SKILLHORIZON: CONSTRUCTING LONG-HORIZON AGENT TASKS THROUGH SKILL COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agents must complete workflows in which later activities depend on earlier outputs and changes to shared state. A key challenge for evaluation is to expand task requirements and their dependencies while keeping tasks executable and outcomes verifiable. We introduce SkillHorizon, a benchmark that constructs long-horizon tasks through skill composition in seven business environments. Its generators use domain-specific rules to turn reusable skill procedures into concrete business activities. These activities are connected through input–output dependencies, shared resources, and state updates, allowing workload and coordination demands to grow together. Successful reference executions validate task feasibility and measure interaction workload, while programmatic evaluators assess local completion and whole-task success. We evaluate five language models on 1,050 tasks with reference interaction workloads ranging from 64K to 1M tokens. Across these scales, mean local-goal completion falls from 90.4% to 46.8%, as completed work fails to keep pace with growing requirements. At larger workloads, agents more often submit unsuccessful final answers with substantial work unfinished and tool-call allowance remaining. Trace analysis also identifies cases where scripts execute successfully but their outputs fail downstream business checks. SkillHorizon provides a configurable testbed for studying sustained progress, dependency coordination, and complete task delivery in growing workflows.

1 INTRODUCTION

Language agents increasingly carry out business workflows that involve multiple activities and sustained interaction with tools and data (Xu et al., 2024; Li et al., 2025). The outputs of one activity become inputs to later stages, while resource commitments shape subsequent choices. Changes in the environment may also require completed work to be reviewed or revised. Agents must therefore maintain valid intermediate results and coordinate activities as they work toward the final goal. Studying this ability requires executable tasks that can grow in required work and dependencies, with persistent state and verifiable outcomes.

Existing research provides foundations for long-horizon evaluation and skill reuse. WorkArena++ combines atomic tasks into enterprise workflows with reference executions and validation functions (Boisvert et al., 2024). TheAgentCompany evaluates work across business applications, code execution, and communication in a simulated workplace (Xu et al., 2024), while AgentLongBench studies information use over extended interaction histories (Fang et al., 2026). In parallel, Voyager develops an executable skill library, and SkillsBench evaluates packaged skills across tasks (Wang et al., 2023; Li et al., 2026c). AgentSkillOS further studies skill retrieval and graph-based coordination (Li et al., 2026a). These studies support complex workflows and procedural reuse.

Despite this progress, existing evaluations offer limited insight into how agents perform as more business activities are added and linked to earlier work. Studying this scaling requires a construction method that expands task goals and dependencies together. Each new activity must specify its required inputs, expected outputs, and use of shared resources. State updates must also define when earlier results need to be revised. At the same time, resource allocations and completion checks must match the expanded task. The key challenge is to build these larger workflows from reusable skills while keeping them executable and their outcomes verifiable.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

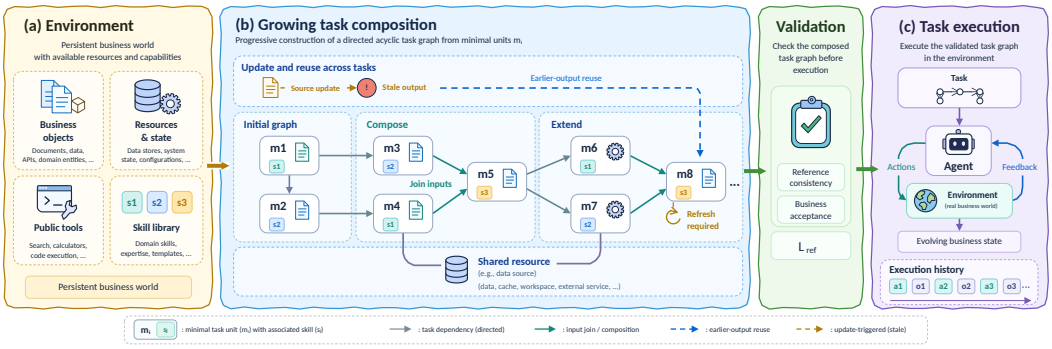

Figure 1: SkillHorizon constructs long-horizon tasks through skill composition. (a) Environments provide objects, resources, tools, and skills. (b) Tasks grow through dependencies, resource constraints, and updates requiring downstream refresh. Public reference executions validate business completion and measure L_{ref}. (c) Agents execute validated tasks through repeated interactions, changing business state and accumulating actions a_i and observations o_i.

We introduce SkillHorizon, a benchmark that constructs long-horizon tasks through skill composition in seven business environments spanning finance, field maintenance, social campaigns, logistics, research, documentation, and manufacturing. Domain-specific generators turn reusable skill procedures into business modules with concrete objects and goals. They connect modules through input–output dependencies and configure shared resources and state updates. Adding modules increases required work, while their connections determine which earlier results must be reused, combined, or revised. The generators create task goals and initial states together. Each environment provides public tools, skill resources, and programmatic evaluators. Successful reference executions validate task feasibility and measure interaction workload; evaluators check local-goal completion and whole-task acceptance. Figure 1 summarizes the pipeline.

We evaluate five language models on 1,050 tasks with reference interaction workloads ranging from 64K to 1M tokens, yielding 5,250 model–task executions. Mean local-goal completion falls from 90.4% to 46.8% across these scales, as completed work fails to keep pace with growing requirements. We also analyze failed final answers with recorded completion and tool use. At 1M, 54.0% of these runs complete less than 25% of local goals while retaining over 50% of the tool-call allowance. The corresponding share is 2.4% at 64K. Trace analysis further identifies cases where scripts execute successfully but their outputs fail downstream business checks. These findings reveal difficulties in sustaining progress and using local results to complete larger workflows.

Our contributions are threefold:

1. Skill-based task construction. We introduce a method that builds business activities from reusable skills and connects them through input–output dependencies, shared resources, and state updates. The method supports configurable expansion of required work and coordination demands.

2. An executable and scalable benchmark. We build seven persistent business environments with task generators, public tools, skill resources, reference executions, and programmatic evaluators. These resources support reproducible evaluation across task instances and workload scales.

3. An empirical study of long-horizon execution. We evaluate five models across increasing workloads and identify gaps between required and completed work, including unfinished execution with unused tool allowance and downstream failures of script outputs.

2 The SkillHorizon Benchmark

2.1 Business Environments and Skills

SkillHorizon comprises seven executable business environments: FinanceOps, FieldOps, SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps. Each environment maintains a persistent business world of objects, resources, and artifacts, governed by domain-specific rules. Operations

produce artifacts, allocate resources, and update business state. Their effects persist across interactions: later activities consume earlier outputs and must satisfy the resource and validity constraints of the current state. Appendix D.1 details the resources and business rules of each environment.

Business operations are exposed as Model Context Protocol (MCP) tools (Model Context Protocol Contributors, 2025). Through domain-specific gateways, agents query business objects, invoke operations, and inspect artifacts. The environment checks requested operations against the current state and applicable business constraints, returning execution results or diagnostic feedback.

Each environment $\mathcal{E}$ provides a library $\mathcal{S}_{\mathcal{E}}$ of reusable skills that organize public tool operations into business procedures. We characterize a skill $s \in \mathcal{S}_{\mathcal{E}}$ as

$$s = (\alpha_s, I_s, P_s, O_s), \tag{1}$$

where α_s specifies applicability conditions, I_s required inputs, P_s the procedure, and O_s expected outputs and state effects. Skills follow the Agent Skills package format (Agent Skills, n.d.), with procedural documentation and optional scripts and reference materials. During task construction, domain-specific rules instantiate these requirements and effects over concrete business objects to define local work. Agents select and apply these procedures through public tools.

2.2 Dependency-Structured Task Construction

A *business module* is the unit of task composition. It specifies a piece of required business work by applying one or more skill procedures to concrete objects and inputs. Domain-specific rules determine its input requirements I_m, required outputs and state effects O_m, and completion conditions g_m:

$$m = (I_m, O_m, g_m). \tag{2}$$

A module may combine several skills to fulfill its business requirements, while the same skill can be reused across modules with different objects and goals.

We connect modules when completing one requires outputs produced by another. The resulting dependencies form a directed acyclic graph $G = (M, D)$. Each edge $e = (m_u, m_v, d_e) \in D$ specifies the outputs that m_v requires from m_u, together with conditions on their content, provenance, quantity, or validity. Environment operations and completion checks enforce these requirements. Shared resources and mutable business state further constrain when modules can proceed and whether previously produced outputs remain usable.

For a construction configuration c and sampling seed ξ, the environment-specific generator produces

$$(G, x_0, g) = \mathrm{Gen}_{\mathcal{E}}(c, \xi). \tag{3}$$

The configuration controls work scope, dependency patterns, resource conditions, and updates. The seed selects concrete objects, attributes, and connections within these rules. Construction couples the dependency graph G with an initial state x_0 that supplies the corresponding business objects and resources, and a goal g that requires the generated work and its joint delivery conditions to be satisfied. Agents pursue this goal through public environment operations.

We extend tasks by adding required business modules and connecting them to existing work through domain-specific composition rules. Sequential extension makes an earlier output a prerequisite for further work; joins require multiple upstream outputs to jointly satisfy a downstream requirement; long-range reuse carries earlier results into later stages. These connections determine which results must be combined and how far their content and validity must be maintained. Task expansion therefore controls both the amount of required work and the dependency structure through which that work contributes to the overall goal.

Task expansion also determines the business state needed to support the generated work. Domain-specific rules derive resource quantities and allocations from task requirements while imposing capacity and scheduling constraints. For tasks with updates, these rules specify when changes occur and which artifacts or subsequent operations they affect. Existing results retain or lose validity according to the domain rules, creating requirements for revision and downstream coordination. Figure 1 summarizes the construction pipeline, and Appendix D.2 details the domain-specific rules. Public reference execution then verifies that each instance's business requirements can be satisfied.

2.3 Reference Validation and Workload Calibration

We validate generated instances with a rule-based reference policy that executes business activities through public environment tools, following the executable-oracle approach used in compositional benchmarks (Boisvert et al., 2024). The policy queries the current state and carries out the required work under the instance's dependency, resource, and update conditions. An instance passes validation when its resulting artifacts and final business state satisfy the local completion requirements and global delivery constraints. We calibrate workload from these successful reference traces.

We measure reference workload as the cumulative tool-interaction volume of a successful reference execution. For a trace containing R tool calls, we define

$$L_{\mathrm{ref}} = \sum_{i=1}^{R} \left[\mathrm{Tokens}(a_i) + \mathrm{Tokens}(o_i)\right], \tag{4}$$

where a_i contains the tool name and arguments, and o_i is the returned observation. Fixed serialization and tokenization count each call–observation pair, including repeated queries. Thus, L_{ref} measures tool interaction as the reference policy fulfills the task's business requirements.

To obtain tasks at different workload scales, we adjust domain-specific work-scale parameters under the selected construction rules and measure the resulting successful reference executions. Instances are assigned to workload bands according to their measured L_{ref}. Alongside this measure, we record business work counts and dependency properties, including depth, joins, and reuse spans where applicable, to characterize the amount and organization of required work. Appendix B details the calibration parameters, workload-band boundaries, and instance-selection rules.

3 Experiments

We evaluate agents on generated tasks that vary in the amount of required business work and the dependencies connecting it. Across seven executable environments, we measure terminal business-goal completion and whole-task success to examine agents' ability to sustain progress toward an overall goal.

3.1 Experimental Setup

Models and harness. We evaluate Grok 4.6, Qwen3.8-Flash, DeepSeek V4.1 Flash, Gemini 3.8 Flash, and GPT-5.6 Sol using a unified Harbor + Inspect framework across all seven environments (Harbor, n.d.; UK AI Security Institute, 2024). Agents interact with persistent business state through public tools, guided by skill documents and supplied scripts. Summary-based context compaction is triggered when estimated input length reaches 200,000 tokens, within an input capacity of 480,000 tokens (Anthropic, 2025). Business state and generated artifacts remain accessible through public tools. Appendix A specifies model configurations and execution settings.

Task set. Tasks span seven business environments and five bands: 64K, 128K, 256K, 512K, and 1M tokens, assigned by measured reference workload L_{ref}. Each band contains 210 tasks generated under approximately 100 parameter configurations spanning business scale, dependency structure, and update conditions. Appendix B details task sampling and construction profiles.

Tool budget. Each task receives a tool-attempt budget based on its public reference execution:

$$B = R + \lceil \lambda A \rceil, \qquad \lambda = 0.5,$$

where R counts all reference tool calls and A counts business-execution calls. The budget scales with reference workload and provides extra allowance for exploration and correction. Queries, business operations, and corrective attempts share this budget. Appendix A details accounting; Appendix A.6 reports the reference-budget audit.

Evaluation metrics. We evaluate task outcomes by checking the final environment state (Trivedi et al., 2024; Yao et al., 2024). Local-goal completion (LGC) measures the fraction of local business goals satisfied at termination, averaged equally across tasks. Task success (TS) measures the proportion of tasks satisfying all final requirements, including cross-artifact dependencies and delivery constraints. Appendix C.2 details the scoring and aggregation rules.

Table 1: Main results across seven environments and reference-workload bands (%). Local-goal completion (LGC) averages terminal native-goal fractions; task success (TS) measures full business acceptance. Both metrics use the same 210 executions per model and band, balanced across environments. Overall averages all 1,050 executions per model.

Model	Metric	64K	128K	256K	512K	1M	Overall
Grok 4.6	LGC ↑	95.5	93.1	89.2	78.7	73.4	86.0
	TS ↑	90.0	81.4	74.8	60.5	52.4	71.8
Qwen3.8-Flash	LGC ↑	94.3	83.1	73.1	61.6	48.9	72.2
	TS ↑	82.9	61.4	45.2	39.5	22.9	50.4
DeepSeek V4.1 Flash	LGC ↑	88.9	79.1	72.5	63.9	56.1	72.1
	TS ↑	74.8	53.8	43.8	37.1	26.2	47.1
Gemini 3.8 Flash	LGC ↑	92.3	81.3	64.1	38.5	21.3	59.5
	TS ↑	78.6	58.6	44.3	25.2	8.1	43.0
GPT-5.6 Sol	LGC ↑	80.9	63.4	56.2	44.9	34.1	55.9
	TS ↑	51.4	23.3	10.5	5.7	1.9	18.6

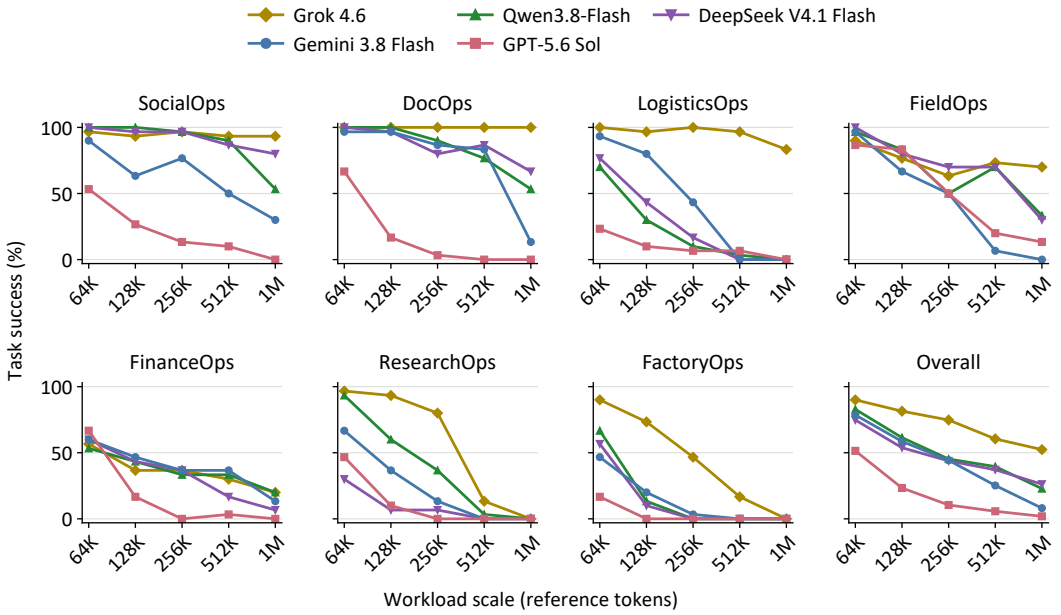

Figure 2: Task success across seven business environments and reference-workload scales. The Overall panel pools all seven environments with equal task weight. Colors and markers identify models; each environment point includes 30 assigned tasks and each Overall point includes 210. Table 5 reports task success and local-goal completion.

3.2 Main Results

Performance declines with workload, and model rankings vary across scales. As shown in Table 1, all five models achieve lower LGC and TS at 1M than at 64K. Grok leads overall with 86.0% LGC and 71.8% TS, but its LGC falls from 95.5% to 73.4%, and its TS from 90.0% to 52.4%. Qwen and DeepSeek have similar overall LGC, at 72.2% and 72.1%, respectively, yet Qwen leads by 5.4 percentage points at 64K while DeepSeek leads by 7.2 points at 1M. Their TS ranking also reverses. Overall averages therefore do not fully characterize performance on longer tasks.

High local-goal completion can coexist with low task success. From 64K to 1M, Gemini's FieldOps LGC falls from 98.6% to 4.6%, while TS declines from 96.7% to zero. GPT's FinanceOps LGC remains high, declining from 93.7% to 85.8%, while TS falls from 66.7% to zero.

Task success follows different workload-scaling patterns across environments. As shown in Figure 2, Grok, DeepSeek, and Qwen each retain task success above 50% at 1M in SocialOps and DocOps. LogisticsOps and FieldOps show wider differences between models: Grok reaches 83.3% and 70.0%, respectively, while the other models achieve zero success in LogisticsOps and at most 33.3% in FieldOps. In FinanceOps, no model exceeds 20.0% task success at 1M. In ResearchOps and FactoryOps, all five models reach zero task success at this scale.

4 Analysis

We analyze execution traces to examine progress under growing workloads, the delivery of local results, recovery after failed attempts, and the use of historical outputs across compaction boundaries. We also examine how task sampling affects the measured success rate.

