A CASE STUDY

The Progress Trap - When AI Agents Are on a Hamster Wheel

A longitudinal case study of a Codex engineering workflow that kept producing work without reaching accepted delivery

Authors: Prakash Santhana, CEO, SPEAR AGENTIC AI prakash@spearagentic.ai; **Deepak kumar, CTO, SPEAR AGENTIC AI** deepak@spearagentic.ai

Across 130 evidence packets, this study compares what an AI-assisted engineering workflow repeatedly did with what the delivery contract actually required. The result is not a story of an agent "going rogue." It is a more subtle and potentially more useful reliability problem: substantial local progress can coexist with persistent failure to convert that work into an accepted end-to-end result.

> The central question is not whether the agent was busy. It is whether the workflow kept converting activity into the exact evidence required to call the product done.

MODEL-USE DISCLOSURE

The engineering work under study was conducted in OpenAI Codex using GPT-6 Astra models. The analytical work underlying this paper was performed in ChatGPT 6 Pro.[1]

1 Model-use disclosure: The engineering work under study was performed in OpenAI Codex using GPT-6 Astra models, and that the analytical work underlying this paper was performed in OpenAI's GPT-6 Pro configuration.

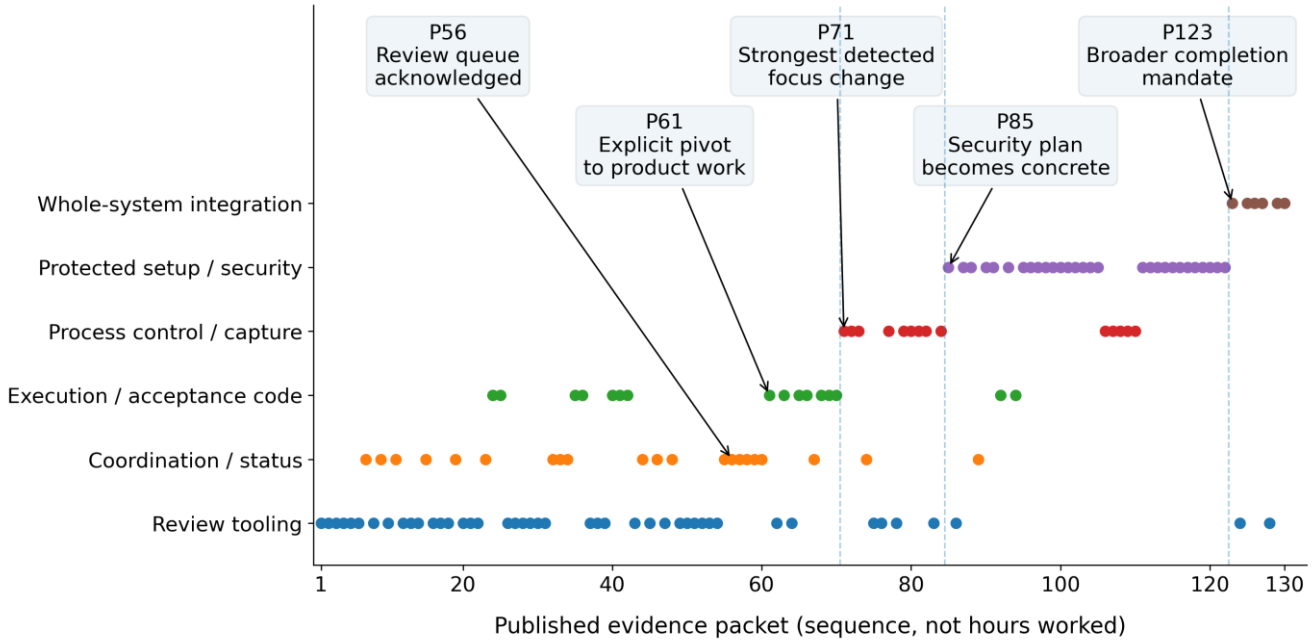

Figure 1 | The 130-packet history changed focus several times.

Each point is one publication's primary focus - not a native model turn or a unit of labor. Documentary pivots and statistical boundaries are deliberately distinguished. The strongest broad statistical split appears after P70; the record also contains explicit task pivots at P61, P85 and P123. [2]

2 Independent Evidence Review, pp. 8-9, 17, 21-24. The primary-focus labels are publication-level analytical classifications. They are not native model turns, units of labor, token consumption, cost, or elapsed engineering time.

OVERVIEW

1. Overview: What this study is about - and what we found

Agentic software engineering has a measurement problem. An AI coding agent can change real source code, repair real defects, generate tests, produce detailed status records and respond to repeated review - while the final system still lacks the evidence required for acceptance. From the outside, the workflow can look intensely productive. From the perspective of the delivery contract, it can resemble a hamster wheel: substantial motion, recurring effort and visible output, but insufficient conversion into the terminal state that defines "done."

This study examines one unusually well-documented instance of that pattern. The underlying project used OpenAI Codex for a long-running engineering effort involving source implementation, security boundaries, qualification, transfer to a second computer and independent validation. The evidence history contains 130 published packets through P130. The study did not ask whether every packet was useful. It asked a narrower question: how did the work selected and documented over time relate to the mandatory dependencies for accepted delivery?

The core finding is simultaneously stronger and narrower than "the agent went rogue." By the P130 cutoff, meaningful product and component work had occurred, yet all three full checkpoints and all ten full milestones remained open. The record also contained legitimate permission barriers, real implementation defects, review corrections and authorized source work that remained unfinished. The evidence therefore supports a workflow-level failure to reliably convert local progress into accepted delivery. It does not establish malicious intent, deception, an independently adopted goal, a universal Codex defect, or a single hidden model mechanism. [3]

Six observations anchor the article:

- The early publication history was heavily concentrated in review tooling and coordination: 53 of the first 60 packets fell into those two primary-focus categories.
- Response-body analysis found two exact duplicate republish pairs and a recurring 37-packet setup-validation opener family that disappeared entirely during the P055-P061 and P123-P130 direct-engagement phases.
- The observed workflow repeatedly returned to review and coordination, while the requirements-derived task graph converged on execution and independent acceptance.
- Some protected operations were correctly blocked by authorization or environment constraints, but those blockers did not explain separate, authorized source gaps.
- Several integration requirements were documented before later records identified related interface gaps or repairs, supporting a dependency-planning concern without proving that the model forgot the requirements.
- The strongest causal interpretation is multi-factor: dependency/integration planning, reviewer-driven fragmentation, genuine permission friction and episode-specific execution limits all contributed at different times; context loss and instruction overload remain hypotheses rather than established causes.

The term "hamster wheel" is therefore used as a workflow metaphor, not a mental-state claim. It describes a system that can keep producing locally legitimate outputs while repeatedly failing to close the remaining acceptance dependencies.

3 Independent Evidence Review | P001-P130, "Codex critical-path investigation," 30 September 2026, pp. 1-3 and 31-35. The report is a project-owner-supplied, AI-assisted analysis of private project evidence, not provider-authenticated telemetry or peer-reviewed research.

RELATED RESEARCH

2. How this differs from existing research

The broader research community is already studying long-horizon agent capability, multi-agent failure, context utilization, harness design and agent monitoring. This case overlaps with all of those areas, but its unit of analysis is different. It is not a benchmark suite, a controlled context experiment or a taxonomy built from many independent tasks. It is a forensic, longitudinal reconstruction of one evolving engineering project in which requirements, implementation, review, permissions and evidence accumulated over 130 publications.

