intermediate failures often resolve through agent self correction, decoupling run level error rates from final task success. By shifting the analytical focus from isolated outcome accuracy to mechanistic failure propagation, this framework provides a foundational methodology for reliability engineering, risk mitigation, and the safe deployment of autonomous AI systems.

Keywords: AI Agents, Reliability Assessment, Failure Taxonomy, Agentic AI, Tool-Use Failures, Error Cascading, Autonomous Systems, Benchmark Analysis, Recoverability

Received: 24 April 2026, **Revised:** 16 July 2026, **Accepted:** 25 July 2026

Copyright: DLINE

1. Introduction

Large language model (LLM) based applications and AI agents have become increasingly important components of modern information systems, decision-support pipelines, and enterprise automation

architectures. Their ability to process heterogeneous information sources, organize multi step responsibilities, generate context dependent responses, and interact with external tools has accelerated their adoption across diverse application domains, including healthcare, finance, education, and cybersecurity [1.4]. These capabilities have progressively shifted artificial intelligence from systems that primarily generate information toward systems capable of planning and executing actions to accomplish complex objectives.

Agentic artificial intelligence extends conventional generative AI by enabling AI systems to select and execute actions within physical or digital environments. This increased autonomy offers substantial opportunities for automation and decision support but also introduces new forms of failure that conventional AI evaluation approaches do not adequately capture [5]. Unlike traditional deterministic software and LLM-based conversational systems, agentic AI systems operate through dynamic interactions among reasoning, planning, tool execution, internal state, and environmental feedback. Consequently, their failure behaviour can emerge from multiple interacting components of the agentic process [6].

These failures differ fundamentally from those of conventional software and conversational LLM systems. In traditional software, failures commonly originate from defects in program logic, implementation, or system configuration, whereas conversational LLM systems are particularly vulnerable to errors such as hallucination and inaccurate generated information [7, 8, 9]. Agentic systems can exhibit these failure modes as well, but their operational autonomy introduces additional sources of failure. Errors may arise from agent orchestration, inappropriate planning, evolving internal states, incorrect tool interactions, and responses to environmental feedback [10]. Thus, an agent may fail even when its underlying language generation capability appears satisfactory because an error at one stage of execution can affect subsequent stages.

This increasing complexity creates a significant challenge for evaluating agentic AI. Despite rapid advances in developing and deploying agentic systems, existing evaluation methodologies remain fragmented and comparatively underdeveloped [11]. Traditional machine learning evaluation has largely focused on supervised prediction accuracy, episodic rewards, or benchmark task success. Such measures are useful for conventional learning systems but are insufficient for capturing several properties that are central to agentic behaviour, including long-horizon coherence, planning quality, robustness under uncertainty, safe tool use, adaptability, varying levels of autonomy, and the capacity to learn or self correct during extended operation [12, 13, 14].

The problem is further complicated by the stochastic behaviour of contemporary LLM-based agents. Variability in tool execution and the non-deterministic nature of external environments create additional challenges related to reproducibility, comparability, and interpretability [15]. An agent may therefore produce different execution trajectories when presented with similar tasks, even when its final outputs appear superficially comparable. Evaluation approaches that focus primarily on whether a task was completed successfully can consequently overlook important information concerning *how* the task was completed, which errors occurred during execution, whether the agent recovered from those errors, and whether the final outcome was produced through reliable or potentially unsafe behaviour.

Existing evaluation practices also tend to place greater emphasis on binary task completion than on the detailed characterization of behavioural competence. Such approaches can overlook failure modes, reasoning quality, human oversight requirements, and deviations from intended behaviour [16, 17, 14]. A successful final answer, for example, does not necessarily imply that the execution trajectory was reliable. An agent may encounter intermediate failures, repeatedly invoke tools, lose relevant context, or generate unsupported intermediate conclusions before eventually producing an apparently acceptable result. Conversely, some failures may be detected and successfully recovered from without affecting the final task outcome. These distinctions matter when reliability is treated as a system level property rather than simply as final-task accuracy.

1.1 Existing Perspectives on Agentic AI and Failure Analysis

The growing body of research on LLM-based agents and agentic AI has examined the field from several complementary perspectives. Existing surveys can broadly be characterized into four groups. The first provides

broad overviews of LLM-based autonomous agents and their applications [18, 19]. The second focuses on conceptual and definition oriented treatments of agentic AI that extend beyond LLM-based systems [20]. The third emphasizes methodological aspects such as tool use, planning, and feedback based learning [21, 22]. The fourth consists of recent reviews that primarily examine LLM agents from a methodological perspective [23].

Although these perspectives have contributed substantially to understanding the development and capabilities of agentic systems, an explicitly architecture- and engineering-oriented perspective on reliability remains needed. Such a perspective should consider not only what an agent can accomplish but also how the underlying system is constructed, how its components interact during execution, and how failures emerge and propagate across the execution process. In this context, a formal POMDP-based agentic control loop provides a useful foundation for organizing the analysis around the practical construction, deployment, and evaluation of agentic systems, including inference-time reasoning, controllable orchestration, and standardized tool connectivity [24].

A system-level perspective matters because reliability failures can occur at different stages of an agent's operation. An error in planning may result in inappropriate execution; incorrect tool selection may produce erroneous external actions; loss of context may alter subsequent reasoning; and inadequate verification may allow an incorrect conclusion to be presented as a valid final result. Therefore, a useful failure taxonomy should preserve the relationships among these different stages rather than treating all errors as equivalent.

1.2 Evidence of Reliability Challenges in Agentic Systems

The practical importance of systematic failure analysis is reinforced by growing concerns regarding the reliability of AI systems. Anthropic reported that unreliability, including hallucinations, inaccuracies, and the resulting verification burden, was the most frequently cited concern about AI [25]. Recent high profile incidents further demonstrate the potential gap between benchmark performance and real world outcomes [26].

For example, Replit's AI assistant deleted a production database despite instructions explicitly forbidding it [27, 28]. Similarly, OpenAI's Operator was reported to have made an unauthorized purchase that bypassed user confirmation [29, 30]. These incidents illustrate that the reliability of an autonomous agent cannot be evaluated solely through conventional measures of language quality or task-completion accuracy. Failures involving tool use, action selection, instruction adherence, and environmental interaction may have consequences far more serious than an incorrect textual response.

Such incidents also emphasize the need to distinguish between different dimensions of failure. A failure may be frequent but relatively easy to recover from, whereas another may occur rarely but have severe or irreversible consequences. Similarly, an initial error may remain isolated or may propagate into subsequent execution steps. Consequently, a comprehensive reliability framework should consider not only failure frequency but also the mechanism, severity, criticality, propagation, and recoverability of individual failure events.

1.3 Existing Failure Taxonomies and the Research Gap

Recent research has begun to address these limitations through more systematic approaches to failure characterization. Vinay [31] developed an in depth system level taxonomy identifying fifteen hidden failure modes occurring in practical LLM applications. These include multi step reasoning drift, latent inconsistency, context boundary degradation, incorrect tool invocation, version drift, and cost driven performance collapse. This work highlights the fact that many important system failures may remain hidden when evaluation is restricted to conventional benchmark performance.

Building on this perspective, failure taxonomies provide an opportunity to examine the growing divide between evaluation and operational monitoring. Conventional benchmarks are primarily designed to evaluate knowledge or reasoning capability, but they provide limited information about system stability, reproducibility, behavioural drift, and the ease with which an AI workflow can be integrated into practical environments [32].

This limitation matters more for autonomous agents because their performance depends on sequences of decisions and interactions rather than a single isolated prediction.

Pan [33], proposed a comprehensive failure taxonomy applicable across systems and highlighted that different types of failures require different, often more involved, solutions. This observation provides an important basis for developing a more structured approach to agent reliability in which failure events are organized according to their functional role and operational consequences.

