Trace-Level Failure Boundaries and Repair for Verifier-Augmented Engineering Agents

Anonymous Authors[1]

Abstract

Verifier-augmented agents can look reliable in clean runs while failing at the boundary where tool outputs, artifact fields, and later checks diverge. We study this problem in SysML v2 link-budget generation and introduce a replayable trace-level protocol for engineering agents: deterministic labels for verifier invocation and result use, fixed gates over emitted artifacts, and tool-free repair tests on frozen failures. The simple faithful-tool calibration is deliberately treated as weak evidence because the exposed tool and the final verifier share the same physics. The boundary probes are the main object: they show failures from schema ambiguity, audit-surface mismatch, and stale artifact memory, and they test whether explicit verifier-visible contracts or repair signals move those failures. The strongest repair lesson is negative as well as positive: more verifier text is not automatically better, while structured quarantine of suspect memory is a useful but bounded intervention. The contribution is not a higher pass rate, but a reproducible way to localize when verifier use stops being faithful and which repair signals actually change the failed artifact.

1. Introduction

Engineering agents increasingly produce artifacts that are parsed, checked, reused, and carried into later decisions. In model-based systems engineering, a generated SysML model may encode mission parameters, derived quantities, requirements evidence, and verification claims. Tool calling should help, since an agent can invoke calculators or simulators before emitting the final model. However, a final artifact can still be wrong for several trace-local reasons: the agent may skip the verifier, ignore its result, attach a tool call to a discarded draft, duplicate attributes so extraction becomes ambiguous, or omit fields needed for later audit. A terminal pass/fail score does not distinguish these cases.

We therefore study *trace-level failure boundaries*: conditions under which a verifier-augmented engineering agent crosses from faithful tool use to a mechanically detectable failure state. The goal is not a universal taxonomy or an industrial SysML benchmark. We instead define deterministic labels over archived trace events, parsed artifact fields, and verifier gates, then replay the exact failed artifacts under controlled repair prompts. This separates three claims that are often conflated: a tool loop can work in a clean calibration, it can fail under small schema, memory, or compute-conflict triggers, and a repair signal may or may not change the frozen failed artifact once verifier tools are removed.

We instantiate the protocol on SysML v2 textual artifacts for CubeSat link-budget generation and related engineering probes. The easy calibration is intentionally weak evidence: `gpt-5.5` passes 50/50 when the exposed tool and final verifier share the same physics. The boundary probes are the main result. Schema-surface contracts remove scoped-attribute and audit-surface failures; redundant tools with one hidden biased compute path do not by themselves induce biased emission; and stale artifact memory can persist across many stages before a late power gate exposes it. In frozen-artifact repair, numeric verifier text repairs 0/21 SysML failures on both tested models, while trigger-aware structured quarantine repairs 21/21 on `gpt-5.5` and 19/21 on `gpt-5.4-mini`. A JSON transfer probe preserves the memory trigger but saturates across repair arms, limiting the repair-shape claim rather than broadening it.

The contributions are: (1) a replayable failure-boundary protocol with status/tool labels, gate-level endpoints, normalized event records, and deterministic coders; (2) a boundary-discovery suite covering schema ambiguity, audit-surface mismatch, redundant compute paths, staged memory contamination, model sensitivity, and JSON transfer; and (3) a stricter repair ablation in which failed artifacts are frozen, verifier tools are withheld, and only the repair text changes. All claims are scoped to the recorded harness, tasks, models, and mechanically visible labels.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related Work

Engineering artifact benchmarks. Recent engineering benchmarks measure whether a model can produce a plausible technical artifact or a design that passes an external check. SysMBench, EngDesign, RocketBench, and SysML v2 semantic-alignment work all make engineering outputs more measurable (Jin et al., 2025; Guo et al., 2025; Simonds, 2025; Li et al., 2025). Our question is narrower: after a verifier is available, did the agent invoke it, preserve its result, and expose the fields needed for later checks? We therefore move the unit of analysis from final design quality to trace-local verifier handling over the emitted artifact.

Industrial MBSE and standards. Industrial MBSE has a broader target: SysML v2 specifications and pilot implementations, commercial tooling, model libraries, requirements traceability, allocation, behavioral verification, and configuration management (Obj, 2025; SysML.org, 2025; Systems Modeling Community, 2026b;a). NASA's 2025 MBSE–MBSA report also emphasizes that practical model-based analysis often still depends on hand-built profiles, plugins, and human review (Shi et al., 2025). We do not claim that our textual SysML subset covers that industrial surface. The narrower harness is deliberate: it lets us archive traces, replay exact artifacts, and assign deterministic verifier-boundary labels.

Trajectory and tool-use evaluation. Agent evaluations increasingly inspect trajectories, tool use, propagation cascades, runtime mitigation, repeated-trial reliability, and fault injection (Bang et al., 2025; Kokane et al., 2025; Gurram, 2026; Zhu et al., 2025; Ma et al., 2025; Barke et al., 2026; Chen et al., 2026a;b; Liu et al., 2026; Cemri et al., 2025; Yao et al., 2024; Barres et al., 2025; Gupta, 2026; Jia et al., 2026; Hui et al., 2026). This work supplies useful vocabulary for process failures. For our purpose, however, the labels must be deterministic gates over one stored engineering artifact. Our S/T rubric is correspondingly narrow: it records verifier invocation, verifier-result honoring, visibility removal, and artifact fabrication. Appendix C maps these endpoints back to broader taxonomies.

Security and guardrails. Adversarial-agent and guardrail work studies malicious instructions, unsafe tools, and runtime enforcement (Debenedetti et al., 2024; Zhan et al., 2024; Andriushchenko et al., 2025; OWASP GenAI Security Project, 2025; Wang et al., 2026; Doshi et al., 2026). Our probes use non-adversarial prompts. Even the hidden biased redundant tool is a compute-disagreement probe rather than prompt injection or malicious memory poisoning. This isolates a pre-security reliability question: can the agent keep tool results, artifact fields, and later gates aligned before adversaries are introduced?

