SKILL2ENV: CAPABILITY-ORIENTED ENVIRONMENT SYNTHESIS FROM SKILLS FOR GENERAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Executable environments are critical for post-training agents on tasks that require tool use and multi-step interaction, but constructing executable tasks together with their environments remains difficult to scale. Skills provide reusable domain knowledge, operational procedures, and tool-use instructions, but a substantial gap remains between the information contained in a skill and a concrete, challenging task with a complete executable environment. To address this gap, we introduce **Skill2Env**, a capability-oriented framework that starts from a skill and uses agent capability demands to guide task and environment synthesis. Skill2Env represents these demands through reusable difficulty patterns and instantiates them into task blueprints that specify objectives, challenges, environment facts, information boundaries, and acceptance criteria. These blueprints guide the joint construction of task instructions, execution substrates, workspaces, and rubric-based evaluators around source skills. We further propose **Iterative Task Hardening**, which uses solver execution evidence to identify insufficiently challenging task designs, strengthen or extend their difficulty-pattern instantiations, and revise the corresponding blueprints and environments. Using 1.5K high-scoring trajectories generated from Skill2Env environments for supervised fine-tuning, we observe consistent improvements across a broad range of agent benchmarks, demonstrating the effectiveness of capability-oriented environment synthesis for agent post-training.

1 INTRODUCTION

As large language models are increasingly deployed in real-world tasks that require tool use and multi-step interaction (Yao et al., 2023; Schick et al., 2023), executable environments have become an important component of post-training for improving agentic capabilities (Pan et al., 2025). Constructing executable tasks and their environments, however, involves more than generating task instructions. It also requires configuring usable tools, preparing a workspace that hosts task resources and state, and establishing evaluation mechanisms aligned with task objectives (Xie et al., 2024). Manually constructing and maintaining these interdependent components on a task-by-task basis is difficult to

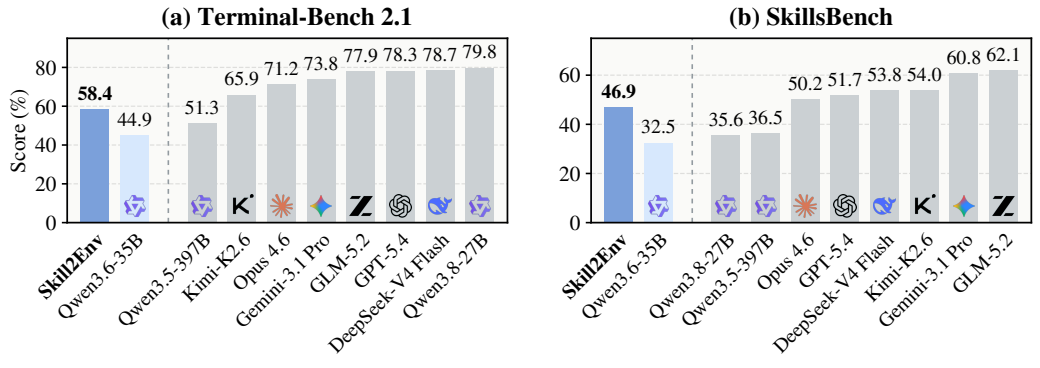


Figure 1: Skill2Env performance on Terminal-Bench 2.1 and SkillsBench.

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.

scale to the volume and diversity required for agent training. Automatically synthesizing executable tasks and their environments is therefore a key approach to scaling agent post-training (Gandhi et al., 2026; Cheng et al., 2026).

Skills provide a starting point for environment synthesis. A skill encapsulates domain knowledge, operational procedures, and tool-use instructions, and may include scripts, templates, and reference resources. Therefore, it can provide useful grounding for task design, tool configuration, and workspace construction. However, a substantial gap remains between the information contained in a skill and the synthesis of a concrete, challenging task and its executable environment. Current work on synthesizing environments from skills has explored several approaches. SkillSynth (Fan et al., 2026) models agent trajectories as sequences of alternating scenarios and skills; Terminal-World (Cheng et al., 2026) uses skills to jointly drive task-instruction synthesis and environment construction; FACET (Shi et al., 2026a) ensures consistency through cross-skill scenario reconstruction. Building on these efforts, we ask a further question: given a skill, how can we construct executable tasks and their environments based on agents' capability demands?

To address this question, we propose **Skill2Env**, a pipeline that starts from a skill and uses capability demands to guide environment synthesis. We structure agents' capability demands along five dimensions: environment understanding, planning, skill usage, long-horizon consistency, and error recovery. We further use difficulty patterns to specify how these demands are translated into concrete environment structures. For each skill, the synthesis agent first selects compatible difficulty patterns and specifies how these patterns should be reflected in a coherent task. It then constructs a task blueprint that jointly specifies the task objective, challenges, environment facts, information boundaries, and acceptance criteria. The blueprint guides the joint construction and refinement of the task instruction, execution substrate, workspace, and rubric-based evaluator around the source skill. For example, a document-processing skill only describes how to process documents using CLI tools, but Skill2Env can turn it into a concrete and challenging reconciliation task that requires the agent to resolve conflicting evidence and maintain consistency across outputs.

We further introduce **Iterative Task Hardening**, which uses solver execution evidence to progressively increase task difficulty. After each rollout, the system identifies aspects of the task that remain insufficiently challenging, strengthens existing difficulty-pattern instantiations or introduces additional compatible patterns, and revises the blueprint and environment accordingly. Reusable challenges revealed during this process can further extend the difficulty pattern pool. Repeating this process produces increasingly demanding tasks based on the solver's observed performance.

Our empirical study characterizes the resulting tasks and their environments and evaluates whether the supervision signals obtained from these environments can transfer. To assess transfer, we select 1.5K high-scoring trajectories for supervised fine-tuning and observe improvements in the model across a broad range of agent benchmarks.

Our main contributions are as follows:

- We introduce **Skill2Env**, a capability-oriented framework for skill-based task and environment synthesis. Through difficulty patterns and an explicit task blueprint, Skill2Env connects agent capability demands to the joint construction of task instructions, execution substrates, workspaces, and evaluators around source skills, yielding 2,963 executable tasks.
- We propose **Iterative Task Hardening**, which uses execution evidence from the solver agent to identify insufficiently challenging task designs and progressively strengthen the corresponding capability demands through coordinated updates to task blueprints and environments, yielding more demanding tasks and more effective training supervision.
- We demonstrate the effectiveness of Skill2Env for agent post-training. Supervised fine-tuning on 1.5K trajectories generated from Skill2Env environments yields consistent improvements across a broad range of agent benchmarks.

2 Related Work

Executable Environment and Task Synthesis. Existing methods for executable environment and task synthesis broadly follow two paradigms. *Workspace-anchored* methods start from an existing or recoverable workspace and derive tasks by restoring, modifying, or perturbing its state, as in

repository-, pull-request-, failure-, or trajectory-based synthesis (Jain et al., 2025; Chen et al., 2026; Lin et al., 2026; Wu et al., 2026). In contrast, *workspace-from-scratch* methods construct the task-specific workspace from higher-level specifications: Endless Terminals materializes containerized environments from generated terminal-task specifications (Gandhi et al., 2026); CLI-Universe and NexForge construct environments from capability or requirement specifications (Hua et al., 2026; Zhao et al., 2026); and skill-based methods synthesize executable environments from reusable agent skills (Fan et al., 2026; Cheng et al., 2026; Shi et al., 2026a; Tan et al., 2026). Skill2Env follows the latter paradigm and studies how to bridge the gap between the information contained in a skill and the synthesis of a concrete, challenging task and its executable environment.

Skill-Based Environment and Task Synthesis. Recent work uses skills to scale the construction of executable training environments and tasks. SkillSynth samples workflow paths from a scenario-mediated skill graph and instantiates them as terminal tasks (Fan et al., 2026). Terminal-World jointly derives task instructions, environments, and teacher trajectories from skills, and extends synthesis through skill teams and graphs (Cheng et al., 2026). FACET reconstructs coherent scenarios from related skills and uses the realized environment state to ground task instructions, reference solutions, and verifiers (Shi et al., 2026a). SKT combines rule-based and agent-based verification with feedback-guided repair to generate skill-grounded tasks and successful training trajectories (Tan et al., 2026). Building on these efforts, Skill2Env further focuses on how to construct tasks and executable environments based on explicit agent capability demands. It translates these demands into reusable difficulty patterns, and uses execution feedback to progressively harden tasks.

3 Preliminaries

Skills, environments, and tasks. An agent skill κ is a reusable package of procedural instructions and optional supporting resources for a class of tasks (Li et al., 2026; Xu & Yan, 2026). Its instructions encode domain-specific knowledge and procedures and may specify how tools should be used within a workflow. Let $\mathcal{X}$ denote the execution substrate that provides the tools and runtime support required for execution. A workspace W consists of task-specific local resources and their organization, which the agent can inspect and modify through the available tools. With initial workspace W_0, these components define an executable environment and a task instance:

$$\mathcal{E} = (\mathcal{X}, \kappa, W_0), \qquad \mathcal{T} = (g, \mathcal{E}, V), \tag{1}$$

where g is the task instruction specifying the objective and explicit requirements, and V is a task evaluator. In Skill2Env, environment synthesis is coupled with the generation of a task instruction and evaluator, yielding an executable task instance $\mathcal{T}$.