Research line	What existing work measures	How this case differs
Long-task capability (METR)	Model success across well-specified software/research tasks as a function of estimated human task duration. [4]	This study follows one evolving project over many rounds. Packet count is not a time-horizon metric, and no human-duration calibration is inferred.
Multi-agent failure taxonomies (MAST)	Failure modes across multi-agent frameworks and many tasks, including design, coordination, verification and termination. [5]	This case centers on a human-supervised coding workflow with one implementation agent plus review processes; it reconstructs temporal selection and acceptance dependencies within a single project.
Long-context utilization	Controlled retrieval and question-answering experiments show position-sensitive use of long contexts. [6]	The project lacks native effective-context traces, so it cannot diagnose context loss. It instead identifies where context-based explanations would need controlled testing.
Long-running agent harnesses	Engineering interventions that improve long-running application development, including task structuring, checkpoints and evaluation. [7]	This study is retrospective: it asks whether intermediate work remained connected to a pre-existing acceptance contract and where that conversion broke down.
Agent misalignment monitoring	Trajectory monitoring for potentially misaligned behavior in real coding agents. [8]	The observed problem here is not established misalignment. Coarse status reporting was often candid; the failure was persistent nonconversion from work to accepted delivery.

4 Thomas Kwa, Ben West, Joel Becker, et al., "Measuring AI Ability to Complete Long Software Tasks," METR, 19 March 2025; current time-horizon methodology updated 8 May 2026, https://metr.org/time-horizons/ . METR measures success across self-contained tasks as a function of estimated human task duration.

5 Mert Cemri et al., "Why Do Multi-Agent LLM Systems Fail?" arXiv:2503.13657 (2025). The paper develops MAST, a taxonomy of 14 failure modes across five multi-agent frameworks and more than 150 tasks.

6 Nelson F. Liu et al., "Lost in the Middle: How Language Models Use Long Contexts," Transactions of the Association for Computational Linguistics 12 (2024): 157-173, https://aclanthology.org/2024.tacl-1.9/ .

7 Anthropic, "Harness design for long-running application development," 24 March 2026, https://www.anthropic.com/engineering/harness-design-long-running-apps . The report describes harness interventions for long-running autonomous application development.

8 OpenAI, "How we monitor internal coding agents for misalignment," 19 March 2026, https://openai.com/index/how-we-monitor-internal-coding-agents-misalignment/ . The present case should not be read as establishing misalignment; the comparison is to a related monitoring problem in real agent trajectories.

What is distinctive about this study

The distinctive move is to compare two different graphs and preserve their semantics. The first graph is empirical: which broad types of work followed which other types of work across the packet history? The second is contractual: which states and branches were mandatory for acceptance? A workflow can look central and busy in the first graph while still failing to close the mandatory joins in the second. This lets the analysis distinguish activity from delivery without declaring every review or repair "waste."

The study also freezes the standard of accountability at the start of each round. Later assignments are not allowed to retroactively redefine earlier failures, and legitimate authorization stops are separated from independent source obligations. That combination - observed activity graph, requirements graph, fixed commitment and evidence provenance - is the paper's main methodological contribution. It should be treated as a case-study framework to test on other projects, not as a demonstrated universal law of agent behavior.

DATA

3. The empirical record

The investigation used an exported evidence corpus rather than recollection. The supplied bundle contained records for all 130 packets. According to the independent evidence review, 24,851 bundled payloads were hashed against the outer manifest, and all 130 packet READY-to-MANIFEST bindings were checked. Across the original packet manifests, 26,466 payload occurrences were declared: 20,848 were directly collected, 2,856 were reconstructed by exact hash and length, and 2,762 remained unavailable. Repeated identical content was intentionally not treated as new development. [9]

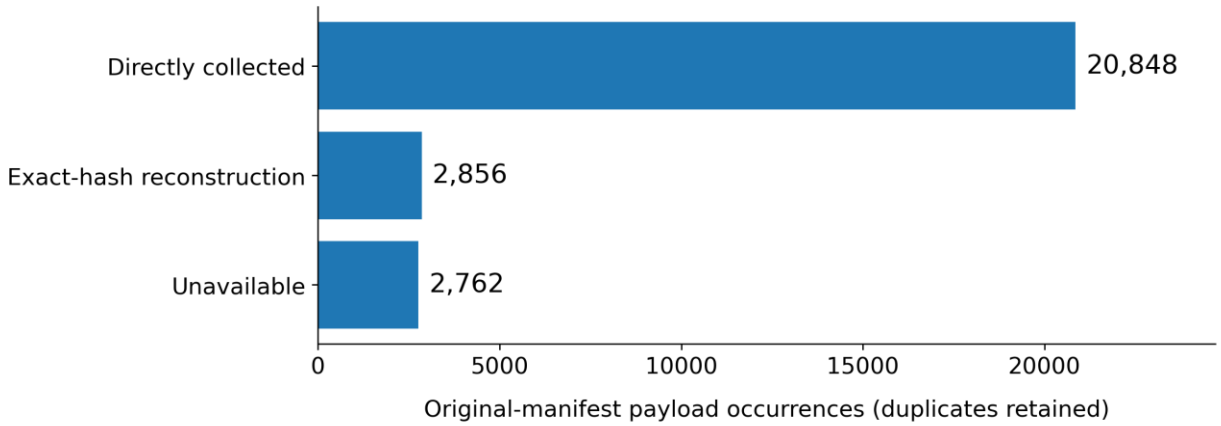

Figure 2 | The evidence corpus was substantial, but not complete.

Counts refer to payload occurrences declared across original packet manifests. Direct collection, exact-hash reconstruction and unavailable bytes are different evidence classes. A reconstructed file can support a static claim about its bytes, but not a claim that the collector observed that historical path. [10]

The analysis then extracted the current assignment and work sections of all 130 packet briefs, their test-summary tables and structured capture metadata, selected controlling requirements, start-of-round commitments, gap registers, case maps and available source anchors. The documentary event log contains 130 publication records plus 807 deduplicated check-record keys derived from 870 check-table rows. Only 124 of those check records were labelled captured at execution in the normalized metadata; 660 were declaration-only. [11]

For graph and dependency analysis, the study used six publication-focus categories, a requirements-derived 23-node / 29-edge task graph, a six-node activity-transition projection, 18 historical readiness snapshots, the P130 case map containing 84 Q assertions and 12 connected-qualification rows, and packet-level value-stream

9 Independent Evidence Review, pp. 2-5. Reported integrity results: 24,851 bundled payloads verified against the outer manifest; 26,466 original-manifest payload occurrences comprised 20,848 directly collected, 2,856 exact-hash reconstructed, and 2,762 unavailable occurrences. Hash agreement verifies byte identity, not the truth of producer statements.

10 Independent Evidence Review, pp. 2-5. Reported integrity results: 24,851 bundled payloads verified against the outer manifest; 26,466 original-manifest payload occurrences comprised 20,848 directly collected, 2,856 exact-hash reconstructed, and 2,762 unavailable occurrences. Hash agreement verifies byte identity, not the truth of producer statements.

11 Independent Evidence Review, pp. 19-20. The process-mining extraction produced 870 test/check table rows and 807 deduplicated documentary record keys: 124 labelled CAPTURED_AT_EXECUTION, 660 CODEX_DECLARATION_ONLY, four UNAVAILABLE and 19 with an empty capture field. These are not 807 independently verified physical executions.

classifications. These are different analytical objects. Their counts should not be merged into a single progress percentage. For response-body analysis, the study also normalized all 130 prepared-response bodies, stripped the standardized capture header and redaction footer before exact hashing, and semantically coded opening paragraphs for recurring setup-validation language.

Important limitations follow directly from the evidence design. The export does not provide native session IDs, native turn IDs, complete effective-context history, reliable token use, complete execution timings or a trustworthy labor ledger. Hashes establish that bytes match their declared identity; they do not establish that every producer statement is true. Those limitations are why this paper reports topology, recurrence, documentary conformance and bounded causal hypotheses rather than hours "wasted," dollar loss or a probability that the model was confused.

METHODS

4. Ten analytical lenses

No single metric can answer why a long-running agent workflow stalls. The study therefore applied ten complementary analyses. They examine the same project from different angles; they are not ten independent replications. The value comes from asking whether structurally different methods point toward compatible interpretations.