4.1 Sustained Progress and Completion

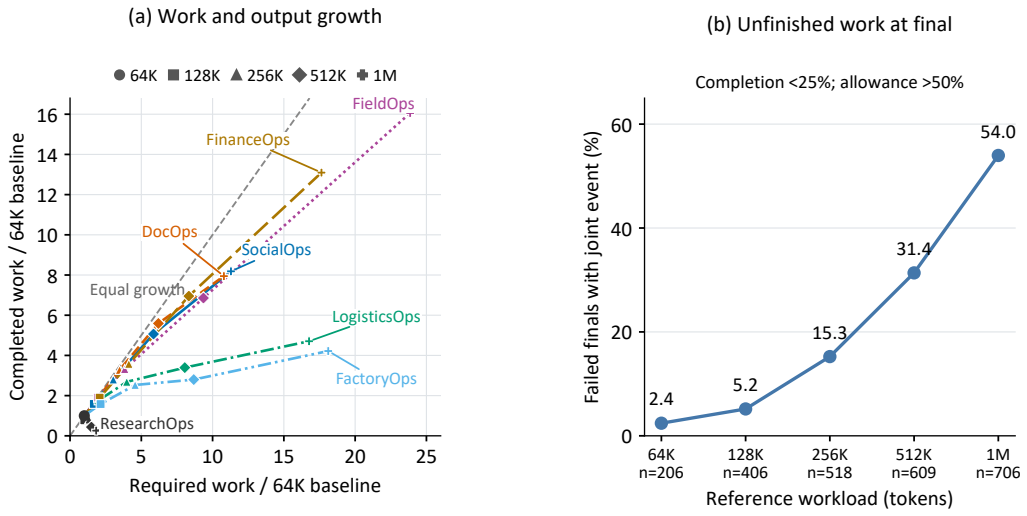

Figure 3: Growing requirements outpace effective output. (a) Required and completed goal counts across five workload bands, averaged over five models and separately normalized to each environment's 64K baseline. The gray diagonal marks equal proportional growth. (b) Percentage of failed finals with both measurements that complete less than 25% of goals while leaving more than 50% of the tool allowance unused.

Effective output does not keep pace with growing task requirements. Figure 3a traces the growth of required and completed work across the five workload bands. Averaged across models, the number of completed local goals increases from 64K to 1M in six environments, but grows more slowly than the number required. In ResearchOps, the average number of completed rounds decreases over this range even as task requirements grow. Mean LGC falls from 90.4% to 46.8%. The shortfall also becomes more widespread: the proportion of executions completing less than one-quarter of their local goals rises from 2.3% to 47.1%. Appendix F.1 reports absolute counts by model.

Agents increasingly stop with substantial work and tool allowance remaining. Among failed agent-submitted final answers with measured completion and tool use, the proportion completing less than 25% of local goals while retaining more than 50% of the tool allowance increases from 2.4% at 64K to 15.3% at 256K and 54.0% at 1M, as shown in Figure 3b. At the largest workload, more than half of these failed finals therefore leave most local goals unfinished without exhausting the tool-call budget. Appendix F.2 reports the counts and threshold sensitivity.

4.2 From Local Execution to Global Delivery

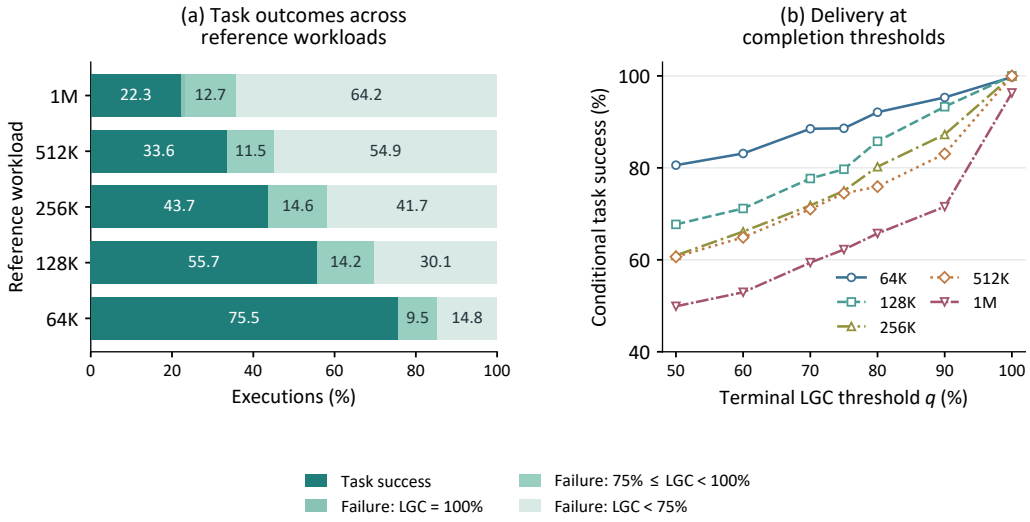

Figure 4: Local completion and global delivery across seven environments and five models. (a) Joint outcomes for all 5,250 executions, with 1,050 per workload band. (b) Task success among executions whose terminal LGC meets threshold q, shown separately for each workload band. Table 29 reports the numerators and denominators.

Substantial local progress does not guarantee complete task delivery. Across seven environments, 656 executions complete at least three-quarters of their local goals but leave the remaining work unfinished. These executions account for 9.5–14.6% of tasks within each workload band. Among executions ending with LGC of at least 80%, task success falls from 92.1% at 64K to 65.7% at 1M. Even at a 90% completion threshold, success falls from 95.3% to 71.6%. Thus, substantial local progress is less consistently accompanied by complete delivery at larger workloads.

Successful local computation does not ensure valid downstream use. In SocialOps, downstream submissions can reproduce the outputs of successful skill-script executions and still be rejected. The task-averaged rejection rate of these submissions rises from 3.4% at 64K to 16.5% at 1M. Most rejections concern historical-source requirements or current audience loads. These failures occur despite agreement between the submitted values and script outputs, showing that skill composition also depends on whether those outputs satisfy the receiving operation's business constraints.

4.3 Recovery over Long Horizons

Local recovery becomes less reliable as workloads grow. Figure 5a shows recovery within three to seven subsequent verification attempts for each model across the five workload bands. Extending this range from three to seven raises task-averaged recovery from 84.4% to 87.9% at 64K and from 69.6% to 73.0% at 1M, leaving a gap of approximately 15 percentage points. All five models exhibit lower recovery at 1M under both limits. Even with seven retries, the decline from 64K to 1M ranges from 8.4 percentage points for Grok to 22.0 points for GPT.

Continued execution yields smaller completion gains at longer horizons. Among executions with a business failure within the reference allowance, extending the observed budget prefix from $\lambda = 0$ to $\lambda = 1/2$ raises LGC from 61.0% to 88.6% at 64K, but from 51.2% to 56.9% at 1M. Task success increases from 13.6% to 70.8% at 64K and from 5.3% to 27.4% at 1M. Thus, longer prefixes capture additional complete deliveries at both scales, while the gain in local-goal completion is substantially smaller at 1M. Figure 5b–d reports completion, delivery, and failed-goal recovery for these executions at $B_\lambda = R + \lceil \lambda A \rceil$, retaining earlier terminal states. Appendix F.4 details the analysis.

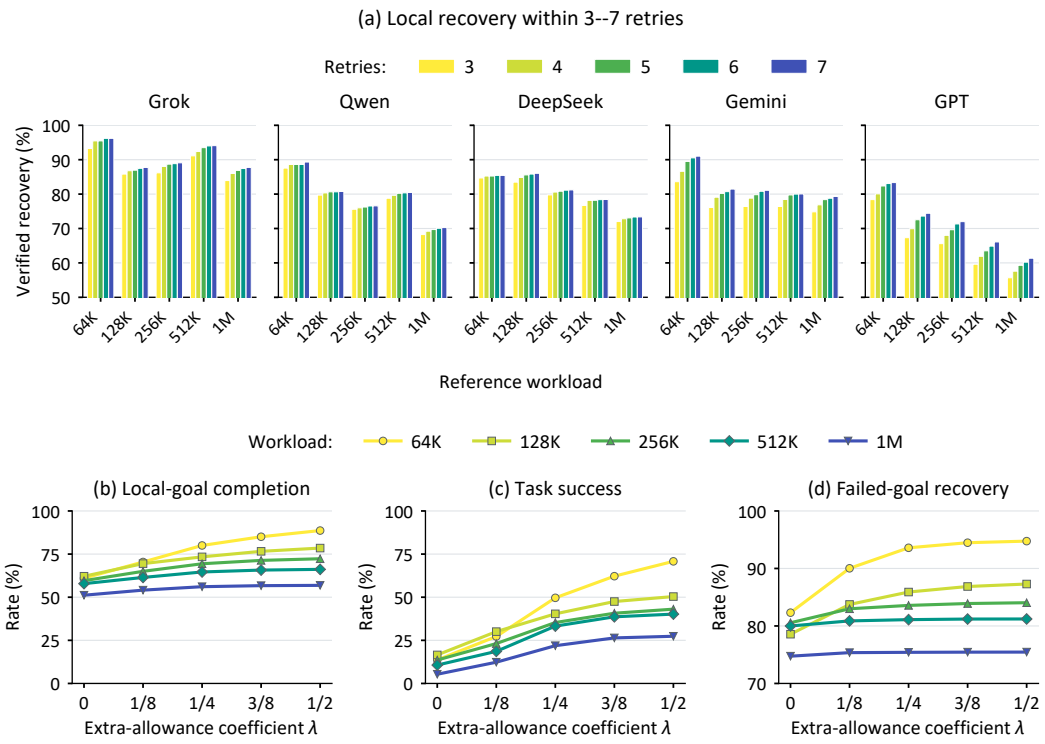

Figure 5: Recovery across workload scales. (a) Task-averaged recovery within 3–7 subsequent checks of the same goal and obligation cycle, shown by model and workload. (b–d) Local-goal completion, task success, and failed-goal recovery along fixed trajectories at $B_\lambda = R + \lceil \lambda A \rceil$, using the same 3,374 executions with public failures by R and retaining earlier terminal states. Appendix F.4 details the populations and counting rules.

4.4 Dependency Continuity across Compaction

Cross-compaction dependencies are associated with lower initial reliability and greater recovery effort. Across seven environments, first-check acceptance is 77.2% when required historical outputs have been removed from the retained context, compared with 84.5% when they remain verbatim available. As shown in Table 2, subsequent attempts narrow the acceptance gap to 1.9 percentage points, but average additional checks increase from 0.169 to 0.344. Recovery therefore closes much of the initial gap while requiring greater verification effort.

Dependency continuity

Metric	Retained	Removed
First-check acceptance (%)	84.5	77.2
Ever accepted (%)	91.8	89.8
Additional checks	0.17	0.34

Task-level outcomes

Metric	No compaction	With compaction
LGC (%)	37.0	71.3
TS (%)	23.4	41.1
Mean tool calls	303	1,161

Table 2: Local dependency checks and task-level outcomes across compaction.

Executions that undergo compaction sustain more business progress. As shown in Table 2, under matched evaluation settings, executions with compaction achieve 71.3% local-goal completion and 41.1% task success, compared with 37.0% and 23.4% for executions without compaction. They also involve substantially more tool calls, averaging 1,161 versus 303. These observations show that agents can continue toward delivery across compaction boundaries, with extra recovery effort to maintain dependency continuity. Appendix F.7 details the comparisons.

4.5 Task-Sampling Sensitivity

Evaluation results remain stable across task samples with a fixed workflow-family composition. We evaluate Gemini 3.8 Flash on 500 reference-validated FinanceOps tasks spanning nine configurations at 128K. Across 1,000 sampled orders, mean task success stays close to the full-set rate of 42.6%, while the central 90% sampling range narrows from 36.0–48.0% at 50 tasks to 41.0–44.3% at 300, as shown in Figure 6. Appendix G details the sampling procedure.

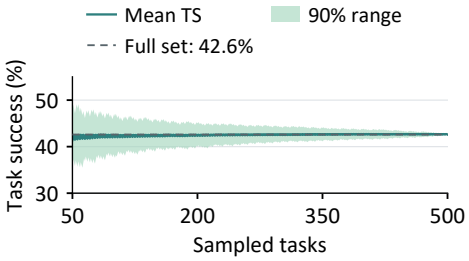

Figure 6: Mean and central 90% range across 1,000 family-balanced task orders.

5 Related Work

Long-horizon benchmarks. AgentLongBench studies information synthesis over extended histories (Fang et al., 2026), while LOCA-bench scales environment states to increase context without changing task semantics (Zeng et al., 2026). WorkArena++ and Toolathlon evaluate compositional workflows and cross-application execution (Boisvert et al., 2024; Li et al., 2025). AgencyBench assesses extended real-world tasks through deliverables and rubrics (Li et al., 2026b), while Long-Horizon-Terminal-Bench provides subtask-level grading (Li et al., 2026d). SkillHorizon focuses on expanding skill-grounded business activities, artifact dependencies, and shared-state constraints to evaluate local progress and whole-task delivery as requirements grow.

Agent skills. Reusable skills encode procedures that agents can apply across tasks. Voyager accumulates and composes executable skills (Wang et al., 2023), while the Agent Skills specification packages instructions with optional scripts and reference resources (Agent Skills, n.d.). SkillsBench compares task performance without skills, with curated skills, and with self-generated skills (Li et al., 2026c). AgentSkillOS combines hierarchical skill retrieval with DAG-based orchestration (Li et al., 2026a). These works study skills for task solving. SkillHorizon instead uses skills to define business activities and task dependencies, leaving execution plans to the agent.

MCP-based environments and benchmarks. MCP standardizes access to external tools and data sources (Model Context Protocol Contributors, 2025). MCP-Universe, MCP-Bench, and MCPMark evaluate agents through real-world tool integrations, multi-step coordination, and programmatic outcome checks (Luo et al., 2025; Wang et al., 2025; Wu et al., 2025). Related API-based benchmarks, including AppWorld and τ-bench, assess task completion through application-state and database-goal checks (Trivedi et al., 2024; Yao et al., 2024). However, expanding these task suites still involves task-specific generation, review, or verifier design, leaving systematic variation of workload and dependencies less directly supported.

6 Conclusion

We introduced SkillHorizon, a benchmark that constructs long-horizon business tasks by composing skill-grounded activities and their dependencies across seven executable environments. Configurable generators expand required work and its connections, while public reference executions establish feasibility and programmatic checks evaluate business outcomes. Across five models and 64K–1M reference workloads, effective progress falls behind growing task requirements. Trace analyses reveal gaps between successful local execution and accepted business delivery, together with heterogeneous recovery and cross-compaction dependency-use patterns. These findings highlight the importance of evaluating sustained progress and downstream validity alongside local execution success.

SkillHorizon provides a configurable testbed for studying how context management, skill use, and execution strategies support complete business workflows. Its generators support evaluation on new instances and larger workloads to assess whether improvements transfer across task scales.

Reproducibility Statement

Code and data are available at https://anonymous.4open.science/r/SkillHorizon-71E8/. The repository provides the Harbor + Inspect execution framework, task generators and procedural resources for seven environments, construction metadata for 1,050 tasks, terminal scores for 5,250 model–task executions, and scripts for recomputing the main results. Sections 2 and 3 describe construction and evaluation. Appendices A–D specify execution settings, workload counting, metrics, and domain construction; Appendix F details process measurements and cases.

Ethics Statement

Tasks use synthetic objects in isolated environments, with no changes submitted to real business services. Private evaluation data are separated from public resources. Raw execution records preserve operational events independently of the paper's outcome aggregation. External code and resources retain their provenance and applicable licensing requirements.

AI Use Statement

LLMs assisted with benchmark construction, language polishing, and organizing and anonymizing the code and data for the anonymous repository. The authors take responsibility for the final manuscript and released artifacts.

References

Agent Skills. Agent skills specification, n.d. URL https://agentskills.io/specification. Accessed September 26, 2026.

Anthropic. Effective context engineering for AI agents, September 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents. Published September 29, 2025.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. WorkArena++: Towards compositional planning and reasoning-based common knowledge work tasks. arXiv:2407.05291v2, 2024. URL https://arxiv.org/abs/2407.05291v2.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Shicheng Fang, Yuxin Wang, Xiaoran Liu, Jiahao Lu, Chuanyuan Tan, Xinchi Chen, Yining Zheng, Xuanjing Huang, and Xipeng Qiu. AgentLongBench: A controllable long benchmark for long-contexts agents via environment rollouts. arXiv:2601.20730v3, 2026. URL https://arxiv.org/abs/2601.20730v3.

Harbor. Harbor documentation, n.d. URL https://docs.harborframework.com/. Accessed September 26, 2026.

Infinite Context Compact. Infinite context compact. Software repository, README, n.d. URL https://github.com/zhaochen0110/Infinite_context_compact.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. arXiv:2603.02176v1, 2026a. URL https://arxiv.org/abs/2603.02176v1.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. arXiv:2510.25726v2, 2025. URL https://arxiv.org/abs/2510.25726v2.

Keyu Li, Junhao Shi, Yang Xiao, Mohan Jiang, Jie Sun, Yunze Wu, Dayuan Fu, Shijie Xia, Xiaojie Cai, Tianze Xu, Weiye Si, Wenjie Li, Dequan Wang, and Pengfei Liu. AgencyBench: Benchmarking the frontiers of autonomous agents in 1m-token real-world contexts. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7422–7440. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.337. URL https://aclanthology.org/2026.acl-long.337/.

Xiangyi Li, Wenbo Chen, Yimin Liu, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. arXiv:2602.12670v1, 2026c. URL https://arxiv.org/abs/2602.12670v1.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-Horizon-Terminal-Bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. arXiv:2607.08964v2, 2026d. URL https://arxiv.org/abs/2607.08964v2.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. MCP-Universe: Benchmarking large language models with real-world model context protocol servers. arXiv:2508.14704v1, 2025. URL https://arxiv.org/abs/2508.14704v1.

Model Context Protocol Contributors. Model context protocol specification, 2025. URL https://modelcontextprotocol.io/specification/2025-06-18. Protocol specification dated June 18, 2025. Accessed September 26, 2026.

OpenAI. tiktoken: A fast BPE tokeniser. Software repository, n.d. URL https://github.com/openai/tiktoken. Accessed September 26, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. arXiv:2407.18901, 2024. URL https://arxiv.org/abs/2407.18901.

UK AI Security Institute. Inspect AI: Framework for large language model evaluations. Software, May 2024. URL https://github.com/UKGovernmentBEIS/inspect_ai.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. arXiv:2305.16291v2, 2023. URL https://arxiv.org/abs/2305.16291v2.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. arXiv:2508.20453v1, 2025. URL https://arxiv.org/abs/2508.20453v1.

Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, Zirui Wang, Jinjie Ni, Yufan Yang, Arvin Xu, and Michael Qizhe Shieh. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use. arXiv:2509.24002v1, 2025. URL https://arxiv.org/abs/2509.24002v1.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. arXiv:2412.14161, 2024. URL https://arxiv.org/abs/2412.14161.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. arXiv:2406.12045, 2024. URL https://arxiv.org/abs/2406.12045.

Weihao Zeng, Yuzhen Huang, and Junxian He. LOCA-bench: Benchmarking language agents under controllable and extreme context growth. arXiv:2602.07962v1, 2026. URL https://arxiv.org/abs/2602.07962v1.

A Experimental Settings and Detailed Results

A.1 Tasks and Models

Table 3 summarizes task-family counts, public skill-library sizes, instance counts, and measured reference interaction lengths across the seven benchmark domains.