Faithfulness and this paper's gap. Faithfulness work asks whether explanations or reasoning traces reflect the computation actually used (Kadavath et al., 2022; Lanham et al., 2023; Turpin et al., 2023). We ask the analogous artifact question: after an external computation returns, do the final SysML values and claims remain consistent with it, and can later gates still see the relevant fields? The gap we target is the combination of fixed verifier surfaces, trace-local labels, controlled boundary triggers, and frozen-artifact repair ablations with verifier tools removed.

3. Methodology

Experimental unit and artifact extraction. The basic unit of the study is a replayable run directory. For the calibration condition, we use five hand-authored S-band CubeSat downlink specifications. The model sees only the natural-language task specification. Verifier names, thresholds, and mission constraints are loaded by the runner from a separate task configuration. A reference SysML model is retained only for package inspection and closure checks; it is never shown to the model and is not used as an answer key. Boundary probes use the same task-directory structure. When a probe needs generated prompts, as in the non-space OMBSE and JSON-transfer settings, the prompt-generation step stores the exact prompt in the trace archive.

We apply the same extraction rule to every model output. The evaluator selects the first fenced block tagged `sysml`, `sysml2`, or `kerml`; if no such block appears, the whole response is treated as the candidate artifact. There is no hand repair between generation and scoring. The subset parser flattens declared attributes into path–value pairs, and parse errors, API errors, and empty final artifacts are all coded as S0. Each run archive stores the complete trace, the raw model text, and the exact artifact that was scored; the replication package gives the repository-level mapping.

Verifier gates. A run is counted as a pass only if the extracted artifact parses and every configured gate accepts it. The link gate recomputes range, free-space path loss, environmental losses, noise, and required E_b/N_0 from standard modulation/coding references (Sklar, 2001; Proakis & Salehi, 2008); the self-consistency gate checks whether the artifact's declared design attributes support its declared derived quantities. The power gate extracts a single verifier-visible occurrence of each required power attribute and derives RF draw, battery capacity, battery mass, and battery-mass fraction. Fixed tolerances, default losses, and task-specific limits are stored in the task configuration and reproduced in the appendix. The F1 tasks are treated as a calibration surface, not as a difficult benchmark distribution. We do not run a full SysML v2/KerML conformance checker, commercial parser, model-library integration, or

viewpoint-completeness audit; every claim is about this verifier harness.

Agent protocol. All interactive conditions use the same chat-based tool-calling harness. The primary experiments use `gpt-5.5`; the matched sensitivity suite uses `gpt-5.4-mini`. We treat repeated runs as stochastic samples from the hosted endpoint rather than bit-identical replays.

The calibration and cross-family conditions compare two prompt contracts over the same verifier tools. In the advertised-tool arm, the link-budget and self-consistency tools are available but optional. In the prompt-mandated arm, the prompt instructs the model to use both tools before emitting final SysML. The harness records each tool call and result, then continues until final emission or a fixed budget. Repair ablations expose no verifier tools and operate only on frozen failed artifacts.

Code	Operational meaning
P	verifier pass; declared values agree within tolerance
S2	visible fabrication; declared values disagree by > 1 dB
S1	visibility removal; required verification attributes absent
S0	parse failure or API failure
T0	required verifier tool not invoked
T1	tool result honored by the final artifact
T2	tool result overridden by the final artifact
T3	tool called, but final artifact lacks relevant fields

Table 1. Trace-label rubric used by the deterministic coder.

Trace labels. The trace is the object being labeled, not the model's explanation of itself. For each completed run, deterministic post-hoc coders reduce the archived prompts, model messages, tool calls, tool results, extracted artifact, verifier gates, and repair attempts to one artifact-status label and one tool-handling label (Table 1). We discuss latent causes such as memory pollution or premature answer defense only when the trace contains an observable witness, and we compare only cases where the verifier endpoint is unchanged and one declared axis varies. If a trace contains several `compute_link_budget` calls, T1/T2 is assigned against the returned candidate whose derived quantities best match the final artifact within the self-consistency tolerance; exploratory calls are not counted as overrides. Appendix C maps the fired labels to AgentDebug and MAST categories.

Probe families and sample sizes. Each boundary probe changes one source of difficulty while preserving the relevant verifier endpoint. The main `gpt-5.5` suite includes the F1 calibration (50 runs), cross-family engineering transfer (120), scoped-attribute ambiguity with and without canonical scoping (50), non-space OMBSE audit-surface tests (30, plus a 40-run subset audit), and redundant-tool disagreement (100). The scope and OMBSE probes test whether making the verifier-visible surface explicit removes failures without changing the underlying physics or audit target. The redundant-tool probe gives the agent two same-prior link-budget compute paths, one silently biased upward by $+3$ dB, and scores only the values emitted in the final artifact. Exact prompts, seeds, and task variants are in the appendix and trace archive.

Memory and repair probes. The SysML memory suite separates persistence from harm with 18-stage chains. Controls carry `downlink_duty_cycle=0.15`; contaminated chains carry `downlink_duty_cycle=0.60`. Stages 1–17 keep the power gate inactive or preparatory, while Stage 18 activates it. We then freeze the 21 contaminated Stage-18 artifacts that both retained the stale value and failed the late power gate, and apply six text-only repair signals: no feedback, gate name only, masked verifier text, numeric verifier text, trigger-blind quarantine, and trigger-aware quarantine naming `downlink_duty_cycle`. Paraphrase, `gpt-5.4-mini`, and JSON-transfer probes test sensitivity to wording, model family, and representation. The JSON probe uses short operational policies with late audits for warehouse speed, infusion bolus rate, and rail headway; it separates memory-trigger transfer from repair-shape transfer.

Statistical analysis. All inferential calculations use archived coder outputs. For zero-failure exchangeable runs, we report the one-sided exact 95% upper bound $1-0.05^{1/n}$, and we also report task-clustered endpoints when repeated calls share task specifications. Frozen-artifact repair arms are paired by source seed and compared with exact McNemar tests, with Holm correction only inside the declared repair-comparison family. Memory-composition checks use two-sided Fisher exact tests; other cross-probe pattern statements are descriptive unless explicitly pre-registered.

4. Results

4.1. Boundary Probes

Table 2 summarizes the main evidence. The 50/50 faithful-tool calibration is a sanity check because the exposed tools share implementation with the final verifier; the boundary probes carry the claim.