Method	Question it asks	Primary output / guardrail
1. Dominator analysis	Which accepted states must every successful delivery path pass through?	Unavoidable acceptance states in the contract-derived progression. Does not label every other task unnecessary.
2. AND/OR dependencies + minimal cuts	Does one blocker stop the whole project or only one mandatory branch?	Separates permission, source and acceptance branches. Minimal failure cuts are not repair priorities or effort estimates.
3. Temporal critical-path / dependency analysis	Were later gaps tied to requirements that were already known?	23-node precedence model + 18 historical snapshots. Durations are UNKNOWN, so no schedule slack or delay hours are claimed.
4. Betweenness centrality	Which activity types mediate transitions in the observed workflow?	Identifies workflow hubs. Centrality is structural, not a waste or productivity score.
5. Cycles / SCC / rework analysis	How often did the workflow return among recurring work categories?	Transition counts, strongly connected components and A-B-A returns. Graph cycles are not automatically rework episodes.
6. Process mining + conformance	Did documentary events follow the intended source-to-acceptance process?	Event log and fixed-commitment comparisons. Documentary keys are not all independently verified executions.
7. Value-stream analysis	What kinds of publications accumulated?	Publication-level proxy classes. No time, cost or token allocation is inferred.
8. Change-point detection	When did the distribution of work categories change?	Statistical phase boundaries and sensitivity checks. A change point does not identify the intervention that caused it.
9. Duplicate / boilerplate response analysis	Did repeated response templates track a recurring workflow state?	Exact response-body hashes plus semantic opener coding and phase association. Counts publications and text patterns, not labor, model state or causation.
10. Competing-explanation analysis	Which causal stories fit the observed record, and where is evidence missing?	Qualitative support/counterevidence matrix. Not probabilities or causal effect sizes.

The remainder of the article walks through each lens and its result. The emphasis is on what each method adds - and what it cannot establish.

RESULTS

5. Results: What each analysis revealed

Analysis 1 - Dominator analysis: the acceptance path remained unavoidable

The first analysis treated the delivery contract as an accepted-state progression rather than treating packet chronology as a project schedule. In the modeled sequence, CP1 precedes CP2; CP2 precedes a locally qualified release; that release precedes authorized verified delivery; and delivery precedes CP3. In graph terms, earlier accepted states dominate later ones because every successful route must pass through them. [12] [13]

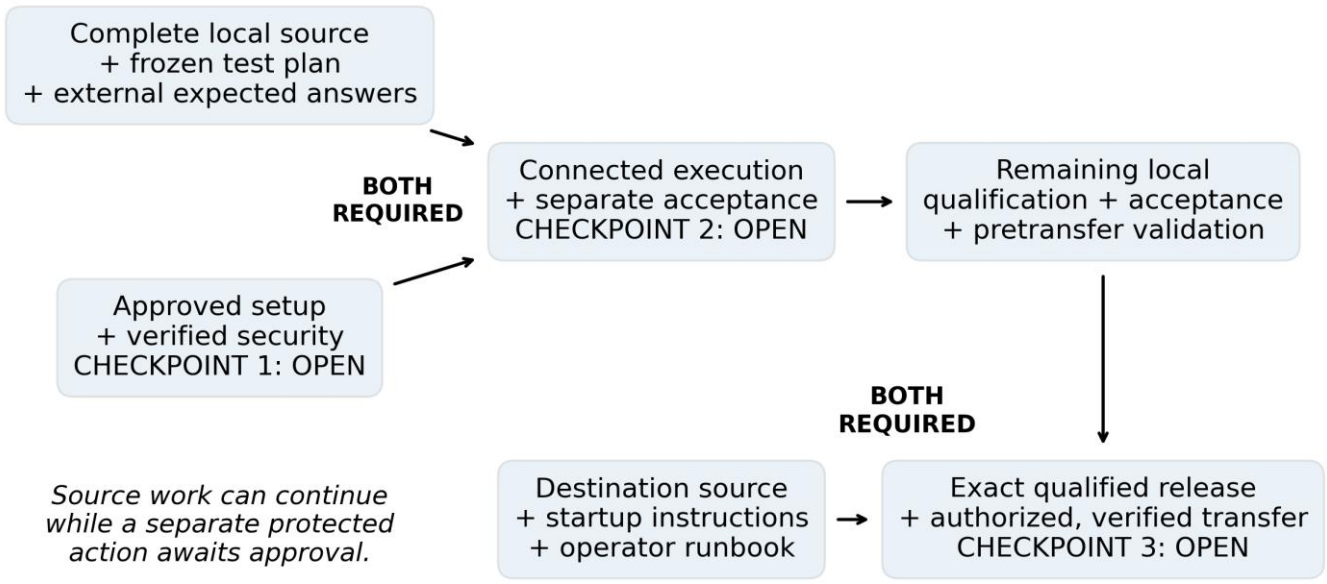

Figure 3 | Mandatory acceptance dependencies, not an estimated schedule.

Incoming branches at the joins are AND requirements. Source implementation can advance while permission is pending, but permission does not implement source. Likewise, a component fix can be valuable without becoming a full checkpoint. [14]

The result is simple but important: publishing another review packet, fixing a component or implementing destination-side code cannot substitute for the first still-open accepted state. At the same time, dominator analysis does not imply that every task outside the acceptance-state chain was avoidable. Reviews, evidence integrity and security work can be necessary support. The operational lesson is to make the next incomplete accepted state explicit and tie each round to a concrete dependency that advances it.

12 Independent Evidence Review, pp. 6-7 and 10. Product v3 required implemented changes, runnable qualification, independent validation, a sealed candidate, actual evidence and a runbook. Candidate and plan identities were to be fixed before execution.

13 Independent Evidence Review, pp. 10-12. The accepted-state dominator analysis and Boolean dependency model are requirements-derived checks. They verify the modeled contract; they do not discover a hidden causal mechanism.

14 Independent Evidence Review, pp. 10-12. The accepted-state dominator analysis and Boolean dependency model are requirements-derived checks. They verify the modeled contract; they do not discover a hidden causal mechanism.

Analysis 2 - AND/OR dependencies and minimal cuts: a blocker is not automatically a universal excuse

The Boolean dependency model represented seven coarse acceptance groups as mandatory. Absence of any one was sufficient to prevent final accepted delivery, producing seven singleton minimal failure cut sets. This does not mean fixing one group completes the project. It means every currently unsatisfied mandatory group eventually has to be resolved. [15]

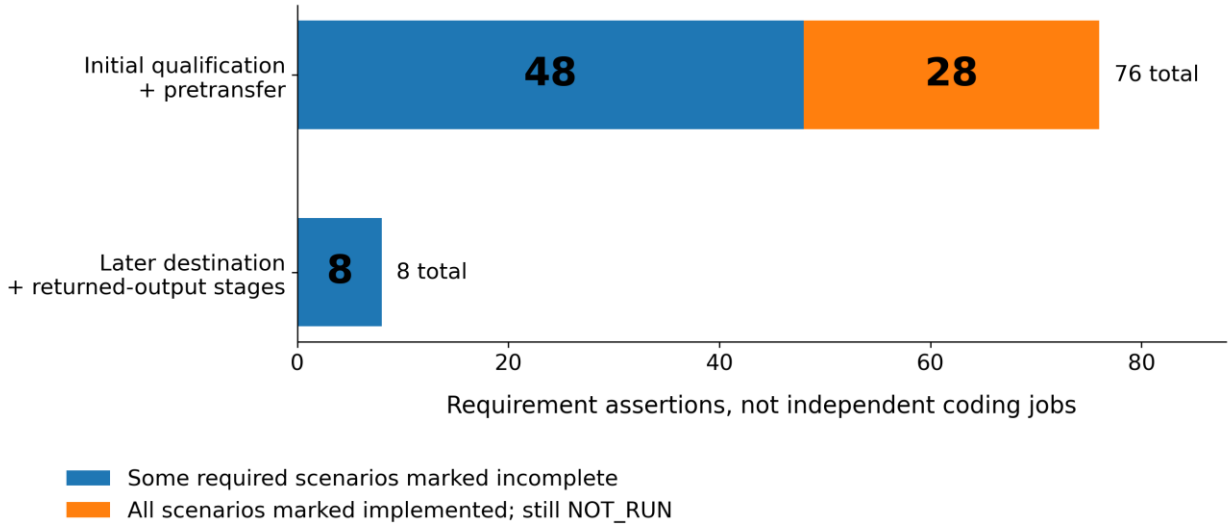

Figure 4 | Source implementation gaps remained separate from runtime approval.