Table 3: Benchmark composition and measured reference interaction lengths. Skills count the complete public library available to each task. Each domain contains 30 instances in each of the five reference-workload bands. FieldOps families denote its two selected scenarios within one maintenance constructor.

Domain	Families	Skills	Instances	Reference tokens (min–max)
FinanceOps	3	12	150	56,667–1,016,542
FieldOps	2	9	150	38,116–1,028,809
SocialOps	3	8	150	62,226–1,036,721
LogisticsOps	3	9	150	54,900–1,044,025
ResearchOps	3	7	150	51,263–1,033,542
DocOps	3	6	150	51,462–1,046,945
FactoryOps	3	7	150	50,611–1,029,448

Construction profiles and per-band business-work ranges appear in Appendix B.1.

The requested model identifiers for the main evaluation matrix are `gemini-3.8-flash`, `gpt-5.6-sol`, `qwen3.8-flash`, `grok-4.6`, and `deepseek-v4.1-flash`. Official DeepSeek evaluations request `deepseek-flash`, the V4.1 Flash route. Each result retains its requested identifier and provider configuration.

Each main model–domain evaluation comprises 150 assigned task instances, with 30 per reference-workload band. Table 4 specifies the Gemini request configuration. GPT-5.6 Sol and Grok 4.6 also use the OpenLux Chat Completions route with requested medium reasoning and an 8,192-token output allowance. Official DeepSeek runs use `deepseek-flash`; requested medium reasoning maps to high on that interface, with a 32,768-token allowance including reasoning and visible output.

Table 4: Gemini request configuration. The model identifier specifies the requested provider route.

Setting	Value
Model identifier	`gemini-3.8-flash`
Provider interface	OpenLux Chat Completions
Reasoning effort	Medium
Maximum output	`max_tokens=8192`
Streaming and samples	`stream=false`, `n=1`
Temperature and top-p	Unspecified in the request
Compaction threshold	200,000 estimated input tokens
Local input capacity	480,000 tokens

OpenLux requests use `https://api.openlux.ai/v1/chat/completions`. DeepSeek uses its official API; Qwen uses the official DashScope API and an OpenLux route for a subset of FinanceOps executions. The evaluation uses the final selected set of 150 tasks in each environment and the same five model families throughout. Task identifiers and configuration records associate each result with its evaluated instance, model route, and execution settings.

A.2 Agent Execution

Actions and feedback. All seven environments are evaluated using the same Harbor + Inspect framework (Harbor, n.d.; UK AI Security Institute, 2024). Harbor provides task orchestration, while Inspect handles agent execution, model generation, and tool dispatch. Domain gateways expose a common action–observation interface and implement business operations and completion checks. Agents submit public tool actions or final answers through the configured text-JSON action protocol.

Arguments are specified before the corresponding tool feedback is returned, and batched actions execute in order. On the first public failure, the executed prefix and its feedback are returned; the unattempted suffix remains uncharged. Subsequent model turns can use newly returned values and revise the action sequence.

Procedural resources and environment access. Agents consult skill documents, invoke business tools, and run supplied scripts. A public gateway exposes these operations while keeping controller logs and hidden evaluation data separate from model-visible resources. Native tool-call conversion, agent-owned workspaces, and arbitrary program execution are disabled in this configuration. Environment operations update persistent business state and artifacts.

Context management. The main context configuration sets a 200,000-token estimated-input compaction threshold and a 480,000-token input capacity. At the threshold, the model summarizes older dialogue using its observed information, following the summary-based compaction pattern described by Anthropic (2025). The controller retains initial instructions and a fitting tail of complete action–result turns. The resulting summary replaces the older dialogue. Business state and artifacts persist across compaction and remain accessible through public tools. Continuation records preserve the message-boundary and summarization settings applied at each stage.

A.3 Budgets and Termination

We set $\lambda = 0.5$ and round the additional allowance upward to an integer. Task i therefore receives a shared tool-attempt budget

$$B_i = R_i + \lceil \lambda A_i \rceil = R_i + \lceil A_i/2 \rceil \,, \tag{5}$$

where R_i is the public reference's counted tool calls and $A_i \leq R_i$ its business execution calls. Budgeted attempts include queries, skill-document reads, fixed-script invocations, exploration, and identifiable rejected actions. Scripts count at their public invocation granularity. The protocol specifies declaration accounting consistently for reference and model execution. The B_ith action remains eligible for evaluation. Automatic transport retries are recorded in the infrastructure account.

Local failures are associated with a task, business object, independently verifiable obligation, and completion cycle. Valid negative observations, public rejections, and content errors detected by offline evaluation have distinct categories. The paper reports success, final failure, and tool-budget exhaustion, with the assignments specified in Appendix A.4. Resumed executions retain their task identity, saved state, and accumulated consumption. Appendix C.4 specifies aggregation across resumptions.

A.4 Detailed Results and Execution Conditions

Table 5 provides the complete environment–model–scale matrix underlying the overall scores in Table 1 and the task-success curves in Figure 2.

Result scope and endpoint accounting. Table 6 reports 2,424 successes, 2,657 final failures, and 169 tool-budget-exhaustion failures across 5,250 model–task executions. Final failure combines 2,558 unsuccessful agent-submitted final answers and 99 closed abnormal endpoints with failed business acceptance. Each execution contributes one terminal evaluation to LGC and TS. Final-answer analyses use the 2,558 observed unsuccessful submissions, with measurement coverage reported in Appendix F.2.

The complete matrix contains 175 environment–model–scale cells. Each cell contains 30 terminal evaluations for both LGC and TS, using the definitions in Appendix C.2.

Execution conditions. Checkpoint continuation preserves business state, dialogue, tool consumption, and usage. Per-execution configuration records specify output allowances, compaction settings, local-failure and time limits, and protocol-recovery feedback. Cumulative accounting counts each executed action once across checkpoints. Terminal budget utilization uses the recorded budget-charged attempts, with declaration accounting applied according to the execution configuration.

Table 5: Local-goal completion / task success (LGC / TS, %) by environment and workload. Both metrics use all 30 executions per cell. LGC averages native terminal-goal fractions; TS requires full business acceptance.

Environment	Model	64K	128K	256K	512K	1M
SocialOps	Grok 4.6	98.9 / 96.7	98.0 / 93.3	96.7 / 96.7	96.8 / 93.3	99.3 / 93.3
	Qwen3.8-Flash	100.0 / 100.0	100.0 / 100.0	97.8 / 96.7	95.6 / 90.0	80.7 / 53.3
	DeepSeek V4.1 Flash	100.0 / 100.0	98.7 / 96.7	97.8 / 96.7	96.9 / 86.7	90.3 / 80.0
	Gemini 3.8 Flash	96.7 / 90.0	92.7 / 63.3	90.7 / 76.7	80.4 / 50.0	47.4 / 30.0
	GPT-5.6 Sol	70.0 / 53.3	54.0 / 26.7	46.9 / 13.3	33.5 / 10.0	20.6 / 0.0
LogisticsOps	Grok 4.6	100.0 / 100.0	99.1 / 96.7	100.0 / 100.0	99.7 / 96.7	89.2 / 83.3
	Qwen3.8-Flash	78.5 / 70.0	58.1 / 30.0	26.5 / 10.0	12.0 / 3.3	3.6 / 0.0
	DeepSeek V4.1 Flash	77.5 / 76.7	70.5 / 43.3	51.4 / 16.7	30.1 / 0.0	12.8 / 0.0
	Gemini 3.8 Flash	96.7 / 93.3	88.5 / 80.0	57.1 / 43.3	2.0 / 0.0	0.4 / 0.0
	GPT-5.6 Sol	75.0 / 23.3	60.6 / 10.0	54.6 / 6.7	38.1 / 6.7	14.3 / 0.0
ResearchOps	Grok 4.6	98.9 / 96.7	97.8 / 93.3	90.8 / 80.0	41.3 / 13.3	22.3 / 0.0
	Qwen3.8-Flash	95.6 / 93.3	84.4 / 60.0	70.0 / 36.7	34.7 / 3.3	18.7 / 0.0
	DeepSeek V4.1 Flash	58.9 / 30.0	38.9 / 6.7	29.2 / 6.7	14.8 / 0.0	8.1 / 0.0
	Gemini 3.8 Flash	86.7 / 66.7	62.2 / 36.7	43.3 / 13.3	17.8 / 0.0	6.2 / 0.0
	GPT-5.6 Sol	76.7 / 46.7	37.8 / 10.0	18.3 / 0.0	10.2 / 0.0	3.7 / 0.0
DocOps	Grok 4.6	100.0 / 100.0	100.0 / 100.0	100.0 / 100.0	100.0 / 100.0	100.0 / 100.0
	Qwen3.8-Flash	100.0 / 100.0	100.0 / 100.0	99.7 / 90.0	93.8 / 76.7	87.2 / 53.3
	DeepSeek V4.1 Flash	100.0 / 100.0	99.5 / 96.7	98.7 / 80.0	99.6 / 86.7	97.2 / 66.7
	Gemini 3.8 Flash	98.9 / 96.7	99.3 / 96.7	92.9 / 86.7	86.7 / 83.3	32.8 / 13.3
	GPT-5.6 Sol	83.9 / 66.7	61.8 / 16.7	62.7 / 3.3	54.4 / 0.0	40.6 / 0.0
FactoryOps	Grok 4.6	98.1 / 90.0	93.8 / 73.3	81.8 / 46.7	60.5 / 16.7	56.3 / 0.0
	Qwen3.8-Flash	90.0 / 66.7	58.3 / 13.3	44.9 / 0.0	31.7 / 0.0	9.2 / 0.0
	DeepSeek V4.1 Flash	89.7 / 56.7	64.1 / 10.0	48.7 / 0.0	22.4 / 0.0	14.4 / 0.0
	Gemini 3.8 Flash	78.7 / 46.7	52.0 / 20.0	26.2 / 3.3	5.5 / 0.0	11.5 / 0.0
	GPT-5.6 Sol	71.1 / 16.7	44.0 / 0.0	33.9 / 0.0	16.9 / 0.0	7.8 / 0.0
FinanceOps	Grok 4.6	81.0 / 56.7	70.1 / 36.7	65.1 / 36.7	58.2 / 30.0	55.8 / 20.0
	Qwen3.8-Flash	96.7 / 53.3	86.4 / 43.3	85.9 / 33.3	79.9 / 33.3	74.5 / 20.0
	DeepSeek V4.1 Flash	96.0 / 60.0	88.5 / 43.3	89.3 / 36.7	89.8 / 16.7	85.0 / 6.7
	Gemini 3.8 Flash	90.3 / 60.0	85.7 / 46.7	76.6 / 36.7	66.7 / 36.7	46.3 / 13.3
	GPT-5.6 Sol	93.7 / 66.7	91.0 / 16.7	90.4 / 0.0	90.7 / 3.3	85.8 / 0.0
FieldOps	Grok 4.6	91.6 / 90.0	92.7 / 76.7	89.9 / 63.3	94.0 / 73.3	91.1 / 70.0
	Qwen3.8-Flash	99.6 / 96.7	94.2 / 83.3	87.1 / 50.0	83.2 / 70.0	68.7 / 33.3
	DeepSeek V4.1 Flash	100.0 / 100.0	93.4 / 80.0	92.3 / 70.0	93.4 / 70.0	84.7 / 30.0
	Gemini 3.8 Flash	98.6 / 96.7	88.4 / 66.7	61.9 / 50.0	10.6 / 6.7	4.6 / 0.0
	GPT-5.6 Sol	95.9 / 86.7	94.4 / 83.3	87.0 / 50.0	70.5 / 20.0	65.6 / 13.3

Final-answer confirmation. A neutral confirmation prompt follows a proposed final answer. A confirming final ends execution, while a counted business-tool attempt restarts the confirmation cycle. The prompt uses public execution information.

A.5 Execution Accounting

Table 7 summarizes the five-environment complete-history cohort used in the budget audit below. Tool counts and utilization cover 3,621 trajectories; adopted-compaction counts are identifiable for 3,616, with five Gemini trajectories unresolved. Tool dispatches, budget-counted attempts, and model API requests have separate counters, accumulated across resumptions. Seven-environment terminal progress and failed-final allowance measurements appear in Appendices F.1 and F.2.

A.6 Budget Scale and Recorded Execution Cost

This reference-budget audit covers SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps. The reference policy supplies both total calls R and business calls A. As shown in Table 8, with

Table 6: Three-category outcome accounting for 150 assigned tasks per model–environment group. S: success; F: final failure; B: tool-budget exhaustion. The assignment rule in Appendix A.4 applies to all groups; each model–task pair contributes one final result.

Environment	Model	S	F	B
SocialOps	Grok 4.6	142	7	1
	Qwen3.8-Flash	132	18	0
	DeepSeek V4.1 Flash	138	12	0
	Gemini 3.8 Flash	93	56	1
	GPT-5.6 Sol	31	112	7
LogisticsOps	Grok 4.6	143	7	0
	Qwen3.8-Flash	34	116	0
	DeepSeek V4.1 Flash	41	109	0
	Gemini 3.8 Flash	65	83	2
	GPT-5.6 Sol	14	122	14
ResearchOps	Grok 4.6	85	61	4
	Qwen3.8-Flash	58	92	0
	DeepSeek V4.1 Flash	13	137	0
	Gemini 3.8 Flash	35	115	0
	GPT-5.6 Sol	17	132	1
DocOps	Grok 4.6	150	0	0
	Qwen3.8-Flash	126	24	0
	DeepSeek V4.1 Flash	129	21	0
	Gemini 3.8 Flash	113	35	2
	GPT-5.6 Sol	26	91	33
FactoryOps	Grok 4.6	68	43	39
	Qwen3.8-Flash	24	123	3
	DeepSeek V4.1 Flash	20	125	5
	Gemini 3.8 Flash	21	126	3
	GPT-5.6 Sol	5	119	26
FinanceOps	Grok 4.6	54	90	6
	Qwen3.8-Flash	55	94	1
	DeepSeek V4.1 Flash	49	100	1
	Gemini 3.8 Flash	58	91	1
	GPT-5.6 Sol	26	113	11
FieldOps	Grok 4.6	112	31	7
	Qwen3.8-Flash	100	50	0
	DeepSeek V4.1 Flash	105	45	0
	Gemini 3.8 Flash	66	84	0
	GPT-5.6 Sol	76	73	1

Table 7: Cumulative execution accounting for SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps. Compactions report adopted replacements over the identifiable-count trajectories shown in parentheses. Utilization is the median percentage of assigned tool allowance used.

Model	Trajectories	Tool attempts	Compactions	Utilization (%)
Grok 4.6	721	644,922	3,070 (721)	78.7
Qwen3.8-Flash	749	487,882	1,285 (749)	74.5
DeepSeek V4.1 Flash	691	534,825	2,347 (691)	76.2
Gemini 3.8 Flash	736	325,119	1,152 (731)	75.9
GPT-5.6 Sol	724	521,087	1,953 (724)	83.3

$B = R + \lceil A/2 \rceil$, the median budget exceeds the reference count by 37.6–44.4% across these environments. All 750 reference records in this audit satisfy the formula, and their 3,750 exported endpoint budgets match their assigned values.

Among the endpoint records, 3,621 have complete merged tool histories whose non-declaration attempts agree with cumulative charged counts. These include 1,717 successes, 1,764 final failures, and 140 budget-exhaustion outcomes. Table 9 reports execution cost on the successful subset. Median

Table 8: Reference execution and assigned budget scales. R ranges span all selected tasks in each environment; A/R and B/R are within-environment medians under $B = R + \lceil A/2 \rceil$.

Environment	Tasks	R range	A/R (%)	B/R
SocialOps	150	132–1,476	87.0	1.435
LogisticsOps	150	176–3,305	75.1	1.376
ResearchOps	150	229–4,638	82.0	1.410
DocOps	150	89–1,173	88.4	1.444
FactoryOps	150	135–2,691	86.5	1.433

Table 9: Execution cost of successful trajectories with complete histories. U is the cumulative charged-attempt count and R the reference call count. All observations use the recorded $\lambda = 0.5$ budget.

Model	n	Median U/R	P90 U/R
Grok 4.6	585	1.12	1.28
Qwen3.8-Flash	374	1.10	1.23
DeepSeek V4.1 Flash	341	1.15	1.28
Gemini 3.8 Flash	324	1.17	1.33
GPT-5.6 Sol	93	1.21	1.38

cost ranges from 1.10 to 1.21 times the reference count across models, including information access, alternative execution choices, and correction.

B Construction Profiles and Workload Measurement

B.1 Reference Workload and Task Selection

Tables 10 and 11 describe the selected release manifests. Required work is measured in financial work packages, devices, posts, bookings, source works, pages, or production orders. Scale controls include financial stages and source width, service-network size, campaign waves, route legs, evidence rounds, document depth, and production levels.

Table 10: Construction profiles of the selected evaluation set. Ranges cover 150 tasks per domain. Work units and structural quantities retain their domain-specific meanings. The coverage column counts tasks with the reported dependency field.

Domain	Work units	Scale setting	Dependency quantity	Coverage
FinanceOps	24–1,152 work packages	4–24 logical stages	0–23 evidence span (stages)	150/150
FieldOps	9–768 devices	8–446 services	4–8 service depth	90/150
SocialOps	9–111 posts	2–13 waves	1–11 parent span (waves)	150/150
LogisticsOps	16–320 bookings	4–8 legs	1–6 custody hops	150/150
ResearchOps	15–395 source works	3–6 rounds	3–6 used calibration depth	150/150
DocOps	6–105 pages	2–35 configured depth	2–89 join nodes	150/150
FactoryOps	12–272 orders	2–5 levels	12–1,740 unit-level material links	150/150

Dependency quantities retain their native meaning: public-evidence stage span in FinanceOps, service-network depth in FieldOps, longest parent-wave span in SocialOps, freight custody hops in LogisticsOps, used calibration-map depth in ResearchOps, join nodes in DocOps, and unit-level material links in FactoryOps. FinanceOps measures the stage span of reused evidence alongside cumulative stage-closure dependencies. FieldOps service depth spans 4–8 in the 90 tasks with this recorded attribute. Multiple material units can link the same pair of FactoryOps production orders. The DocOps references record 200 update events across 100 tasks, affecting 2–29 published pages

Table 11: Business work, stage settings, and reference execution across workload scales. Entries give minimum–maximum ranges within each environment–scale group. Stage settings retain their domain-specific meanings. Business calls (A) count reference business-execution invocations, including supplied scripts at their public call granularity. Table 10 reports additional dependency attributes.