Interaction. Initializing $\mathcal{E}$ yields state s_0. At step t, state s_t includes the current workspace W_t and other runtime state relevant to execution. The agent receives a partial observation o_t and selects an action a_t using policy π. Given history $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ and environment transition mechanism $P_{\mathcal{E}}$,

$$a_t \sim \pi(\cdot \mid g, h_t), \qquad s_{t+1} \sim P_{\mathcal{E}}(\cdot \mid s_t, a_t). \tag{2}$$

An execution lasting L steps produces a rollout $\tau = (o_0, a_0, \ldots, a_{L-1}, o_L)$. After execution, the evaluator assigns a task reward $r = V(s_L) \in [0, 1]$.

4 Skill2Env

We introduce Skill2Env, a capability-oriented pipeline for jointly synthesizing tasks and executable environments from skills. As shown in Figure 2, Skill2Env curates executable skills, operationalizes capability demands through difficulty patterns, and organizes environment construction based on task blueprints. The blueprint serves as an author-side contract for the task instruction, execution substrate, workspace, and rubric-based evaluator. Initial synthesis establishes the intended challenges. Iterative Task Hardening then uses execution evidence to progressively strengthen challenges that the solver already handles well.

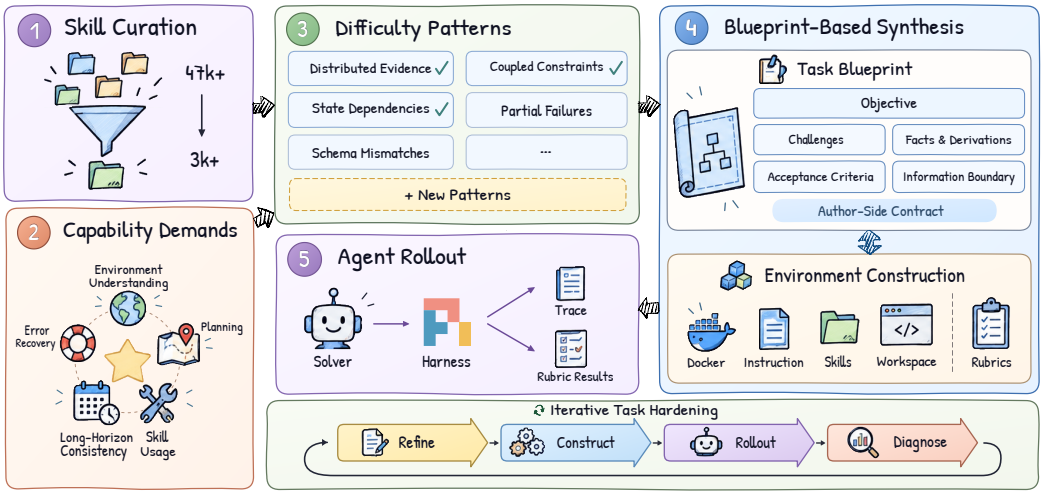


Figure 2: **Overview of Skill2Env.** Curated skills and capability demands are connected through difficulty patterns and organized into task blueprints, which guide the construction of executable tasks and their environments. Solver rollouts provide rubric results, and behavioral evidence for environment validation and task diagnosis. This feedback further supports difficulty-pattern expansion and Iterative Task Hardening through coordinated updates to task blueprints and environments.

4.1 Skill Curation

Skills provide reusable domain knowledge and operational procedures for environment synthesis. We curate skills from public skill libraries that connect this knowledge to practical tool operations, providing both domain context for task design and operational support for execution.

We downloaded more than 47K public skill folders from ClawHub (OpenClaw, 2026) and screened them using 3 criteria: *Practical Utility*, *Runtime Compatibility*, and *Setup Feasibility*. Practical Utility requires the skill to serve a clear and substantive purpose. Runtime Compatibility requires its hardware, operating system, permission, and compute requirements to be compatible with the target Linux sandbox. Setup Feasibility requires that the inputs, dependencies, data, and services needed for execution can be feasibly provisioned. Applying these criteria, we retained more than 3K skill folders for subsequent environment synthesis.

4.2 Capability-Oriented Environment Synthesis

We next describe how Skill2Env translates agent capability demands into concrete task and environment designs grounded in the curated skills.

From agent capability demands to task difficulty patterns. Completing complex tasks typically involves a continuous process of understanding, planning, execution, and adjustment. Throughout the process, an agent needs to continuously interpret task-relevant facts, objectives, constraints, and evolving environment states, and use this evolving understanding to formulate and revise its execution plan. It also needs to interpret and appropriately apply the tool-use instructions and execution procedures provided by the skill according to the current task requirements. As the task progresses across multiple steps, the agent needs to maintain consistency between intermediate results and the environment state. When errors occur, it should use execution feedback to locate problems, adjust its operations, and continue making progress. Based on this execution process, we identify five core capability demands to guide subsequent environment design: environment understanding, planning, skill usage, long-horizon consistency, and error recovery (Table 2 in Appendix A).

Capability demands specify which agent capabilities a task is intended to challenge. Constructing an executable environment further requires specifying how task design can place demands on these capabilities. We therefore introduce difficulty patterns, each describing a reusable class of task chal-

lenges and how to construct them. Each agent capability demand is instantiated through multiple task difficulty patterns, each capturing a distinct source of difficulty. For environment understanding, *distributed evidence* spreads facts across sources, while *implicit constraints* requires inferring unstated rules. For planning, *state dependencies* links later operations to earlier results, while *resource budgets* limits cumulative resource use.

Let $\mathcal{P}$ denote the difficulty pattern pool. We initialize $\mathcal{P}$ with 100 reusable patterns, organized into five capability dimensions (Appendix B). The initial patterns are distilled from recurring challenge structures observed during pilot synthesis.

To apply these patterns to a skill κ, the synthesis agent identifies the workflows it supports and selects a compatible subset $\mathcal{P}_\kappa \subseteq \mathcal{P}$. Selection depends on whether these challenges can be naturally integrated into a common task objective and whether the skill supports the operations needed to address them. Different combinations of patterns allow the same skill to support tasks with different capability demands.

Blueprint-guided environment construction. Given a skill κ and selected difficulty patterns $\mathcal{P}_\kappa$, the synthesis agent drafts a task blueprint B that instantiates the intended challenges within a coherent task. As an author-side contract, B links task requirements and challenges to supporting facts, acceptance criteria, execution requirements, and information boundaries. Execution requirements specify the necessary tool capabilities, runtime dependencies, and initial resources and state. Information boundaries distinguish what the instruction provides, what the solver must discover in the workspace, and what it must infer. Facts and evidence necessary for solving remain accessible to the solver, while reference outcomes and derivations, where applicable, remain on the author side.

Given the source skill κ, the synthesis agent uses the blueprint to construct the task instruction g, execution substrate $\mathcal{X}$, initial workspace W_0, and rubric-based evaluator V, forming the executable task instance $\mathcal{T}$:

$$(\kappa, B) \xrightarrow{\text{synthesis}} \mathcal{T} = \big(g, (\mathcal{X}, \kappa, W_0), V\big). \tag{3}$$

Workspace construction combines synthesized task materials with retrieved real-world materials, organized according to the blueprint's facts and evidence requirements. Blueprint design and environment construction proceed iteratively: the synthesis agent may revise the blueprint to address construction issues or incorporate improved task designs, then update the affected components under the revised contract.

Rubric-based evaluation. The evaluator translates the blueprint's acceptance criteria into a set of rubric items $\mathcal{C}$. Each item $c \in \mathcal{C}$ is assigned a positive weight w_c during task synthesis, with weights summing to one. Each item yields a score $q_c \in [0, 1]$, where $q_c = 1$ denotes full satisfaction. Rubric items amenable to programmatic checking use deterministic verification, which typically returns binary scores. The remaining items use an LLM judge, which may assign continuous partial-credit scores according to the item's scoring criteria. Each result is recorded with supporting evidence. The task reward is the weighted average of the rubric results:

$$r = V(s_L) = \sum_{c \in \mathcal{C}} w_c q_c(s_L), \qquad \sum_{c \in \mathcal{C}} w_c = 1, \quad w_c > 0. \tag{4}$$

Consistency validation. After a solver rollout, a validation agent checks task conditions and rubric results against the blueprint and execution evidence. Issues requiring changes to the task contract are addressed through coordinated updates to the blueprint and affected components. Implementation-only issues are corrected against the existing blueprint. Evidence attributable to inconsistent task conditions or evaluator errors is excluded from capability diagnosis, so task hardening is grounded in valid task demands and rubric results.

4.3 Iterative Task Hardening

Initial synthesis instantiates the intended capability demands through difficulty patterns, but some resulting challenges may still be readily handled by the solver agent. We therefore introduce **Iterative Task Hardening**, which uses execution evidence to progressively strengthen task challenges according to the solver's observed performance.