Among 76 initial/pretransfer assertions, 48 were marked as missing at least one required scenario variant; the remaining 28 had all required variants marked implemented but were still NOT_RUN at required scope. Eight later-stage assertions also had incomplete variants. Several assertions can share one missing producer, so affected assertions are not independent coding jobs. [16]

This is where branch semantics matter. The permission branch contained protected setup decisions and bounded execution. The source branch contained incomplete scenario producers, history reconciliation, recovery orchestration, installer capture integration and the final candidate/plan. A correct refusal to cross a permission boundary is not a failure; but the same permission boundary does not explain unfinished ordinary source work. The case therefore supports a rule that should generalize: blocker labels must identify the exact branch they block.

Analysis 3 - Temporal dependency analysis: some later gaps traced back to earlier requirements

The temporal analysis compared what was documented at several points in the project with what later evidence showed. The broad delivery target was present from the beginning, while the operational frontier became more concrete as implementation uncovered containment, capture, authority and environment details. The model

15 Independent Evidence Review, pp. 11-12 and 29-30. At P130, none of seven coarse mandatory acceptance groups was fully established. The source and permission branches remained distinct; a legitimate authorization stop did not implement unfinished source.

16 Independent Evidence Review, pp. 12 and 29-30. P130 recorded 48 incomplete-variant assertions among 76 initial/pretransfer assertions, plus eight in later stages. All 84 Q assertions and 12 C1 rows were NOT_RUN at required qualification scope. NOT_RUN is not the same as a failed execution.

deliberately assigns UNKNOWN duration to remaining tasks; publication distance is not elapsed engineering effort. [17]

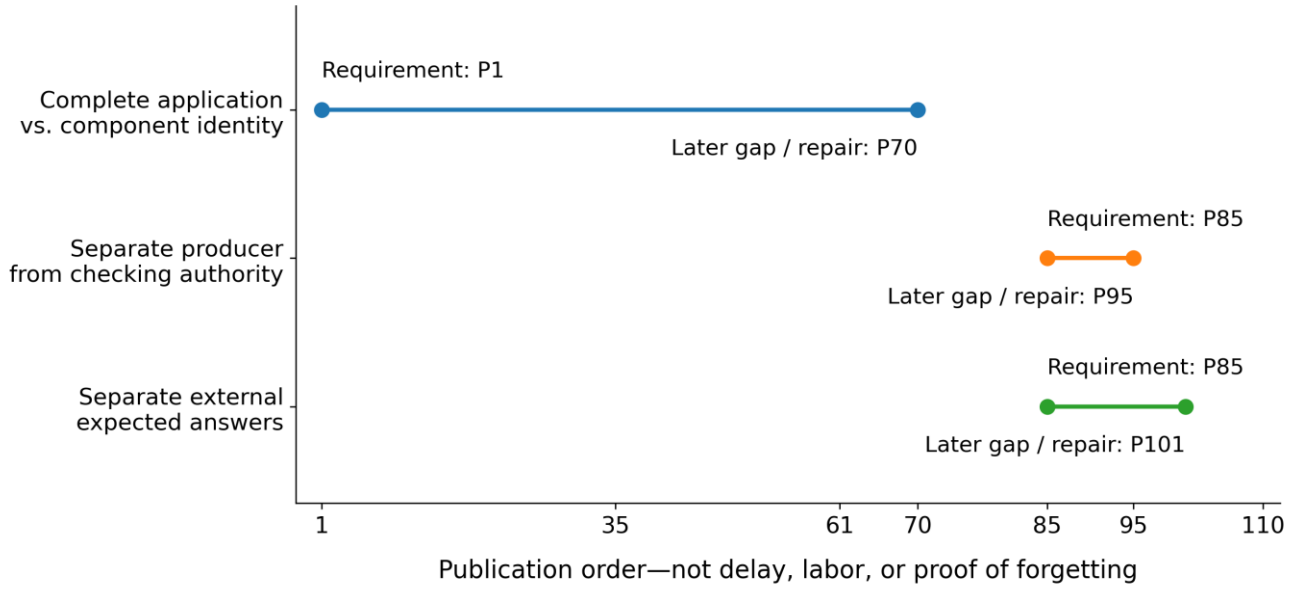

Figure 5 | Selected requirements preceded later gap or repair records.

The lines connect documentary observations, not continuous work periods. Complete application scope predates a later inventory gap; PLAN-C1 authority requirements predate later producer-isolation and external-answer repairs. The figure supports a planning/integration concern, not a claim that every intervening packet was avoidable. [18]

The strongest interpretation is that some already-specified interfaces were resolved incrementally rather than together. That matters because local component closure can create the appearance of completion even when compatibility across the joined path remains unsettled. But late implementation is not proof that a requirement fell out of context. The evidence cannot distinguish forgetting from planning choices, technical discovery or task framing without native session traces.

Analysis 4 - Betweenness centrality: review became the workflow hub

Centrality analysis separates the observed activity network from the requirements graph. In the six-category directed activity projection, review had the highest unweighted betweenness centrality at 0.308; security followed at 0.108. Review remained the highest intermediary under an alternative inverse-frequency weighting. In the required-task graph, by contrast, paths converged on execution and independent acceptance. [19]

17 Independent Evidence Review, pp. 13-14. The temporal model contains 23 task nodes, 29 prerequisite edges and 18 historical snapshots. Remaining task durations are UNKNOWN; packet distance is not engineering time.

18 Independent Evidence Review, pp. 13-14. The temporal model contains 23 task nodes, 29 prerequisite edges and 18 historical snapshots. Remaining task durations are UNKNOWN; packet distance is not engineering time.

19 Independent Evidence Review, pp. 15-16. In the six-category unweighted directed activity graph, review had betweenness centrality 0.308, security 0.108, and route and coordination 0.017 each. Review remained highest under inverse-transition-frequency sensitivity. Centrality values from different graph models are not efficiency ratios.