The contract probes show that several failures are surface failures rather than physics failures. Cross-family transfer is 117/120 on `gpt-5.5`; the three failures duplicate power attributes across scopes. Canonical scoping then moves `gpt-5.5` from 10/25 to 25/25 and `gpt-5.4-mini` from 5/25 to 15/25. The non-space OMBSE audit has the same

Probe	Main result
calibration	50/50 `gpt-5.5` pass; T0/T2/T3 all 0; sanity check only
boundary contracts	cross-family 117/120; scope ambiguity 10/25 → 25/25; non-space OMBSE 0/15 → 15/15
redundant tools	both compute tools 100/100; biased path emitted 0/100; link gates 84/100
18-stage memory	controls 10/10 on both models; polluted value 24/24 on both; late failures 21/24 on `gpt-5.5`, 24/24 on mini
SysML repair	numeric text 0/21 on both; trigger-aware quarantine 21/21 on `gpt-5.5`, 19/21 on mini; paraphrases: aware 19–21, blind 12–17
non-SysML transfer	polluted JSON memory 24/24; unsafe final 15/24; all six repair arms 15/15, so repair-shape transfer is negative
pre-registered repair	fresh seeds: trigger-blind quarantine 16/20 vs no-feedback 4/20; exact McNemar $p = 9.16\times10^{-4}$

Table 2. Main boundary and repair results.

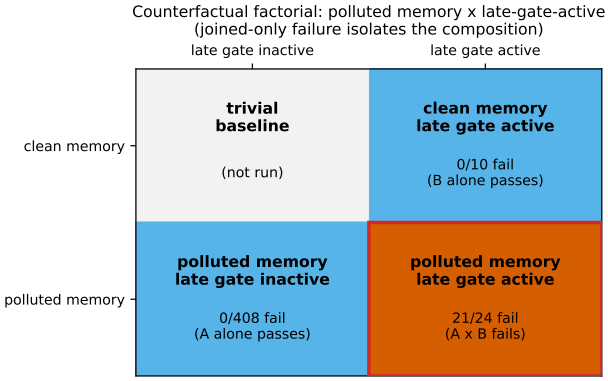

Figure 1. Memory counterfactual.

Repair signal	5.5	5.5 para.	mini
no-feedback re-roll	5/21	–	4/21
gate-name-only	5/21	–	6/21
masked verifier text	1/21	–	9/21
numeric verifier text	0/21	–	0/21
trigger-blind quarantine	18/21	12–17/21	5/21
trigger-aware quarantine	21/21	19–21/21	19/21

Table 3. Frozen-artifact repair with verifier tools withheld.

pattern: plain prompts fail all 15 runs, while scaffolded prompts that declare parser-visible evidence forms pass 15/15. This is evidence for explicit verifier-visible contracts, not for independent industrial MBSE discovery.

The redundant-tool probe gives the agent one unbiased and one silently $+3$ dB-biased link-budget path. Across descriptive and neutral labels, `gpt-5.5` calls both tools in 100/100 runs, emits the biased path in 0/100, and passes link gates in 84/100. The remaining failures are schema, parse, or visibility failures rather than blind trust in the biased compute output. The `gpt-5.4-mini` boundary suite is kept separate: it preserves the memory endpoint but collapses on several SysML/schema surfaces, so it is model-sensitivity evidence rather than a second leaderboard row.

4.2. Memory Contamination and Repair

The strongest trigger is staged memory. A contaminated duty cycle can persist through Stages 1–17 while the power gate is inactive; when Stage 18 activates that gate, polluted memory plus the late gate fails 21/24 on `gpt-5.5` against 10/10 clean late-gate controls (two-sided Fisher exact $p = 2.18\times10^{-6}$). The matched `gpt-5.4-mini` run strengthens the endpoint: 24/24 polluted chains retain the stale value and 24/24 fail the late gate.

We then freeze the 21 failed `gpt-5.5` Stage-18 artifacts and rerun repair with no verifier tools. Table 3 and Figure 2 show the main pattern: numeric verifier text repairs 0/21 on both models, while trigger-aware quarantine repairs 21/21 on `gpt-5.5` and 19/21 on `gpt-5.4-mini`. The middle of the curve is model-sensitive: masked text improves on mini, and trigger-blind quarantine falls from 18/21 to 5/21. The robust claim is therefore not that one prompt shape is universal; it is that full numeric feedback can be actively bad, and that naming the suspect memory field makes structured quarantine much more stable. The statistics and paraphrase sweep support this narrower reading: trigger-aware quarantine separates from trigger-blind quarantine on mini (Holm $p = 1.46\times10^{-3}$), trigger-aware paraphrases remain near ceiling (19–21/21), and a pre-registered fresh-seed comparison gives 16/20 repairs for trigger-blind quarantine versus 4/20 for no-feedback (one-sided exact McNemar $p = 9.16\times10^{-4}$).

A separate non-SysML JSON memory-transfer probe uses warehouse speed, infusion bolus rate, and rail headway policies on `gpt-5.4-mini`. The contamination trigger transfers: polluted values reach the late gate in 24/24 and unsafe final designs appear in 15/24. The repair-shape contrast does not: all six text-only repair arms pass 15/15 on the failed JSON artifacts. We therefore count this as transfer evidence for contamination and a negative cross-domain result for the structured-quarantine repair shape.

Cost and trade-offs. The interventions also change process cost (Table 4). Canonical scoping is cheaper than ambiguous scoping in this harness because it removes repeated repair attempts. The memory chain is expensive by construction, while text-only repair uses no verifier calls; numeric verifier text and trigger-aware quarantine spend comparable tokens but have opposite pass rates. We do not measure human review or downstream rework cost.

4.3. Faithful-Tool Calibration

The calibration result is intentionally demoted. Both prompting arms pass all runs: 25/25 advertised-tool and 25/25 prompt-mandated, with T0=0, T1=50, T2=0, and

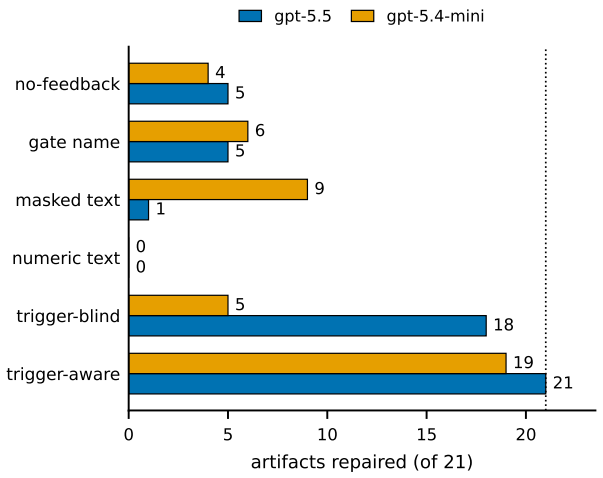

Figure 2. Text-only repair on the same 21 failed SysML artifacts.