Execution-based diagnosis. At iteration j, the solver executes task $\mathcal{T}^{(j)}$ constructed from blueprint $B^{(j)}$, producing trajectory $\tau^{(j)}$ and rubric results $\mathbf{q}^{(j)}$. We use this reward as a *hardening trigger*: tasks above a predefined threshold undergo further hardening, while the remaining tasks are retained. For tasks selected for further hardening, the diagnosis agent compares the execution with the challenges specified in the blueprint, examining how the solver handles the instantiated difficulty patterns, whether intended challenges are bypassed through shortcuts, and where residual failures occur:

$$D^{(j)} = \text{Diagnose}\left(B^{(j)}, \tau^{(j)}, \mathbf{q}^{(j)}\right). \tag{5}$$

When diagnosis reveals a generalizable challenge not represented in the current pattern pool, the challenge is abstracted into a reusable difficulty pattern and added to $\mathcal{P}$ for subsequent task design.

Hardening proposal and task revision. Guided by $D^{(j)}$, the system proposes how to strengthen aspects of the task that are already well handled by the solver. The proposal may increase the instantiation strength of existing difficulty patterns or introduce additional patterns from $\mathcal{P}$ that are compatible with the current skill and workflow. The resulting hardening proposal $H^{(j)}$ is incorporated into the task blueprint, after which the revised blueprint guides coordinated updates to the task instruction, execution substrate, workspace, and evaluator:

$$\begin{aligned} H^{(j)} &= \text{ProposeHardening}\left(\kappa, B^{(j)}, D^{(j)}, \mathcal{P}\right), \\ B^{(j+1)} &= \text{ReviseBlueprint}\left(B^{(j)}, H^{(j)}\right), \\ \mathcal{T}^{(j+1)} &= \text{Construct}\left(\kappa, B^{(j+1)}\right). \end{aligned} \tag{6}$$

Applying the complete Skill2Env pipeline yields 2,963 executable tasks. A comprehensive analysis of environment diversity is provided in Appendix C.

Appendix E provides an end-to-end Docling case study illustrating the task blueprint, workspace, evaluator, and successive hardening revisions.

5 Experiments

We evaluate whether supervision from Skill2Env environments improves agent performance across a range of benchmarks.

5.1 Experimental Settings

Baselines. We compare Skill2Env with frontier foundation models and a prior skill-based environment synthesis method. The closed-source references are GPT-5.4 (OpenAI, 2026), Claude Opus 4.6 (Anthropic, 2026), and Gemini-3.1 Pro (Google DeepMind, 2026). Open-weight foundation models include DeepSeek-V4-Flash (0731) (DeepSeek-AI, 2026), GLM-5.2 (GLM-5 Team et al., 2026), Kimi-K2.6 (Moonshot AI, 2026), Qwen3.8-27B (Qwen Team, 2026c), Qwen3.5-397B-A17B (Qwen Team, 2026a), and our backbone, Qwen3.6-35B-A3B (Qwen Team, 2026b). We include FACET (Shi et al., 2026a) as a skill-based environment synthesis baseline.

Benchmarks and evaluation protocol. Our main comparison covers seven benchmarks. Terminal-Bench 2.1 (Merrill et al., 2026) evaluates terminal-task execution; we report task success rate on the complete task set using Terminus-2 with a single trial per task and a 10,800 s timeout. SWE-bench Multilingual (Yang et al., 2025) evaluates repository-level issue resolution across programming languages; we report resolved-instance rate over all 300 instances using mini-SWE-agent with a single trial per instance and a 7,200 s timeout. SkillsBench (Li et al., 2026) evaluates reusable skill use across domains; we report Avg@3 task reward using OpenHands with skills enabled and a 10,800 s timeout. Claw-Eval (Ye et al., 2026) measures autonomous task execution; we evaluate 199 non-multimodal tasks using its native agent loop with three trials per task and report Pass^3. τ^3-Banking (Shi et al., 2026b), AutomationBench (Shepard & Salimans, 2026), and VitaBench (He et al., 2025) evaluate knowledge-grounded banking support, cross-application workflows, and interactive service tasks, respectively; we report Pass^1, pass rate (version 1.0.6), and mean score. FACET results are source-reported.

Implementation details. We use Kimi-K3 for environment synthesis and execution diagnosis, Qwen3.6-35B-A3B as the fixed solver during Iterative Task Hardening, and Qwen3.5-397B-A17B for LLM-based rubric judging. During task hardening, we set the hardening-trigger threshold to 0.7, such that tasks with reward above this threshold undergo further hardening, while the remaining tasks are retained.

For supervised fine-tuning, the teacher model DeepSeek-V4-Flash generates trajectories on the resulting tasks, from which we retain 1.5K trajectories satisfying the SFT quality filter $r > 0.9$. We fine-tune Qwen3.6-35B-A3B on these trajectories for 5 epochs with a global batch size of 32. Training uses Megatron with a peak learning rate of 10^{-5}, a minimum learning rate of 10^{-6}, cosine decay, and a 10% warmup fraction. Unless otherwise specified, the main results are reported using the resulting SFT checkpoint.

5.2 Main Results

Table 1 shows that Skill2Env improves Qwen3.6-35B-A3B on all seven downstream benchmarks, increasing the unweighted average from 36.6 to 45.0 (+8.4 points) with only 1.5K SFT trajectories. The largest gains are observed on SkillsBench (+14.34) and Terminal-Bench 2.1 (+13.5), while substantial improvements also appear on SWE-bench Multilingual (+7.7), AutomationBench (+7.34), VitaBench (+5.87), Claw-Eval (+5.51), and τ^3-Banking (+4.47). Figure 1 highlights the Terminal-Bench 2.1 and SkillsBench results.

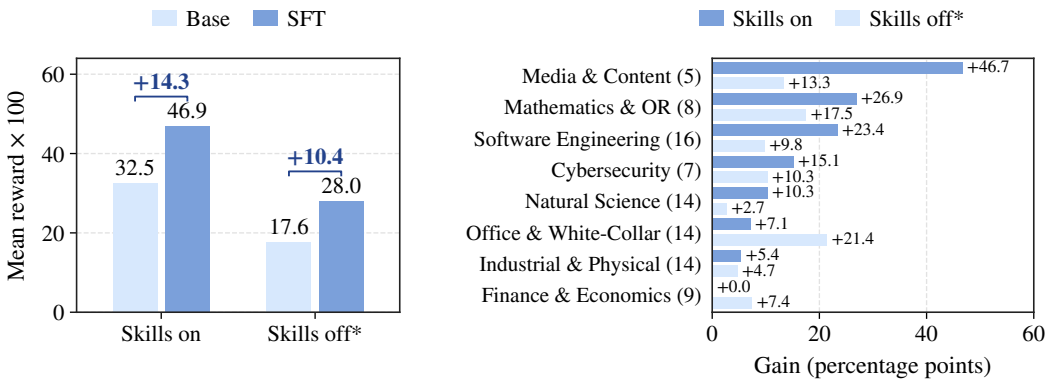
Table 1: Main results on seven agent benchmarks.

Model	Terminal-Bench 2.1	SWE-bench Multilingual	Skills Bench	Claw-Eval	τ^3-Banking	Automation Bench	VitaBench	Avg.
Frontier Closed-Source Models								
GPT-5.4	78.3	71.7	51.7	60.3	28.5	27.7	47.1	**52.2**
Claude Opus 4.6	71.2	77.8	50.2	70.4	20.3	25.5	38.3	**50.5**
Gemini-3.1 Pro	73.8	44.0	60.8	57.8	23.7	28.2	52.7	**48.7**
Open-Weight Models								
DeepSeek-V4-Flash-0731	78.7	76.0	53.8	49.3	30.3	36.3	56.3	**54.4**
GLM-5.2	77.9	81.7	62.1	65.8	28.9	26.3	50.3	**56.1**
Kimi-K2.6	65.9	76.7	54.0	62.3	19.9	26.3	43.9	**49.9**
Qwen3.5-397B-A17B	51.3	66.0	36.5	56.8	16.2	5.5	42.1	**39.2**
Qwen3.6-35B-A3B	44.9	63.3	32.5	55.8	10.7	10.3	38.9	**36.6**
Qwen3.8-27B	79.8	73.8	35.6	69.2	33.7	38.5	41.8	**53.2**
Skill-Based Environment Synthesis								
FACET-Terminal-Qwen3.5-27B	47.6	-	-	-	-	-	-	**-**
Skill2Env (ours / 35B-A3B)	58.4	71.0	46.9	61.3	15.1	17.7	44.8	**45.0**
Gain over baseline	**(+13.5)**	**(+7.7)**	**(+14.3)**	**(+5.5)**	**(+4.5)**	**(+7.3)**	**(+5.9)**	**(+8.4)**

Scores are rounded to one decimal place for display. Reported gains are computed from the unrounded scores before rounding.