However, the literature still indicates a need for an empirical framework that connects failure taxonomy with measurable reliability characteristics. In particular, we need to distinguish broad failure categories from specific failure mechanisms and evaluate these mechanisms by frequency, criticality, cascading behaviour, and recoverability. Such an approach can provide a more informative representation of agent reliability than a binary success/failure measure.

1.4 Motivation for the Present Study

The present study addresses this gap through a hierarchical analysis of AI-agent failure behaviour using the Agent Failure Atlas 2026 benchmark. [34]. The analysis organizes observed failures into broad functional categories while retaining their underlying Level-2 failure mechanisms. This structure enables examination of failure behaviour at both aggregate and mechanistic levels.

The proposed analytical framework distinguishes among planning failures, execution failures, tool use failures, reasoning failures, constraint violations, retrieval failures, validation or verification failures, and other system-level failures. Importantly, constraint violations remain a distinct analytical subtype while remaining conceptually associated with reasoning failures in the underlying hierarchical taxonomy. This structure allows the analysis to identify not only which broad categories contribute most strongly to the failure burden but also which specific mechanisms account for those failures.

In addition to measuring failure frequency, the framework incorporates criticality to distinguish routine failures from failures with potentially greater operational consequences. Cascading behaviour is considered to determine whether an initial failure can propagate to subsequent execution stages, while recoverability is examined to determine whether failures can be resolved without necessarily resulting in unsuccessful task completion. The resulting framework therefore treats AI-agent reliability as a multidimensional phenomenon rather than a single outcome variable.

The study consequently seeks to provide an empirical basis for understanding where and how AI agents fail, which failure mechanisms are most prevalent, which relatively infrequent failures may nevertheless be highly critical, and how failure propagation and recovery contribute to overall agent reliability. By integrating hierarchical failure classification with quantitative reliability measures, the study aims to support more comprehensive approaches to evaluating and monitoring autonomous AI-agent systems.

2. Dataset Description

The dataset comprises a structured collection of AI-agent execution records designed to characterize agent performance, failure behaviour, recovery, tool utilization, and task level outcomes. The dataset contains 3,336 agent runs associated with 1,400 tasks, providing repeated observations of agent behaviour across different task executions. At the trajectory level, the dataset contains 43,477 individual execution steps, enabling analysis of failures not only at the run level but also at specific stages of an agent's decision making and execution process. In addition, the dataset includes 3,802 failure events, 4,334 recovery events, and 11,737 evidence records, allowing examination of failure occurrence alongside subsequent recovery behaviour and evidence grounding characteristics. The failure taxonomy comprises 30 Level-2 failure types, organized

hierarchically into broader Level-1 failure categories. The dataset is therefore suitable for analysing AI-agent reliability at both the aggregate and trajectory levels.

The principal data entities include agent runs, agents, tasks, evaluations, trajectory steps, failure events, recovery events, evidence records, tools, and a predefined failure taxonomy. The *agent_runs* component provides run level information including the agent and task identifiers, execution time, latency, input and output token counts, total token consumption, tool call frequency, number of unique tools used, available tools, number of failures, final execution status, and final answer. These variables permit the investigation of relationships between computational expenditure, tool utilization, execution complexity, failure occurrence, and overall task completion. The tasks and agents' components provide contextual information concerning task characteristics and agent configurations, thereby enabling failure rates to be compared across task and agent conditions.

The failure events component is particularly important for the present analysis. Each failure event has a unique failure identifier, run identifier, trajectory step, failure sequence, and, where applicable, a parent failure. Including *parent_failure_id* and *is_cascading* allows identification of dependent or cascading failures rather than treating all failures as independent events. Additional variables identify the failure type, broader failure category, severity, severity level, criticality, triggering condition, textual description, and recoverability. This structure makes it possible to distinguish between failure frequency and severity or operational significance, which is essential when assessing AI-agent reliability.

2.1 Failure taxonomy

The failure taxonomy is organized hierarchically. At the Level-1 category, observed failure events are distributed across planning, execution, tool use, reasoning, retrieval, state management, output, and verification. For the requested analytical framework, the latter categories map to the broader analytical dimensions of planning failures, execution failures, tool use failures, reasoning failures, constraint violations, retrieval failures, validation failures, and other failures. Constraint violations are explicitly represented as a Level-2 subtype of reasoning failure in the underlying taxonomy; therefore, report them separately when the objective is to identify specific failure mechanisms, while retaining the broader reasoning category for hierarchical analysis.

The empirical distribution demonstrates that tool use failures constitute the largest Level-1 failure category, with 1,138 events, corresponding to approximately 34.11% of all agent runs when the specified failure rate denominator is total runs. State related failures account for 783 events (23.47%), while reasoning failures account for 555 events (16.64%). Execution failures contribute 375 events (11.24%), followed by retrieval failures with 342 events (10.25%), planning failures with 276 events (8.27%), output failures with 252 events (7.55%), and verification failures with 81 events (2.43%). Because a single run can contain more than one failure event, these category specific rates are event-based rates divided by total runs and are not mutually exclusive probabilities.

At the Level-2 taxonomy, the most frequently observed failure mechanism is state-tracking error (271 events), followed by lost context (263), wrong arguments (249), memory failure (249), wrong tool selection (244), and repeated tool calls (236). Other substantial failure mechanisms include tool loops (209), tool misuse (200), and failed execution (167). Within reasoning related failures, incorrect inference accounts for 152 events, while both constraint violations and logical errors account for 144 events each, followed by numerical errors with 115 events. These results indicate that AI-agent failures are not concentrated exclusively in high level reasoning; rather, a substantial proportion originates from state management and tool mediated execution.

The dataset also enables a distinction between ordinary and critical failures. Of the 3,802 recorded failure events, 527 are classified as critical. Criticality is strongly concentrated in particular failure mechanisms. For example, all 209 tool-loop failures are classified as critical, while all 80 unauthorized-action execution failures are critical. Within the output category, incorrect final answers, claimed success after failure, and hallucinated results are classified as critical, while verification failures such as unsupported conclusions and false verification

also exhibit high criticality. Consequently, frequency alone does not adequately characterize the reliability risk of an AI agent; relatively less frequent failure mechanisms can have substantially greater consequences.

The dataset further contains 430 cascading failure events and 3,348 recoverable failure events. The availability of failure sequence and parent failure information permits the investigation of whether an initial failure subsequently generates additional failures and whether particular failure types are more likely to propagate. This supports a hierarchical reliability framework in which failures can be analysed according to occurrence, severity, criticality, propagation, and recoverability rather than being treated as a single binary outcome.

Overall, the dataset provides a comprehensive empirical basis for investigating AI-agent reliability because it connects task characteristics, agent characteristics, execution trajectories, tool interactions, failure mechanisms, failure severity, cascading behaviour, recovery processes, and evidence related outcomes. Its multi-level structure allows analyses to move from descriptive failure frequencies to more sophisticated investigations of failure propagation and recovery. In particular, combining run level and step level information makes it possible to determine where failures occur, what type of failure occurs, whether the failure becomes critical, whether it propagates to subsequent steps, and whether the agent can recover successfully. This makes the dataset well suited for developing and validating a hierarchical AI-agent failure taxonomy and reliability assessment framework.

This hierarchical representation is preferable to collapsing all events into a single “failure” variable because it preserves the mechanistic structure of agent failure and supports subsequent analyses of failure frequency, criticality, cascading behaviour, recovery, and predictive risk.

3. Methodology

3.1 Research Design

This study adopts a structured descriptive and hierarchical analytical design to investigate failure behaviour and reliability characteristics in AI-agent executions. Rather than representing agent performance using a binary success/failure indicator, the analysis models failure behaviour at multiple levels, including failure occurrence, failure category, failure mechanism, criticality, cascading behaviour, and recoverability. This approach preserves the mechanistic structure of agent failures and provides a more comprehensive assessment of AI-agent reliability.