Probe / arm	Tool calls	Tokens
neutral redundant-tool run	4.44	13.9k
scope ambiguity, ambiguous	2.84	18.6k
scope ambiguity, canonical	2.16	13.2k
`gpt-5.5` contaminated chain	37.1	331.6k
repair, numeric verifier text	0	5.39k
repair, trigger-aware quarantine	0	6.28k

Table 4. Representative process costs.

T3=0. This establishes that the harness can observe faithful tool handling under an easy shared oracle; it is not evidence that the same behavior survives schema ambiguity, missing audit evidence, multi-tool disagreement, stale memory, or zero-tool repair. The prompt-mandated arm adds no measurable gain here, but because both arms use `tool_choice="auto"`, this is not a claim about API-level tool-use enforcement.

5. Discussion

The main methodological point is separation. Calibration, boundary discovery, and repair are different evidentiary objects. A clean calibration shows that the tool loop can work under an easy shared oracle; it does not imply that the same agent will remain faithful under schema ambiguity, missing audit evidence, multi-tool disagreement, stale memory, or a zero-tool repair budget. The boundary probes are therefore the main evidence, and the repair ablations answer a narrower follow-up question: once a failed artifact is frozen, which textual signals move that specific boundary when verifier tools are withheld?

The clearest boundary is the memory chain. It appears in both tested OpenAI model families and is isolated by the counterfactual composition check: neither polluted memory before the late gate nor the late gate with clean memory is sufficient, while their combination fails. The clearest negative result is the repair-transfer boundary. The contamination trigger transfers to JSON operational artifacts, but the SysML structured-quarantine repair contrast does not, because the JSON repair surface is a local threshold check rather than a coupled physics chain. The smaller-model suite adds the same caution from a different angle: it preserves the memory endpoint while breaking several schema-surface probes. These results support a boundary map, not a model ranking or a domain-general repair claim.

The repair results also constrain feedback design. Simply showing more verifier text is not uniformly helpful: numeric verifier feedback repairs 0/21 frozen SysML failures on both models. The successful structured prompts instead change the editing operation by quarantining early operations memory, preserving interface values, and re-deriving the smallest feasible power-subsystem edit. The paraphrase sweep suggests that this shape matters beyond one exact wording, especially in the trigger-aware arm, but the JSON null result shows that the shape is meaningful only when the failure surface has enough coupled structure for that operation to act on.

The reproducibility stance is artifact-first rather than bit-identical replay. Hosted model behavior can change behind a stable model string, and seeds are at best sampling hints. The auditable objects are the archived traces, frozen failed artifacts, result JSONs, deterministic post-hoc coders, and scripts that regenerate tables and figures. What transfers most directly is the protocol: freeze the artifact, expose the trace, state the verifier gate, vary one axis, and report both endpoint and process metrics.

6. Threats to Validity

The main construct threat is the shared oracle: the calibration uses the same physics implementation for the exposed tool and final verifier, so it measures agent-side fidelity rather than tool correctness. The boundary probes reduce but do not remove that threat because the same repository still owns the parser, verifier, and audit gates. A commercial SysML v2/KerML parser, richer requirements traceability, behavioral verification, and external analysis tools could expose different failure modes. Bugs in the verifier implementation would also be high-impact, which is why the claim rests on stored artifacts and re-runnable post-hoc coders rather than on prose summaries alone.

The SysML surface itself is a separate limitation. The repository verifier reads a constrained textual subset and normalizes some model text before physics checking; it is not a proof that raw generated artifacts are accepted by the SysML v2 Pilot Implementation or by commercial tools. In pilot conformance checks, raw artifacts can fail official syntax

even when the subset verifier can extract the fields needed for physics gates. We therefore interpret every physics result as a claim about verifier-visible textual artifacts under this harness, not about full SysML v2 conformance. This limitation is orthogonal to the agentic failure question: the paper is about trace-level verifier handling, not industrial MBSE validity.

External validity is narrow. The primary model is `gpt-5.5`, the matched stress suite covers only `gpt-5.4-mini`, and open-weight, Claude, and Gemini models are absent. The smaller model is useful because it breaks some schema surfaces while preserving the memory endpoint, but two OpenAI families are not a model ecosystem. The tasks are engineering mini-domains, and the non-SysML JSON transfer probe has only three domains and 30 chains. It reduces domain narrowness for the contamination trigger but not for the repair-shape effect.

The memory chain is also synthetic. It is an 18-stage unit-test-style trigger with a handoff field designed to persist until the late gate activates; it is not evidence about natural failure rates in hundreds-step autonomous workflows. Its value is counterfactual isolation, not deployment realism. Statistical claims are similarly bounded: most boundary rows are descriptive; inferential claims are limited to declared repair-comparison families and one pre-registered fresh-seed replication. Finally, verifier pass is not engineering acceptance. Human review cost, guardrail-induced task degradation, PLM workflows, configuration management, and commercial MBSE-tool integration remain outside the measurement.

7. Conclusion

We present a trace-level failure-boundary protocol for verifier-augmented engineering agents. The calibration shows that faithful tool use is observable; the boundary probes show where it breaks; and the frozen-artifact repair ablations show which interventions move one memory-contamination boundary under a zero-tool budget. The result is a reproducible boundary map, not a deployment claim or a general repair guarantee. The same protocol can be reused with stronger parsers, hosted tools, broader model families, and longer workflows.

References

Andriushchenko, M., Souly, A., Dziemian, M., Duenas, D., Lin, M., Wang, J., Hendrycks, D., Zou, A., Kolter, Z., Fredrikson, M., Gal, Y., and Davies, X. Agentharm: A benchmark for measuring harmfulness of llm agents. In *International Conference on Learning Representations*, 2025.