Importantly, Skill2Env does not optimize for any particular downstream benchmark. Instead, its synthesized environments are organized around general agent capabilities. The consistent gains across benchmarks with different domains, interfaces, and task structures therefore suggest that supervision constructed around these shared capability demands transfers beyond the synthesized training environments.

Skill2Env also exceeds Qwen3.5-397B-A17B on six of the seven benchmarks, despite using the substantially smaller Qwen3.6-35B-A3B backbone. This result provides additional evidence that the improvement comes from the transferable supervision supplied by the synthesized environments rather than model scale alone. At the same time, frontier-model references remain stronger on several benchmarks, indicating substantial remaining headroom. Overall, the results support the central hypothesis of Skill2Env: executable environments constructed around reusable capability demands can provide supervision that generalizes across heterogeneous downstream agent tasks.



(a) Overall performance

(b) Gains across domains

Figure 3: **SkillsBench improvements after SFT under OpenHands.** (a) Overall scores before and after fine-tuning; blue annotations show absolute gains. (b) Absolute score gains across eight domains; parentheses give the number of tasks. Scores are mean task reward $\times 100$; gains are differences in percentage points. Skills off* disables automatic loading while leaving skill files accessible.

5.3 Analysis of SkillsBench Improvements

Skill2Env constructs environments around five capability demands (Section 4.2); we examine whether the resulting supervision improves performance on SkillsBench tasks that exercise these capabilities across domains. Figure 3(a) shows gains of 14.3 percentage points with skills on and 10.4 with skills off[1] under OpenHands. Gains span seven of eight domains with skills on, with finance unchanged, and all eight with skills off (Figure 3(b)). Media, mathematics and operations research, and software engineering lead the skills-on gains (+46.7, +26.9, and +23.4 percentage points).

Qualitative trajectory analysis suggests corresponding behavioral improvements. In media production, the fine-tuned model better coordinates dependent processing stages while maintaining requirements for the final output, reflecting stronger planning and long-horizon consistency. In office tasks, it more completely identifies and processes content distributed across document structures, reflecting improved environment understanding and reducing omissions that prevent task completion. Across the examined trajectories, we also observe better adaptation of available guidance and helper scripts to task requirements, as well as more effective use of execution feedback to diagnose failures, repair operations, and resume progress. These behaviors correspond to the five capability demands targeted by Skill2Env and suggest that the resulting supervision improves reusable execution capabilities rather than benchmark-specific behaviors.

5.4 Terminal-Bench 2.1 Performance Across Harnesses

We further examine whether the improvements extend across different harnesses. On Terminal-Bench 2.1, the fine-tuned model improves over the base model under all four harnesses (Figure 4). Scores increase from 46.1 to 57.3 with `pi` (+11.2 percentage points), from 44.9 to 58.4 with Terminus-2 (+13.5), from 38.2 to 55.1 (+16.9) with OpenCode and from 40.5 to 52.8 (+12.3) with Claude Code harness, respectively. The consistent gains across harnesses support the generalization of agent capability improvements.

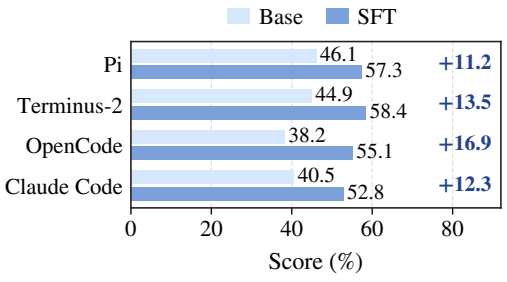


Figure 4: **Terminal-Bench 2.1.** Blue labels show gains (percentage points).

[1] Skills off disables automatic loading; skill files remain accessible in the environment.

5.5 Effectiveness of Iterative Task Hardening

We evaluate Iterative Task Hardening from two complementary perspectives: whether successive revisions increase task difficulty and whether the resulting task collections provide more effective training supervision.

Effect on task difficulty. To isolate the effect of hardening while controlling for task identity, we use 500 paired task identities that undergo hardening in both rounds and compare their versions at iterations 0, 1, and 2. With DeepSeek-V4-Flash under the `pi` harness, the full-credit rate ($r = 1$) decreases from 48.40% to 24.80% and 15.40%, while the mean number of assistant turns increases from 25.91 to 36.54 and 38.85 (Figure 5(a–c)). From iteration 0 to 2, this corresponds to a 33.0-percentage-point reduction in full-credit rate and approximately 50% more interaction steps. Together, these paired results indicate that successive hardening revisions make the affected tasks substantially more demanding.

Training utility across hardening stages. We next evaluate the training utility of the task collections after zero, one, and two hardening rounds. Tasks that do not undergo further hardening are carried forward unchanged, while hardened tasks are replaced by their revised versions. For each iteration, we apply the same SFT quality filter $r > 0.9$ and randomly sample 500 qualifying trajectories, thereby holding both the training-set size and quality threshold fixed. We use random rather than matched samples because the set of qualifying trajectories changes across iterations.

The average score across seven benchmarks increases from 40.58 to 42.51 and 42.87 for iterations 0, 1, and 2, respectively (Figure 5(d)), yielding a 2.29-point improvement from iteration 0 to 2. These results indicate that, under the same data budget and quality threshold, later hardening stages improve average downstream training utility. Appendix D reports the individual benchmark scores.

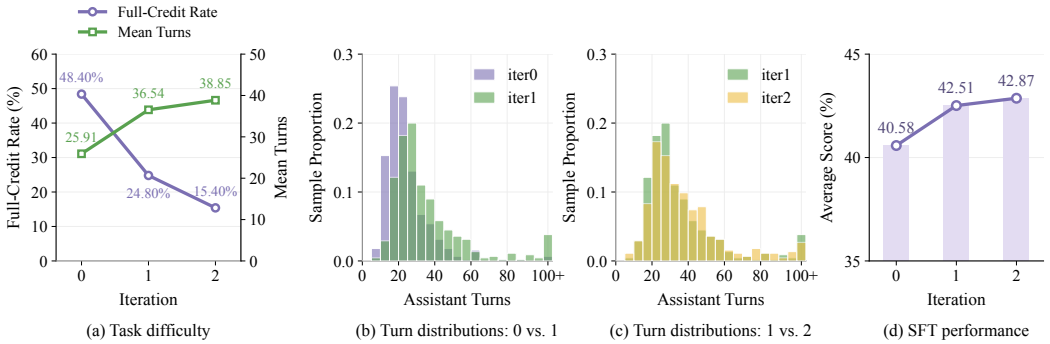


Figure 5: **Task hardening and downstream training utility.** (a) DeepSeek-V4-Flash full-credit rate ($r = 1$, left axis) and mean recorded assistant turns (right axis) over 500 tasks. (b–c) Assistant-turn distributions for 500 tasks at all three iterations, comparing iterations 0–1 and 1–2, respectively. (d) Seven-benchmark mean after SFT on 500 trajectories per iteration.

6 Conclusion

We presented Skill2Env, a capability-oriented framework for synthesizing executable tasks and environments from skills. Skill2Env uses reusable difficulty patterns to translate agent capability demands into concrete task challenges, and organizes their realization through task blueprints that coordinate task instructions, execution substrates, workspaces, and evaluators. Iterative Task Hardening further uses execution evidence to progressively revise task designs toward more demanding capability requirements. Across a broad range of agent benchmarks, supervised fine-tuning on 1.5K high-scoring trajectories generated from Skill2Env environments consistently improves agent performance. These results suggest that skill-based environment synthesis can move beyond expanding skill and task coverage by explicitly organizing task construction around the capabilities agents are required to exercise, providing a foundation for generating increasingly demanding environments as agent capabilities improve.

AI Use Statement

Large language models are used as components of our experimental pipeline, including Kimi-K3 for environment synthesis and execution diagnosis, Qwen3.6-35B-A3B as the solver and training backbone, Qwen3.5-397B-A17B for LLM-based evaluation, and DeepSeek-V4-Flash for generating training trajectories. Their roles and evaluation settings are described in the main text and appendix. During manuscript preparation, generative AI tools were additionally used to assist with English translation and language polishing, as well as to improve the presentation of figures, tables, and captions. The authors retain full responsibility for the paper's technical content, experimental design and results, citations, conclusions, and the accuracy and integrity of the final manuscript.

References

Anthropic. Claude Opus 4.6 System Card. System card, 2026. URL https://www-cdn.anthropic.com/14e4fb01875d2a69f646fa5e574dea2b1c0ff7b5.pdf.