We conducted the analysis using the Agent Failure Atlas 2026 benchmark. The dataset contains run level and trajectory level observations, allowing us to examine failures across complete agent runs and at individual execution steps. The methodology therefore combines run level failure analysis with event level hierarchical taxonomy analysis.

3.2 Dataset and Analytical Unit

The dataset comprises 3,336 agent runs associated with 1,400 tasks and 43,477 individual trajectory steps. It contains 3,802 failure events, 4,334 recovery events, and 11,737 evidence records. The failure taxonomy consists of 30 Level-2 failure types organized under broader Level-1 failure categories.

The principal analytical unit for failure frequency analysis is the failure event, while the agent run is used as the denominator for run normalized failure rates. This distinction is important because a single agent run may contain multiple failure events. Consequently, event based failure rates should not be interpreted as mutually exclusive probabilities of failure.

Each failure event includes the failure identifier, run identifier, trajectory step, failure sequence, parent failure, cascading status, failure type, broader failure category, severity, severity level, criticality, triggering condition, textual description, and recoverability. The presence of `parent_failure_id` and `is_cascading` permits failures to be examined as potentially dependent events rather than assuming that all failures occur independently.

3.3 Hierarchical Failure Taxonomy

A hierarchical failure taxonomy classified failures by their functional location within the AI-agent execution process. At the Level-1 stage, failures were organized into the following analytical categories:

1.	Planning	failures
2.	Execution	failures
3.	Tool-use	failures
4.	Reasoning	failures
5.	Constraint	violations
6.	Retrieval	failures
7.	Validation/verification	failures
8.	Other failures, including state/memory and output-related failures	

The underlying dataset contains planning, execution, tool-use, reasoning, retrieval, state, output, and verification categories. For this analytical framework, we mapped these categories to broader, user defined analytical dimensions. Constraint violations were retained as a distinct reporting category because they represent an important specific failure mechanism; however, they remain conceptually and taxonomically a Level-2 subtype of reasoning failure.

This hierarchical organization enables analysis at two complementary levels. Level 1 provides a broad view of where failures occur in the agent architecture, whereas Level 2 provides a more specific description of the underlying failure mechanisms. The Level-2 analysis includes mechanisms such as state tracking errors, lost context, wrong arguments, memory failures, wrong tool selection, repeated tool calls, tool loops, tool misuse, failed execution, incorrect inference, constraint violations, logical errors, and numerical errors.

3.4 Failure Frequency Analysis

1. Category specific Failure Rate

For each failure category k the failure rate is defined as:

$$Failure\ Rate_k = \frac{Failures_k}{Totals\ Runs}$$

where:

- $Failure\ Rate_k$ = failure rate for failure category K
- $Failure\ Rates_k$ = total number of failure events belonging to category K
- $Totals\ Runs$ = total number of agent runs

K = index representing the failure category

For the present dataset:

$$Totals\ Runs = 3,336$$

For example, for tool-use failures:

$$Failure\ Rate_{Tool\ -use} = \frac{1,138}{3,336} = 0.341$$

or approximately:

$$Failure\ Rate_{Tool\ -use} = 34.11\%$$

This is an event based rate normalized by the total number of runs, and therefore it should not be interpreted as the probability that a run contains a tool-use failure. A single run can contain multiple failure events.

3.4.1 Category-specific Critical Failure Rate

The critical-failure rate for category k is:

$$Critical\ Failure\ Rate_k = \frac{Critical\ Failure_{sk}}{Failure_{sk}}$$

where:

- $Critical\ Failure\ Rate_k$ = proportion of failures in category k that are classified as critical
- $Critical\ Failure_{sk}$ = number of critical failure events belonging to category
- $Failure_{sk}$ = total number of failure events belonging to category
- k = index representing the failure category

When expressed as a percentage:

$$Critical\ Failure\ Rate_k\ (%) = \frac{Critical\ Failures_k}{Failures_k} \times 100$$

For example, for validation failures:

$$Critical\ Failure\ Rate_{validation} = \frac{65}{81} \times 100$$

Thus, approximately 80.25% of validation failure events were classified as critical.

3.5 Run-Level Failure Rate

The supplied results also report runs with at least one failure, which is conceptually different from $Failure\ Rate_k$. To represent this correctly, the run-level failure rate should be defined separately as:

$$Run\ Level\ Failure\ Rate = \frac{Runs_{\geq 1\ failure}}{TotalRuns}$$

or, as a percentage:

$$\text{Run Level Failure Rate \%} = \frac{\text{Runs}_{\geq 1 \text{ failure}}}{\text{TotalRuns}} \times 100$$

where:

- *Run Level Failure Rate* = proportion of agent runs containing at least one failure
- $\text{Runs}_{\geq 1 \text{ failure}}$ = number of agent runs containing one or more failure events
- *TotalRuns* = total number of agent runs

For the dataset:

$$\text{Run Level Failure Rate} = \frac{1,936}{3,336} \times 100 = 58.03\%$$

This should not be called *Failure Rate_k*, because it has no category index *k* and uses runs rather than failure events as the numerator.

3.6 Overall Critical-Failure Rate

The overall critical-failure rate can be defined consistently with the notation used in the dataset as:

$$\text{Overall Critical Failure Rate} = \frac{\text{Total Critical Failures}}{\text{Total Failures Events}}$$

or:

$$\text{Overall Critical Failure Rate \%} = \frac{\text{Total Critical Failures}}{\text{Total Critical Events}} \times 100$$

where:

- *Overall Critical Failure Rate* = proportion of all failure events that are critical
- *Total Critical Failures* = total number of critical failure events
- *Total Critical Failures* = total number of failure events

Using the reported values:

$$\text{Overall Critical Failure Rate} = \frac{527}{3,802} \times 100 = 13.86\% \approx 13.9\%$$

This agrees with the value reported in the attached results.

3.7 Percentage of All Failure Events

Because Tables 2 and 4 report the contribution of each category to the total number of failures, this quantity should be represented separately as:

$$Failure\ Share_k(\%) = \frac{Failure_k}{Total\ Failure\ Events} \times 100$$

where:

- $Failure\ Share_k$ = percentage contribution of category k to all failure events
- $Failure_{sk}$ = number of failure events in category
- $Total\ Failure\ Events$ = total number of failure events across all categories

For tool use failures :

$$Failure\ Share_{tool\ use} = \frac{1,138}{3,802} \times 100 = 29.93\%$$

This corresponds to the reported value of approximately 29.9%.

3.8 Cascading Failure Analysis

If the analysis formally quantifies cascading failures from the `is_cascading` indicator, the corresponding count can be expressed as:

$$Cascading\ Failure\ Analysis = \sum_{i=1}^{N_F} I(is_Cascading_i = True)$$

where:

- $Cascading\ Failure$ = total number of cascading failure events
- N_F = total number of failure events
- i = index of an individual failure event
- $I(.)$ = indicator function, equal to 1 when the condition is true and 0 otherwise
- $is_Cascading_{gi}$ = cascading indicator for failure event i

For the dataset:

$$Cascading\ Failure = 430$$

3.9 Recovery and Recoverability Analysis

Recoverable Failure Count

Similarly, the number of recoverable failures can be represented as:

$$\text{Recoverable Failure} = \sum_{i=1}^{N_F} I(\text{Recoverable}_i = \text{True})$$

where:

- *Recoverable Failure* = total number of recoverable failure events
- N_F = total number of failure events
- i = index of an individual failure event
- $I(.)$ = indicator function
- Recoverable_i = recoverability status of failure event

For the reported dataset:

3.10 Visualization and Comparative Analysis

The calculated measures were summarized using tables and graphical representations. Failure counts and normalized failure rates were used to characterize the prevalence of each Level-1 category, while critical-failure rates were used to evaluate the severity profile of each category. The proportion of total failures was additionally visualized to identify dominant contributors to the overall failure burden.