Bang, Y., Ji, Z., Schelten, A., Hartshorn, A., Fowler, T., Zhang, C., Cancedda, N., and Fung, P. Hallulens: Llm hallucination benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24128–24156, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1176.

Barke, S., Goyal, A., Khare, A., Singh, A., Nath, S., and Bansal, C. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

Chen, M., Wang, J., Mu, F., Wang, Y., Liu, Z., Feng, H., and Wang, Q. Seeing the whole elephant: A benchmark for failure attribution in llm-based multi-agent systems. *arXiv preprint arXiv:2604.22708*, 2026a. Accepted by ACL 2026.

Chen, Y., Jiang, J., Liu, J., Zhang, Y., Guo, X., and King, I. Trace: Trajectory-aware comprehensive evaluation for deep research agents. *arXiv preprint arXiv:2602.21230*, 2026b. Accepted by WWW 2026.

Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M., and Tramer, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Doshi, A., Hong, Y., Xu, C., Kang, E., Kapravelos, A., and Kästner, C. Towards verifiably safe tool use for llm agents. In *International Conference on Software Engineering, New Ideas and Emerging Results Track*, 2026.

Guo, X. et al. Toward engineering agi: Benchmarking the engineering design capabilities of llms. *arXiv preprint arXiv:2509.16204*, 2025. To appear in NeurIPS 2025 Datasets and Benchmarks Track.

Gupta, A. Reliabilitybench: Evaluating llm agent reliability under production-like stress conditions. *arXiv preprint arXiv:2601.06112*, 2026.

Gurram, B. Evaluating tool-using language agents: Judge reliability, propagation cascades, and runtime mitigation in agentprop-bench. *arXiv preprint arXiv:2604.16706*, 2026.

Hui, M., Li, X., Wang, L., Wan, C., Wang, Y., Wang, Y., Song, F., Shi, B., Li, Y., and Li, Y. Flare: Agentic coverage-guided fuzzing for llm-based multi-agent systems. *arXiv preprint arXiv:2604.05289*, 2026.

Jia, J., Deng, Z., Chen, Z., Wang, Y., and Zheng, Z. Mas-fire: Fault injection and reliability evaluation for llm-based multi-agent systems. *arXiv preprint arXiv:2602.19843*, 2026.

Jin, D., Jin, Z., Li, L., Fang, Z., Li, J., and Chen, X. A system model generation benchmark from natural language requirements. *arXiv preprint arXiv:2508.03215*, 2025.

Kadavath, S. et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kokane, S., Zhu, M., Awalgaonkar, T., Zhang, J., Prabhakar, A., Hoang, T., Liu, Z., Murthy, R. R. N., Yang, L., Yao, W., Tan, J., Liu, Z., Niebles, J. C., Wang, H., Heinecke, S., Xiong, C., and Savarese, S. Toolscan: A benchmark for characterizing errors in tool-use llms. In *ICLR 2025 Workshop on Building Trust in LLMs and LLM Applications*, 2025.

Lanham, T. et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Li, Z., Husung, S., and Wang, H. Llm-assisted semantic alignment and integration in collaborative model-based systems engineering using sysml v2. In *2025 IEEE International Symposium on Systems Engineering (ISSE)*, 2025. doi: 10.1109/ISSE65546.2025.11369983.

Liu, Y., Zhang, C., Han, Z., Liu, H., Wang, Y., Yu, Y., Wang, X., and Yin, Y. Trajad: Trajectory anomaly detection for trustworthy llm agents. *arXiv preprint arXiv:2602.06443*, 2026.

Ma, X., Xie, X., Wang, Y., Wang, J., Wu, B., Li, M., and Wang, Q. Diagnosing failure root causes in platform-orchestrated agentic systems: Dataset, taxonomy, and benchmark, 2025. URL https://openreview.net/forum?id=Gc1zx6G4HP. OpenReview manuscript, withdrawn ICLR 2026 submission.

OMG Systems Modeling Language (SysML), Version 2.0. Object Management Group, September 2025. URL https://www.omg.org/spec/SysML/2.0/.

OWASP GenAI Security Project. Owasp top 10 for agentic applications 2026, December 2025. URL https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/. Resource page: OWASP Top 10 for Agentic Applications for 2026.

Proakis, J. G. and Salehi, M. *Digital Communications*. McGraw-Hill, 5th edition, 2008.

Shi, M., Mokotoff, P., and Cinar, G. Model-based systems analysis and engineering: The development of enhanced model-based systems analysis (mbsa) and model-based systems engineering (mbse) couplings for practical applications. Technical Report NASA/CR-20250007048, NASA Langley Research Center, August 2025.

Simonds, T. Llms for engineering: Teaching models to design high powered rockets. *arXiv preprint arXiv:2504.19394*, 2025.

Sklar, B. *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2nd edition, 2001.

SysML.org. Sysml specifications: Current version - omg sysml v2.0, 2025. URL https://sysml.org/sysml-specs/.

Systems Modeling Community. Sysml v2 pilot implementation, 2026a. URL https://github.com/Systems-Modeling/SysML-v2-Pilot-Implementation.

Systems Modeling Community. Omg systems modeling language (sysml) v2 release, 2026b. URL https://github.com/Systems-Modeling/SysML-v2-Release.

Turpin, M., Michael, J., Perez, E., and Bowman, S. R. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, 2023.

Wang, H., Poskitt, C. M., and Sun, J. Agentspec: Customizable runtime enforcement for safe and reliable llm agents. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 1–12, 2026.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Zhan, Q., Liang, Z., Ying, Z., and Kang, D. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024. doi: 10.18653/v1/2024.findings-acl.624.