Mouxiang Chen, Lei Zhang, Yunlong Feng, Xuwu Wang, Wenting Zhao, Ruisheng Cao, Jiaxi Yang, Jiawei Chen, Mingze Li, Zeyao Ma, Hao Ge, Zongmeng Zhang, Zeyu Cui, Dayiheng Liu, Jingren Zhou, Jianling Sun, Junyang Lin, Binyuan Hui, and Zhongxin Liu. SWE-Universe: Scale real-world verifiable environments to millions, 2026. URL https://arxiv.org/abs/2602.02361v2.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-World: Scaling terminal-agent environments via agent skills. *arXiv preprint arXiv:2605.20876*, 2026. URL https://arxiv.org/abs/2605.20876.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026. URL https://dblp.org/rec/journals/corr/abs-2606-19348.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, and Lilin Wang. Toward scalable terminal task synthesis via skill graphs. *arXiv preprint arXiv:2604.25727*, 2026. URL https://arxiv.org/abs/2604.25727.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless terminals: Scaling RL environments for terminal agents. *arXiv preprint arXiv:2601.16443*, 2026. URL https://arxiv.org/abs/2601.16443.

GLM-5 Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen

Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.1 Pro Model Card. Model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, Man Gao, Xi Su, Xiaodong Cai, Xunliang Cai, Yu Yang, and Yunke Zhao. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications, 2025. URL https://arxiv.org/abs/2509.26490.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, Letian Zhu, Xinping Lei, Han Li, Zhiyuan Ma, Zili Wang, Zhaoxiang Zhang, and Jiaheng Liu. CLI-Universe: Towards verifiable task synthesis engine for terminal agents. *arXiv preprint arXiv:2606.22883*, 2026. URL https://arxiv.org/abs/2606.22883.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7evvwwdo3z.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026. URL https://arxiv.org/abs/2602.12670.

Yusong Lin, Haiyang Wang, Shuzhe Wu, Lue Fan, Feiyang Pan, Sanyuan Zhao, and Dandan Tu. CLI-Gym: Scalable CLI task generation via agentic environment inversion. *arXiv preprint arXiv:2602.10999*, 2026. URL https://arxiv.org/abs/2602.10999.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

Moonshot AI. Kimi K2.6: Advancing open-source coding. Technical blog, 2026. URL https://www.kimi.ai/blog/kimi-k2-6.

OpenAI. GPT-5.4 Thinking System Card. System card, 2026. URL https://openai.com/index/gpt-5-4-thinking-system-card/.

OpenClaw. ClawHub. https://clawhub.ai/, 2026. Accessed September 21, 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026c. URL https://qwen.ai/blog?id=qwen3.8.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997.

Daniel Shepard and Robin Salimans. Automationbench, 2026. URL https://arxiv.org/abs/2604.18934.

Kou Shi, Zun Wang, Qisheng Su, Shiting Huang, Ziao Zhang, Zhen Fang, Qingnan Ren, Jin Liu, Yu Zeng, Yiming Zhao, Lin Chen, Zehui Chen, and Feng Zhao. FACET: Preserving source intent and executable state in terminal task synthesis. *arXiv preprint arXiv:2608.18580*, 2026a. URL https://arxiv.org/abs/2608.18580.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-Knowledge: Evaluating conversational agents over unstructured knowledge, 2026b. URL https://arxiv.org/abs/2603.04370.

Zelin Tan, Yiqun Zhang, Hao Li, Zhiyao Cui, Hejia Geng, Shao Zhang, Hangfan Zhang, Yang Chen, Xiaosong Wang, Lilong Wang, Zhenfei Yin, Shuyue Hu, Chen Zhang, and Lei Bai. SKT: Skill-use training at scale via verified synthetic data generation. *arXiv preprint arXiv:2608.02287*, 2026. URL https://arxiv.org/abs/2608.02287.

Jie Wu, Zhenru Zhang, Beichen Zhang, Xuwu Wang, Yuhui Su, Mouxiang Chen, Peng Wang, Zhihai Wang, Que Shen, Hao Zhou, An Yang, Fei Huang, Yujiu Yang, and Dayiheng Liu. Terminal-Universe: Turning agent trajectories into scalable terminal environments. *arXiv preprint arXiv:2609.04148*, 2026. URL https://arxiv.org/abs/2609.04148.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026. URL https://arxiv.org/abs/2602.12430.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-3239. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8b86cf5ace600c48fd188efbb8dedec8-Abstract-Datasets_and_Benchmarks_Track.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-eval: Towards trustworthy evaluation of autonomous agents, 2026. URL https://arxiv.org/abs/2604.06132.

Jiarong Zhao, Zhikai Lei, Zhiheng Xi, Rui Zheng, Hang Yan, Jie Zhou, Qin Chen, and Liang He. NexForge: Scaling agent capabilities through requirement-driven task synthesis for LLMs. *arXiv preprint arXiv:2607.14186*, 2026. URL https://arxiv.org/abs/2607.14186.

A Capability Demands

Table 2 summarizes the five capability demands in Section 4.2; Appendix B operationalizes them as concrete difficulty patterns.

Table 2: Capability demands used to guide environment synthesis.

Capability Demand	**Agent Requirements**
Environment Understanding	Identify task-relevant facts from the information provided by the environment and understand the task objectives and constraints.
Planning	Identify dependencies between operations and continually adapt the execution plan based on evolving environment states.
Skill Usage	Understand and appropriately apply the tool-use instructions and execution procedures provided by the skill according to the current task requirements.
Long-Horizon Consistency	Continuously track state changes during multi-step execution and maintain consistency between intermediate results and the environment state.
Error Recovery	Locate errors using execution feedback, adjust operations, and resume task execution.

B Difficulty Patterns

B.1 Environment Understanding: Patterns

Table 3: Difficulty patterns for environment understanding.

Pattern	**Construction and hardening control**
U01 Target discrimination	Provide near-identical files or records; relational clues uniquely identify the requested target. *Control: candidate similarity.*
U02 Distributed evidence	Split complementary facts across known sources so no single source supports the complete answer. *Control: source count.*
U03 Nested discovery	Place accessible evidence inside nested archives, linked records, or embedded objects. *Control: access depth.*
U04 Scope enumeration	Require all qualifying objects across partitions or pages, with a discoverable completeness criterion. *Control: partition count.*
U05 Sparse-signal retrieval	Embed a small set of relevant facts among topically similar but irrelevant passages. *Control: distractor ratio.*
U06 Authority resolution	Provide conflicting claims with an explicit, accessible hierarchy of authoritative sources. *Control: authority tiers.*
U07 Temporal validity	Provide individually accurate historical records, only some of which apply to the requested date. *Control: revision count.*
U08 Provenance reconstruction	Provide copied or aggregated claims whose reliability depends on tracing them to their original source. *Control: attribution depth.*
U09 Observation completeness	Expose a truncated preview or sampled view; require checking its coverage before interpreting missing items. *Control: omission extent.*
U10 Measurement reliability	Supply noisy observations with documented error bounds; require a justified estimate rather than trusting one reading. *Control: noise level.*
U11 Contextual disambiguation	Use an ambiguous request whose intended meaning is uniquely resolved by supplied context. *Control: plausible meanings.*

Continued on next page

Table 3 (continued)

Pattern	Construction and hardening control
U12 Entity resolution	Refer to the same entity using aliases or incompatible identifiers across sources; provide linking evidence. *Control: alias multiplicity.*
U13 Vocabulary grounding	Use local business terms whose definitions differ from common usage and are documented in the workspace. *Control: definition count.*
U14 Implicit constraints	Make a rule inferable from accessible configurations or consistent examples without restating it in the instruction. *Control: inference depth.*
U15 Instruction provenance	Include quoted instructions inside task data; require distinguishing them from the authorized task contract. *Control: distractor salience.*
U16 Temporal inference	Require event ordering or interval overlap to be inferred from several partial temporal statements. *Control: relation-chain length.*
U17 Spatial interpretation	Require the intended coordinate frame or orientation to be inferred from local landmarks and calibration evidence. *Control: frame ambiguity.*
U18 Quantitative inference	Provide observations from which an unstated rate, denominator, or dimensional relationship must be derived. *Control: unknown relations.*
U19 Causal attribution	Provide controlled comparisons that distinguish causal effects from associations in an otherwise functioning process. *Control: alternative explanations.*
U20 Negative-evidence reasoning	Require an absence-based conclusion only after establishing that the relevant observation process was complete. *Control: coverage conditions.*

B.2 Planning: Patterns

Table 4: Difficulty patterns for planning.