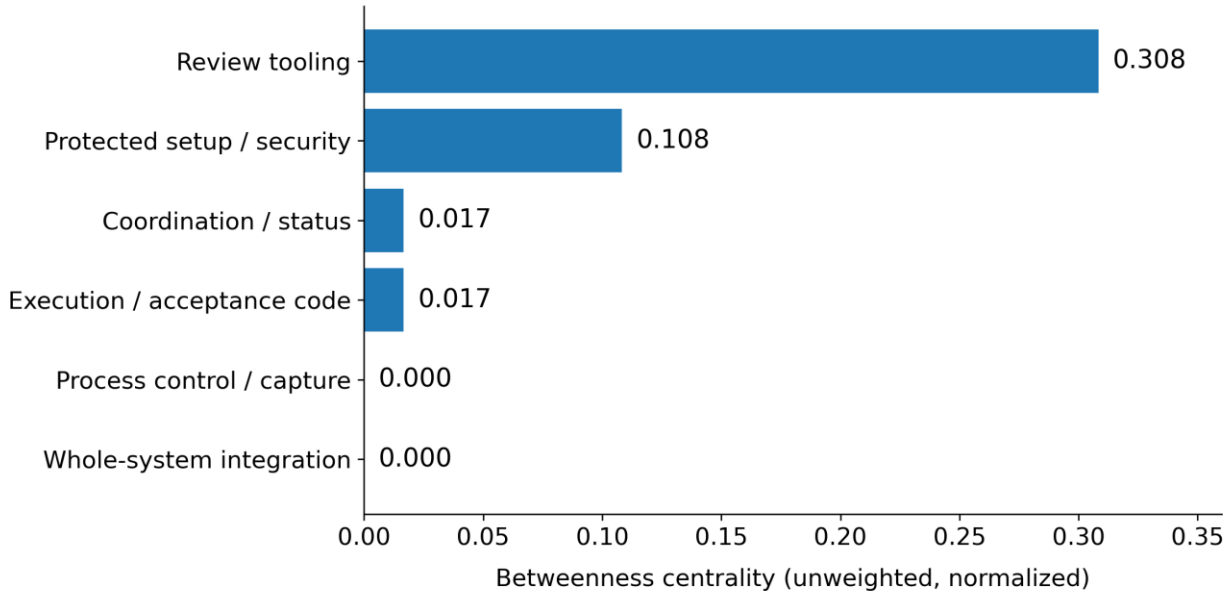

Figure 6 | Review was the strongest intermediary in the observed activity graph.

Betweenness centrality measures shortest-path mediation in the selected graph. It is not an efficiency, labor or waste score. The required-task graph has different nodes and semantics, so its values should not be compared numerically as a productivity ratio. [20]

The useful result is the structural mismatch: the work history was review-centered, while the delivery model was execution-and-acceptance-centered. That does not imply "remove review." Independent review can be mandatory and can catch genuine defects. The management question is whether each return to review produces a necessary decision or accepted dependency while other independent implementation continues.

Analysis 5 - Cycles and recurrence: the workflow repeatedly returned to the same categories

Chronology itself is acyclic - packet P90 cannot literally travel back to P30. But collapsing the history into recurring activity types creates a strongly connected network. The projection contained 56 cross-category transitions and 73 same-category transitions. Review and coordination exchanged focus 20 times, and the history contained 26 contiguous A-B-A returns, including eight review-coordination-review patterns. [21]

20 Independent Evidence Review, pp. 15-16. In the six-category unweighted directed activity graph, review had betweenness centrality 0.308, security 0.108, and route and coordination 0.017 each. Review remained highest under inverse-transition-frequency sensitivity. Centrality values from different graph models are not efficiency ratios.

21 Independent Evidence Review, pp. 17-18. The activity projection had 56 cross-category and 73 same-category transitions, with all six categories in one strongly connected component; review and coordination exchanged focus 20 times, and 26 contiguous A-B-A returns were observed.

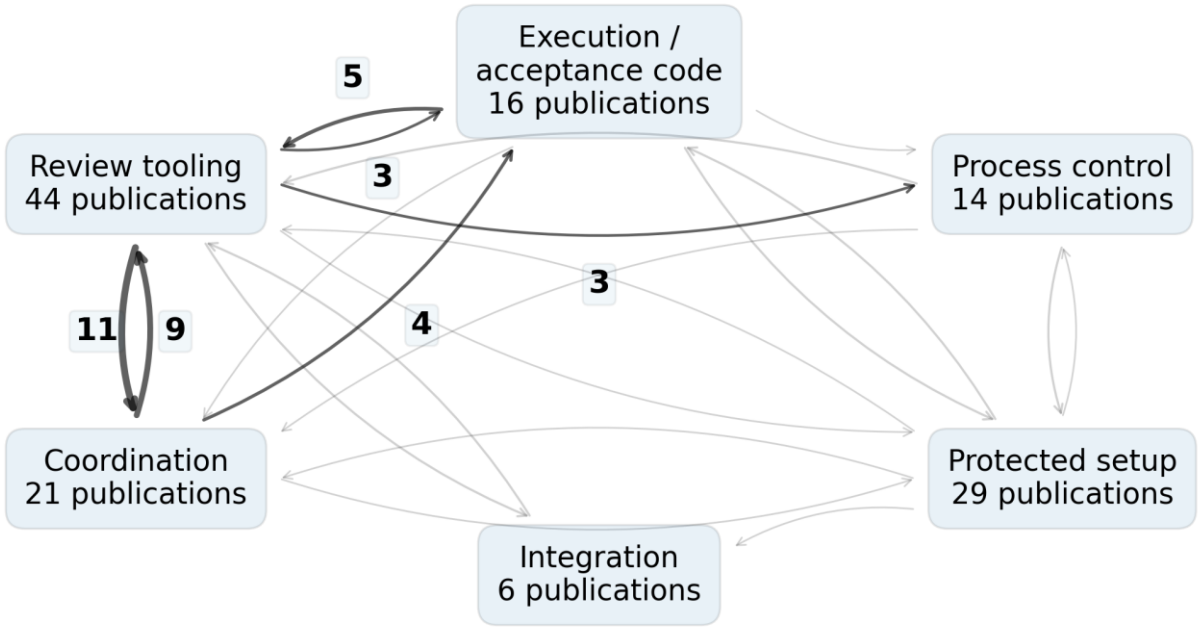

Figure 7 | The activity network shows repeated returns, not literal time loops.

Boxes report publication counts. Thick arrows mark frequent cross-category transitions; lower-frequency links are retained more lightly. The full projected network forms one strongly connected component across all six categories. This is recurrence, not proof that every return was unnecessary rework. [22]

Episode-level review explains why the recurrence cannot be reduced to one failure mode. Some loops corrected evidence provenance; others fixed lifecycle defects; security runs included actual prerequisite work and legitimate authorization stops. The pattern that remains after those qualifications is incomplete conversion: component closure and support work repeatedly exposed or deferred broader contract obligations.

Analysis 6 - Process mining and conformance: the missing transition was product conversion

Process mining reconstructed a documentary event log and compared it with the intended flow: retain the same commitment, implement source, run eligible meaningful checks, preserve identities and failures, obtain the required independent review, execute the exact approved connected qualification against frozen bytes, independently accept it, then release and deliver those same qualified bytes. [23]

22 Independent Evidence Review, pp. 17-18. The activity projection had 56 cross-category and 73 same-category transitions, with all six categories in one strongly connected component; review and coordination exchanged focus 20 times, and 26 contiguous A-B-A returns were observed.

23 Independent Evidence Review, pp. 19-20. The process-mining extraction produced 870 test/check table rows and 807 deduplicated documentary record keys: 124 labelled CAPTURED_AT_EXECUTION, 660 CODEX_DECLARATION_ONLY, four UNAVAILABLE and 19 with an empty capture field. These are not 807 independently verified physical executions.

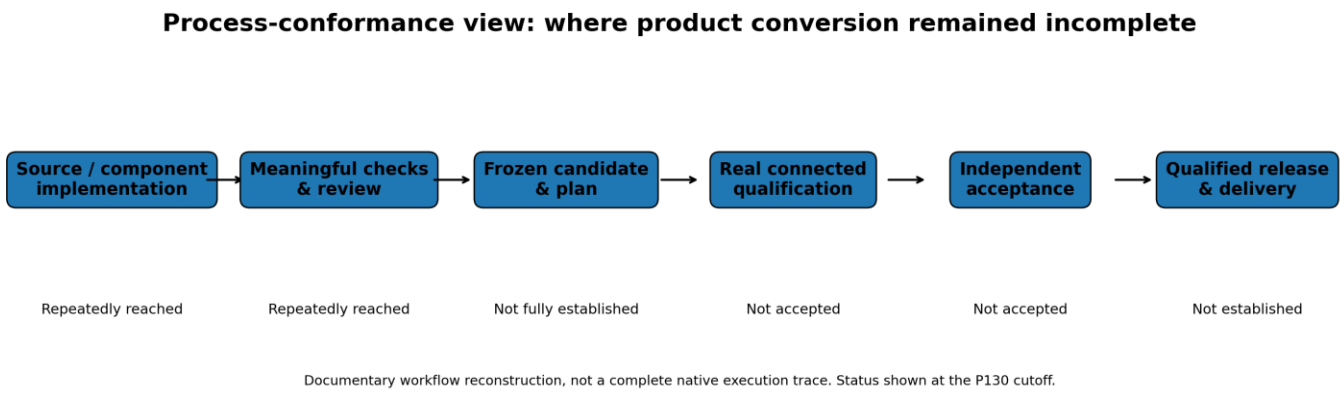

Figure 8 | The documentary process repeatedly reached work and review states without closing the full conversion chain.