Environment	Attribute	64K	128K	256K	512K	1M
SocialOps	Posts	9	15	27–30	48–57	90–111
	Waves	2	3	3–4	6–7	10–13
	Business calls	109–117	180–188	316–361	563–672	1,055–1,306
LogisticsOps	Bookings	16	32	60–70	120–144	224–320
	Legs	4	4	5	6	8
	Business calls	122–139	248–288	519–623	1,074–1,187	2,232–2,560
ResearchOps	Source works	15–21	39	76–92	165–185	365–395
	Rounds	3	3	4	4–5	5–6
	Business calls	183–232	379–434	784–900	1,635–1,825	3,498–3,850
DocOps	Pages	6–10	10–19	19–36	31–63	52–105
	Configured depth	2–4	4–7	7–12	11–21	18–35
	Business calls	73–99	121–195	225–355	385–603	681–1,083
FactoryOps	Orders	12–14	20–28	48–60	65–120	140–272
	Levels	2	3–4	3–5	4–5	4–5
	Business calls	109–132	173–276	413–598	586–1,163	1,214–2,373

and retaining 3–56 per event. FactoryOps references record 0–60 rework attempts per task. Agents encounter updates according to their own execution state.

B.2 Generation Rules

Tables 12 and 13 connect configuration parameters to business requirements and workload calibration. Each sampling slot specifies an environment, family, and target band. Candidates retain their configuration, seed, reference outcome, and exact measured length; selection uses business-valid reference executions within the target interval.

FinanceOps contains 150 tasks across three workflow families: heterogeneous integration, receivables reconciliation, and release maintenance, with 50 tasks per family. Nine variants yield 45 configurations after removing seeds. A seeded generator creates source records, contracts, identity mappings, adjustment evidence, invoices, and remittances. Public feed-selection metadata supports contract and mapping selection. Calibration adjusts source width, and the selected tasks contain 4, 6, 10, 16, and 24 logical stages across the five bands.

FieldOps contains 150 tasks, with 15 normal and 15 recovery tasks per band. Its generator creates devices, service networks, maintenance windows, crews, and spare stock. Configurations vary device counts, repair properties, parent joins, supply conditions, and local faults. The selected 1M tasks span 176–446 services and 294–768 devices. Public references encounter no updates in the 75 normal tasks, two supply corrections in 50 recovery tasks, and one local fault in the other 25. These scenarios assess routine network completion and maintenance under changing operating conditions.

SocialOps calibrates brief counts under a fixed reference policy and uses deterministic successor seeds for out-of-band candidates. LogisticsOps predeclares five candidate seeds per slot to accommodate integer cohort sizes. DocOps, ResearchOps, and FactoryOps select among configured size candidates. Evaluation seeds are disjoint from development seeds. The evaluated releases are SocialOps V3 with audience-set scoring V2, LogisticsOps V3, ResearchOps V2, DocOps V2, and FactoryOps V1. ResearchOps uses compact public returns and retains at least one original study per topic in the terminal study set.

B.2.1 Length Bands and Counting

Reference calls and results are serialized separately as compact UTF-8 JSON with sorted keys and tokenized with the `o200k_base` encoding in tiktoken (OpenAI, n.d.). The workload counter

Table 12: Construction controls and the execution requirements they introduce. Controls are instantiated through domain-specific business rules.

Factor	Configuration or sampled facts	Execution requirement
Work scope	Financial work packages, devices, posts, bookings, studies, pages, or production orders	Complete the required local work and global handoff
Dependency organization	Stages, rounds, waves, parent links, joins, and shared resources	Bind predecessor outputs and schedule feasible operations
Business decisions	Audience rights, evidence status, document purposes, and operating conditions	Select applicable procedures and instance-specific inputs
State changes	Correction, withdrawal, expiry, disruption, or rework conditions	Reassess affected work and preserve applicable results

Table 13: Generation rules for the selected releases. Fixed factors apply within a sampling slot. Calibration changes required business work and records the resulting configuration.

Domain	Dependency construction	Fixed factors and scale calibration
FinanceOps	Stage closures join source outputs and the preceding closure. Family-specific links reuse identity mappings, original invoices, or component decisions; source corrections can invalidate dependent outputs and closures.	Fix workflow family, variant, stage count, and seed during calibration; adjust source width using the public reference workload. Business facts and opaque identifiers vary independently.
FieldOps	Service parents determine pressure and certification requirements. Configurations vary joins, physical device facts, and supply or local-fault updates.	Fix profile and scenario; calibrate service/device counts with the public reference. The selected set includes 75 normal and 75 recovery tasks; 1M tasks contain 176–446 services.
SocialOps	Prior campaign reports provide winning content and audiences. Staggered, distributed, alternating fork–join, and protected-branch rules vary parent origins and joins across waves.	Fix scenario and dependency pattern. Campaigns, waves, and the final wave's campaign count determine brief count. The V3 recalibration adjusts brief counts and successor seeds only for out-of-band slots.
LogisticsOps	Multi-leg bookings share ground transfers across cohorts. Staggered or distributed rules vary baggage delivery spans and cohort joins; physical custody connects successive movements.	Fix family, topology, route legs, and a five-seed candidate pool. Adjust cohort count using reference length; bookings grow with cohorts and legs.
ResearchOps	Acquisition rounds introduce calibration anchors and studies. Studies cite current or earlier anchors; shared evidence, exclusions, and withdrawals affect later synthesis.	Fix family, seed, sections, rounds, aliases, exclusions, and topology. Adjust studies per section; terminal-study trimming provides fine-grained adjustment while preserving topic evidence.
DocOps	Each noninitial page consumes its previous-stage counterpart. Chain–join adds periodic cross-lane parents; braided adds eligible cross-lane parents at each level; distributed also adds older parents. Amendment scenarios preserve an unaffected lane.	Fix family, seed, and topology. Adjust document count n; configured depth is derived as $\lceil n/3 \rceil$. Pages carry purpose, expiry, and shared-capacity conditions.
FactoryOps	Every order in a preceding level supplies a later order. A second parent comes from the preceding level, any earlier level, or the root level, according to the topology. Material quantities propagate upstream.	Fix family, seed, levels, topology, and terminal quantities. Adjust orders per level. Required upstream quantities and material-unit consumption follow the generated supply graph.

Table 14: Reference-workload bands, with an exclusive lower bound and inclusive upper bound. K denotes 1024 tokens and M denotes 1024^2 tokens. Each instance retains its measured reference length.

Band	Lower bound	Upper bound
64K	0	65,536
128K	65,536	131,072
256K	131,072	262,144
512K	262,144	524,288
1M	524,288	1,048,576

includes public tool names, arguments, and returned observations, counting each occurrence once, including repeat queries. Its scope excludes initial instructions, schemas, closing narration, compaction summaries, retransmitted request prefixes, hidden reasoning, and fixed-script internals. Reference policies act on publicly observed state; each reference records its policy version and query strategy.

Query strategy. Reference policies reuse valid observations and query additional information as needed. After state changes, they reassess affected dependencies before reusing prior results.

B.3 Agent History and Active Context

The reference count L_{ref} determines a task's fixed band. Applying the same call-and-result counter to the evaluated execution gives L_{agent}, including skill-document reads and repeated queries. Active context C_t counts the contents of request t, including instructions, actions, observations, and summaries; $C_{\mathrm{peak}} = \max_t C_t$. Provider usage accumulates input and output across requests, including retransmitted prefixes and summary generation. Compaction changes active context while leaving the reference-workload band unchanged.

C Procedural Resources and Measurement Definitions

C.1 Skill Packages and Execution Identity

Procedural resources use the `SKILL.md` convention with optional scripts and references (Agent Skills, n.d.). Each `SKILL.md` specifies applicability, required inputs, procedure, and expected effects. Optional fixed scripts compute over inputs explicitly supplied by the agent. The agent selects those inputs and uses returned results in later business operations. Script-return status, consumer acceptance, and terminal business completion are recorded separately.

Where the declaration protocol is enabled, `skill_id` names a reusable procedure and `run_id` names a declared execution. Business calls may carry `skill_run_id`. A `finish_skill` declaration records the agent's status claim: `completed`, `blocked`, or `abandoned`. Business acceptance is determined by executed checks. Script analyses use observed invocation events; recovery analyses reconstruct business-goal cycles from object identities, actions, and verification outcomes. Explicit run identities support attribution where available.

C.2 Outcomes and Business Obligations

For N evaluated tasks, let $s_i \in \{0, 1\}$ indicate business success, and let g_i and G_i be completed and required native goals, including unattempted goals. Both measurements are available for every execution:

$$\mathrm{TS} = \frac{100}{N} \sum_{i=1}^{N} s_i, \qquad \mathrm{LGC} = \frac{100}{N} \sum_{i=1}^{N} \frac{g_i}{G_i}. \tag{6}$$

Each task contributes equally. Native goals are financial work packages, restored devices with current passing tests, campaign briefs, network-verified bookings, evidence rounds, document pages, and quantity-complete production orders. FinanceOps counts one work package per source, requiring its correct current outputs and any required historical view; stage accounting and final publication remain additional task-level requirements. FieldOps counts device-level verdicts, not commissioned services:

full acceptance additionally checks closed isolations, resource accounting, and current service commissions. The other environments' completion checks are specified in Appendix F. All 5,250 executions contribute to both metrics; each environment–model–band cell contains 30 executions.

Local and global completion. A required object–goal pair is counted once at termination, under its applicable version. A page may have an accepted review and current publication while its required tracker still points to an older publication. Local operation success therefore leaves a separate delivery obligation. Conditional obligations enter the denominator when applicable; terminal evaluation includes required work that the agent never attempted.

Artifact flow. Realized producer–consumer links connect the artifact versions actually produced and consumed. Input validity depends on identity, scope, provenance, content, version, and current business state. We retain operation order and model-decision boundaries: actions in one precommitted batch execute sequentially, but later arguments cannot have been chosen in response to earlier receipts from that batch. Shared-resource constraints are checked in business state. Appendix D.2 gives the domain rules.

C.3 Failure Correction and Update Recovery

A business-goal cycle groups attempts toward the same object and independently checkable requirement. Parameter edits, new artifacts, and repeated failures remain in that cycle until acceptance or observation cutoff. A new update cycle requires a completed goal followed by actual invalidation. Failure correction begins with a failed business attempt; update recovery begins with invalidation. Recovery requires a corrective action followed by verified local acceptance.

Recovery fractions use all observed opportunities of the stated kind, including unrecovered cycles. Unresolved verdicts and fully classified subsets are reported separately. Public rejections, content errors detected offline, and provider retries retain distinct categories. Event intervals report the relevant executed operations; elapsed calls can include interleaved work and are not automatically attributable repair cost. Appendix F.5 specifies the measured recovery cohorts.

C.4 Aggregation and Observation Coverage

Each assigned task contributes one outcome under Appendix A.4. Continuations preserve task identity, saved state, dialogue, and accumulated consumption; executed prefixes are counted once. Transport recovery and provider usage have separate accounts. Event-level metrics use observed identities and checks, with their measurement coverage reported alongside the results.

Process ratios are computed within tasks before aggregation. Native units are not summed across environments. Each analysis states its eligible population, weighting, and resampling unit. Stopping analyses use actual failed final answers; context visibility uses verified replacements and subsequent requests.

D Domain Construction and Dependency Analysis

Domain generators instantiate executable business objects and initial states from resource templates and seeded configurations. These include financial sources, service-maintenance networks, campaign assets, itineraries, research evidence, documents, and manufacturing resources. Table 15 summarizes the seven environments and their representative dependency mechanisms.

Table 15: Seven executable business environments and representative dependency mechanisms. Skill-supported activities produce results that constrain the inputs, choices, and resources of subsequent work.

Environment	Skill-supported work	Dependency mechanism
FinanceOps Sources, invoices, stage closures	Normalize and reconcile Close and publish	**Versioned financial evidence.** Stage closures combine required source outputs and preceding closures. Corrections invalidate affected results while immutable historical views remain distinct from current outputs.
FieldOps Devices, services, test records	Plan and repair Test and commission	**Pressure and certification dependencies.** Current upstream records determine downstream inlet conditions. Supply changes require recertification; local device faults require affected repairs and certifications to be renewed.
SocialOps Campaigns, audiences, reports	Create campaigns Publish and evaluate	**Outcome-conditioned inputs.** Later campaigns use winning content and audiences from specified parent reports, respect permissions, and incorporate newly consented audiences.
LogisticsOps Itineraries, bookings, baggage	Book and transfer Transport baggage	**Custody chains.** Prior releases or manifests establish baggage location and readiness; subsequent transport must satisfy connection times, handling eligibility, and shared capacity.
ResearchOps Evidence, studies, calibration results	Acquire evidence Calibrate and synthesize	**Calibration composition.** New calibrations compose earlier mappings. Downstream synthesis binds current ancestor evidence under metric, population, and unit compatibility requirements.
DocOps Pages, revisions, publications	Reconcile and revise Review and publish	**Contract composition.** Downstream publications combine parent contracts under purpose, expiry, and allocation constraints. Publication validity and downstream binding validity are checked separately.
FactoryOps Orders, materials, inspection results	Produce and inspect Deliver outputs	**Material-flow dependencies.** Downstream demand determines upstream quantities and initial stock. Later orders consume produced units subject to lineage and quality requirements.

D.1 Environment Resources and Instantiation

The seven environments instantiate business objects, dependencies, shared constraints, and update conditions from domain resources. SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps use seeded databases containing 89,094 background records, documented as synthetic or mixed (Infinite Context Compact). FinanceOps generates financial sources and reconciliation evidence; FieldOps generates devices, service networks, maintenance windows, crews, and spares. Table 16 summarizes these resources, and Table 10 reports task-specific workloads.

Each domain supplies skill documents describing reusable procedures and, where applicable, fixed scripts over explicit inputs. Agents select inputs and invoke public operations; the resulting business state determines completion. Appendix C describes the skill packages, and Appendix B.1 specifies the evaluated task families and configurations.

D.2 Domain-Specific Stage Construction

Business modules group local work around required outcomes. A module can contain several obligations, and satisfying an obligation can require multiple operations. The construction graph specifies required upstream results; agents choose operation sequences that meet these dependencies and shared business constraints. Table 17 summarizes these relationships; the paragraphs below give their domain-specific construction rules. Native scoring units are defined in Appendix C.2.

Table 16: Construction resources for the seven environments. Numeric counts describe the five background databases, including relationship and event records; generated instance populations are reported separately.

Environment	Representative resources	Background records
FinanceOps	Generated feeds, identity mappings, adjustment evidence, invoices, and remittances	Generated
FieldOps	Generated devices, service networks, maintenance windows, crews, and spares	Generated
SocialOps	4,200 posts; 180 campaigns; 900 assets	20,734
LogisticsOps	850 passengers; 320 flights; 4,100 bookings	21,150
ResearchOps	1,800 papers; 900 authors; 2,000 citations	11,160
DocOps	140 projects; 1,600 pages; 6,400 document blocks	22,612
FactoryOps	18 robot cells; 180 programs; 1,800 waypoints	13,438

Valid extension. A valid extension introduces required work into the task's dependency structure. Source modules consume initial resources; other modules use required upstream results. Input identities and receiving requirements remain compatible, resource assignments and calendars admit a feasible execution, and the extended goal includes the new outcomes. Dynamic families specify how updates affect existing results and subsequent work. Public reference execution validates the resulting instance.

Financial sources and stage closures. FinanceOps adds source work packages within logical stages. Its three families cover heterogeneous data integration, release maintenance, and receivables reconciliation. Agents select format and identity rules, commit evidence-backed component adjustments, or reconcile remittances against original invoices and credit evidence. Stage closures combine the required current outputs with the preceding closure, and final review and publication cover the whole release. Local completion requires semantically correct current source outputs, including all required consumer views, to be incorporated into the stage closure. Checks cover both numerical results and identity or payment bindings. Source corrections can require affected outputs and downstream closures to be rebuilt, while immutable historical views and unrelated valid work are retained. Business facts and opaque identities vary independently, and compact public decision metadata supports input selection.

Maintenance and service commissioning. FieldOps expands networks of device-maintenance and service-commissioning obligations. A downstream service consumes current upstream commissioning records whose return pressures and remote-feed flows determine its inlet conditions. Repairs, tests, restoration, and commissioning respect maintenance windows, qualified crews, compatible spares, and shared flow capacity. Local completion counts restored devices with current passing tests; final system acceptance also checks the service network and its resource commitments. Configurations vary devices per service, parent joins, and physical requirements. The selected set contains 75 normal and 75 recovery tasks. Recovery scenarios introduce two progress-triggered supply changes or a local device fault. Supply changes require renewed certification; local faults revise repair conditions and can invalidate downstream certifications. Repairs remain reusable under the applicable device revision, and unrelated valid work is retained. Tasks with corrective re-isolation additionally require scheduling and dependency checks when reopening restored boundaries.

Campaign waves. SocialOps links later campaign briefs to specified measured reports. New variants combine the actual parent winners' audiences, apply arm-specific permissions, and include newly consented audiences. Content decisions use the winning calls to action. A wave adds variant publication, measurement, reporting, and deployment obligations. Its reports supply inputs to later waves. Scheduling and exposure conditions depend on the wave configuration, and rights updates require affected unpublished drafts to obtain valid assets and approvals.

Itinerary and custody chains. LogisticsOps creates bookings for additional route legs and baggage obligations connecting release and delivery points. A freight movement consumes a release or a preceding manifest that places the bag at its departure hub; its manifest supports the next movement. The generator supplies route-specific services, schedules, handling permissions, and shared capacities,

Table 17: Stage construction and the business conditions governing connected work. Conditions apply to the corresponding task families.

Environment	Added work and upstream inputs	Feasibility and validity conditions
FinanceOps	Logical stages add source work packages. Stage closures combine required current source outputs with the preceding closure; family-specific links reuse identity mappings, component decisions, or original invoices.	Exact semantic bindings and current versions govern acceptance. Corrections can invalidate affected outputs and downstream closures; immutable historical views remain distinct from current results. Final publication covers the complete release.
FieldOps	Service networks add device maintenance and commissioning obligations. Current upstream commissioning records determine downstream inlet conditions and admissible maintenance windows.	Restored devices require current passing tests. Commissioning also checks current upstream identities, supply revision, and shared flow capacity. Supply changes require recertification; local faults additionally revise affected device conditions. Unaffected repairs and services remain usable.
SocialOps	A campaign wave creates variants, measured reports, and deployment work. Later variants use specified parent reports and their actual winning content and audiences.	Audience permissions, currently licensed assets, exposure constraints, and publication requirements apply. Rights changes affect unpublished drafts; published history remains valid.
LogisticsOps	Additional itinerary legs introduce bookings and transfers. Through-baggage plans bind releases and preceding freight manifests across hubs.	Plans respect readiness, connection times, handling eligibility, and shared capacity. Carrier withdrawals require affected custody plans to be revised while retaining valid arrangements.
ResearchOps	Evidence rounds add calibration sources, studies, and synthesis contributions. Derived calibrations reference earlier maps; compatible studies use their results.	Contributions bind current ancestor evidence and satisfy metric and population compatibility. Corrections affect derived interpretations; prior committed reviews retain their historical identity.
DocOps	Document stages add pages with required parent publications. Cross-lane joins combine predecessor contracts, while older links carry inputs across stages.	Publications satisfy schema, purpose, expiry, and capacity rules. Public bulletins identify amended requirements; artifact and binding validity follow their applicable rules. A current publication can coexist with a stale tracker binding. Valid publications and accepted history are retained.
FactoryOps	Production levels add work orders whose material inputs include units produced by earlier orders. Demand is propagated upstream through the bill of materials.	Executions conserve material lineage and satisfy quality, equipment, and calendar constraints. Tooling drift changes requirements for future work; accepted completed units retain their status.

including capacities based on transported baggage weights. Completing the extended goal requires both passenger arrangements and the specified baggage deliveries. Feasibility depends jointly on capacity, handling eligibility, readiness, and connection times.