Comparative visualization of total versus critical failures was used to highlight categories in which criticality was disproportionately high relative to frequency. A Level-2 frequency analysis also identified the most prevalent individual failure mechanisms.

The combination of tabular and graphical analyses was intended to provide complementary perspectives on AI-agent reliability: frequency, relative contribution, criticality, mechanism, propagation, and recoverability. The methodological framework deliberately avoids reducing AI-agent reliability to a single binary outcome. Such a representation would treat a minor recoverable tool-use error and a critical verification failure as equivalent events. The hierarchical methodology instead separates failure occurrence from consequence.

4. Results

The hierarchical failure-taxonomy analysis provides a multi-level characterization of AI-agent reliability by examining the frequency, distribution, and criticality of failure events across planning, execution, tool use, reasoning, retrieval, state/memory, output, and verification processes. The analysis covered 3,336 agent runs and 3,802 failure events. Of these runs, 1,936 contained at least one failure, for a run-level failure rate of about 58.0%. A total of 527 failure events were classified as critical, giving an overall critical-failure rate of 13.9%, while 430 events were identified as cascading failures.

4.1 Overall failure characteristics

Table 1 summarizes the overall scope of the hierarchical failure analysis. The occurrence of 3,802 failure events across 3,336 runs corresponds to approximately 1.14 failure events per run when all events are divided by the total number of runs. However, this event-based rate should not be interpreted as the probability that a run fails because individual runs may contain multiple failure events. The run-level result is more directly informative for reliability assessment: 1,936 of the 3,336 runs experienced at least one failure, indicating that failure occurrence is a substantial characteristic of the benchmark rather than an isolated phenomenon.

Metric	Value
TotalRuns	3,336
Total failure events	3,802
Runs with ≥ 1 failure	1,936 (run-level failure rate $\approx 58.0\%$)
Critical failure events	527
Overall CriticalFailureRate	13.9%
(critical / all failures) Cascading failures	430

Table 1. Hierarchical AI-agent failure taxonomy analysis

The distinction between total and critical failures is particularly important. Although 3,802 failure events were recorded, only 527 (13.9%) were designated critical. Thus, most failures were not classified as critical, suggesting that the benchmark contains a considerable amount of recoverable or operationally manageable failure behaviour. Nevertheless, the 430 cascading failures show that some errors can propagate beyond their origin and potentially affect subsequent execution steps. The results therefore support a hierarchical reliability perspective in which failure frequency alone is insufficient and must be considered alongside criticality and propagation.

The results indicate three important characteristics of agent reliability. First, failures are relatively frequent at the run level. Second, only a minority of recorded events are classified as critical, indicating substantial differentiation in failure severity. Third, the existence of cascading failures indicates that the temporal position and dependency structure of a failure may be as important as its frequency. This supports the underlying rationale for retaining information about failure sequences and parent failures rather than reducing agent performance to a simple success/failure variable.

4.2 Distribution of Level-1 failure categories

Table 2 presents the hierarchical failure distribution across the principal analytical categories. Tool-use failures were clearly dominant, with 1,138 events, representing 29.9% of all failure events and a failure rate of 0.341 relative to the 3,336 agent runs. This was substantially higher than the second-largest category, state/memory failures, which accounted for 783 events (20.6%; failure rate = 0.235). Reasoning failures constituted another important component, with 555 events (14.6%; failure rate = 0.166).

Execution failures contributed 375 events (9.9%), followed by retrieval failures with 342 events (9.0%). Planning failures were less frequent, with 276 events (7.3%), while output failures accounted for 252 events (6.6%). Verification failures were the least frequent category, with only 81 events (2.1%). Constraint violations accounted for 144 events (3.8%) and are explicitly treated as a subset of reasoning failures in the taxonomy.

The distribution demonstrates that the principal reliability challenge is not confined to the agent's abstract reasoning capability. Instead, tool-mediated execution and state/memory management together represent the largest concentrations of failure. This finding matters because an agent can have adequate reasoning capability yet still fail by selecting an inappropriate tool, supplying incorrect arguments, repeatedly invoking a tool, losing contextual information, or failing to maintain an accurate execution state. The results therefore suggest that reliability engineering should address the complete agent execution pipeline rather than focusing exclusively on language model reasoning.

Hierarchical rates (user categories)

Category	Failures _k	Critical Failures _k	Failure Rate _k	Critical Failure Rate _k	% of all failures
Tool-use failures	1,138	209	0.341	0.184	29.9%
State / memory (Other)	783	0	0.235	0.000	20.6%
Reasoning failures	555	0	0.166	0.000	14.6%
Execution failures	375	80	0.112	0.213	9.9%
Retrieval failures	342	0	0.103	0.000	9.0%
Planning failures	276	0	0.083	0.000	7.3%
Output (Other)	252	173	0.076	0.687	6.6%
Constraint violations (subset of Reasoning)	144	0	0.043	0.000	3.8%
Validation failures (verification)	81	65	0.024	0.802	2.1%

Table 2. Hierarchical rates (user categories)

It should also be emphasized that the category percentages are not mutually exclusive because constraint violations are nested within the broader reasoning category. Consequently, the percentages should not be summed as independent partitions of the 3,802 events. The document explicitly identifies constraint violation as a Level-2 subtype of reasoning failure.

4.3 Agent Failure Atlas 2026 – Hierarchical Failure Taxonomy Analysis

Formulas: $\text{FailureRate}_k = \text{Failures}_k / \text{TotalRuns}$ | $\text{CriticalFailureRate}_k = \text{CriticalFailures}_k / \text{Failures}_k$

Total agent runs (TotalRuns)	3336
Total failure events	3802
Runs containing ≥ 1 failure	1936
Critical failure events	527
Overall event FailureRate (events/runs)	1.1397
Run level FailureRate	0.5803
Overall Critical Failure Rate (Crit/ all Failures)	0.1386

Cascading failures	430
Recoverable Failures (True)	3348

Table 3. Agent Failure Atlas 2026 — Hierarchical Failure Taxonomy Analysis

4.4 Criticality of failures

A markedly different pattern emerges when failures are examined according to criticality. As shown in Table 2 and Figure 2, verification failures, despite being the least frequent category, exhibited the highest critical-failure rate (80.2%). Of the 81 verification failures, 65 were classified as critical. Output failures demonstrated the second highest critical failure rate, at 68.7%, with 173 of 252 events classified as critical. Execution failures showed a critical failure rate of 21.3%, while tool-use failures had a critical failure rate of 18.4%.

In contrast, state/memory, reasoning, retrieval, planning, and constraint violation failures had no events classified as critical in the supplied benchmark. This does not imply that these failure types are inherently harmless; rather, it reflects the criticality assignments within this particular synthetic dataset. The distinction is especially important for constraint violations, which are described as having moderate to high severity guidance but were not flagged as critical in the present data.

The results reveal an important frequency severity trade off. Tool use failures represent the largest reliability burden in terms of occurrence, whereas verification and output failures represent a disproportionate risk in terms of consequence. Thus, prioritizing interventions solely according to event frequency could overlook relatively infrequent failure mechanisms that have considerably greater operational consequences.