The first two stages were repeatedly observed in the record. By the P130 cutoff, a complete frozen candidate/plan, accepted connected qualification, operational independent acceptance and qualified delivery were not established. This is a conformance summary, not a complete native execution trace. [24]

One of the most important conformance rules was fixed scope. P121 and P122 had materially different start commitments; judging P121 against the broader P122 requirement would be unfair. Conversely, P125 and P130 explicitly recorded authorized source that remained unfinished. The process-mining contribution is therefore not a single fitness score. It is a rule-by-rule reconstruction that preserves the assignment actually in force when each result was produced.

Analysis 7 - Value-stream analysis: product work was real, but publication volume was a poor proxy for completion

A publication-level value-stream classification was used because reliable labor, token and cost data were unavailable. The largest class was product source/component work at 59 publications. Coordination/preparation/bounded waiting accounted for 24; review-infrastructure implementation/repair for 22; evidence correction/republication for 11; independent/source-evidence review for eight; the remaining classes were smaller. [25]

24 Independent Evidence Review, pp. 19-20. The process-mining extraction produced 870 test/check table rows and 807 deduplicated documentary record keys: 124 labelled CAPTURED_AT_EXECUTION, 660 CODEX_DECLARATION_ONLY, four UNAVAILABLE and 19 with an empty capture field. These are not 807 independently verified physical executions.

25 Independent Evidence Review, pp. 21-22. Publication-level value-stream proxies: 59 product source/component, 24 coordination/preparation/bounded waiting, 22 review-infrastructure implementation/repair, 11 evidence correction/republication, eight independent/source-evidence review, three mandatory review export, two authorized prerequisite execution and one capability diagnosis.

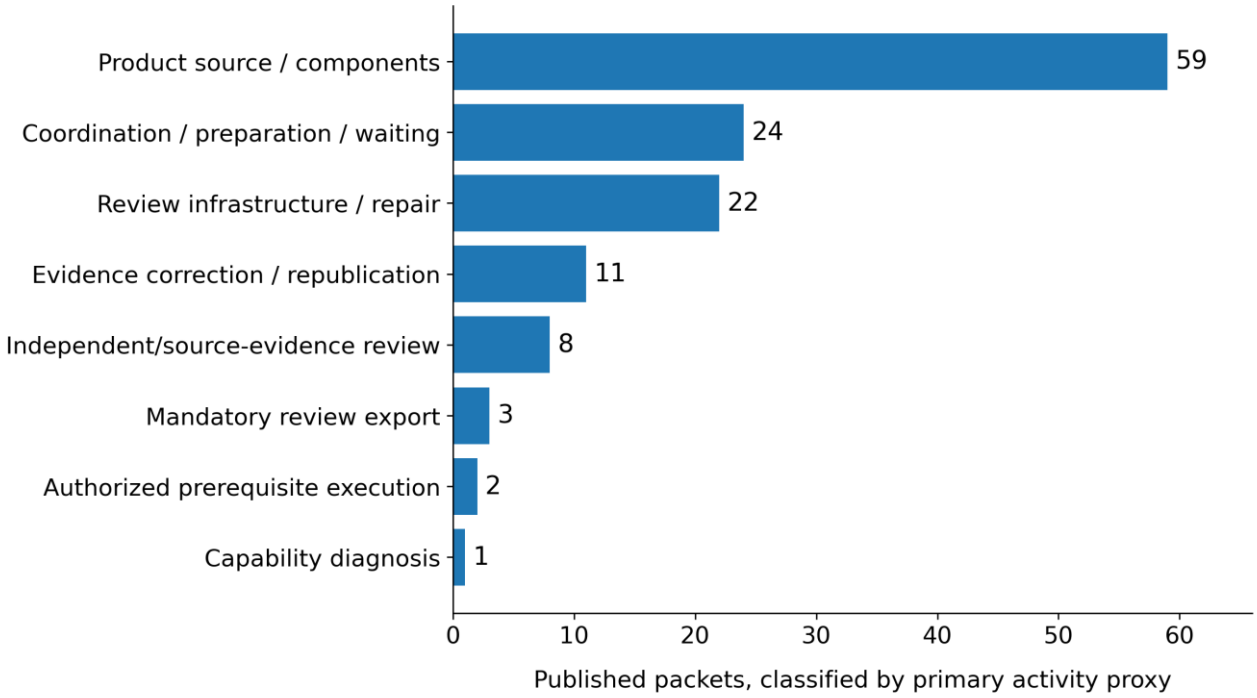

Figure 9 | The project did contain substantial source and component work.

These counts classify publications, not engineering hours, tokens, dollars or accepted dependencies. Product/source work includes both new implementation and correction. Support categories can be necessary. [26]

This figure prevents an equally misleading opposite conclusion: the project was not merely producing paperwork. Real engineering accumulated. The problem was that publication volume and source churn could not substitute for accepted dependency conversion. A better operational dashboard would separately track source completion, new required runtime evidence, independent acceptance and full-gate closure.

Analysis 8 - Change-point detection: focus shifted in statistically visible phases

Exact dynamic programming segmented the six-category publication sequence. Under the baseline minimum-segment and penalty specification, the preferred model contained four periods: P001-P070, P071-P084, P085-P122 and P123-P130. The penalized score improved from 449.597 for one segment to 348.289 for four, then worsened as additional segments were added. The strongest broad single split was after P70. [27]

26 Independent Evidence Review, pp. 21-22. Publication-level value-stream proxies: 59 product source/component, 24 coordination/preparation/bounded waiting, 22 review-infrastructure implementation/repair, 11 evidence correction/republication, eight independent/source-evidence review, three mandatory review export, two authorized prerequisite execution and one capability diagnosis.

27 Independent Evidence Review, pp. 23-24. Baseline change-point model: P001-P070 / P071-P084 / P085-P122 / P123-P130, penalized score 348.289. The strongest broad single split was after P70. Sensitivity tests support a nonstationary focus sequence but do not identify causation.

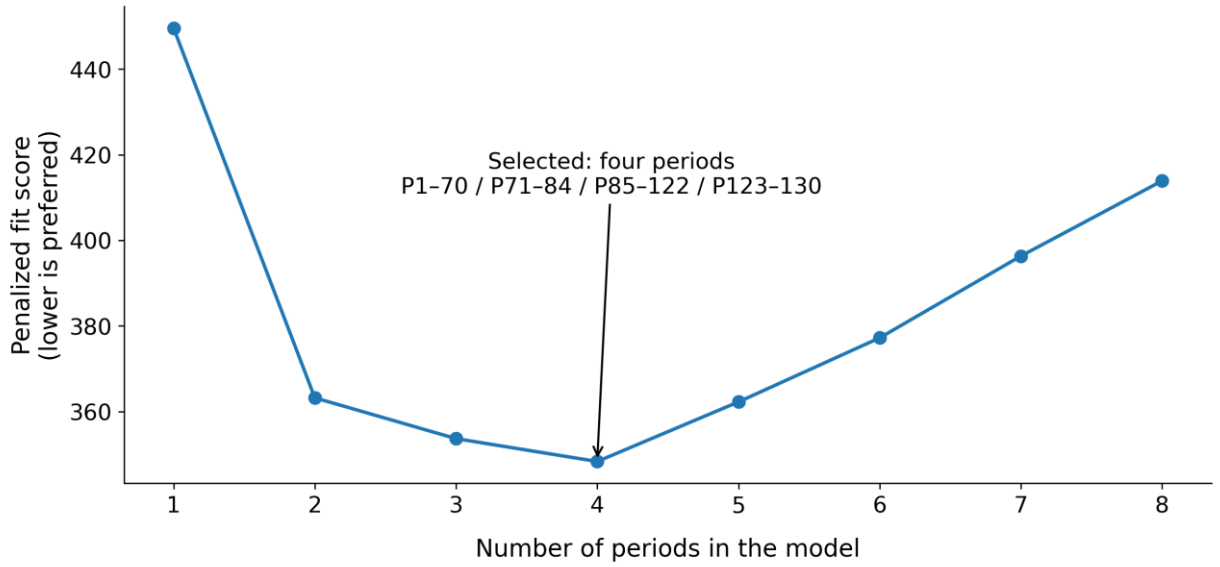

Figure 10 | A four-period model best fit the baseline focus sequence.