Calibration and evidence rounds. ResearchOps adds a calibration anchor and studies in each acquisition round. Derived anchors name earlier calibrations and compose their effective maps. At least one compatible study uses each new anchor, making its derivation relevant to synthesis. Later contributions bind the current annotations of the named calibration and its ancestors. Metric, population, and unit rules determine usable evidence; corrections change derived interpretations and withdrawals alter eligible contributions. New rounds supply further anchors and studies for subsequent synthesis.

Document stages. DocOps adds pages whose required parents include the previous-stage page in the same lane. Topology rules introduce cross-lane joins and older predecessors. Each page

supplies a local grant with allowed purposes and an expiry stage; parent requirements enumerate downstream consumers, weights, and allocation limits. Publication combines the required parent contracts with local fields and conditions, creating an input for later pages. Local grants sustain available inputs across stages, while reservations, expiry, and amendments determine which inherited allocations remain usable.

Production levels. FactoryOps assigns each new order an upstream supplier from the preceding level and an additional compatible supplier according to the topology. Terminal demand propagates backward through the bill of materials to determine required upstream quantities and initial raw stock. Produced units carry their lineage and quality measurements into downstream orders; the task goal requires dispatch of the terminal units. Equipment occupancy, maintenance, calibration, and selective rework govern execution feasibility. Extending a production graph therefore requires recomputing upstream demand and checking batch and scheduling conditions along with the added orders.

Construction checks. Legal executions and targeted counterexamples test prerequisites, input compatibility, validity, update propagation, and retention of unaffected results. Table 18 summarizes representative checks. Successful public reference executions establish feasibility for the selected instances.

Table 18: Construction checks pair legal execution and artifact retention with targeted counterexamples. Detection may occur at an interface, business review, or independent evaluation.

Domain	Legal execution and retention	Counterexample
SocialOps	Complete sources and audience; repeated observation adds no conversion	Missing source or audience rejected
LogisticsOps	Two-stage custody and two-source join; legal transport alternatives	Missing predecessor handoff rejected
ResearchOps	Valid evidence composition and historical review retention	Withdrawn or stale ancestor evidence detected
DocOps	Target-page repair; independent pages remain valid	Expiry and purpose mismatch detected under conservation
FactoryOps	Qualified batch peers retained through constrained rework	Maintenance conflict or acceptance of quarantined units rejected

E Dependency Semantics and Execution Behavior

E.1 Dependency Requirements

The construction mechanisms induce four overlapping requirements. *Content and evidence* dependencies require the preservation and combination of specified upstream information. *Shared-state* dependencies couple feasibility through resources, allocations, and business time. *Version and revision* dependencies require affected results to be refreshed while applicable work is retained. *Commitment and delivery* dependencies require outputs to be associated with the intended consumers and evaluated handoff points. A task can exhibit several requirements simultaneously.

Their business semantics differ across environments. DocOps amendments can invalidate affected page outcomes; SocialOps asset changes primarily alter unpublished drafts; FactoryOps drift changes later processing conditions while accepted physical outputs remain valid. ResearchOps commits preceding reviews into the history evaluated by subsequent rounds. These rules determine which states an agent must revisit and which earlier outcomes remain usable.

E.2 Complete Task-Family Results

Table 19 includes the twenty selected families across seven environments. FinanceOps and the five database-backed domains each contain three families of fifty tasks; FieldOps contains two scenarios of seventy-five tasks. Each model uses the same assigned task labels. Counts retain all assigned tasks and follow the three-category rule in Appendix A.4. Family comparisons describe the evaluated constructions and recorded protocols; parameters and other mechanisms can vary together.

Table 19: Complete task-family results. Each cell reports success / final failure / tool-budget exhaustion over 50 assigned tasks, or 75 for each FieldOps scenario. The assignment rule in Appendix A.4 applies throughout. Model labels abbreviate the full names in the experimental setup.

Family	Gemini	GPT	Qwen	Grok	DeepSeek
SocialOps					
Overlapping conversions	46/4/0	11/36/3	46/4/0	49/1/0	46/4/0
Cooldown reactivation	15/34/1	10/38/2	45/5/0	47/3/0	45/5/0
Overlap + reactivation	32/18/0	10/38/2	41/9/0	46/3/1	47/3/0
LogisticsOps					
Carrier disruption	25/25/0	10/35/5	17/33/0	48/2/0	22/28/0
Restricted handling	22/27/1	2/45/3	9/41/0	47/3/0	11/39/0
Specialist + disruption	18/31/1	2/42/6	8/42/0	48/2/0	8/42/0
ResearchOps					
Identity compatibility	11/39/0	6/44/0	22/28/0	27/22/1	3/47/0
Study retraction	12/38/0	8/41/1	20/30/0	29/21/0	7/43/0
Ambiguity + withdrawal	12/38/0	3/47/0	16/34/0	29/18/3	3/47/0
DocOps					
Expiring reservations	37/12/1	7/33/10	41/9/0	50/0/0	38/12/0
Independent amendments	35/15/0	6/32/12	42/8/0	50/0/0	49/1/0
Expiry + amendments	41/8/1	13/26/11	43/7/0	50/0/0	42/8/0
FactoryOps					
Equipment windows	6/43/1	1/39/10	8/41/1	24/12/14	9/40/1
Selective rework	6/43/1	2/42/6	9/39/2	21/17/12	7/40/3
Constrained rework	9/40/1	2/38/10	7/43/0	23/14/13	4/45/1
FinanceOps					
Heterogeneous integration	17/33/0	3/39/8	16/33/1	20/29/1	13/37/0
Receivables reconciliation	1/48/1	14/36/0	0/50/0	4/44/2	3/46/1
Release maintenance	40/10/0	9/38/3	39/11/0	30/17/3	33/17/0
FieldOps					
Mesh convergence	33/42/0	44/31/0	55/20/0	58/14/3	56/19/0
Mesh supply shocks	33/42/0	32/42/1	45/30/0	54/17/4	49/26/0

SocialOps illustrates model-specific differences across families. Gemini succeeds on 46/50 overlapping-conversion tasks, 15/50 cooldown-reactivation tasks, and 32/50 combined tasks. Its success counts across the five bands are $(10, 10, 10, 9, 7)$, $(7, 2, 5, 1, 0)$, and $(10, 7, 8, 5, 2)$, respectively. The first two families have median reference business-call counts of 320 and 324. DeepSeek records 46/50 and 45/50 successes on the same two families. These observations show different family-level performance profiles under similar median work sizes.

Each of the five database-backed environments includes two component families and one combined family. Across their twenty-five environment–model comparisons, nine combined families have lower recorded success than both component families. Combination effects vary with the mechanisms, task construction, and model.

E.3 DocOps Process Coverage and Progress

The DocOps analysis covers the same thirty 1M tasks for each of the five models. Terminal statistics use all 150 evaluations; review-attempt statistics use the 149 complete histories. The remaining successful Gemini execution has a partial event history. Table 20 reports terminal outcomes and failed-final summaries.

Final page completion is the number of currently completed pages divided by the required page count. It includes the page's required delivery conditions; separate change-window obligations also contribute to global success. Budget use is the cumulative tool attempts divided by the task's original B. Medians in Figure 7(a) condition on a valid final answer with failed business acceptance. Grok has no endpoints in that subset.

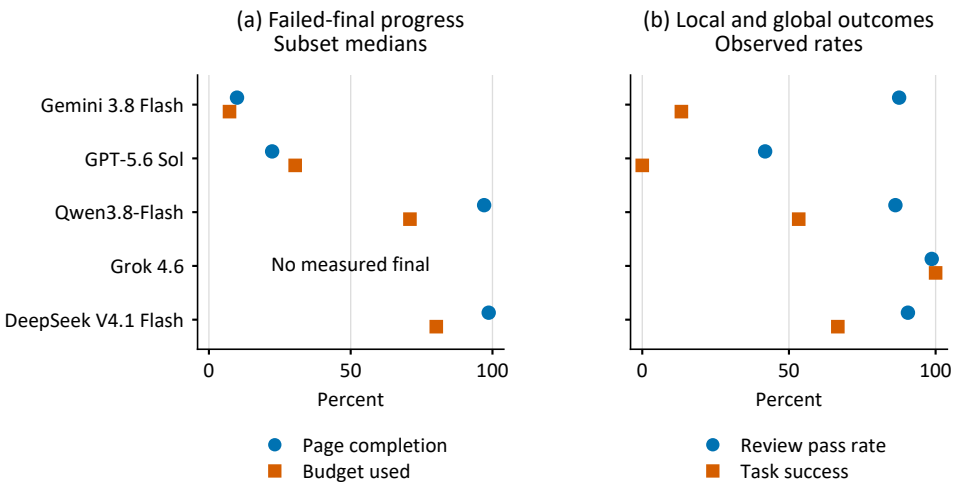

Figure 7: **Local progress, execution correctness, and global completion separate in DocOps's 1M workload band.** (a) Median page completion and budget use among observed valid-final failures with measured states; Grok has no such measured final. (b) Observed review pass rate over actual attempts with complete logs, alongside task success over all 30 assigned tasks per model. Review attempts reflect each model's reached stages. Page completion is a domain-local progress measure. Appendix E.3 gives counts and cohort definitions.

Table 20: DocOps 1M endpoints and failed-final progress. Endpoint counts use all 30 assigned tasks per model: S success, F final failure, and B_f tool-budget failure. The final columns are median page completion (PC) and budget use (BU), in percent, among observed final-answer failures with measured business states.

Model	S	F	B_f	PC	BU
Grok 4.6	30	0	0	—	—
Qwen3.8-Flash	16	14	0	97.1	70.9
DeepSeek V4.1 Flash	20	10	0	98.7	80.2
Gemini 3.8 Flash	4	26	0	9.9	7.3
GPT-5.6 Sol	0	26	4	22.3	30.4

For Figure 7(b), review pass rate counts attempts without a recorded public failure or request for changes from `run_document_review`. These include repeated reviews and span the stages reached by each model. Task success uses all thirty assigned tasks. Table 21 gives process coverage and counts.

Table 21: Observed DocOps 1M review outcomes on recorded endpoints with complete logs. Counts include repeated attempts. Pass rate is one minus the fraction of attempts returning a public failure or requiring changes.

Model	Tasks	Reviews	Failed	Pass (%)
Grok 4.6	30	3323	44	98.7
Qwen3.8-Flash	30	3065	420	86.3
DeepSeek V4.1 Flash	30	3475	328	90.6
Gemini 3.8 Flash	29	1013	126	87.6
GPT-5.6 Sol	30	2636	1532	41.9

Shared-task comparison. Table 22 examines the 75-page independent-amendments task ending in identifier `04`, with $B = 1601$ for every model. Gemini stops after three completed pages and

three accepted reviews. GPT reaches the same page count after thirteen unsuccessful reviews across eighteen attempts. Qwen, DeepSeek, and Grok complete all pages, with different amounts of local correction and tool use.

Table 22: Execution on the same 75-page DocOps task (independent amendments, identifier 04). Each model has $B = 1601$. Reviews and failed reviews count actual attempts.

Model	Pages	Tool attempts	Reviews	Failed
Grok 4.6	75/75	1199	126	0
Qwen3.8-Flash	75/75	1297	136	10
DeepSeek V4.1 Flash	75/75	1272	135	7
Gemini 3.8 Flash	3/75	43	3	0
GPT-5.6 Sol	3/75	119	18	13

Compaction and recorded conditions. Twenty of Gemini's twenty-six valid-final failures occur before any adopted compaction. Successful Grok, DeepSeek, and Qwen tasks use means of 9.17, 11.10, and 9.69 adopted compactions, respectively. Adoption requires an explicit history-replacement event. Appendix F.7 compares dependency visibility at consumption.

E.4 Delivery and Version Consistency

Seven DeepSeek valid-final failures are exactly one page short. Each incomplete page has an accepted write and publication in its history. Terminal checks identify explicit tracker identity or version failures on three pages and missing or malformed evidence on four. Of five near-complete Qwen failures, two have an unbound tracker and two an old-version binding; the fifth has two affected pages, one with an unbound publication and one with inconsistent decision links. Two DeepSeek and two Qwen cases additionally have unfinished change windows. All twelve final responses claim completion or no remaining work.

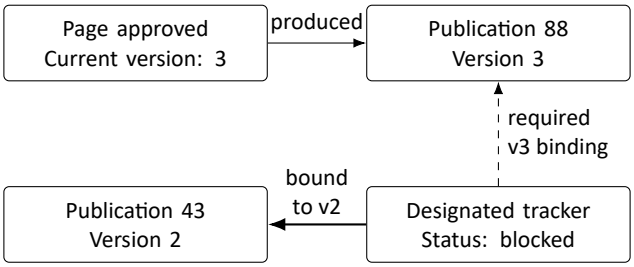

Figure 8: **A correct publication exists while its downstream delivery remains stale.** The recorded DeepSeek DocOps case has an approved version-3 page and publication, but the task's designated tracker still references version 2. Solid arrows show recorded relationships; the dashed arrow shows the required current-version association. Numeric suffixes abbreviate the recorded publication identifiers.

In the DeepSeek expiring-amendments task ending in `03`, the affected page is approved at version 3 and publication `88` also represents version 3. As shown in Figure 8, its designated tracker remains blocked and points to publication `43`, which represents version 2. In Qwen's expiring-amendments task ending in `06`, 69 of 70 pages are complete; the remaining page is approved and has a publication, but its tracker has no publication binding. A public update poll reports that initial delivery is still awaited. The agent interprets that response as having no changes left to process. Both cases expose the need to inspect current consumer bindings when deciding whether work is complete.

F Execution Progress, Composition, and Recovery

We apply common definitions of local execution, business-goal recovery, dependency use, and compaction across all seven environments. The analysis uses 5,250 terminal business evaluations; 4,949

Table 23: Number of extracted model–task measurements across all bands. S, L, R, D, Fa, Fn, and Fd denote SocialOps, LogisticsOps, ResearchOps, DocOps, FactoryOps, FinanceOps, and FieldOps. Each row reports its measurement population.

Measurement	S	L	R	D	Fa	Fn	Fd	Total
Terminal records	750	750	750	750	750	750	750	5,250
Complete process histories	737	750	699	749	686	689	639	4,949
Local completion / paired success	750	750	750	750	750	750	750	5,250
Explicit tool-error fraction	737	750	699	749	685	689	639	4,948
Paired dependency visibility	137	159	120	277	181	342	302	1,518
Identifiable compaction counts	744	750	750	750	747	738	748	5,227

Table 24: Measured terminal progress and actual failed-final stopping states. Completion is averaged per execution; the two stopping-state medians use the same failed-final subset in each row.

Workload	Scored n	Mean goals (%)	Below 25% (%)	Failed finals n	Final goals median (%)	Allowance median (%)
64K	1050	90.4	2.3	206	66.7	14.7
128K	1050	80.0	5.9	406	57.1	18.7
256K	1050	71.0	11.4	518	46.5	23.7
512K	1050	57.5	31.0	609	25.0	40.0
1M	1050	46.8	47.1	706	16.7	60.4

executions have complete merged event histories. Continued executions contribute one cumulative record. Both terminal outcome metrics cover every execution. Actual failed-final analyses select observed final responses. Event-based rates use recorded operations, business-object identities, and verification outcomes. Table 23 reports the extracted observations by environment.

Native local-goal scoring. Native goals are source work packages, restored device cases, campaign briefs, network-verified bookings, research rounds, document pages, and quantity-complete production orders. LGC divides the evaluator's completed count by its required count. Counts satisfy integer bounds and reconcile with per-object verdicts where exported. Separate accounting, service-network, freight, integrity, update, and final-handoff requirements contribute to full task acceptance. The main scores and absolute-output analysis use the same completed/required counts for all 5,250 executions.

F.1 Sustained Progress: Fractions and Absolute Outputs

Figure 3 uses native terminal scores for all 5,250 assigned task–model executions. Each execution contributes equally within its model–workload cell; each band contains 1,050 executions. Completion and the fraction below 25% use this full population. Failed-final medians use the 2,445 submissions with both completion and allowance measurements, as detailed in Appendix F.2. Pointwise intervals use 4,000 percentile bootstrap resamples within each plotted model–band cohort (seed 20260926).

Native counts and completion fractions. We extract native completed and required counts at the recorded terminal endpoint; their per-task ratios give the main LGC scores. Available per-object inventories reconcile to the aggregate counts. Absolute-output summaries cover all 5,250 assignments, including executions without complete event histories.

FinanceOps uses `completed_work_packages/total_work_packages`: semantically correct source work packages incorporated into stage-close outputs. FieldOps uses `completed_cases/total_cases`: devices restored with a passing current test. Full task acceptance additionally checks stage accounting, publication, and released obligations in FinanceOps, and resource, service-network, and system requirements in FieldOps. SocialOps uses `completed_briefs/required_briefs`, DocOps `completed_pages/required_pages`, ResearchOps `completed_rounds/required_rounds`, and FactoryOps `completed_orders/required_orders`. LogisticsOps uses booking counts from `network_evaluation`, with cargo and freight requirements assessed by full task acceptance. FactoryOps counts include quantity-complete intermediate orders; terminal handoff is checked sepa-

Table 25: Domain-specific local-goal units and terminal count coverage. Global delivery requirements are assessed separately by TS.