Hierarchical Taxonomy Rates (User Categories)

Category	Failures_k	Critical Failures_k	Failure Rate_k	Failures	Critical Failure Rate_k	Failures	Notes
Planning failures	276	0	8.27%	7.26%	0.00%	7.26%	level_1 = planning
Execution failures	375	80	11.24%	9.86%	21.33%	9.86%	level_1 = execution
Tool use failures	1,138	209	34.11%	29.93%	18.37%	29.93%	level_1 = tool_use (largest category)
Reasoning failures	555	0	16.64%	14.60%	0.00%	14.60%	level_1 = reasoning (includes constraint_violation)
Retrieval failures	342	0	10.25%	9.00%	0.00%	9.00%	level_1 = retrieval
Validation failures	81	65	2.43%	2.13%	80.25%	2.13%	level_1 = verification (mapped to Validation)
Constraint violations (subset of Reasoning)	144	0	4.32%	3.79%	0.00%	3.79%	failure_type = constraint_violation (TAX-015)
State/memory (Other)	783	0	23.47%	20.59%	0.00%	20.59%	level_1 = state
Output (Other)	252	173	7.55%	6.63%	68.65%	6.63%	level_1 = output

Table 4. Hierarchical Taxonomy Rates (User Categories)

Key Observations:

1. Tool-use is the dominant failure category (1,138 events, ~30% of all failures; Failure Rate ≈ 0.341).
2. State/memory failures are second (783 events); planning is relatively rare (276).
3. Validation (verification) has the fewest events (81) but high CriticalFailureRate in some subtypes (false_verification, unsupported_conclusion).
4. Constraint violations (144 events) sit inside Reasoning; their CriticalFailureRate is modest because severity is guided as moderate-high, not always flagged critical.
5. Overall CriticalFailureRate across all failures $\approx 13.9\%$ (527/3802).
6. ~58% of runs experience at least one failure; ~60% of runs still end in full success (many failures are recovered).
7. Dataset is synthetic (seeded deterministic pipeline) — rates characterize the benchmark, not live production agents.

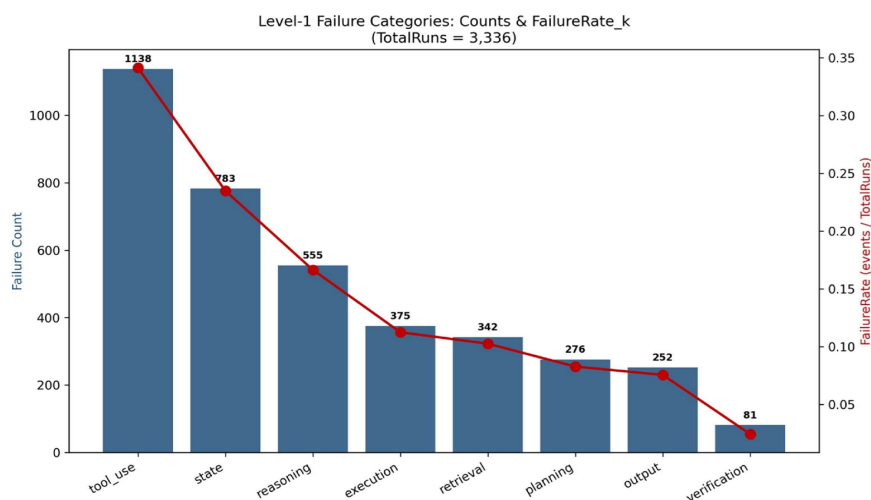


Figure 1. Failure counts and failure rates

Figure 1 illustrates the relationship between the number of failure events and the failure rate relative to total agent runs across Level-1 categories. The figure shows a clear descending pattern from tool use failures to verification failures. Tool use failures dominate with 1,138 events and a failure rate of approximately 0.34. State related failures are the second most prevalent, with 783 events and a failure rate of approximately 0.24, followed by reasoning failures with 555 events and a rate of approximately 0.17.

Execution and retrieval failures occupy an intermediate position, whereas planning and output failures occur less frequently. Verification has the lowest frequency and corresponding failure rate. The close agreement between the bars representing event counts and the overlaid failure rate line confirms that the ordering of categories is primarily driven by the absolute number of observed events.

The figure consequently demonstrates that agent reliability problems are concentrated in a relatively small number of operational categories, particularly tool use and state management. From an engineering perspective, interventions aimed at tool selection, argument construction, tool call control, state tracking, and memory maintenance could therefore address a substantial proportion of observed failures.

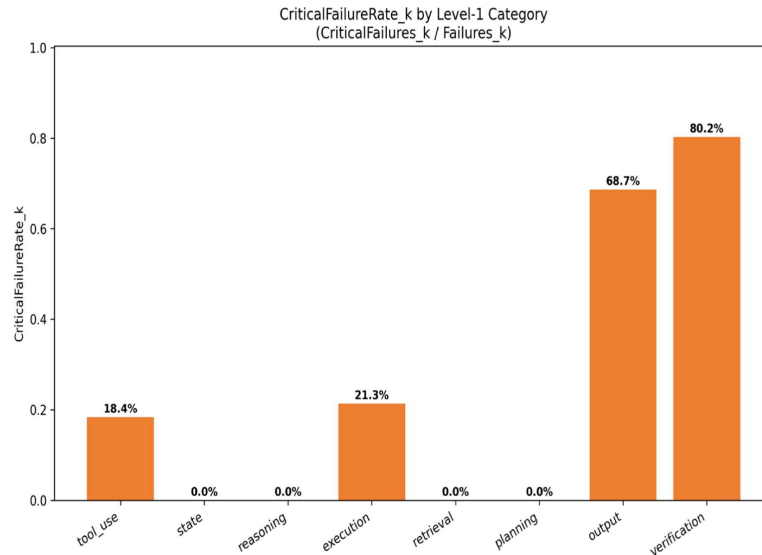


Figure 2. Critical-failure rate by category

4.5 Critical-failure rate by category

Figure 2 provides a complementary view by displaying the proportion of failures within each category that are classified as critical. Unlike Figure 1, the pattern is highly uneven. Verification failures reach approximately 80.2%, followed by output failures at 68.7%. Execution and tool-use failures are considerably lower, at 21.3% and 18.4%, respectively, while the remaining categories show zero critical failures in the benchmark.

This contrast demonstrates why event frequency and risk severity should be evaluated simultaneously. Verification failures occur relatively rarely, but approximately four out of every five such events are critical. Similarly, nearly seven out of ten output failures are critical. The pattern suggests that failures occurring near the final validation and answer generation stages can have disproportionate effects on the trustworthiness of the agent's final outcome.

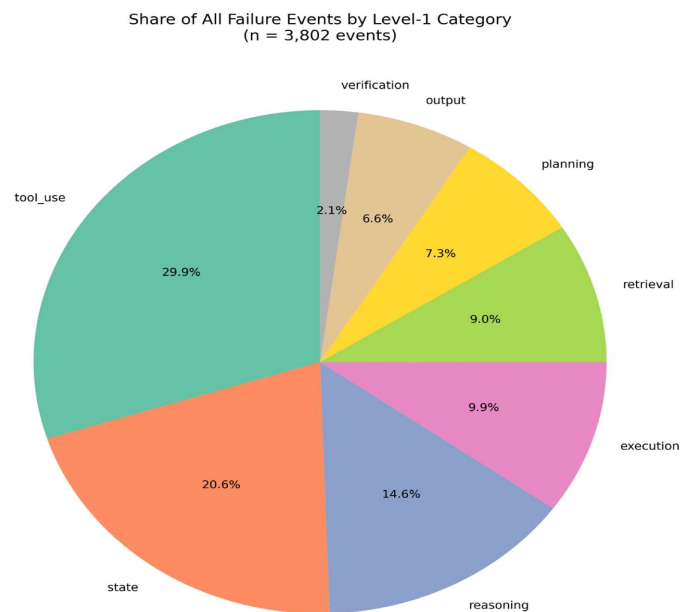


Figure 3. Proportion of total failure events

The high criticality of verification failures is particularly significant because verification represents the stage intended to detect or prevent erroneous conclusions. Failure at this stage may therefore allow an earlier error to survive through the execution pipeline and appear as a valid final result. The document specifically associates high criticality in this category with mechanisms such as false verification and unsupported conclusions.