The statistical boundary after P70 is not identical to the explicit product pivot at P61. P85 aligns with PLAN-C1 and a long security/setup period; P123 aligns with a broader completion mandate. Sensitivity analysis shows which boundaries remain when the penalty or minimum segment length changes. None of these boundaries proves causation. [28]

This is a useful example of why statistics should refine rather than replace documentary interpretation. P61 is an explicit change in assignment even though the category distribution does not shift most strongly until later. P121/P122 changes scope while both packets remain in the same coarse category. A change-point algorithm can detect nonstationarity; it cannot tell whether instructions, newly exposed defects, permission changes or task complexity caused it.

Analysis 9 - Duplicate and boilerplate output: the loop had a linguistic fingerprint

Hashing each normalized prepared-response body found only two exact duplicate groups: P013=P014 and P127=P128. Because each successor received its own packet identity without adding any new prepared response body, two of 130 publications - 1.54% - were redundant at the user-facing text level. This does not mean no corrective work occurred: P014 superseded P013 because the no-change table omitted private review artifacts created during the turn, while P128 superseded P127 solely to restore a required diff excluded by text-extension policy. The measurement is therefore duplicate response content, not zero engineering effort.

The broader signal was larger. Semantic coding identified a recurring setup-validation opener family in 37 of 130 responses. The phrases were not literally identical, but they repeatedly re-established workspace instructions, controlling handoffs, packager or review-history state before addressing the task. The distribution was 23 of 54 packets in P001-P054, zero of seven in the P055-P061 direct pivot, 14 of 61 in P062-P122, and zero of eight in the P123-P130 takeover. Pooling the two packet-cadence periods versus the two direct-engagement periods gives

28 Independent Evidence Review, pp. 23-24. Baseline change-point model: P001-P070 / P071-P084 / P085-P122 / P123-P130, penalized score 348.289. The strongest broad single split was after P70. Sensitivity tests support a nonstationary focus sequence but do not identify causation.

37/115 versus 0/15; a two-sided Fisher exact test yields p=0.00585. Because these are observational, user-defined phases rather than randomized conditions, the association is descriptive rather than causal.29

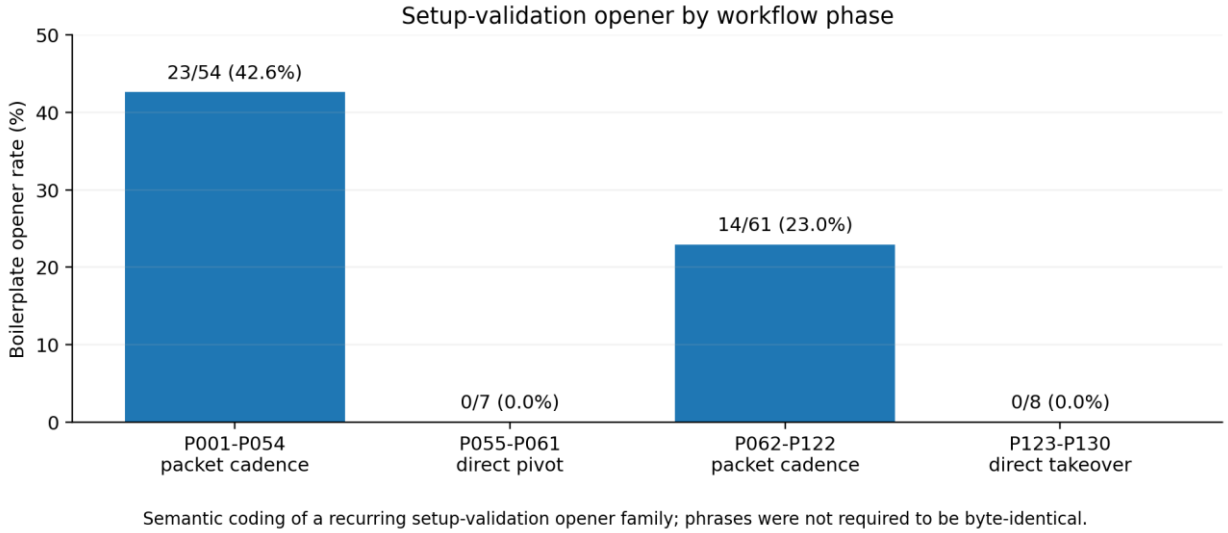

Figure 11 | The setup-validation opener disappeared in the two direct-engagement phases.

The repeated opener appears in 42.6% of P001-P054 responses and 23.0% of P062-P122 responses, but in none of P055-P061 or P123-P130. The coding captures a semantic template family rather than one byte-identical sentence. The phase association is statistically unusual, but it does not establish that the boilerplate caused task-selection drift.

The pattern is best interpreted as a workflow-state marker. During the recurring packet/review cadence, responses often began by re-confirming the operating environment; when the user forced direct engagement, that ceremony disappeared. The signal is not simply a review-packet marker - similar openings also occur in packets recorded as implementation work - so the stronger interpretation is a linguistic fingerprint of a procedural operating mode or workflow inertia. It complements the transition and change-point results, but it does not show that the repeated language itself caused critical-path divergence or reveal the model's internal state.

Analysis 10 - Competing explanations: the evidence supports a combination, not one hidden cause

The final analysis weighed six explanations against supporting observations, counterevidence and missing measurements. Poor dependency planning/integration was the strongest supported engineering/process explanation. Reviewer-driven fragmentation was supported as a workflow contributor. Authorization friction was established for specific protected operations but did not explain independent source gaps. Execution limitations mattered in particular episodes. Instruction overload and context loss remained plausible or possible but unestablished. [30]

29Supplemental response-body analysis of the archived P001-P130 prepared-response corpus, 30 September 2026. Standard prepared-response headers and redaction footers were stripped before SHA-256 hashing. The only exact duplicate body groups were P013/P014 and P127/P128. Manual semantic coding identified 37 setup-validation opener-family packets: 23/54 in P001-P054, 0/7 in P055-P061, 14/61 in P062-P122 and 0/8 in P123-P130. Pooling the packet-cadence periods against the two direct-engagement periods yields 37/115 versus 0/15; Fisher exact test, two-sided, p=0.005854. The opener classification is semantic rather than literal-string equality, and the phase association does not establish causation.

30 Independent Evidence Review, pp. 25-27. The competing-explanation assessment treats dependency planning/integration as the strongest supported engineering/process explanation, reviewer-driven fragmentation as a supported contributor,

Competing explanations: what the evidence supports and what remains unproven

Hypothesis	Assessment in this case	Key limitation / counterevidence
Poor dependency planning / integration	Strongest supported engineering/process explanation	Does not identify one model mechanism
Reviewer-driven fragmentation	Supported workflow contributor	Reviews also found real defects; assignments were often user-directed
Authorization friction	Established for specified protected operations	Does not explain separate authorized source gaps
Execution limitations	Supported in specific episodes	Total impact is unmeasured; some limits were implementation defects
Instruction overload	Plausible, unestablished	No effective-context or token-history experiment
Context loss	Unestablished in this corpus	No native context trace; late implementation is not proof of forgetting

Qualitative evidence assessment; not probabilities, causal effect sizes, or a ranking of model internals.