Pattern	Construction and hardening control
P01 State dependencies	Make later actions require specific earlier outputs or state transitions, forming a nontrivial dependency graph. *Control: dependency depth.*
P02 Bootstrap dependencies	Introduce an apparent dependency cycle with a documented bootstrap action that makes the workflow feasible. *Control: cycle size.*
P03 Conditional workflows	Make the correct continuation depend on an earlier result, requiring an explicit contingent plan. *Control: branch depth.*
P04 Subgoal decomposition	Specify one coherent objective that must be decomposed into intermediate milestones and actionable work units. *Control: milestone count.*
P05 Probe scheduling	Offer several informative probes with different costs; require choosing which uncertainty to resolve next. *Control: probe alternatives.*
P06 Coupled constraints	Make individually feasible requirements share decision variables, so they must be solved jointly. *Control: coupling density.*
P07 Priority hierarchies	Specify mandatory obligations and ordered preferences; lower-priority gains cannot override higher-priority requirements. *Control: priority levels.*
P08 Utility tradeoffs	Provide competing feasible solutions and an explicit utility rule requiring a justified compromise. *Control: objective count.*
P09 Obligation coverage	Let each candidate action satisfy a subset of requirements; require selecting a set covering all mandatory obligations. *Control: coverage overlap.*
P10 Optional-goal selection	Offer optional deliverables under a fixed capacity; require selecting which goals to pursue, not merely their order. *Control: optional-goal count.*

Continued on next page

Table 4 (continued)

Pattern	Construction and hardening control
P11 Deadline scheduling	Assign durations and deadlines to dependent actions; require a feasible schedule with limited slack. *Control: deadline slack.*
P12 Resource budgets	Fix the required goals but limit cumulative expenditure, such as queries, tokens, or simulated cost. *Control: budget tightness.*
P13 Shared capacity	Make concurrent actions compete for reusable workers, memory slots, or exclusive equipment. *Control: capacity contention.*
P14 Locality placement	Make execution cost depend on where data and actions are placed; moving inputs incurs a documented cost. *Control: locality asymmetry.*
P15 Shared-work factoring	Provide related subtasks with reusable intermediate work; independent execution wastes a constrained resource. *Control: reusable overlap.*
P16 Commitment timing	Include an irreversible but valid action that removes future options; require deciding when enough evidence exists to commit. *Control: lost options.*
P17 Outcome contingencies	Give actions several legitimate outcomes with known bounds; require reserving feasible continuations before observing the outcome. *Control: outcome uncertainty.*
P18 Exogenous replanning	Introduce observable changes in legitimate goals or availability, requiring revision of a previously feasible plan without a fault. *Control: change extent.*
P19 Adaptive stopping	Permit iterative refinement with measurable benefit and cost; require choosing when further refinement is no longer justified. *Control: marginal benefit.*
P20 Scale-sensitive strategy	Provide workloads where the appropriate task-level strategy changes, such as manual inspection versus indexed or programmatic processing. *Control: workload scale.*

B.3 Skill Usage: Patterns

Table 5: Difficulty patterns for skill usage.

Pattern	Construction and hardening control
S01 Capability routing	Provide several documented skill entry points; require selecting the one implementing an already chosen operation. *Control: entry-point similarity.*
S02 Applicability checks	Make a skill procedure valid only under stated input assumptions that must be checked before applying it. *Control: precondition count.*
S03 Procedural exceptions	Provide a default procedure with explicit exceptions; some inputs require the exception-specific local handling. *Control: exception density.*
S04 Example adaptation	Supply a worked skill example whose incidental constants differ from the current case; require transferring the procedure correctly. *Control: example mismatch.*
S05 Version-specific behavior	Fix a known installed version whose documented operation differs from another version or common example. *Control: version divergence.*
S06 Parameter binding	Require exact binding of arguments, flags, or option values for a selected operation whose meaning is already known. *Control: parameter coupling.*
S07 Payload conformance	Require a structured request with nested fields, exact types, and meaningful distinctions between omitted and null values. *Control: schema depth.*
S08 Invocation context	Require the correct working directory, environment variables, or scoped process context for an otherwise valid invocation. *Control: context conditions.*
S09 Interaction protocols	Require a documented request-response sequence, such as a prompt handshake or paginated interface, within one operation. *Control: protocol states.*

Continued on next page

Table 5 (continued)

Pattern	Construction and hardening control
S10 Completion protocols	Make an accepted or started operation differ from a completed one; expose a documented readiness or completion signal. *Control: lifecycle stages.*
S11 Native type fidelity	Require native dates, formulas, identifiers, or typed values; visually similar strings do not preserve their semantics. *Control: type distinctions.*
S12 Syntax boundaries	Include nested syntax, escaping, or embedded delimiters that require structural parsing rather than naive substitution. *Control: nesting depth.*
S13 Schema mismatches	Provide valid source and target schemas with different fields, nesting, or grouping; map between them while preserving relationships, values, and units. *Control: structural mismatch.*
S14 Unit conversion	Provide explicitly known units or coordinate frames; require correct numerical conversion into the target representation. *Control: conversion depth.*
S15 Precision control	Specify tolerances and rounding points so premature rounding or unstable computation produces an incorrect local result. *Control: precision demand.*
S16 Rule realization	Provide an explicit domain rule that must be implemented faithfully, including exact thresholds and operator semantics. *Control: rule complexity.*
S17 Recursive generality	Require a selected procedure to handle arbitrary valid nested compositions rather than only the supplied examples. *Control: composition depth.*
S18 Boundary handling	Include valid empty, singleton, extreme, or degenerate inputs whose behavior is defined by the procedure contract. *Control: boundary variety.*
S19 Tie semantics	Require a selected ranking or selection procedure to resolve equal-valued candidates using a documented secondary rule. *Control: tie density.*
S20 Stochastic reproducibility	Require a randomized procedure to honor specified seeds, sampling semantics, or repeatability conditions. *Control: random-state scope.*

B.4 Long-Horizon Consistency: Patterns

Table 6: Difficulty patterns for long-horizon consistency.

Pattern	Construction and hardening control
L01 Identity continuity	Rename or relocate an object during a valid workflow while later steps must continue tracking the same logical object. *Control: identity transitions.*
L02 Revision coherence	Produce several valid revisions; downstream steps must consistently consume the designated current revision. *Control: revision distance.*
L03 Derivation lineage	Require intermediate and final artifacts to retain traceable links to the exact source records used to generate them. *Control: lineage depth.*
L04 Assumption continuity	Choose a convention or mapping early in the workflow and require later steps to preserve that choice. *Control: retention span.*
L05 Configuration continuity	Run valid steps across sessions or tools whose defaults differ; maintain the intended shared configuration throughout. *Control: context switches.*
L06 Reference integrity	Update linked objects while preserving valid cross-references, keys, paths, or foreign-key relationships. *Control: reference fan-out.*
L07 Cross-format agreement	Maintain the same facts in several already-created formats after updates, such as a workbook and a report. *Control: representation count.*
L08 Implementation agreement	Change implemented behavior and keep its interface documentation, examples, and declared contract synchronized. *Control: coupled artifacts.*

Continued on next page

Table 6 (continued)

Pattern	Construction and hardening control
L09 Aggregate agreement	Revise detailed records and keep all derived totals, summaries, or indexes consistent with the current details. *Control: aggregation levels.*
L10 Claim-evidence agreement	Maintain consistency between final narrative claims and the actual evidence or results available at completion. *Control: claim dependencies.*
L11 Non-target preservation	Perform a scoped edit while leaving semantically unrelated content unchanged across the workflow. *Control: protected content.*
L12 Structural preservation	Change target values while retaining required formatting, nonrelational metadata, or layout across saves and reloads. *Control: protected structures.*
L13 Monotonic obligations	Satisfy requirements incrementally, but later valid edits can invalidate earlier achievements unless they are rechecked. *Control: interacting obligations.*
L14 Policy continuity	Maintain explicit privacy, permission, or scope restrictions across intermediate artifacts and final delivery. *Control: propagation paths.*
L15 Accounting conservation	Maintain conserved quantities or cumulative accounting across otherwise valid updates, including debits, credits, and allocations. *Control: update count.*
L16 Exactly-once coverage	Process a fixed known worklist across multiple successful steps without omissions or duplicate effects. *Control: worklist length.*
L17 Deferred obligations	Interrupt a subgoal with other valid work, then return to unresolved obligations rather than forgetting them. *Control: deferral span.*
L18 Checkpoint fidelity	Persist an intentional checkpoint that faithfully records all state needed for a planned later continuation. *Control: state dimensionality.*
L19 Interleaving consistency	Allow valid operations to interleave; maintain invariants through ordering, version checks, or isolated updates. *Control: interleaving count.*
L20 Resource ownership	Track ownership and lifetime of temporary files, processes, locks, or handles across a normally completing workflow. *Control: ownership transfers.*

B.5 Error Recovery: Patterns

Table 7: Difficulty patterns for error recovery.