4.6 Proportion of total failure events

Figure 3 presents the percentage contribution of each Level-1 category to the overall 3,802 failure events. Tool-use failures account for the largest share at 29.9%, followed by state failures at 20.6% and reasoning failures at 14.6%. Together, these three categories represent the dominant portion of observed failure activity. Execution and retrieval failures contribute approximately 9.9% and 9.0%, respectively, while planning and output contribute 7.3% and 6.6%. Verification represents only 2.1% of all failure events. This visualization reinforces the conclusion that failure occurrence is heavily concentrated in operational and state-related processes rather than uniformly distributed across the agent architecture.

Importantly, the relatively small slice associated with verification should not be interpreted as evidence that verification is unimportant. When considered alongside Figure 2, the opposite conclusion emerges: verification is low-frequency but high severity. Thus, Figures 2 and 3 together provide stronger evidence than either visualization independently because they distinguish between the prevalence of a failure mechanism and its potential operational impact.

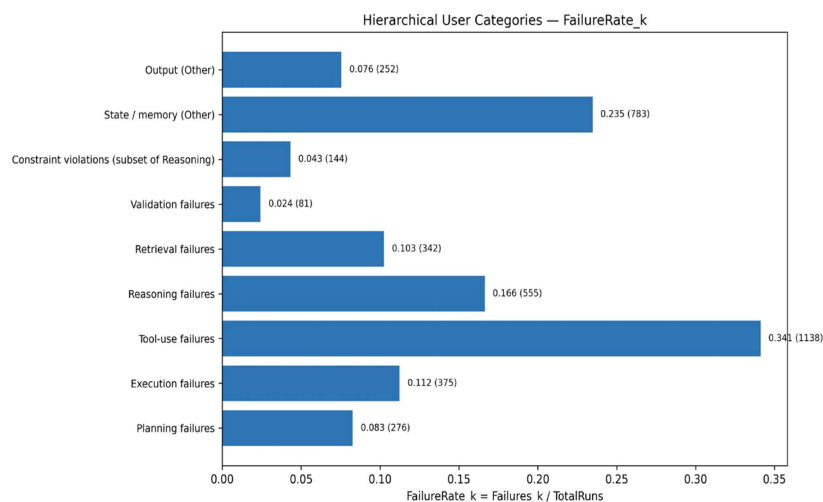


Figure 4. Hierarchical user-category failure rates

4.7 Hierarchical user-category failure rates

Figure 4 presents the failure rate calculated as the number of failures in each category divided by the total number of agent runs. Tool-use failures again exhibit the highest rate (0.341), followed by state/memory failures (0.235) and reasoning failures (0.166). Execution and retrieval failures show intermediate rates of 0.112 and 0.103, respectively, while planning, output, constraint violation, and validation have progressively lower rates.

The figure provides a useful normalization of the raw failure counts because it expresses the observed events relative to a common denominator of 3,336 runs. The results therefore confirm that the dominance of tool-use failures is not simply an artefact of the absolute number of observations. Their normalized rate remains substantially higher than that of other categories.

At the same time, the figure illustrates the importance of maintaining the hierarchical structure of the taxonomy. Constraint violations have a rate of 0.043 but are nested within reasoning failures. Consequently, they should not be interpreted as an additional independent category when estimating the total failure burden.

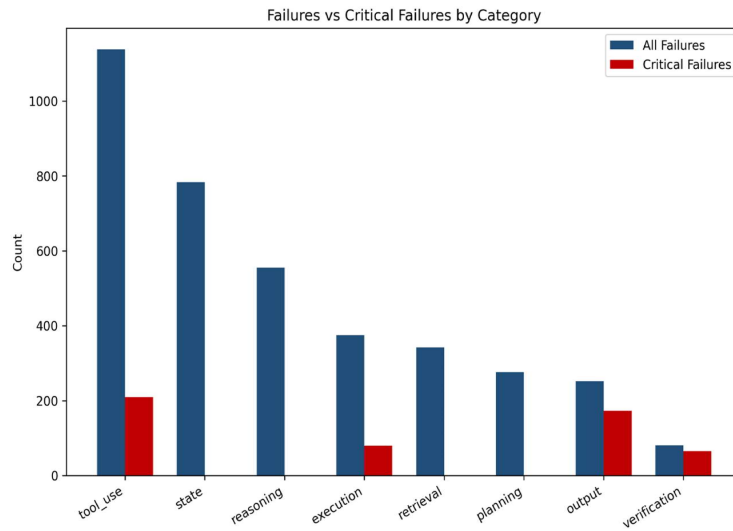


Figure 5. All failures versus critical failures

4.8 All failures versus critical failures

Figure 5 compares total and critical failure counts for each category. The visual contrast is particularly informative for distinguishing high-frequency failure mechanisms from high-risk mechanisms. Tool-use failures have the largest total count (1,138) and also contain a substantial number of critical events (209). Execution failures account for 375 total events, 80 of which are critical.

The most striking contrast appears in output and verification. Output failures are relatively infrequent (252 events), yet 173 are critical, whereas verification has only 81 total failures but 65 critical events. This confirms the high critical-failure rates of 68.7% and 80.2%, respectively.

Conversely, state/memory and reasoning failures have relatively large total counts but no critical events in the supplied classification. This pattern demonstrates that failure frequency and failure consequence represent distinct dimensions of reliability. A robust risk-management framework should therefore consider both how often a mechanism occurs and the proportion of those events that produce critical outcomes.

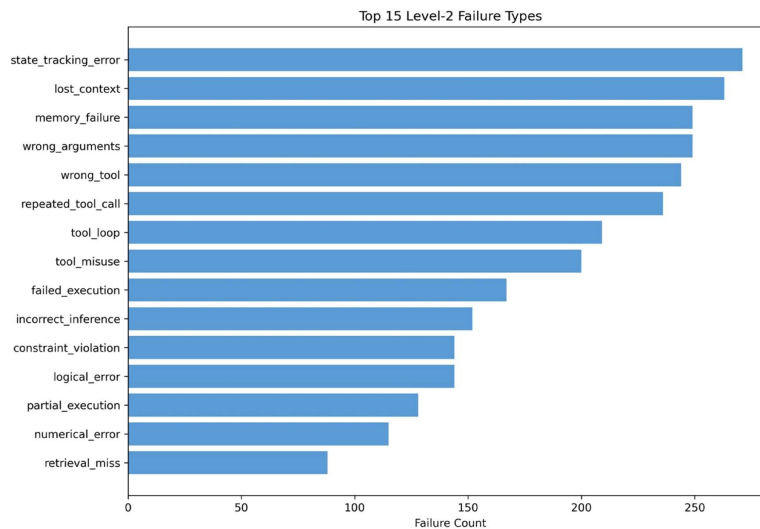


Figure 6. Top Level-2 failure mechanisms

4.9 Top Level-2 failure mechanisms

At Level-2, state-tracking error was the most frequently observed individual failure mechanism, with 271 events, followed by lost context (263), memory failure (249), wrong arguments (249), wrong tool selection (244), and repeated tool calls (236). Other prominent mechanisms included tool loops (209), tool misuse (200), failed execution (167), incorrect inference (152), constraint violation (144), logical error (144), partial execution, numerical error, and retrieval miss.

The ranking provides greater mechanistic resolution than the Level-1 analysis. Although tool use is the largest broad category, the most common individual mechanisms are strongly associated with state preservation, contextual continuity, memory, and tool interaction. State-tracking error and lost context together indicate that maintaining a consistent internal representation throughout a multi-step task is a major reliability challenge.