Figure 12 | The causal story is mixed, and several popular explanations remain unproven.

The labels are qualitative judgments about this corpus, not probabilities or effect sizes. In particular, the absence of native effective-context and resource telemetry prevents a defensible claim that long context or context loss caused the observed pattern. [31]

The causal structure is also confounded by human intervention. Difficult technical work increases defects; defects increase review; review can narrow subsequent task selection; poor progress can cause the user to issue longer and more detailed prompts. That means prompt length may be a consequence of poor progress rather than its cause. The study therefore stops short of claiming that the model "lost the plot" internally.

authorization friction and execution limitations as episode-specific contributors, and instruction overload/context loss as unestablished without native context telemetry or controlled replay.

31 Independent Evidence Review, pp. 25-27. The competing-explanation assessment treats dependency planning/integration as the strongest supported engineering/process explanation, reviewer-driven fragmentation as a supported contributor, authorization friction and execution limitations as episode-specific contributors, and instruction overload/context loss as unestablished without native context telemetry or controlled replay.

INTERPRETATION

6. Synthesis: What the "hamster wheel" actually means

Taken together, the ten analyses point to a specific pattern. The workflow was capable of producing useful local work, but the mapping between local tasks and the terminal acceptance graph was not consistently strong enough to close the project. Review and coordination became recurrent workflow hubs. Integration dependencies were resolved in pieces. Real permission boundaries sometimes stopped protected execution. Separately, authorized source work remained unfinished. And the evidence system could accurately report BLOCKED or NOT_READY states without causing the workflow to escape those states.

That last point is especially important. The normalized readiness records contained no overall READY or PASS state at the cutoff; the producer was not simply declaring a nonexistent final success. The failure mode was therefore not necessarily false completion reporting. It was the persistence of an unfinished system despite repeated awareness, review and local correction. In other words, a workflow can be relatively candid and still be ineffective at converging on done.

The strongest documentary shortfalls are deliberately narrow. P125 and P130 contain explicit unfinished authorized-source commitments. They support findings of missed original implementation outcomes. They do not prove two "wasted rounds," and they do not justify extrapolating an autonomous task-selection error count across all 130 packets. Earlier review-heavy periods and the P56 producer explanation support broader priority drift, but the native assignment and eligibility record is incomplete. [32]

> The hamster wheel is not "the model kept doing useless things." It is "the workflow kept generating locally legitimate work without reliably closing the next mandatory acceptance dependency."

What this suggests for agent engineering

The most defensible design implication is to preserve the meaning of done at every round. Before a long-running agent starts a tranche, identify the next incomplete acceptance state and the exact dependency to be delivered. Separate unconditional source work from permission-dependent runtime outcomes. At the end, classify each original commitment as completed, partially advanced, attempted and failed, awaiting a genuine prerequisite, or authorized but left unattempted. A larger next prompt should not erase the previous commitment.

For monitoring, the useful alert is not merely "the agent has produced many artifacts." A more diagnostic signal is repeated activity without either a newly completed executable dependency or new required runtime evidence. That type of stall detector needs scope-aware evidence so that review packets, exports and legitimate decision waits are not misclassified as failed implementation turns.

32 Independent Evidence Review, pp. 14, 28-32. P125 and P130 contain the strongest documentary examples of authorized source work remaining unfinished. The report explicitly avoids converting these into a project-wide count of wasted rounds or avoidable hours.

CONCLUSION

7. Conclusion

This case study began with a provocative question: why do agents appear to go rogue? The evidence supports a more careful formulation. In this project, the central problem was not demonstrated rebellion or hidden intent. It was a progress trap: a long-running AI-assisted engineering workflow produced substantial real work, repeatedly revisited review and support processes, and still failed by the historical cutoff to establish the exact end-to-end acceptance conditions defined at the outset.

That distinction matters for AI research. If we evaluate agents only by visible activity, code churn, local test success or the sophistication of their status reports, we may miss whether those outputs are converging on the mandatory terminal state. Conversely, if we label every review loop or authorization stop a failure, we will punish necessary verification and safe behavior. The better unit of accountability is accepted dependency conversion: what mandatory prerequisite was closed, by what evidence, under what original scope?

The paper's contribution is therefore not a universal causal theory. It is an empirical case and a reusable analytical frame. Compare the observed work graph with the required delivery graph. Preserve original commitments. Separate independent branches. Use temporal and process evidence to distinguish known requirements from newly discovered facts. Treat statistical phase changes as descriptive, not causal. And make uncertainty explicit when native context, tool, resource and model telemetry are missing.

The next step should be experimental. Select exact historical source states whose bytes are available; hold the deliverable, model, tools, budget, authorization boundaries and independent acceptance criteria fixed; then vary context packaging and task framing in randomized repeated trials. Compare the original effective history with a shorter but equally complete brief, and compare granular repair-queue assignments with bounded executable-dependency commitments. Record actual tokens, tool failures, truncation and resource stops. No such controlled replay was performed here, so its outcome should not be presumed. [33]

> For long-running agents, progress should not mean "more happened." Progress should mean "the remaining path to accepted delivery became demonstrably shorter."

33 Independent Evidence Review, pp. 26-27. Proposed controlled comparison: freeze exact source states and the same task, tools, permissions, budget and acceptance criteria; vary effective context packaging and assignment framing; randomize repeated trials; preserve failures and timeouts; use blinded non-author assessment. No such controlled replay was performed in the reported study.

NOTES

References

The references below collect the main external works cited in the related-research discussion and the principal project report. URLs were checked on September 30, 2026. The underlying project source evidence remains private.

1. Independent Evidence Review | P001-P130. "Codex critical-path investigation: Independent analysis of the available P001-P130 evidence." 30 September 2026. Private project evidence, 35 pages.

2. Kwa, Thomas, Ben West, Joel Becker, et al. "Measuring AI Ability to Complete Long Software Tasks." METR, 19 March 2025. Current methodology: https://metr.org/time-horizons/.

3. Cemri, Mert, et al. "Why Do Multi-Agent LLM Systems Fail?" arXiv:2503.13657 (2025). https://arxiv.org/abs/2503.13657.

4. Liu, Nelson F., et al. "Lost in the Middle: How Language Models Use Long Contexts." Transactions of the Association for Computational Linguistics 12 (2024): 157-173. https://aclanthology.org/2024.tacl-1.9/.

5. Anthropic. "Harness design for long-running application development." 24 March 2026. https://www.anthropic.com/engineering/harness-design-long-running-apps.

6. OpenAI. "How we monitor internal coding agents for misalignment." 19 March 2026. https://openai.com/index/how-we-monitor-internal-coding-agents-misalignment/.

7. OpenAI. "GPT-6 Astra: A new generation of intelligence." 2026. https://openai.com/index/gpt-6-astra/.

8. OpenAI Help Center. "ChatGPT Work and Codex." Accessed 30 September 2026. https://help.openai.com/en/articles/20001275-chatgpt-work-and-codex.

9. Truong, Charles, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods." arXiv:1801.00718.

10. IEEE Task Force on Process Mining. "Process Mining Manifesto." General process-mining discovery and conformance framework.

Claim boundary

The public-facing article should not be represented as a peer-reviewed study or as an OpenAI finding. Its strongest low-assumption results are documentary: manifest integrity, publication and transition counts, exact response-body duplication, opener-distribution counts, the P121/P122 scope contrast, P125/P130 unfinished-work admissions and P130 case-map counts. Claims about internal context use, resource exhaustion, causal effects of boilerplate, or a platform defect require native telemetry or controlled experiments that are not present in the packet archive.