Environment	Native unit	Count records / assigned	Exact histories
FinanceOps	Correct source work package	750/750	689
FieldOps	Restored device case	750/750	639
SocialOps	Accepted brief	750/750	737
LogisticsOps	Network-verified booking	750/750	750
ResearchOps	Accepted review round	750/750	699
DocOps	Accepted page	750/750	749
FactoryOps	Quantity-complete order	750/750	686

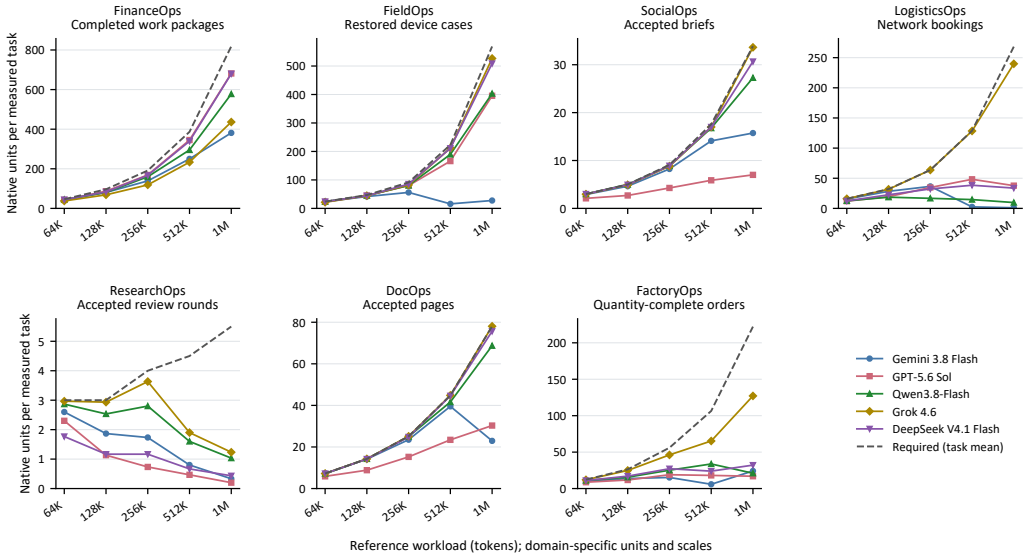

Figure 9: Absolute native output at terminal endpoints, by model, on all 5,250 assigned executions. Solid lines show mean completed units per execution; dashed lines show mean required units pooled across models. Each environment–model–band cell contains 30 executions. Each panel uses its domain-specific unit and scale.

rately. ResearchOps counts accepted rounds, retaining creation-time judgments for committed rounds and reevaluating the current round under its current state.

Mean absolute output increases across bands in six environments, but requirements grow faster overall. ResearchOps differs: accepted-round count falls overall, with a small increase at 256K; within-round work is not measured by this count. In LogisticsOps from 512K to 1M, Grok increases from 128.40 to 239.83 completed bookings per execution; GPT declines from 48.27 to 37.73, Gemini from 2.53 to 1.03, DeepSeek from 38.40 to 33.70, and Qwen from 14.60 to 9.70. The aggregate increase is therefore not a shared model trend. Full-assignment summaries give equal weight to executions and, within each environment–band cell, equal weight to the five models.

Incomplete-goal counts include both blocked and unattempted work.

Table 26: Mean completed / required native units per assigned execution, pooled across models. Each environment–band cell contains all 150 assignments. These means describe different task instances in each workload band.

Environment	64K	128K	256K	512K	1M
FinanceOps	42.10/46.40	79.96/96.40	150.61/191.33	292.67/386.67	551.29/818.40
FieldOps	23.19/23.83	44.09/46.87	77.00/91.37	158.96/223.00	372.39/568.67
SocialOps	2.79/3.00	4.43/5.00	7.82/9.10	14.15/17.57	22.86/33.87
LogisticsOps	13.69/16.00	24.12/32.00	36.76/63.67	46.44/128.80	64.40/268.27
ResearchOps	2.50/3.00	1.93/3.00	2.01/4.00	1.09/4.50	0.65/5.50
DocOps	6.94/7.23	13.15/14.27	22.74/25.13	38.81/44.83	55.13/78.07
FactoryOps	10.49/12.27	16.59/26.30	26.54/56.00	29.36/106.47	44.27/222.20

F.2 Joint Progress and Stopping States

We join terminal local-goal completion and remaining allowance by environment, task label, model, and selected continuation endpoint. Of the 2,558 unsuccessful agent-submitted final answers defined in Appendix A.4, 2,445 have both measurements and 113 have unmeasured allowance. The table below gives joint coverage by model and workload. Progress is measured at termination, and remaining allowance is the fraction of unused tool attempts, $(B - A_{\text{used}})/B$.

Across workload bands, the jointly observed failed-final counts are 206, 406, 518, 609, and 706, out of 208, 412, 543, 653, and 742 actual failed finals, respectively. Their median completion rates are 66.7%, 57.1%, 46.5%, 25.0%, and 16.7%; corresponding median unused allowances are 14.7%, 18.7%, 23.7%, 40.0%, and 60.4%. These are paired measurements within each execution, with different task and failure subsets across bands. Table 24 collects the same stopping summaries alongside all 5,250 terminally scored executions.

Low progress with unused allowance. We count executions that jointly satisfy terminal completion below 25% and unused allowance above 50%. Table 28 gives the event counts and their proportions among measured failed finals and among all assigned executions. At 64K, five of 206 measured failed finals satisfy both conditions; at 1M, 381 of 706 do. These observed events account for 0.5% and 36.3% of all assigned executions in the respective bands. Keeping the completion threshold fixed and changing the unused-allowance threshold to 40% gives failed-final event rates of 2.4% at 64K and 58.9% at 1M; using 60% gives 2.4% and 47.5%. Each threshold yields an increasing event rate across the five workload bands.

Explicit errors and business acceptance. Across the seven environments, 4,948 executions have a measurable tool-error fraction, and 448 have no explicit tool errors: 243 succeed and 205 fail business acceptance. The 4,500 positive-error executions comprise 2,054 successes, 2,278 final failures under the outcome table's reporting convention, and 168 budget-exhaustion outcomes. These measurements use complete histories with a nonzero tool-attempt denominator. Explicit errors record public rejections; omitted work or content errors detected only by terminal evaluation can occur without a tool rejection. This separates interface-level execution from satisfaction of the full business goal.

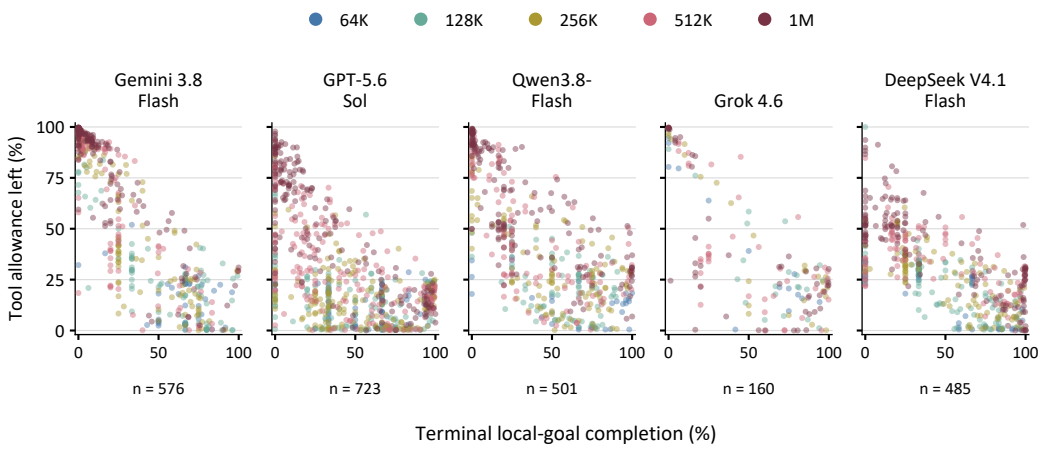

Figure 10: Terminal progress and unused tool allowance in the same failed-final execution. Each point is one jointly measured model–task record; colors denote reference workload. Remaining allowance is $(B - A_{\text{used}})/B$. Counts below each panel give its sample size. All seven environments contribute to the pooled analysis; Appendix F.2 gives sample sizes by band.

Table 27: Joint progress–allowance observations / actual failed finals. All seven environments are pooled; each model–band has 210 assigned tasks.

Workload	Gemini	GPT	Qwen	Grok	DeepSeek
64K	41/42	70/71	32/32	15/15	48/48
128K	82/85	123/126	81/81	24/24	96/96
256K	112/115	166/172	112/112	27/27	101/117
512K	153/156	181/188	124/124	46/54	105/131
1M	188/192	183/189	152/153	48/56	135/152

Table 28: Actual failed finals with completion below 25% and unused allowance above 50%. The final column divides observed events by all assigned executions in the workload band.

Workload	Jointly measured	Events	Event rate (%)	All assigned (%)
64K	206	5	2.4	0.5
128K	406	21	5.2	2.0
256K	518	79	15.3	7.5
512K	609	191	31.4	18.2
1M	706	381	54.0	36.3

F.3 Skill Execution and Stateful Composition

Local completion and whole-task delivery. Figure 4(a) partitions all 5,250 executions into four mutually exclusive outcomes: task success, task failure with all local goals complete, task failure with $75\% \leq \mathrm{LGC} < 100\%$, and task failure with $\mathrm{LGC} < 75\%$. Each workload band contains 1,050 executions and gives each execution equal weight. From 64K through 1M, the category counts are respectively $(793, 2, 100, 155)$, $(585, 0, 149, 316)$, $(459, 0, 153, 438)$, $(353, 0, 121, 576)$, and $(234, 9, 133, 674)$. The 75% boundary groups partial completion for display; full completion uses equality of the native integer completed and required goal counts. All successful tasks satisfy this equality.

Panel (b) reports $P(\mathrm{TS} = 1 \mid \mathrm{LGC} \geq q)$ at terminal completion thresholds $q \in \{50, 60, 70, 75, 80, 90, 100\}\%$. Each curve pools all seven environments and five models within one workload band, with equal weight per eligible execution. Eligibility is evaluated directly on integer goal counts, using $100\,\mathrm{completed} \geq q\,\mathrm{required}$ with q expressed as a percentage. Both metrics are measured at termination. The eligible model and environment composition varies with workload and threshold. Because successful tasks have LGC = 100%, raising the threshold retains the successful numerator and removes partially complete failures from the denominator. Across all bands, 2,424 of 2,435 executions with full local completion succeed. The eleven failures comprise two FinanceOps executions at 64K and nine FieldOps executions at 1M.

Units and event ordering. Script and matched-output analyses use 3,621 complete histories from SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps, with continuation prefixes counted once. Of these, 2,436 trajectories contain an explicit `run_skill_script` attempt, totaling 61,304 attempts and 59,295 successful returns. A successful return records normal execution of a supplied program on the submitted inputs. Consumer acceptance and terminal delivery are measured through their business checks. The FinanceOps and FieldOps cases use native terminal evaluations from the seven-environment outcome cohort.

Figure 12(a) averages each task's successful-script-return fraction; panel (b) reports business success on the same script-attempting tasks. Table 30 gives task-level outcomes. The cohort includes 904 trajectories with a successful script return but no terminal business acceptance, 864 of which already succeed on their first script attempt. Among 1,654 trajectories whose script attempts all succeed, 641 fail business acceptance. Seven trajectories achieve business success without a successful script return. Script execution and full business delivery therefore measure distinct outcomes.

Subsequent invocation. There are 1,896 trajectories with another script call in a later model decision after the script's first successful return. Of these, 612 contain 1,567 subsequent rejections across 639 task–skill pairs. Figure 12(c) averages the rejected-call fraction within each eligible trajectory. Repeated calls can address different objects, inputs, and state versions.

Matching producer outputs to consumers. We compare public script outputs with fields submitted to later business operations. A producer must return in an earlier model decision, ordered by continuation stage and decision index. Same-decision batched calls are recorded separately. All compatible earlier producers are retained when several return the same values; each consumer is counted once. Matching establishes observable field consistency. The provenance relation can remain many-to-one when values are recomputed or returned repeatedly.

Complete device goals and unfinished service delivery. The selected FieldOps endpoints contain nine unsuccessful 1M executions whose completed-device count equals the required-device count. Seven also have every required service active: one GPT execution, one Grok execution, four DeepSeek executions, and one Qwen execution. The other two, one GPT and one Qwen execution, each leave one of 384 required services inactive. None of the nine endpoints contains the final acceptance artifact, and all have zero open isolations. These counts join native device-goal scores to the service and acceptance state of the same selected endpoint. They distinguish device restoration, service activation, and final workflow acceptance.

Financial work packages and publication validation. At 1M in FinanceOps, GPT has mean native work-package completion of 85.8% over 30 executions. All 30 retain incomplete required

Table 29: Task success conditional on terminal LGC meeting threshold q. Each cell reports successful tasks / eligible executions, pooling seven environments and five models.

Workload	$q = 50\%$	60%	70%	75%	80%	90%	100%
64K	793/984	793/954	793/896	793/895	793/861	793/832	793/795
128K	585/864	585/822	585/753	585/734	585/682	585/627	585/585
256K	459/753	459/694	459/639	459/612	459/572	459/526	459/459
512K	353/582	353/544	353/497	353/474	353/465	353/425	353/353
1M	234/469	234/442	234/394	234/376	234/356	234/327	234/243

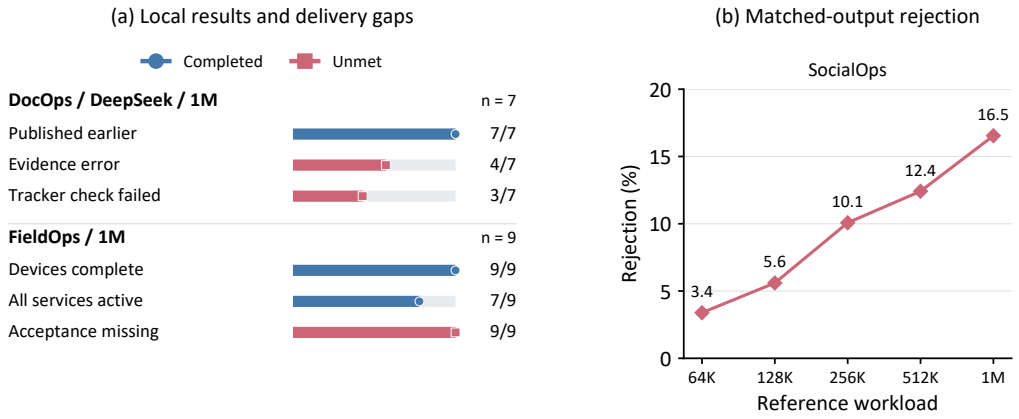

Figure 11: Delivery-state and matched-output audits. (a) Historical publication and terminal evidence checks for seven one-page-short failures among DeepSeek's 30 DocOps executions at 1M, and terminal delivery checks for nine FieldOps failures with all device goals complete. Counts share a denominator within each group; two DocOps cases also have unfinished change windows. (b) Task-averaged rejection of SocialOps submissions matching earlier successful skill-script outputs.

packages; 29 contain a publication artifact, and none passes the native publication check. DeepSeek has mean package completion of 85.0%; two executions complete all required packages and pass whole-task acceptance, while the other 28 retain incomplete packages. Across all five models at 1M, 18 of 150 executions complete every required work package, and all 18 pass whole-task acceptance. Work-package acceptance includes the required semantic views and their inclusion in stage closure. These endpoint counts distinguish publication-artifact existence from publication validity while retaining incomplete local obligations in the explanation of whole-task failure.

For exposure bindings, matching requires the same post identity, historical publication identities, and audience-load mapping. Identity lists are order-normalized while preserving duplicates, and numeric value types are retained. For production finalization, matching requires the script input's production-run identity and the returned accepted/quarantined unit partitions. Document matching parses contract JSON, preserving arrays and values while ignoring object-key order and formatting. A matching accepted write must return page identity and version before it can be linked to a later publication with the same page, version, and contract.

Input-history consistency. The SocialOps exposure contract yields 24,713 matched binding attempts across 730 trajectories: 21,326 are accepted and 3,387 are rejected. Table 32 separates their public rejection reasons. Of the source-set and load-value rejections, 1,421 and 811 respectively occur at the next non-declaration tool attempt after a compatible script return, in a later model decision. Under the three-category outcome assignment, the affected 371 trajectories end in 249 successes, 115 final failures, and seven budget failures. Per-trajectory matched-binding rejection fractions average 3.39%, 5.60%, 10.07%, 12.42%, and 16.54% from 64K through 1M, over 142, 147, 148, 147, and 146 trajectories. These measurements describe this history-sensitive contract.

Table 30: Script-return and business outcomes on the same 2,436 script-attempting trajectories from the five-environment cohort. Each trajectory contributes once to each row.

Observed outcome	Trajectories	Fraction (%)
First script attempt returns successfully	2,297	94.3
At least one script returns successfully	2,396	98.4
Final business acceptance	1,499	61.5

Table 31: Script-attempting / subsequent-invocation trajectory counts, pooling the five environments. Model abbreviations follow the figure legend.

Workload	Gemini	GPT	Qwen	Grok	DeepSeek
64K	112/81	82/44	91/75	145/108	66/63
128K	116/89	99/66	89/78	143/107	70/67
256K	120/86	86/64	84/69	145/108	73/64
512K	105/78	77/58	82/65	133/110	70/62
1M	98/66	83/63	72/60	129/103	66/62

The corresponding production-quality contract has 1,495 matched finalization attempts across 111 FactoryOps trajectories: 1,494 are accepted and one is rejected for missing mandatory observations.

Observed input correction. In a GPT-5.6 Sol 128K execution, the binding operation at tool attempt 163 matches a successful exposure calculation returned one tool attempt earlier, but rejects the historical source set. At attempt 167, a binding for the same post matches another successful calculation returned one tool attempt earlier and is accepted. The accepted binding names eight historical publications, including publication 77. This sequence shows a historical-input contract failure after successful script execution, followed by an accepted recalculation for the same business object.

Publication and terminal binding. Across 749 DocOps trajectories, we identify 22,410 later-decision writes matching successful script contracts, of which 22,293 are accepted. These matches cover 713 trajectories. The accepted-write and same-version publication chains reach 16,295 distinct task–page pairs. At termination, 16,056 of these pages pass evaluation; 147 pages across 38 trajectories have explicit tracker identity, status, or version failures. The remaining 92 pages have broader evidence-resolution errors and are kept separate from the tracker-specific count. Historical publication verifies that the output reached that stage; its continued validity depends on subsequent edits and bindings.

The seven DeepSeek 1M near-completion cases in Appendix E.4 all contain a matching accepted write and publication history on their incomplete page. Three pages fail explicit tracker identity and version checks; four have missing or malformed evidence. In one versioned case, publication 43 records page version 2 and the later publication 88 records version 3, with the latter linked to a matching script-write chain. The page nevertheless fails terminal publication and tracker identity/version checks. Two of the seven executions also leave change windows unfinished. These observations distinguish historical artifact creation from valid terminal delivery.

ResearchOps calculation route. ResearchOps exposes an additional calculation route through `calculate_review_synthesis`, which runs the fixed synthesis procedure and stores its result in a draft. Among 699 ResearchOps histories, 303 attempt this operation without an explicit script call. Considering either calculation route gives 680 attempting trajectories, 679 with a successful calculation return and 207 with final business acceptance. This named alternative is tabulated separately from explicit script attempts; its automatic result storage is part of the provided interface.