Zhu, K. et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

A. Detailed Threats to Validity

T1, shared oracle. The faithful-tool calibration uses the same physics module for tool output and final verification, so it is a fidelity sanity check, not tool-correctness evidence. **T2, model coverage.** The main probes use the recorded `gpt-5.5` configuration. A matched `gpt-5.4-mini` suite adds F1, cross-family, operational-MBSE, scope-ambiguity, redundant-tool, memory-chain, and frozen-artifact repair stress tests. It preserves the strongest contamination endpoint (control 10/10 pass; polluted late-gate failure 24/24), but it also exposes substantial model sensitivity: cross-family tasks pass 89/120, the operational-MBSE subset passes 2/40, and redundant-tool variants pass link gates 0/100 because schema-surface failures dominate before trust failures can appear. This is a boundary map across two OpenAI families, not a ranking of model families and not evidence about open-weight, Claude, or Gemini systems. **T3, task distribution.** The tasks remain spacecraft-heavy and use intentionally minimal triggers; the non-SysML JSON memory-transfer probe reduces this concern but has only three domains and 30 chains, and the cross-domain six-arm repair saturates at 15/15 so transfer evidence covers the contamination trigger only, not the structured-quarantine shape effect. This improves reproducibility but risks trigger overfitting. **T4, sample size.** Run-level exact bounds are descriptive because repeated calls share task specifications; task-clustered bounds are wider. **T5, tool descriptions.** Sparse, ambiguous, or adversarial tool schemas could reintroduce non-invocation or override; a 50-run sparse-tool-description ablation on `gpt-5.5` that strips the `compute_link_budget` description to a one-sentence stub passes 50/50 on F1 with 0/50 S1 and 0/50 T3, so this trigger is weak on `gpt-5.5` at this scale and is not a substitute for adversarial-schema or open-weight tests. **T6, hosted tools.** In-process function calling omits serialization, authorization, transport, and prompt-injection boundaries. **T7, multi-tool conflict.** The redundant-tool probe covers one compute conflict, not a broad ecology of stale or adversarial tools. **T8, loss model.** The environmental loss budget is defensible but not unique; alternative budgets change magnitudes.

T9, SysML v2 surface. The primary harness uses a subset parser rather than commercial-tool conformance. **T10, verifier scope.** Numeric gates and scaffolded raw-text OMBSE audit surrogates do not establish unconstrained MBSE modeling competence. **T11, iterative authoring and chain length.** The 18-stage SysML memory chain and 8-stage JSON transfer probe are not real library patch/merge workflows under PLM or configuration management, and the chains are unit-test instantiations of a chain-length axis along which the protocol is intended to compose; they are designed to expose contamination-onset, not to claim hundreds-step deployment-length operation. **T12, engineering acceptance.** Gate pass is not human review acceptance; guardrail-induced task degradation and rework cost remain unmeasured. **T13, partial external trace-schema cross-walk.** The paper records traces in its own event schema and codes the S/T axes deterministically. We provide a definitional cross-walk to AgentDebug (Zhu et al., 2025) and MAST (Cemri et al., 2025) categories in Appendix C (Table 6) so each fired S/T endpoint and each trigger family is named under at least one external schema. We do not re-code the underlying 446 traces under AgentDebug, AgentFail (Ma et al., 2025), AgentRx (Barke et al., 2026), MAST, TraceElephant (Chen et al., 2026a), TRACE (Chen et al., 2026b), or TrajAD (Liu et al., 2026), and we do not claim inter-rater agreement with any of these schemas. The cross-walk is therefore a schema-level correspondence, not a recoded benchmark, and the rubric should still be read as a within-paper coding instrument until a successor study performs a per-trace recoding. The recorded run/task/seed/arm, event-index, actor, event-type, tool-name, input/output hashes, gate, and status fields remain a strict superset of what the cross-walk maps need.

T14, paper-wide multiple-comparisons control. Inferential statistics in this paper are limited to two declared families: (a) the within-family Holm correction over the five repair-arm contrasts on the original n=21 frozen artifacts, and (b) a single pre-registered confirmatory McNemar comparison on a disjoint fresh seed range (`paper/preregistration_repair_replication.md`, locked at git tag `preregistration-2026fmai`). All other probe results in Table 2 are reported as descriptive failure-boundary characterizations; we do not assert paper-wide family-wise error control across the boundary suite, and cross-probe pattern statements ("model is fragile under X but robust under Y") are not claims of joint statistical significance. The pre-registered confirmation is therefore the only probe whose inferential claim is robust to a paper-wide multiple-comparisons challenge by construction, since its inferential family was declared as a single comparison before data were collected.

B. Boundary and Repair Probes

The boundary and repair runs are separate from the 50-run F1 calibration. They test where the clean result stops holding and whether the resulting failure reports are actionable.

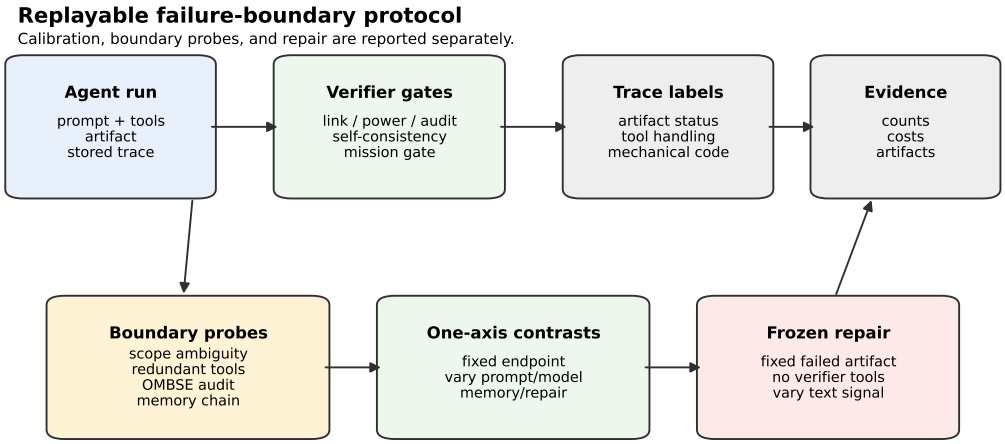

Figure 3. Replayable protocol separating calibration, boundary probes, one-axis contrasts, and frozen-artifact repair.

We first added three boundary tasks: `bp_power_01` adds a coupled `power_budget` gate; `bp_xband_01` transfers to X-band high-rate downlink; and `bp_caps_01` places the design near mission caps. We run 20 seeds per arm for each task (120 runs). This cross-family extension passes 117/120. All three failures occur in `bp_power_01`: the artifact passes link, self-consistency, and mission constraints but fails `power_budget` because power attributes such as `pa_efficiency` and `downlink_duty_cycle` are duplicated across spacecraft and transmitter scopes, making power extraction ambiguous.