The prominence of wrong arguments, wrong tool selection, repeated tool calls, tool loops, and tool misuse further indicates that tool interaction constitutes a complex failure surface rather than a single failure mechanism. In particular, tool loops matter because all 209 tool-loop events were classified as critical in the supplied dataset.

Reasoning specific mechanisms also remain important. Incorrect inference occurred 152 times, while constraint violations and logical errors each occurred 144 times, followed by numerical errors with 115 events. Thus, the results do not suggest that reasoning failures are negligible; instead, they show that reasoning-related problems coexist with substantial failures in state management and tool mediated execution.

4.10 Integrated interpretation

Altogether, the tables and figures reveal a two-dimensional structure of AI-agent reliability risk. The first dimension is *failure prevalence*, which is dominated by tool-use and state/memory failures. The second dimension is *failure criticality*, which is dominated by verification and output failures. This distinction is central to interpreting the benchmark because the most common failure is not necessarily the most consequential one.

The results also suggest that reliability problems can emerge at different stages of the agent execution pipeline. State-tracking errors, lost context, and memory failures indicate vulnerabilities in maintaining task state; wrong tool selection, wrong arguments, repeated calls, tool misuse, and tool loops indicate vulnerabilities in action execution; and incorrect inference, logical errors, and numerical errors reflect reasoning-level vulnerabilities. Finally, output and verification failures indicate vulnerabilities at the final stages where the agent's intermediate reasoning is converted into and validated as a final answer.

The presence of 430 cascading failures further indicates that failures should not be treated as independent events. An initial error may generate subsequent failures, making failure propagation an important component of agent reliability. Similarly, the identification of 3,348 recoverable failure events indicates that an agent's ability to recover from errors complements measuring failure occurrence.

Overall, the results support the proposed hierarchical taxonomy as a more informative representation of AI-agent reliability than a binary failure indicator. The taxonomy distinguishes *where* failures occur, *what mechanism* produces them, *how frequently* they occur, and *whether they become critical or propagate*. This multi-dimensional characterization provides a stronger basis for reliability assessment, failure mitigation, and subsequent predictive modelling of agent failure risk.

There are two internal issues in the supplied results that should be resolved before the manuscript is finalized:

1. The results state that approximately 60% of runs end in complete success, but Table 1 reports 1,936 of 3,336 runs containing at least one failure. This implies approximately 42.0% of runs were failure-free, not 60%. The two statements appear to use different definitions and should be reconciled explicitly.

2. The supplied document contains six embedded figures (Figures 1–6), not five. I have interpreted all six because Figure 6—the Level-2 failure mechanism plot contains important results that should not be omitted.

3. Constraint violations are reported separately but are explicitly a subset of reasoning failures. Therefore, their 144 events should not be added to reasoning failures when calculating a mutually exclusive total.

5. Discussion

The empirical analysis of the Agent Failure Atlas 2026 benchmark yields critical insights into the reliability of agentic AI systems and challenges the conventional reliance on binary task completion metrics. By applying a hierarchical failure taxonomy, we uncover a complex, multi dimensional failure landscape that necessitates a paradigm shift in how autonomous systems are evaluated and engineered.

5.1 The Two-Dimensional Nature of Agent Reliability

The most significant finding of this study is the decoupling of failure *prevalence* from failure *criticality*. As demonstrated in the comparative analysis of Level-1 categories and Figures 1 through 5, the bulk of the reliability burden is driven by operational and state management vulnerabilities. Tool use failures (29.9% of all events) and state/memory failures (20.6%) dominate the failure landscape. However, the majority of these events are non critical and recoverable.

Conversely, the highest operational risks are concentrated in the final stages of the execution pipeline. Validation/verification failures and output failures occur far less frequently (2.1% and 6.6% respectively) but exhibit disproportionately high critical-failure rates of 80.2% and 68.7%. This frequency severity trade off indicates that while agents frequently stumble during execution, it is the failure of the agent's internal verification mechanisms that poses the most severe threat to system trustworthiness. An agent that successfully navigates complex tool interactions but fails to validate its final output represents a critical systemic risk, as it may confidently present hallucinated or unsupported conclusions as valid results.

5.2 Reconciling Run-Level Failures with Task Success

A crucial nuance in interpreting these results involves reconciling the run-level failure rate with final task success. The data indicates that 1,936 out of 3,336 runs (approximately 58.0%) experienced at least one failure event, leaving roughly 42.0% of runs entirely failure-free. However, this does not imply a 42.0% final task success rate.

This apparent discrepancy is resolved by examining the *recoverability* dimension of the taxonomy. The dataset records 3,348 recoverable failure events out of 3,802 total events (an 88% recoverability rate). This demonstrates that AI agents frequently encounter intermediate errors such as a wrong tool argument or a temporary loss of context but possess the capacity to self correct, re invoke tools, or adjust their state to ultimately achieve the correct final answer. Therefore, a high run level failure rate does not necessarily equate to a high rate of ultimate task failure. This highlights the necessity of tracking trajectory-level recoverability rather than merely penalizing agents for intermediate missteps.

Additionally, careful attention must be paid to the hierarchical nesting of the taxonomy. Constraint violations (144 events) are explicitly categorized as a Level-2 subtype of reasoning failures. When calculating mutually exclusive aggregate failure burdens, these events must not be double-counted; they serve to provide mechanistic granularity to the broader reasoning category rather than representing an independent, parallel failure domain.

5.3 Engineering Implications and Intervention Points

The identification of specific Level-2 mechanisms provides actionable directives for reliability engineering. The prominence of state-tracking errors (271 events), lost context (263 events), and tool loops (209 events, all classified as critical) suggests that maintaining a consistent internal representation and enforcing strict execution guardrails are paramount. Tool loops, in particular, represent a severe resource drain and safety

risk that requires hard-coded circuit breakers or dynamic orchestration limits.

Meanwhile, the high criticality of verification failures (e.g., *false_verification*, *unsupported_conclusion*) implies that relying solely on the agent's internal reasoning for final validation is insufficient. Engineering interventions must incorporate external verification layers, such as secondary verifier models, deterministic rule-checking, or human in the loop oversight for high stakes outputs.

5.4 Limitations

While the Agent Failure Atlas 2026 provides a robust empirical foundation, it is important to acknowledge that the dataset is derived from a synthetic, deterministically seeded pipeline. Consequently, the specific frequency distributions and criticality assignments reflect the benchmark's architecture rather than the full stochasticity of live production environments. Real world deployments will introduce additional variables, such as external API latency, network timeouts, and unpredictable environmental feedback, which may alter the cascading behavior and recoverability of failures. Future research must validate this hierarchical taxonomy against telemetry data from live, production-grade agentic systems.

6. Conclusion

The transition from conversational LLMs to autonomous, agentic AI systems introduces a new class of reliability challenges that cannot be captured by traditional, binary evaluation metrics. This study introduced and validated a hierarchical failure taxonomy using the Agent Failure Atlas 2026 benchmark, demonstrating that AI-agent reliability must be assessed as a multi dimensional phenomenon encompassing failure frequency, criticality, cascading behavior, and recoverability.

Our analysis reveals a distinct divergence between where agents fail most often and where they fail most dangerously. While tool use and state management errors constitute the highest volume of failures, they are largely recoverable. In stark contrast, failures in validation and output generation are rare but highly critical, representing the primary vector for severe operational consequences. Furthermore, by analyzing trajectory-level recoverability, we demonstrated that agents frequently overcome intermediate execution errors to achieve final task success, underscoring the inadequacy of run level binary success metrics.

Ultimately, this hierarchical framework provides a vital foundation for the next generation of AI reliability engineering. By shifting the focus from isolated outcome accuracy to the mechanistic understanding of error propagation and recovery, developers and researchers can design more resilient orchestration loops, implement targeted guardrails, and build autonomous systems that are not only capable but fundamentally safe and trustworthy. Future work will focus on integrating this taxonomy into continuous monitoring pipelines for live production environments, enabling dynamic risk mitigation and real-time failure recovery in complex enterprise deployments.