Pattern	Construction and hardening control
R01 Failure recognition	Return a success-like status despite a violated postcondition; expose independent evidence enabling detection of the deviation. *Control: signal discrepancy.*
R02 Fault localization	Surface an error downstream from its cause; provide logs or checks that identify the faulty component or step. *Control: propagation distance.*
R03 Cause discrimination	Make several causes produce the same symptom within a localized component; provide probes distinguishing the actual cause. *Control: competing causes.*
R04 Persistence diagnosis	Provide observable evidence distinguishing a transient interruption from a persistent defect requiring an altered action. *Control: diagnostic delay.*
R05 Diagnostic preservation	Make inspection or repair alter ephemeral recovery evidence; require preserving a faithful copy before intervening. *Control: evidence lifetime.*
R06 Input repair	Supply recoverably malformed input with sufficient repair evidence; require restoring validity without inventing missing semantics. *Control: defect extent.*
R07 Environment repair	Expose a correctable dependency, permission, or configuration fault and a permitted local route to restore execution. *Control: fault interactions.*

Continued on next page

Table 7 (continued)

Pattern	Construction and hardening control
R08 Data restoration	Corrupt mutable task state while retaining a valid backup or redundant evidence from which it can be reconstructed. *Control: recovery distance.*
R09 Transactional rollback	Interrupt an atomic operation after partial mutation; require restoring its pre-operation state before proceeding. *Control: mutation footprint.*
R10 Compensating actions	Create partial effects that cannot be rolled back directly; provide allowed counteractions that restore the required business invariant. *Control: compensation chain.*
R11 Idempotent retries	Lose an acknowledgement after an operation may have committed; require checking its effect before safely retrying. *Control: commit ambiguity.*
R12 Partial failures	Fail only part of a batch and record per-item outcomes; resume failed work without repeating successful effects. *Control: failure dispersion.*
R13 Checkpoint recovery	Interrupt execution unexpectedly; resume from the latest valid checkpoint and reconcile work performed after that checkpoint. *Control: lost-work interval.*
R14 Fallback substitution	Disable a selected implementation after failure while leaving another available implementation that satisfies the same contract. *Control: fallback mismatch.*
R15 Conflict repair	Expose a detected write or version conflict; require reconciling competing changes before retrying the blocked operation. *Control: conflict scope.*
R16 Repair validation	Require repeating the original failing check after repair; an apparently successful repair command is insufficient evidence. *Control: failure conditions.*
R17 Regression validation	Make a repair capable of breaking previously working behavior; require checking the broader preserved contract after repair. *Control: regression surface.*
R18 Degraded-result qualification	Permit a documented degraded result after failure; require identifying its limitations and meeting the explicit fallback acceptance criteria. *Control: fallback conditions.*
R19 Cascading repair	Make repairing one defect reveal another previously masked defect, requiring successive diagnosis until the specified postcondition holds. *Control: masking depth.*
R20 Safe termination	Provide an explicit stop condition when further recovery is unsafe; require bounded termination, preserved state, and an accurate status report. *Control: stop conditions.*

C Environment Diversity

The synthesis process yields a final collection of 2,963 executable tasks. We analyze environment diversity through source-skill resources, execution substrates, skill-domain coverage, and initial workspaces.

C.1 Skill Resources

We first examine the resources contained in the 2,963 source-skill packages. Across the 2,963 task packages, 2,613 contain auxiliary skill resources beyond `SKILL.md`, comprising 19,649 files after excluding catalog metadata. As shown in Figure 6, these files cover diverse resource types, including source code, text, structured data, configuration files, and images. They also span 13 programming or markup languages, with Python and Shell being the most common, and appear in different package locations such as scripts, references, tests, examples, and templates.

Beyond static resources, many skills also expose executable interfaces. Static analysis identifies references to 101 distinct external programs, including commonly used CLI tools such as `git` and `curl`. Together, these statistics show that the source skills provide heterogeneous resources, file structures, and executable interfaces for environment construction.

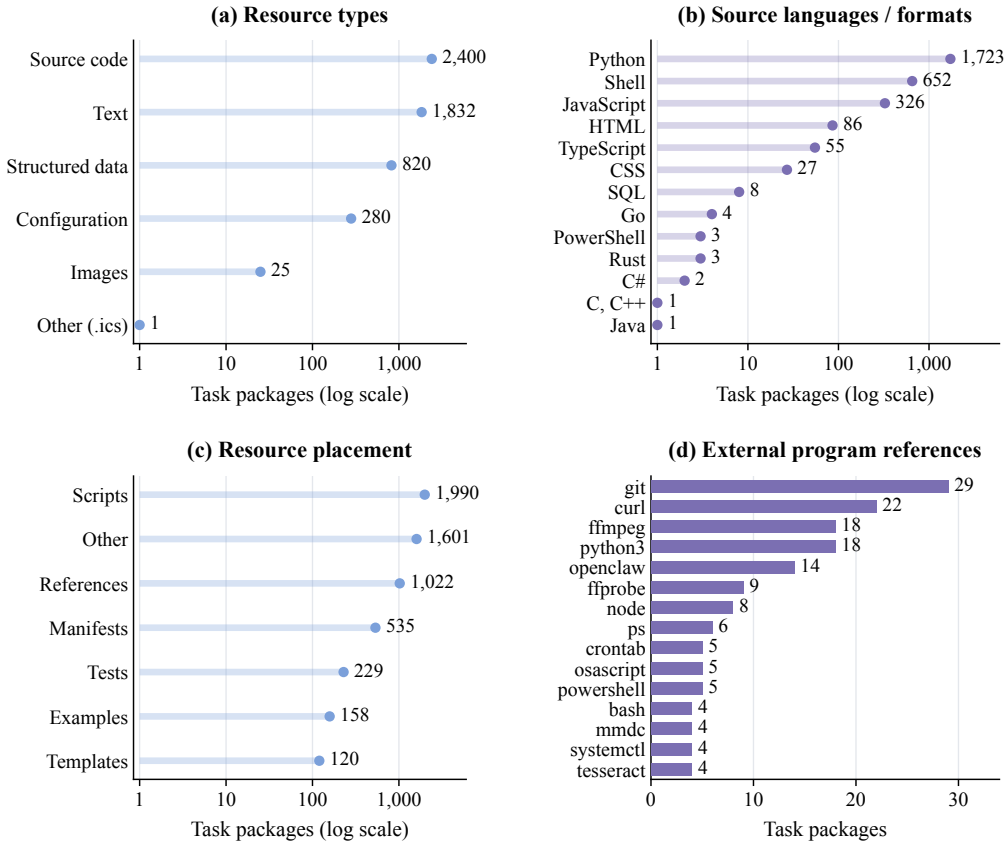


Figure 6: **Skill-resource diversity across 2,963 task-associated source-skill packages.** (a) Resource types after removing catalog auxiliary files. (b) All 13 source-language or format labels. (c) Resource placement under fixed path and filename rules. (d) The 15 most frequently referenced external programs in the Python subprocess analysis, ranked by the number of packages containing a literal program-name reference; ties are ordered alphabetically. Panels (a)–(c) use logarithmic count axes; (d) uses a linear axis.

C.2 Execution Substrate

We next examine the software dependencies and command-line interfaces associated with the 2,963 tasks. Across setup scripts, source-skill packages, and initial workspaces, we identify 4,242 distinct dependency packages spanning eight ecosystems and 823 distinct CLI names. As shown in Figure 7, npm contributes the largest number of package identities, followed by pip and Cargo, while both package and CLI usage exhibit long-tailed distributions across tasks. These statistics characterize the diversity of the software and executable interfaces available to the synthesized environments.

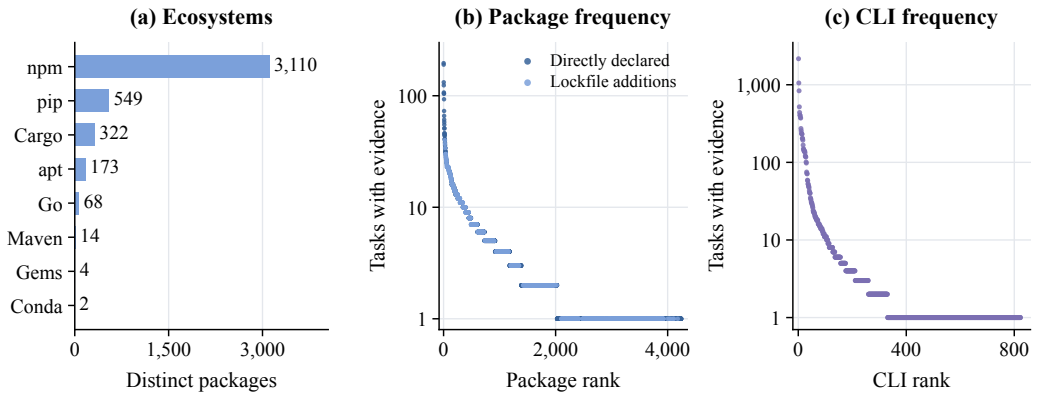


Figure 7: **Execution-substrate diversity.** (a) Number of distinct dependency packages across eight package ecosystems. (b) Task occurrence frequencies of the 4,242 identified packages, ranked by frequency. (c) Task occurrence frequencies of the 823 identified CLI names, ranked by frequency.

C.3 Skill Domains and Workspaces

Figure 8 summarizes source-skill domain coverage and its relationship to initial workspace assets.

Skill-domain coverage. Each task inherits one primary domain label from its source skill under a model-generated taxonomy. The collection spans 25 domains. The largest domain is software engineering and code quality, containing 320 tasks (10.8%). Other prominent domains include agent memory and self-improvement, financial markets, office documents, scientific computing, media processing, and everyday applications.

Workspace diversity and domain–workspace combinations. Initial workspaces contain 46,295 files, covering 113 recognized extensions across 12 asset families and 812 distinct nonempty extension sets. These families include source code, structured data, text and markup, office documents, images, audio/video, and scientific assets. Of the 2,963 tasks, 2,707 (91.4%) contain at least two recognized extensions, while 254 contain one and two contain none. File types are identified by their final extensions, excluding configured skill mounts, caches, and dependency directories.