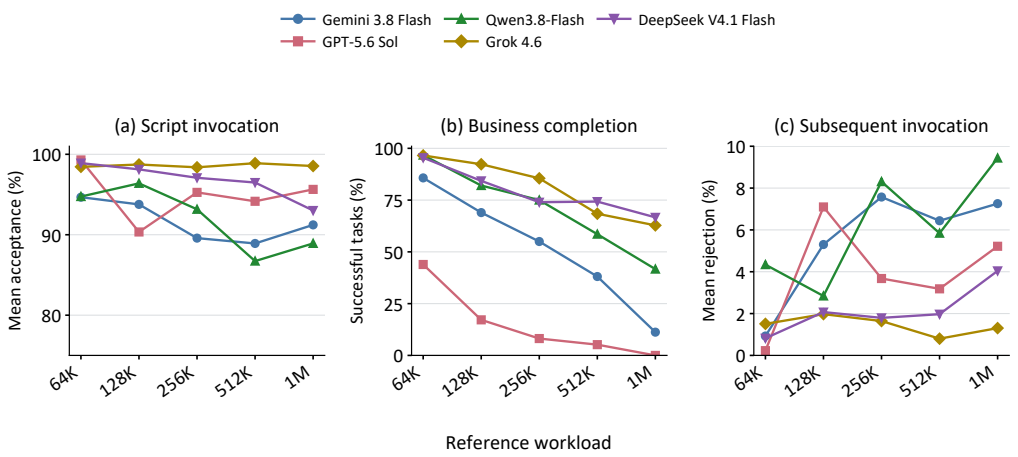

Figure 12: Script execution, task completion, and subsequent invocation in SocialOps, LogisticsOps, ResearchOps, DocOps, and FactoryOps. Panels (a) and (b) use 2,436 script-attempting trajectories; panel (c) uses 1,896 trajectories with a later-decision call to a previously successful skill script. Each model–workload ratio gives one vote per eligible trajectory. Table 31 reports both sample sizes.

Table 32: Public rejection reasons for exposure-binding attempts whose fields match an earlier successful script output.

Consumer check	Rejected attempts
Required historical source identities	1,796
Current decayed audience loads	1,222
Due-post time or status condition	366
Public schema	2
Post identity lookup	1
Total	3,387

F.4 Recovery after Public Business Feedback

Recovery within a retry allowance. We evaluate fixed recorded trajectories at progressively larger retry and tool-budget cutoffs. The analysis measures outcomes along the observed action sequences, with the original execution policy held fixed. Figure 5a uses the same 35,265 eligible public first-failure cycles across seven environments as the recovery-opportunity audit, spanning 3,471 selected task–model executions. For each failed business goal and obligation cycle, a retry is a subsequent check-bearing attempt. The initial failed check is excluded. A pass at the jth subsequent check is counted as recovery within allowance k when $j \leq k$. Checks stop at the first verified pass; unresolved goals retain all recorded subsequent checks through the cycle cutoff. Reads, unrelated goals, and edits without a corresponding goal check do not increment this count. A previously completed obligation that is invalidated belongs to a new cycle. Thus, this measure counts verification attempts, not all repair actions or model requests.

For each $k \in \{3, 4, 5, 6, 7\}$, the denominator includes every eligible first-failure cycle, including those with no recheck. We compute recovery fractions within each execution and average them with equal execution weight within model and workload. The same population is used for all five allowances. Table 33 gives support for each model–workload cell. All seven environments contribute, with environment and failure-type composition varying across cells. Grouped bars use equal execution weights; the allowance-matched comparisons below use shared-stratum weights.

The three mutually exclusive outcomes at each cutoff are recovery within k checks, reaching k checks without a pass, and no pass with fewer than k recorded checks. The second group includes goals that recover after the cutoff; the third includes zero-recheck goals and goals edited without subsequent

Table 33: Executions / first-failure goal cycles supporting the retry-allowance analysis. The population of each cell stays fixed for retry limits 3–7.

Model	64K	128K	256K	512K	1M
Grok	68/131	91/375	107/853	118/1443	129/3484
Qwen	122/335	150/709	160/1048	162/2097	166/2251
DeepSeek	135/335	140/897	152/1138	158/1348	170/3432
Gemini	107/333	132/793	143/891	97/426	92/1174
GPT	159/602	174/1283	179/2069	180/3584	180/4234

Table 34: Observed recovery within a retry allowance, pooled with equal weight per eligible execution (%). Reached allowance includes goals recovered only after k checks; fewer checks means fewer than k recorded rechecks without a pass by the cycle's end. Intervening edits may occur in either group.

Retries	Outcome	64K	128K	256K	512K	1M
3	Recovered	84.4	77.5	75.7	75.2	69.6
	Reached allowance, no pass	6.1	9.4	8.0	9.1	8.6
	Fewer checks, no pass	9.5	13.1	16.3	15.6	21.8
4	Recovered	86.0	79.2	77.3	76.8	71.1
	Reached allowance, no pass	4.0	5.9	5.0	5.7	5.7
	Fewer checks, no pass	10.0	14.9	17.7	17.5	23.2
5	Recovered	87.1	80.3	78.1	77.7	72.0
	Reached allowance, no pass	2.6	4.0	3.7	3.7	3.7
	Fewer checks, no pass	10.3	15.7	18.2	18.6	24.3
6	Recovered	87.6	80.8	78.8	78.2	72.5
	Reached allowance, no pass	2.0	2.7	2.7	2.8	3.0
	Fewer checks, no pass	10.4	16.4	18.5	18.9	24.5
7	Recovered	87.9	81.3	79.1	78.6	73.0
	Reached allowance, no pass	1.7	2.1	2.2	2.3	2.3
	Fewer checks, no pass	10.4	16.6	18.7	19.1	24.7

verification. Table 34 reports these outcomes. At 1M, increasing k from three to seven raises pooled recovery from 69.6% to 73.0%; the corresponding rates at 64K are 84.4% and 87.9%.

Fixed-trajectory budget sensitivity. We evaluate recorded trajectory prefixes at $B_\lambda = R + \lceil \lambda A \rceil$ for $\lambda \in \{0, 1/8, 1/4, 3/8, 1/2\}$, where R is the reference tool-call count and A is the reference business-execution call count. Cutoffs follow cumulative charged attempts under each execution's accounting rules, including charged controller refusals and excluding exempt declarations. For trajectories that end before a cutoff, the observed terminal state is retained.

The recovery analysis includes eligible goal–obligation cycles whose first public business failure occurs by R: 30,452 failed cycles from 3,374 executions across all seven environments. At each cutoff, a cycle is recovered if its first subsequent verified pass occurs by that cutoff and before the cycle ends. We calculate the recovered fraction within each execution and average across executions within each workload band. The five bands contribute 558, 659, 722, 704, and 731 executions, respectively, keeping the failure population fixed across λ.

We reconstruct each cutoff's business state in an isolated workspace using the corresponding domain implementation and apply its native completion checks. Charged actions are counted individually, including actions within a batched response. Recorded ResearchOps failures follow the runtime's atomic rollback contract: business state is unchanged and the sequence clock advances; these failed calls are not reexecuted. Reconstructed terminal scores are checked against the selected results; when all cutoffs follow an execution's end, its observed score applies throughout.

Figure 5b–d and Table 35 report LGC and TS for the same 3,374 executions used for recovery. Recovery measures eligible failed-goal cycles; LGC measures all required local goals, and TS measures whole-task delivery. All three metrics use a fixed execution population across cutoffs and capture subsequent correction and workflow progress.

Table 35: Fixed-trajectory budget sensitivity within the same 3,374-execution failure cohort for all three metrics. The 64K, 128K, 256K, 512K, and 1M bands contain 558, 659, 722, 704, and 731 executions, respectively, at every cutoff. Entries are task-averaged percentages.

Workload	$\lambda = 0$	$1/8$	$1/4$	$3/8$	$1/2$
Local-goal completion					
64K	61.0	70.4	80.0	85.1	88.6
128K	62.2	69.5	73.4	76.6	78.5
256K	59.6	65.0	69.4	71.3	72.3
512K	57.9	61.5	64.6	65.7	66.1
1M	51.2	54.1	56.1	56.7	56.9
Task success					
64K	13.6	27.2	49.6	62.2	70.8
128K	16.5	30.0	40.4	47.5	50.4
256K	13.6	23.1	35.3	40.7	43.1
512K	10.7	18.6	33.2	38.6	40.2
1M	5.3	12.3	21.9	26.4	27.4
Failed-goal recovery					
64K	82.3	90.0	93.6	94.5	94.8
128K	78.6	83.7	85.9	86.9	87.3
256K	80.6	83.0	83.6	83.9	84.0
512K	80.0	80.9	81.1	81.2	81.2
1M	74.7	75.4	75.4	75.4	75.5

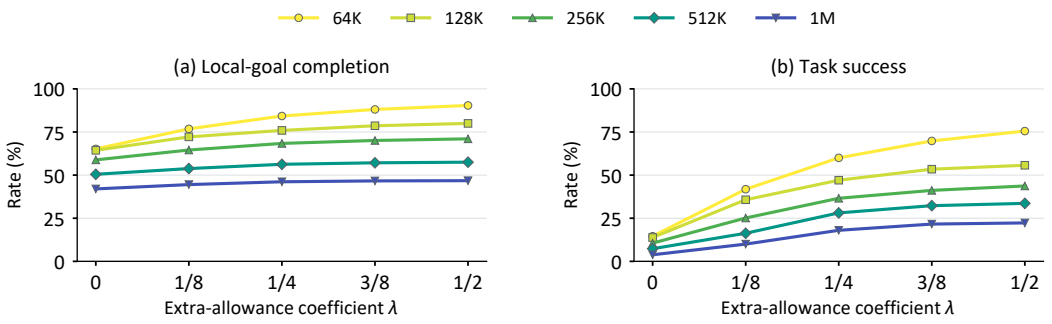

Figure 13: Fixed-trajectory budget sensitivity across all 5,250 selected executions, including trajectories without an eligible public failure. Each workload band contains 1,050 executions at every cutoff. These completion and delivery curves use a broader population than the failure-cohort curves in Figure 5b–d; they do not isolate the contribution of error repair.

Figure 13 and Table 36 extend the same prefix scoring to all 5,250 executions, with 1,050 per workload band, covering progress both with and without an observed business failure.

Goals, feedback, and subsequent outcomes. We follow object-specific business obligations after a publicly observable failed check. Each record identifies the selected task–model endpoint, business object, operation, state cycle, and feedback source. Repeated checks of an unresolved obligation belong to one cycle; a previously satisfied obligation that becomes invalid opens an update cycle. A correctly executed diagnostic operation may report an unmet business condition. Tool-return status and business-goal verdict are recorded separately. Subsequent outcomes are a verified pass, another check without a pass, or no further check. No recheck describes the verification sequence, including cases with intervening edits. The observation ends at the first later pass or the end of the cycle's recorded history.

Seven-environment coverage. The audit contains 42,671 first-failure records joined to the selected endpoints. Of these, 35,265 have an identifiable business goal, public business-failure feedback,

Table 36: Fixed-trajectory budget sensitivity across all 5,250 selected executions (1,050 per workload band), including executions without an eligible public failure. Entries are task-averaged percentages.

Workload	$\lambda = 0$	$1/8$	$1/4$	$3/8$	$1/2$
Local-goal completion					
64K	65.2	76.8	84.3	88.1	90.4
128K	64.4	72.2	75.9	78.6	80.0
256K	58.8	64.6	68.4	70.1	71.0
512K	50.5	53.8	56.3	57.1	57.5
1M	42.0	44.5	46.1	46.6	46.8
Task success					
64K	14.6	41.8	60.0	69.8	75.5
128K	13.8	35.7	47.0	53.4	55.7
256K	10.6	25.1	36.6	41.1	43.7
512K	7.3	16.3	28.1	32.3	33.6
1M	3.8	10.0	18.0	21.6	22.3

Table 37: Public business-failure observations with measurable allowance, before shared-support selection. Each environment has 750 assigned executions. Counts retain domain-specific goal types.

Environment	Tasks	Cycles	Later pass	Rechecked, no pass	No recheck
DocOps	449	2574	2462	88	24
FactoryOps	618	15424	10690	2088	2646
FieldOps	609	8494	7707	392	395
FinanceOps	392	1495	816	659	20
LogisticsOps	606	5530	4585	301	644
ResearchOps	179	229	55	172	2
SocialOps	618	1519	1507	2	10

exact state/history measurements, and measurable remaining allowance. Table 37 reports coverage before shared-support selection. Goal types remain separate, and eligibility is determined at failure, including goals with no subsequent execution.

ResearchOps contributes 229 rounds with actual public business refusals, identified from saved submission receipts and round histories. Later content correctness is checked against the recorded artifact state. The other 1,184 failed rounds have only offline content findings or schema feedback and remain outside this comparison. FinanceOps contributes 1,495 business-goal cycles, distinguishing source outputs, stage closure, and publication. Intermediate business operations are analyzed separately. Where a schema refusal precedes a real business refusal, the latter anchors the comparison; 26 FinanceOps cycles are re-anchored in this way. Another 22 business-goal cycles have unknown allowance at the business refusal. FieldOps contributes 8,494 failed cycles, separating device tests from service certification and initial obligations from update-induced invalidation. Known same-decision checks are marked separately from later decisions; their ordering alone does not establish that the actor read a prior receipt.

Allowance at the first public failure. Remaining allowance is the original task budget minus cumulative charged tool calls immediately after the failed check. Both the absolute count and its fraction of the original budget are retained. Continuations preserve accumulated charges. Model requests and declaration-exempt calls are not treated as charged business-tool attempts. We use absolute bins of 0–99, 100–249, 250–499, 500–999, 1,000–1,999, 2,000–3,999, and at least 4,000 calls, and relative bins of 0–20%, 20–40%, 40–60%, 60–80%, and 80–100%. Relative bins are left-closed and right-open except that the final bin includes 100%. Boundaries were fixed before inspecting supported-cell recovery outcomes in this reanalysis.

Shared populations and averaging. For a workload comparison, each model–environment–operation–cycle-type stratum must contain at least three distinct tasks in each workload band and in the same joint allowance bin. We first average goal outcomes within each task, then average tasks within a stratum. Common strata receive equal weights across the compared bands; supported relative-

Table 38: Shared-support coverage for adjacent workload comparisons. Task counts are deduplicated across allowance bins within each band of a comparison.

Workloads	Tasks: lower	Tasks: higher	Cycles: lower	Cycles: higher
64K–128K	287	302	751	1463
128K–256K	401	348	1516	1403
256K–512K	355	287	1698	2467
512K–1M	328	333	2110	2788

allowance bins receive equal weights within each stratum at a given absolute-allowance bin. The same weights are applied to both bands. Table 38 gives the union of supported tasks for each adjacent-band comparison. Tasks contributing to several allowance bins are counted once in this table.

Requiring shared support across all five workload bands leaves only the FactoryOps/Grok initial-output stratum with fewer than 100 calls and less than 20% allowance remaining. Its five task counts are 9, 16, 14, 10, and 12. We therefore report four adjacent-band comparisons with broader pair-specific support. Together they include all seven environments. Their populations differ across pairs and allowance bins, so points are not connected into a single workload or allowance trajectory. Absolute-only and fraction-only analyses are retained as sensitivity exports.

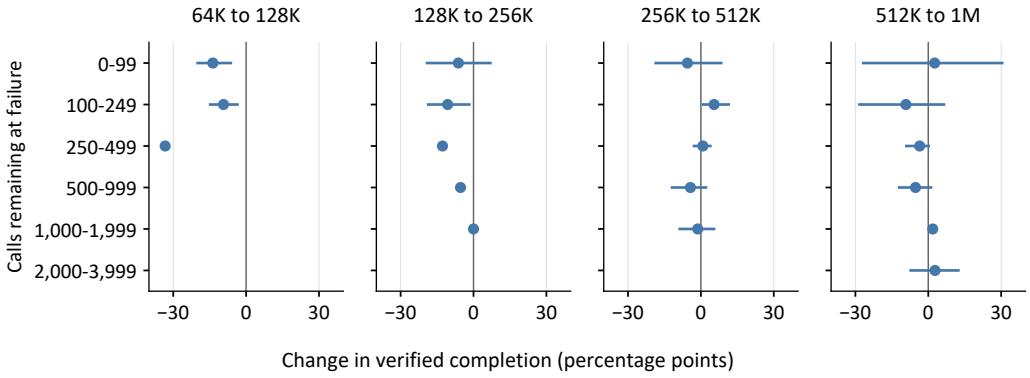

Figure 14: Recovery differences vary across workload ranges. Points show higher- minus lower-workload verified completion, using shared model, environment, operation, cycle-type, and joint remaining-allowance strata. All 19 supported adjacent-band comparisons are shown; each point has its own supported population. Whiskers show estimable pointwise 95% base-task-bootstrap intervals, sharing draws across models; points without whiskers retain their estimates. Appendix F.4 reports support and interval rules.

Recovery and verification. Figures 15 and 16 show subsequent verified completion and the absence of a further check. Table 39 reports supported adjacent-band differences. At 0–99 remaining calls, the 64K-to-128K completion difference is -13.7 percentage points with a pointwise 95% interval from -20.5 to -5.8; the no-recheck difference is $+4.7$ points. The 256K-to-512K completion difference at 100–249 calls is $+5.4$ points, with an interval from $+0.4$ to $+12.0$. The 512K-to-1M difference at 1,000–1,999 calls is $+1.8$ points; 379 resamples retain all required cells, below the 400-draw interval threshold. Recovery differences therefore vary across the matched workload and allowance comparisons.

Uncertainty and interpretation. Pointwise intervals use nonparametric bootstrap resampling (Efron, 1979), with 500 seeded draws of base-task instances within each environment and workload band. Each base task's draw is shared across all models and goal/cycle cells within a plotted allowance bin, preserving cross-model dependence on shared task content. Intervals require at least 400 draws retaining every required cell and nonzero percentile width. Degenerate or insufficient-draw intervals are omitted while retaining the point estimate; interval status, valid draw counts, and base-task counts

Table 39: Adjacent-workload differences at jointly matched absolute and relative allowances. Values are higher minus lower workload, in percentage points, with pointwise 95% task-bootstrap intervals.