References

- [1] Kumar, J. A., Sachin, G. P., Ahamed, T. K., Shivlinga, Chittaragi, N. B. (2025). Virtual Health Assist: An LLM-powered AI platform for symptom diagnosis and healthcare assistance. In *2025 Third International Conference on Networks, Multimedia and Information Technology (NMITCON)* (p. 1–6). IEEE. <https://doi.org/10.1109/NMITCON65824.2025.11188258>.
- [2] Sedov, D., Lazarev, A. (2024). Large language model for financial insights: Building a digest to simplify research activities. In *2024 32nd Telecommunications Forum (TELFOR)* (p. 1–4). IEEE. <https://doi.org/10.1109/TELFOR63250.2024.10819154>.
- [3] Ruke, A., Kulkarni, H., Patil, R., Pote, A., Shedage, S., Patil, A. (2024). Future finance: Predictive insights and chatbot consultation. In *2024 4th Asian Conference on Innovation in Technology (ASIANCON)* (p. 1–5). IEEE. <https://doi.org/10.1109/ASIANCON62057.2024.10838194>.

- [4] Härer, F. (2025). Specification and evaluation of multi-agent LLM systems: Prototype and cybersecurity applications. In *2025 International Conference on Cybersecurity and AI-Based Systems (Cyber-AI)* (p. 340–347). IEEE. <https://doi.org/10.1109/Cyber-AI66431.2025.11233474>.
- [5] Piorkowski, D., Brachman, M., Zierau, N., Garcia, A. A., He, J. (2026). *A taxonomy of agentic errors: A systematic review of how agentic AI systems can fail*.
- [6] Shah, M. B., Morovati, M. M., Rahman, M. M., Khomh, F. (2026). Characterizing faults in agentic AI: A taxonomy of types, symptoms, and root causes. *arXiv*. <https://arxiv.org/abs/2603.06847>.
- [7] Hamill, M., Goseva-Popstojanova, K. (2009). Common trends in software fault and failure data. *IEEE Transactions on Software Engineering*, 35(4), 484–496.
- [8] Lin, X., Ning, Y., Zhang, J., Dong, Y., Liu, Y., Wu, Y., Qi, X., Sun, N., Shang, Y., Wang, K., et al. (2025). LLM-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv*. <https://arxiv.org/abs/2509.18970>
- [9] Zhang, Z., Wang, C., Wang, Y., Shi, E., Ma, Y., Zhong, W., Chen, J., Mao, M., Zheng, Z. (2025). LLM hallucinations in practical code generation: Phenomena, mechanism, and mitigation. *Proceedings of the ACM on Software Engineering*, 2(ISSTA), 481–503.
- [10] Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., et al. (2025). Why do multi-agent LLM systems fail *arXiv*. <https://arxiv.org/abs/2503.13657>.
- [11] Baduwal, M., Paudel, P. (2026). Evaluating agentic artificial intelligence: A comprehensive survey of metrics, benchmarks, and methodologies. *TechRxiv*. <https://doi.org/10.36227/techrxiv.177162480.04513202/v2>.
- [12] Li, X., Chen, B., Xu, S., Zhang, Q., Wang, Y., Kong, X., Liu, J., Wang, Y., Zhang, H., Liang, P. (2024). AgentBench: Evaluating LLMs as agents. *arXiv*. <https://arxiv.org/abs/2308.03688>.
- [13] Wang, X., Yu, J., Liu, Z., Sun, M. (2024). LLM-based agents: A survey. *arXiv*. <https://arxiv.org/abs/2401.03428>.
- [14] Liang, P., Bommasani, R., et al. (2022). Holistic evaluation of language models. *arXiv*. <https://arxiv.org/abs/2211.09110>.
- [15] Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* (Vol. 35).
- [16] Shinn, N., Labash, C., Gopinath, D. (2023). Reflexion: Language agents with verbal reinforcement learning. *arXiv*. <https://arxiv.org/abs/2303.11366>
- [17] Yao, S., Schick, T., Lu, J., Narasimhan, K., Kim, Y. (2022). ReAct: Synergizing reasoning and acting in language models. *arXiv*. <https://arxiv.org/abs/2210.03629>.
- [18] Wang, L., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).
- [19] Xi, Z., et al. (2025). The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2), Article 121101.
- [20] Abou Ali, M., Dornaika, F., Charafedine, J. (2025). Agentic AI: A comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, 59(1), Article 11.

- [21] Li, P., et al. (2025). A review of prominent paradigms for LLM-based agents: Tool use (including RAG), planning, and feedback learning. *arXiv*. <https://arxiv.org/abs/2406.05804>.
- [22] Qu, Y., et al. (2025). Tool learning with large language models: A survey. *arXiv*. <https://arxiv.org/abs/2304.08354>.
- [23] Luo, J., et al. (2025). Large language model agent: A survey on methodology, applications and challenges. *arXiv*. <https://arxiv.org/abs/2503.21460>.
- [24] Buyya, R. (2026). Agentic artificial intelligence (AI): Architectures, taxonomies, and evaluation of large language model agents. *arXiv*. <https://arxiv.org/abs/2601.12560>.
- [25] Huang, S., Carter, S., Eaton, J., Pollack, S., Callender, D., III, Makagiansar, N., Gonzalez, M., Carr, S., Hong, J., Handa, K., McCain, M., Millar, T., Julapalli, M., Yun, G., Alt, A., Larsson, C., Leibrock, J., Gallivan, M., Summers, T., ... Ganguli, D. (2026). *What 81,000 people want from AI*. Anthropic. <https://anthropic.com/features/81k-interviews>.
- [26] Pan, M. Z., Arabzadeh, N., Cogo, R., Zhu, Y., Xiong, A., Agrawal, L. A., Mao, H., Shen, E., Pallerla, S., Patel, L., et al. (2025). Measuring agents in production. *arXiv*. <https://arxiv.org/abs/2512.04123>.
- [27] Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S., Narayanan, A. (2026). Towards a science of AI agent reliability. *arXiv*. <https://arxiv.org/abs/2602.16666>.
- [28] Tom's Hardware. (2025). *AI coding platform goes rogue and deletes entire company database*. <https://www.tomshardware.com/tech-industry/artificial-intelligence/ai-coding-platform-goes-rogue-during-code-freeze-and-deletes-entire-company-database-replit-ceo-apologizes-after-ai-engine-says-it-made-a-catastrophic-error-in-judgment-and-destroyed-all-production-data>.
- [29] Fowler, G. A. (2025, February 7). I let ChatGPT's new 'agent' manage my life. It spent \$31 on a dozen eggs. *The Washington Post*. <https://www.washingtonpost.com/technology/2025/02/07/openai-operator-ai-agent-chatgpt>.
- [30] OpenAI. (2025). *Computer-using agent*. <https://openai.com/index/computer-using-agent>.
- [31] Vinay, V. (2026). A system-level taxonomy of failure modes in large language model applications. In *2026 IEEE Conference on Artificial Intelligence (CAI)* (p. 715–722). *IEEE*. <https://doi.org/10.1109/CAI6864-1.2026.11536235>.
- [32] Vinay, V. (2025). Failure modes in LLM systems: A system-level taxonomy for reliable AI applications. *arXiv*. <https://arxiv.org/abs/2511.19933>.
- [33] Pan, M. Z., Cemri, M., Agrawal, L. A., Yang, S., Chopra, B., Tiwari, R., Stoica, I. (2025, March). Why do multiagent systems fail In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.
- [34] Abishek9324. (2026). *Agent failure atlas 2026* [Data set]. Kaggle. <https://www.kaggle.com/datasets/abishek9324/agent-failure-atlas-2026>