We then scale that observation into five scoped-attribute trigger variants: duty-cycle operations memory versus power, PA efficiency component versus power, transmitter power RF versus power, payload mass versus spacecraft mass, and receiver noise versus ground-station noise. Ambiguous prompts pass 10/25 for `gpt-5.5`; canonical scoping passes 25/25. The matched `gpt-5.4-mini` stress test is weaker but directionally consistent: ambiguous prompts pass 5/25 and canonical scoping passes 15/25, so the contract helps but does not close the boundary on the smaller model.

The nontrivial tool-trust probe in §4.1 uses redundant compute paths rather than a single biased service. This makes the failure boundary behavioral: the agent can compare a biased primary service against an independent cross-check, and the trace reveals whether it detects, ignores, or mis-resolves the disagreement.

Finally, `ombse_01` adds a scaffolded operational-MBSE audit: requirement declarations with `satisfy`/`verify` chains, port/item-flow compatibility, allocation completeness, a state/transition behavior graph, patch evidence against a named baseline model, and configuration/change-control IDs. The task prompt supplies the expected evidence forms, so this is closer to textual audit-evidence copying/completion than unconstrained MBSE synthesis. It is a subset audit, not Cameo, Capella, Rhapsody, Pilot Implementation, or PLM integration. Across 20 seeds per arm, 34/40 runs pass all gates. The six failures are trace-local: one behavior graph references a state that is not declared, one artifact omits a complete requirement–satisfy–verify chain, and four artifacts leave allocations incomplete.

To test whether this is only a spacecraft notation artifact, we add three non-space OMBSE domains: warehouse robotics, infusion-pump operation, and rail signaling. Plain generation passes 0/15. A scaffolded audit-surface contract that supplies parser-compatible metric attributes and exact requirement/interface/ allocation/behavior/patch/configuration evidence forms passes 15/15. This is best interpreted as evidence that the audit surface is portable, not that the agent independently models those domains.

We also add a non-SysML JSON transfer probe for the memory-contamination shape. The domains are warehouse robot aisle-speed policy, hospital infusion-pump bolus rate policy, and rail-signaling headway policy. Each chain has eight JSON artifact stages: seven handoff stages where the safety gate is inactive, then a late safety audit. With `gpt-5.4-mini`, controls pass 6/6. Contaminated chains carry the polluted value to the late gate in 24/24 (eight contaminated seeds per domain) and fail 15/24 final audits: warehouse 8/8, infusion 6/8, rail 1/8. The rail result is informative because the value is carried but the model self-corrects in 7/8 of those chains before the final gate. The full six-arm text-only repair ablation

(no-feedback, gate-name-only, masked verifier text, numeric verifier text, trigger-blind structured quarantine, trigger-aware structured quarantine) saturates at 15/15 across every arm. The OMBSE repair surface is a single $\leq/\geq$ check against one threshold, which is too local for the structured-quarantine *shape* contrast (originally 18/21 vs 5/21 on SysML) to discriminate. We therefore use this probe only as domain-transfer evidence for contamination, and report a *negative* cross-domain transfer result for the repair-shape effect: the trigger transfers across domain and base model, but the shape-vs-no-feedback gap that holds in the coupled-physics SysML setup does not transfer to a single-comparator OMBSE surface.

Memory contamination and repair ablation. The current memory suite uses 18 artifact stages per chain. In the contaminated condition, `downlink_duty_cycle = 0.60` reaches the late Stage 18 power gate in 24/24 chains and causes `power_budget` failure in 21/24; controls pass 10/10. Three contaminated chains pass despite retaining the polluted value, which keeps the trigger from being a deterministic tautology. A matched `gpt-5.4-mini` rerun of the same 34 chains is stricter: controls again pass 10/10, but polluted chains fail the late power gate in 24/24. We report this as a model-sensitivity strengthening of the contamination endpoint, not as evidence about natural deployment rates. The counterfactual factorial (polluted memory $\times$ late-gate-active) is shown in Figure 1: only the joined condition fails, isolating the composition as the trigger. We then freeze the 21 failed artifacts and run six text-only repair arms with no verifier tools: no-feedback re-roll, gate-name-only, masked verifier text, numeric verifier text, structured trigger-blind quarantine, and trigger-aware structured memory quarantine. Pass counts are 5/21, 5/21, 1/21, 0/21, 18/21, and 21/21. This is useful as a stricter repair result: the failure boundary is reproducible, no-feedback re-rolls are not enough, and structured quarantine helps even when it does not name the suspect field or threshold. The archived CFP audit (`paper/results/cfp_alignment_results.json`) records the corresponding primitive-composition table and repair cost / failure-gate side effects.

The two structured quarantine prompts are intentionally disclosed because they are interventions, not a learned general repair policy. The trigger-blind arm uses the following prompt.

Trigger-blind quarantine prompt.

```
Structured systems intervention: before editing, quarantine any early
operations-memory or bookkeeping field that may have been recorded before
the currently failing verifier gate existed. Use the mission specification
and masked verifier feedback to identify the stale power-subsystem input.
Preserve mission interface values unless the mission specification
explicitly requires a change. Then re-derive the smallest feasible
power-subsystem edit and update only directly derived power bookkeeping
values.
```

The trigger-aware arm adds the suspect field and target range:

Trigger-aware quarantine prompt.

```
Structured systems intervention: quarantine early operations-memory fields
before editing. Treat `downlink_duty_cycle` as suspect because it was
recorded before the power gate existed. Keep `pa_efficiency`,
`bus_avionics_power_W`, `depth_of_discharge`, and
`battery_specific_energy_Wh_per_kg` at their existing interface values
unless a mission constraint explicitly says otherwise. Re-derive a feasible
`spacecraft.power.downlink_duty_cycle` at or below 0.20, then update only
directly derived power bookkeeping values.
```

This wording is why the repair claim is restricted to finite-sample interventions on frozen artifacts, not presented as a domain-general memory repair mechanism.