Workloads	Remaining calls	Later pass	No recheck
64K–128K	0–99	-13.71 [-20.51, -5.76]	+4.66 [-1.36, +10.66]
	100–249	-9.31 [-15.33, -3.03]	+0.97 [-2.96, +3.81]
	250–499	-33.33 [·]	+0.00 [·]
128K–256K	0–99	-6.23 [-19.66, +7.49]	+4.30 [-6.85, +15.74]
	100–249	-10.65 [-19.19, -1.20]	+3.92 [-2.52, +9.87]
	250–499	-12.76 [·]	+4.57 [·]
	500–999	-5.35 [·]	-1.77 [·]
	1,000–1,999	+0.00 [·]	+0.00 [·]
256K–512K	0–99	-5.54 [-19.16, +8.89]	-5.43 [-17.89, +6.70]
	100–249	+5.41 [+0.41, +11.97]	-4.44 [-10.45, +0.05]
	250–499	+0.82 [-3.38, +4.42]	+2.31 [-1.24, +8.24]
	500–999	-4.31 [-12.42, +2.64]	+2.69 [-2.56, +7.69]
	1,000–1,999	-1.30 [-9.27, +5.97]	+3.03 [+0.00, +9.09]
512K–1M	0–99	+2.64 [-27.29, +30.98]	-4.68 [-14.96, +7.38]
	100–249	-9.21 [-28.87, +6.99]	+1.85 [+0.00, +5.56]
	250–499	-3.53 [-9.58, +0.72]	+0.00 [·]
	500–999	-5.26 [-12.54, +1.65]	+1.12 [-4.56, +7.86]
	1,000–1,999	+1.85 [·]	+2.83 [·]
	2,000–3,999	+2.78 [-7.79, +12.92]	-4.65 [-12.97, +1.65]

· marks an interval
with fewer than 400 valid draws or a degenerate bootstrap distribution. Base-task resampling shares weights across models. All supported comparisons are shown; intervals are not adjusted for multiple comparisons.

are exported. Intervals are marginal, without adjustment for multiple comparisons. The analysis conditions on observed public failure and measurable history. Matching coarse allowance bins does not equalize repair difficulty, failure types, or prior task progress and does not identify a causal workload effect. Additional checks stop at the first later pass for recovered goals; unresolved goals retain checks through the observed cutoff. These counts measure verification effort rather than total repair cost.

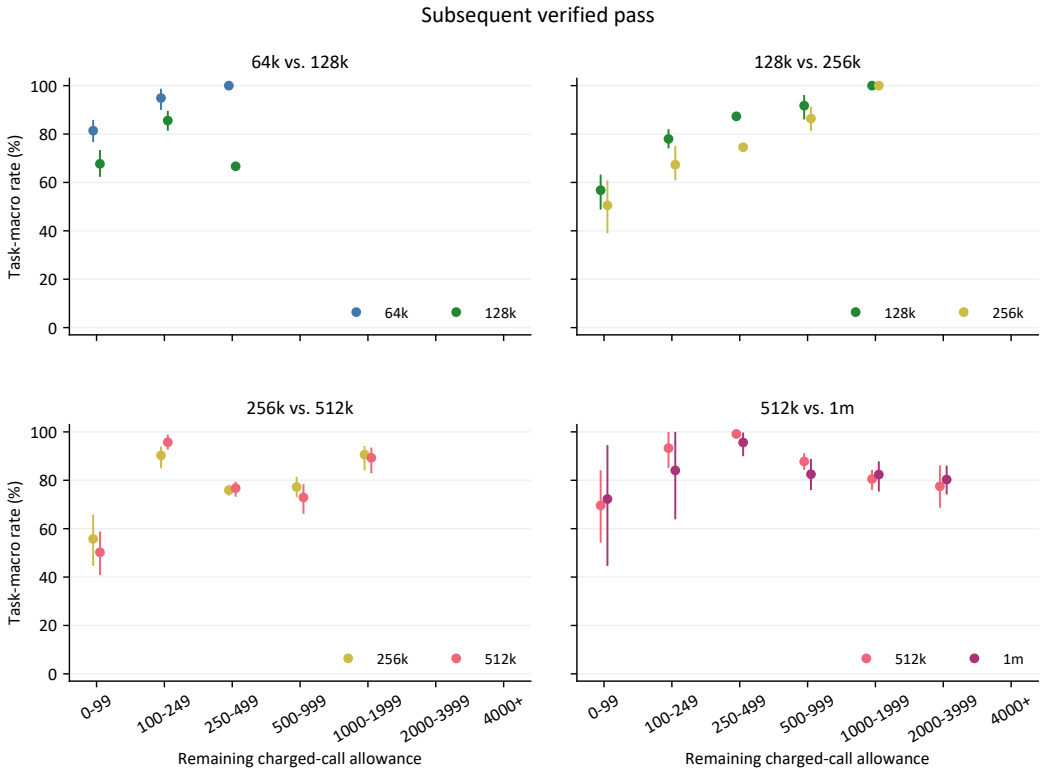

Figure 15: Subsequent verified completion after public business failure. Panels compare adjacent workload bands within shared model, environment, operation, cycle-type, and joint absolute/relative allowance strata. Points average task-level fractions using common weights within each comparison. Error bars are pointwise 95% task-bootstrap intervals. Each panel and allowance bin has its own supported population.

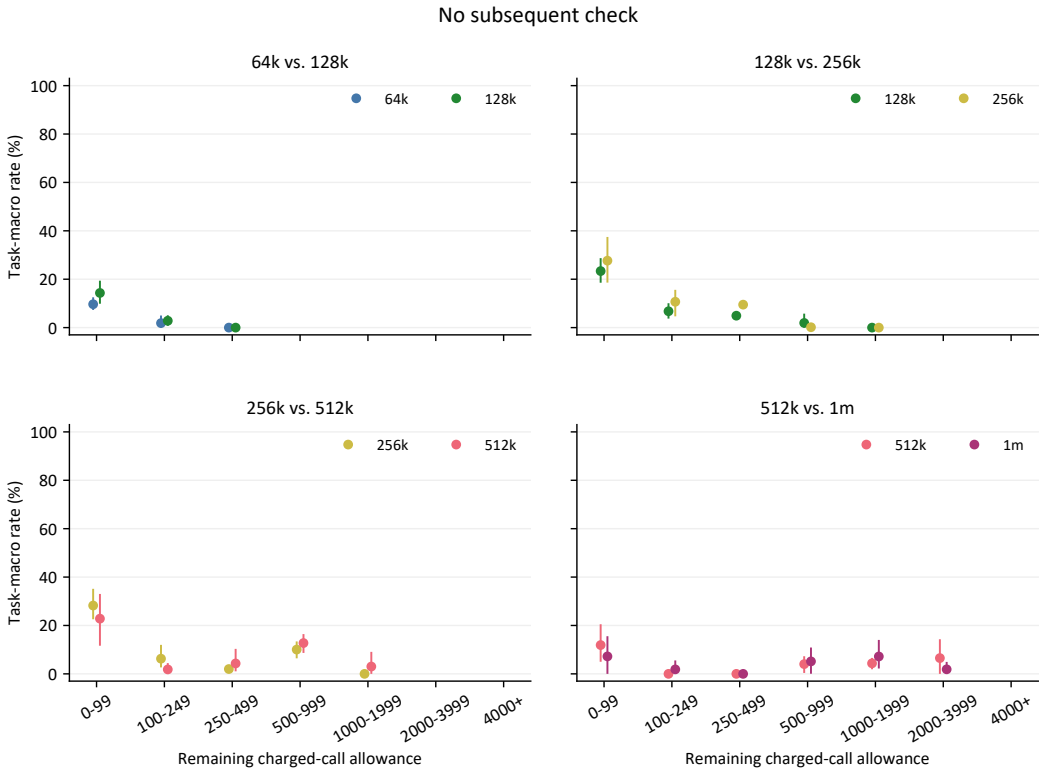

Figure 16: Goals with no subsequent semantic check after public business failure, using the same populations and weights as Figure 15. This category includes goals edited without further verification.

F.5 Verified Business-Goal Composition and Recovery

We link actual producer outputs to consumer arguments, object identity and version, executed business checks, input or state changes, and observed outcomes. The following trajectories illustrate predecessor selection, resource availability, and dependency repair during skill composition.

Passenger-specific predecessor identity. In Gemini's LogisticsOps `128k-handling_restricted-01` trajectory, two accepted leg-2 clearances, returned at attempts 207 and 208, belong to different passengers but both appear in the same leg-3 ground trip returned at 332. The target booking is revised at 350. At 401, the agent supplies clearance 10 as predecessor and receives `WRONG_PREVIOUS_LEG`; the actual predicate requires the target passenger's previous booking, not merely the numeric leg. At 410, changing only the predecessor to clearance 11 yields an accepted target clearance. The same booking and transfer trip are retained. This local join is corrected, while terminal business acceptance remains false with 24/32 bookings complete.

Historical acceptance versus current availability. In GPT's FactoryOps `128k-constrained_rework-08` trajectory, accepted material units are produced at 211 and 266. Scheduling one order at 357 reserves six units; a different order's scheduling call at 359 reuses those six and receives `MATERIAL_NOT_ACCEPTED_AND_AVAILABLE`. Both calls belong to the same model decision, so runtime order does not imply that the actor saw the first reservation receipt before choosing the second. A changed material selection at 385 is rejected again. No accepted schedule for that target occurs before the cutoff; the task ends with 6/24 orders complete. Different planned start times do not license duplicate material consumption.

Selective replacement after invalidation. In GPT's DocOps `128k-expiring_amendments-06` trajectory, one page's initial contract fails at 129. Six actual same-block edits lead to a full review pass at 188, followed by publication and tracker synchronization. Attempt 223 then invalidates that completed page. Its replacement contract at 239 combines an unchanged parent publication from 90 with a new parent publication from 237, preserving their exact page/version identities. It adds the amended provenance field, uses one-day retention, and changes origin-scoped capacity and withholding. All checks pass at 246, publication and synchronization follow, and the same page remains accepted at the final observation. Correct restoration requires updating the changed dependency while retaining still-valid evidence, not discarding every old artifact.

In the contrasting GPT DocOps `128k-expiring_amendments-08` trajectory, the child correctly binds three publications produced at 52, 208, and 218, and writes the correct aggregate incoming capacity of 243 units. Reviews at 249, 252, 255, and 262 nevertheless reject capacity and batch-size constraints. One real edit corrects two origin-level withheld amounts from 16/8 to 22/2 but does not restore the whole contract. Eight actual correction writes follow the first failed check. The last, at the observation cutoff 266, has no subsequent review and cannot be credited as recovery. The same page's native terminal acceptance is false. Partial repair and successful writes do not substitute for a passing business check.

F.6 Object-Level Execution around Compaction

These cases locate business operations relative to adopted summaries, verified in subsequent actor requests. A producer at attempt p precedes a consumer at c across replacement boundary b when $p \leq b < c$. This temporal ordering is distinct from consumption-time visibility: Appendix F.7 reconstructs consuming requests and rereads to identify required outputs removed from verbatim context. The cases below illustrate execution and correction around those boundaries.

Successful QC continuity and actual recovery. In DeepSeek's FactoryOps `128k-selective_rework-09` trajectory, the same output slot fails quality checks at 283, 351, and 381, before verified compaction at 394. Observation 394 reports fit 91 in the allowed range $[89, 93]$ but torque 56 outside $[51, 55]$. The disposition at 395 correctly quarantines that unit. Rework produces fit 92 and torque 57 at 404, followed by another quarantine. Further rework consumes that descendant and yields fit 91 and torque 54; finalization at 411 accepts it. Fit remains within range throughout these observations, while successive corrections bring torque into range and preserve unit ancestry. The task ends with 27/27 orders complete.

The physical cycle starts at 248 and first fails at 283, before compaction 394; its later 381-to-395 check interval crosses the boundary. The 128-call failure-to-recovery span includes interleaved work. Two canceled runs produce no physical output and are excluded from quality-repair counts.

A missed field returned after replacement. In GPT's DocOps `128k-expiring_amendments-07` trajectory, parent publications at 199 and 231 lie on opposite sides of verified compaction 205. The write at 234 binds both correct page/version triples, but review 240 rejects the field union while accepting binding, capacity, batch size, and retention. The missing field, `field_3814ba0e35`, appears in the parent payload returned at 231, in an earlier decision than the consuming write. Edit 241 adds only that field; review 244 passes, later publication succeeds, and the page is accepted at termination. The correction concerns content freshly returned after compaction.

The DocOps `06` trace above supplies a further timing contrast: first failure 129 precedes compaction 155, further actual edits follow it, and completion occurs at 188. Its first-check interval is not cross-boundary, while its 129-to-179 failed-check interval is. The LogisticsOps predecessor correction and FactoryOps competing reservation traces also cross verified producer-to-consumer boundaries (356 and 289, respectively). These temporal crossings establish event order; Appendix F.7 separately tests whether required outputs were actually removed from the consuming request.

The case records retain task identities, source lines, attempt numbers, consumed fields, and native outcomes.

F.7 Required Dependencies across Compaction

Consumption-time visibility. Both local comparison groups execute after an adopted summary. We determine exposure immediately before the first attributable business attempt, using required artifact roles reconstructed from the task contract and preceding public state. A required role can have alternative producer candidates. The retained group has all identified candidates fully available as verbatim structured tool returns in the consuming request. The removed group has at least one required role for which every identified candidate was present in a compaction request, removed when its summary was adopted, and still absent at consumption. Complete rereads restore visibility. Artifact identity and payload must occur in the same returned object; an identifier in a summary alone does not establish verbatim availability. Mixed alternatives and unresolved visibility are recorded separately.

Business scope. The required-input projections cover operation-specific artifacts in FinanceOps and FieldOps, report–brief slots in SocialOps, clearance/release/transfer roles in LogisticsOps, previous-round review candidates in ResearchOps, required parent publications in DocOps, and bill-of-materials or reservation objects in FactoryOps. The comparison measures visibility of required artifacts at attempted business checks. Eligibility requires an adopted compaction, an identifiable required role, and a resolved retained or removed classification. Submitted errors and missing inputs remain eligible when these conditions hold. Of the 5,250 executions, 130 have no required-object audit targets. Before paired-support selection, 434,833 cycles are classified as retained and 100,390 as removed. Other cycles are categorized by absent targets, pre-compaction timing, mixed alternatives, or unresolved observations.

Local outcomes and aggregation. The comparison includes 1,518 executions from 599 base tasks, with 2,610 matched execution–operation–cycle-kind cells across seven environments. First-check acceptance records the first observed business-check outcome. Ever accepted records at least one acceptance before the cycle ends, not necessarily continued validity at termination. Additional checks count subsequent check-bearing attempts after the first rejection until acceptance or cycle cutoff, including attempts on unresolved goals; first-pass goals contribute zero. We average targets within each group and cell, cells within an execution, executions within each environment, and finally the seven environments equally. Bootstrap intervals use 2,000 resamples of base-task labels within environments, keeping model observations of the same task together. Matching compares different goals within the same execution and operation; temporal position, goal difficulty, and business state can still differ. Table 40 reports support and environment-level contrasts. Equal weighting of the 1,518 executions instead yields removed-minus-retained differences of -7.43 percentage points in first-check acceptance, -1.63 points in ever accepted, and $+0.171$ additional checks.

Table 40: Strict dependency-visibility support and contrasts by environment. Counts are paired executions, distinct base tasks, and execution–operation–cycle-kind cells. Differences are removed minus retained; acceptance differences are percentage points and checks are counts. Aggregate uncertainty is reported in Table 41.

Environment	Executions	Tasks	Cells	First check	Ever accepted	Checks
FinanceOps	342	94	603	-3.51	+0.06	+0.121
FieldOps	302	89	841	-1.91	-0.15	+0.016
SocialOps	137	70	137	-8.55	+0.03	+0.141
LogisticsOps	159	90	332	-10.10	-5.62	+0.553
ResearchOps	120	68	120	-0.14	+0.00	+0.001
DocOps	277	96	365	-19.65	-1.41	+0.367
FactoryOps	181	92	212	-7.01	-6.49	+0.021

Task-level comparison. Compaction counts include summaries verified to enter a later saved request, including an ensuing compaction request. Rejected or unused summaries are excluded, and duplicated continuation prefixes are counted once. Counts are identifiable for 5,227 of the 5,250 final executions; 23 retain their terminal scores but have unresolved compaction counts. We compare zero with at least one adopted compaction in the 95 environment–workload–model strata containing both groups, with 1,116 and 1,719 executions, respectively. Each stratum receives the same weight in both groups. LGC, TS, and charged tool-call counts use these fixed weights. TS bounds combine Bonferroni-adjusted Clopper–Pearson cell intervals into conservative simultaneous 95% bounds. Compaction count is an observed execution property: early stopping can prevent compaction, while longer execution creates more opportunities for it. The comparison therefore describes task outcomes alongside compaction exposure.

Table 41: Local dependency visibility after compaction, averaged equally across seven environments. Differences are removed minus retained; intervals are 95% task-cluster bootstrap intervals.

Metric	Retained	Removed	Difference [95% CI]
First-check acceptance (%)	84.51	77.25	-7.27 [-8.59, -6.00]
Ever accepted (%)	91.76	89.82	-1.94 [-2.74, -1.15]
Additional checks	0.169	0.344	+0.174 [+0.132, +0.219]

Table 42: Task-level compaction comparison with fixed equal weights over 95 shared environment–workload–model strata. TS bounds are conservative simultaneous 95% bounds.

Compaction	Executions	LGC (%)	TS (%)	TS bounds	Mean calls
None	1,116	37.0	23.4	[9.6, 81.1]	303.4
At least one	1,719	71.3	41.1	[12.6, 75.4]	1,161.1

G Task-Sampling Sensitivity

We use a supplementary 500-task FinanceOps evaluation of Gemini 3.8 Flash at 128K: 167 heterogeneous-integration tasks, 167 receivables-reconciliation tasks, and 166 release-maintenance tasks. A total of 213 tasks pass final business evaluation, giving a success rate of 42.6%. The three families contribute 50, one, and 162 successes, corresponding to 29.9%, 0.6%, and 97.6%. We evaluate sampling sensitivity while holding family composition fixed.

We generate 1,000 task orders using NumPy's PCG64 generator with seed 20260927. For each order, we independently shuffle the sorted task labels within each family, then take one task per family in the fixed cycle of heterogeneous integration, receivables reconciliation, and release maintenance. We continue until all tasks have been included, skipping an exhausted family. Each task appears exactly once, and family counts differ by at most one in every prefix. At any given sample size, all orders therefore contain the same number of tasks from each family.

For every integer sample size from 50 to 500, Figure 6 reports the mean success rate over the 1,000 prefixes and their pointwise 5th–95th percentile range, computed with linear interpolation. These ranges measure subsampling variation within the fixed pool of evaluated tasks. The 90% range narrows from 36.0–48.0% at 50 tasks to 41.0–44.3% at 300; the corresponding means are 41.4% and 42.7%. Family counts are 17/17/16 at 50 tasks and 100/100/100 at 300, so integer allocation also affects the mean. Prefixes share tasks across sample sizes; as their coverage of the pool increases, the range converges to the full-set result of 42.6% at 500 tasks.