Paraphrase ablation

To distinguish wording from shape, we ran five shape-preserving paraphrases of each quarantine prompt against the same 21 frozen artifacts (`paper/results/text_repair_ablation_paraphrase_results.json`). Each paraphrase preserves the "quarantine early operations-memory field $\rightarrow$ re-derive smallest feasible power-subsystem edit" structure but varies surface phrasing (procedure-style, two-step, suspect-list, workflow, re-statement; full text in `scripts/run_text_repair_ablation.py`). The trigger-aware paraphrases pass 19, 20, 20, 21, 21 / 21 (mean 20.2, SD 0.84), each paired-McNemar Holm-equivalent to the original 21/21 (Holm-adjusted $p = 1.0$). The trigger-blind

paraphrases pass 12, 13, 14, 14, 17 / 21 (mean 14.0, SD 1.87); none cross Holm-adjusted $\alpha = 0.05$ against the original 18/21, but the original sits on the high end of the wording distribution. All five trigger-blind paraphrases beat no-feedback (5/21) by $\geq$7 cases and uniformly beat masked-verifier text (1/21) and numeric-verifier text (0/21). The trigger-aware shape is paraphrase-robust; the trigger-blind shape is qualitatively but not quantitatively wording-invariant. The headline "shape carries the effect" claim should therefore be read as: structured-quarantine *shape* robustly beats unstructured feedback under this budget, but the specific 18/21 trigger-blind point estimate is wording-driven.

Trace excerpt. The contaminated seed 0 trace localizes the failure without relying on post-hoc narrative. The early polluted value remains irrelevant until the late power gate appears; text-only repair then separates unstructured feedback from the structured quarantine intervention.

```
stage 1..17: downlink_duty_cycle = 0.60, power gate not yet rejecting it
stage 18 power_gate: downlink_duty_cycle = 0.60
stage 18 power_budget: FAIL in 21/24 contaminated chains
gpt-5.4-mini matched rerun: FAIL in 24/24 contaminated chains
text repair: no_feedback 5/21; gate_only 5/21; masked 1/21;
             numeric 0/21; trigger_blind 18/21;
             trigger_aware_quarantine 21/21
```

Probe	Condition	n	pass	Failure gate
F1 calibration	seeds 0–4 per arm/task	50	50	–
cross-family	power / X-band / caps	120	117	power
operational MBSE	trace/interface/allocation/behavior	40	34	mixed
scope ambiguity	`gpt-5.5`, ambiguous vs canonical	50	35	power / schema
non-space OMBSE	plain vs scaffolded	30	15	audit gates
redundant tool	descriptive labels	50	42	schema / vacuity
redundant tool	neutral A/B labels	50	42	schema ambiguity
memory contamination	18 stages, 10 control + 24 polluted	34	13	late power
text-only repair	six arms over 21 failed artifacts	126	50	mission / power
paraphrase ablation	10 paraphrases over 21 failed artifacts	210	171	mission / power
`gpt-5.4-mini` memory chain	18 stages, 10 control + 24 polluted	34	10	late power
`gpt-5.4-mini` text repair	16 arms over 21 failed artifacts	336	169	mission / power
non-SysML memory transfer	8 stages, 6 control + 24 polluted	30	15	late safety
non-SysML repair null	six arms over 15 failed JSON artifacts	90	90	–

Table 5. Boundary and repair runs used in the current claim.

C. External Trace-Schema Cross-Walk

Table 6 records the definitional correspondence from each fired S/T endpoint and each trigger family to AgentDebug (Zhu et al., 2025) and MAST (Cemri et al., 2025) categories. AgentDebug categories follow the trajectory-level error taxonomy (tool selection, tool argument, tool-result handling, planning/decomposition, memory, and termination errors). MAST categories follow the multi-agent-system failure taxonomy (specification & system-design, inter-agent misalignment, and task-verification failures); because our setting is single-agent, MAST inter-agent-misalignment categories collapse to the single-agent slice that maps stage-to-stage handoff in the memory chain. The cross-walk is at the schema level: it names where each S/T label and trigger would land if the same trace were re-coded under the external schema. We do not re-code the underlying 446 traces and do not report inter-rater agreement, so this closes the comparable-protocols gap at the schema-correspondence level only (§6 T13).

Our endpoint / trigger	AgentDebug category	MAST category
T0 (tool not invoked)	tool-selection error (omission)	specification: missing verification step
T1 (result honored)	success path; n/a	success path; n/a
T2 (result overridden, >1 dB)	tool-result-handling error (override)	task-verification failure (verifier ignored)
T3 (call without relevant attrs)	tool-result-handling error (misapplication)	specification: artifact–tool decoupling
S1 (visibility removal)	planning / termination error (premature emit)	specification: missing verification artifact
S2 (visible fabrication, >1 dB)	final-answer / planning error	task-verification failure (false-positive emit)
scope-ambiguity trigger	tool-argument error (ambiguous schema)	specification & system-design failure
redundant-tool conflict	tool-selection / tool-result-handling under conflict	task-verification failure (cross-check)
non-space OMBSE audit gap	tool-argument error (evidence-form mismatch)	specification & system-design failure
memory-contamination chain (counterfactual factorial)	memory error (stale handoff)	inter-stage handoff failure (single-agent slice of inter-agent misalignment)
text-only repair under withheld tools	planning error under reduced action budget	task-verification failure under tool-budget collapse

Table 6. Schema-level cross-walk to AgentDebug and MAST categories.

D. Verifier Implementations

The executable verifier definitions are in `speccraft/verifier.py`. The link gate extracts exactly one value for each required link-design attribute, computes slant range from Earth radius 6378.137 km, computes Friis path loss with $c = 2.99792458 \times 10^8$ m/s, uses -198.6 dBm/Hz as the Boltzmann noise-density constant at 1 K, and applies the environmental losses and required margin from the task configuration. The self-consistency gate recomputes the same derived quantities and checks any declared candidate quantities with the tolerances stated in §3; it fails artifacts with no declared derived link-budget quantity. The power gate derives transmitter RF power, DC draw, per-orbit energy, battery capacity, battery mass, and battery-mass fraction from the artifact's declared power attributes and the configured orbit period and mass fraction cap. The operational-MBSE gates are regex-based raw-text subset audits for requirement traceability, interfaces, allocation, behavior, patch/merge evidence, and configuration IDs.

E. Per-Task Designs

F1 instances 01–05: NL specs, parametric ranges, reference designs, and closure margins. Source: `tasks/f1_01..f1_05`.