The joint distribution shows domain-specific combinations: office documents and PDFs occur in 52.1% of office-domain tasks, audio/video assets in 40.9% of media tasks, and geospatial/scientific assets in 24.0% of scientific-computing tasks. Thus, domain coverage is accompanied by varied workspace materials within domains.

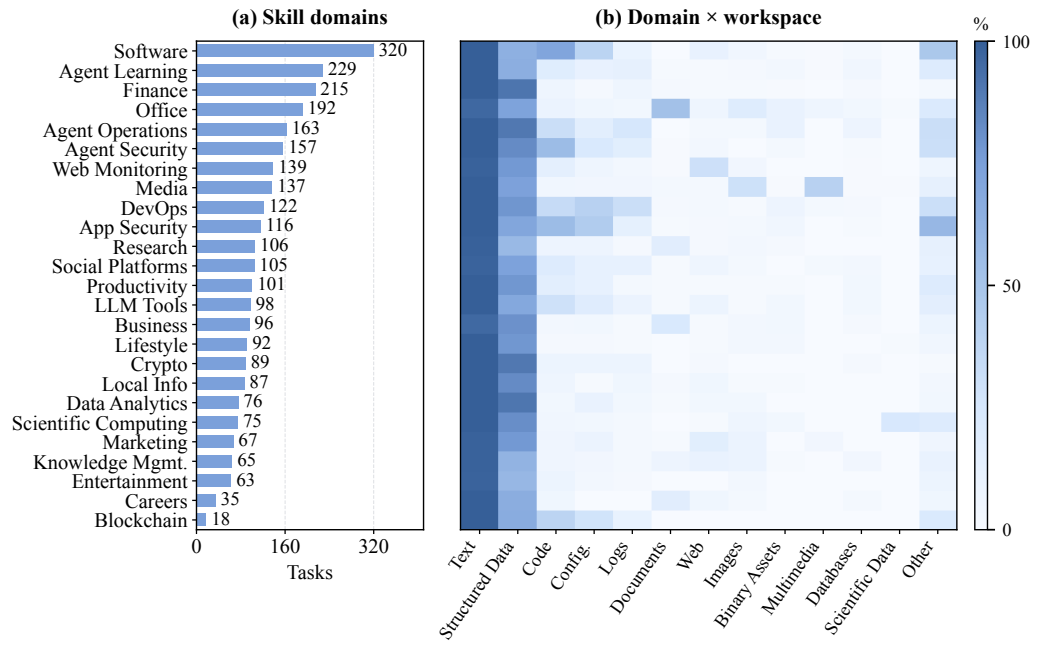


Figure 8: **Diversity of 2,963 synthesized tasks.** (a) Task counts across 25 source-skill domains, shown with abbreviated labels. (b) Percentage of tasks in each domain containing each workspace asset family; rows align with (a).

D Per-Benchmark Results Across Iterations

Figure 9 provides the individual benchmark scores underlying the average in Figure 5(d). The seven-benchmark unweighted mean is 40.58, 42.51, and 42.87 at iterations 0, 1, and 2, respectively.

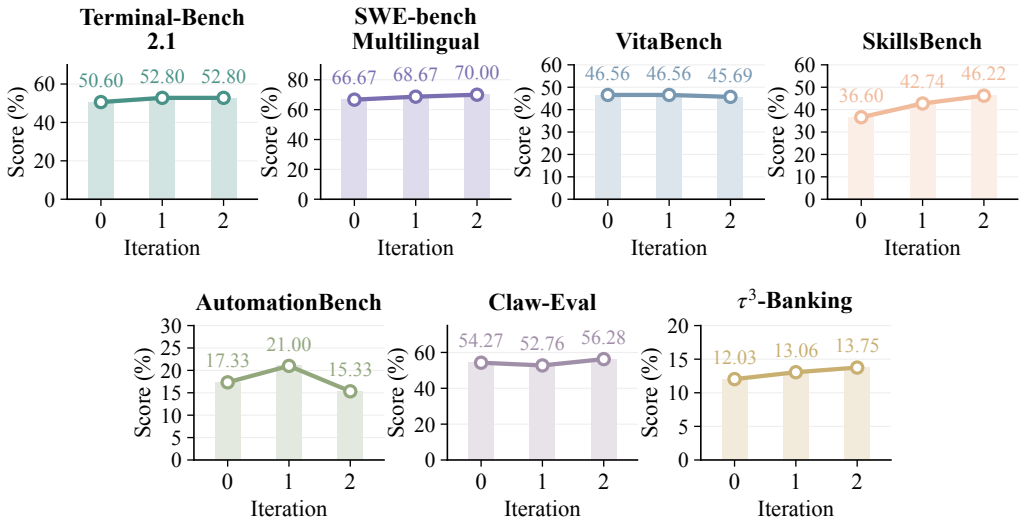


Figure 9: **Per-benchmark scores across hardening iterations.** Bars show scores and lines connect successive iterations. Labels report the scores to two decimal places.

E Docling Case Study: From a Skill to a Reconciliation Environment

This case shows how a document-processing skill becomes a quarterly vendor-reconciliation environment, and how execution evidence guides two rounds of task hardening. We present the complete user request, finance brief, and setup script, followed by the blueprint's requirements, challenges, information boundaries, reference facts, and evaluation criteria. The task materials define what the solver receives; the blueprint and evaluation criteria describe the author-side contract.

Task setting: complete user request

I need your help closing out a vendor engagement for our Q3 2024 books. I'm the finance analyst at Meridian Field Institute; our finance manager has sent a close-out request for the consultancy engagement **MSA-2024-HV07** with Halden & Verweij Consultancy BV.
The engagement file is in this workspace:

- `correspondence/2024-10-03_finance_closeout_request.md` — the close-out brief. **This is the authoritative specification**: it defines the scope, the checks to perform, and the exact structure of the three deliverables.
- `correspondence/2024-07-02_vendor_rate_increase_notice.md` — a rate notice from the vendor; assess it against the contract before relying on it.
- `contract/MSA-2024-HV07.docx` — the consultancy agreement (fees, caps, travel policy, VAT, corrections).
- `vendor_rate_card_annexB.html` — the contractual Annex B rate card (HTML export).
- `invoices/INV-2024-0731.pdf` and `invoices/INV-2024-0930.pdf` — the Q3 2024 invoices.
- `receipts/` — scanned travel receipts for the quarter.
- `ledger/2024_q3_payments.xlsx` — this quarter's outgoing payments.
- `archive/` — material from a closed 2023 engagement, attached for reference only and out of scope for the 2024 reconciliation.

Please reconcile the vendor's Q3 2024 billing against the contract terms: check every billed line against the agreement and Annex B, check travel claims against the receipts, verify invoice arithmetic and VAT, and cross-check settlement against the payments ledger. The source documents are in different formats (DOCX, PDF, HTML, scanned images, XLSX) — extract them reliably and watch that you do not lose table structure or receipt values in conversion.
Produce exactly the three deliverables specified in the brief, at these paths:

1. `reconciliation/line_items.csv`
2. `reconciliation/summary.json`
3. `reconciliation/findings.md`

All amounts are EUR. Where the brief specifies a schema or format, follow it exactly — the files feed our audit tooling. If your checks surface corrections, quantify them per the contract and recommend the resulting recovery action.

Execution environment: the skill, tools, and initial workspace

The Docling skill supplies document-conversion and OCR guidance. The task starts with **10 files in six formats**; the three requested output files do not yet exist.

Initial material	Files	Role in the task
Contract and Annex B	DOCX + HTML	Rates, quarterly day cap, travel rules, VAT.
July/September invoices	2 PDF	Seven billed rows to extract and reconcile.
Taxi and rail receipts	2 PNG	Scanned evidence of actual travel costs.
Payment ledger	1 XLSX	Establishes that both valid invoices were paid.
Finance brief and rate notice	2 MD	Output schemas; a rate change effective in 2025.
Archived invoice	1 PDF	Similar-looking material outside the 2024 engagement.

E.1 The Complete Finance Brief

Workspace document: finance close-out request

Close-out request: Halden & Verweij (MSA-2024-HV07) - Q3 2024
From: Marieke van Dijk, Finance Manager **To:** Finance analyst (vendor close-out) **Date:** 2024-10-03
Hi,
The Q3 2024 books are closing and internal audit wants the Halden & Verweij engagement reconciled before we release the file. Both Q3 invoices have already been paid in full (see `ledger/2024_q3_payments.xlsx`), so if the reconciliation surfaces corrections we will need a quantified credit-note request.

Scope

- Engagement **MSA-2024-HV07**, calendar quarter **Q3 2024** (1 July - 30 September 2024) only.
- In-scope invoices: `invoices/INV-2024-0731.pdf` and `invoices/INV-2024-0930.pdf`.
- Contract terms: `contract/MSA-2024-HV07.docx`; the contractual Annex B rate card is also exported at `vendor_rate_card_annexB.html`.
- Travel receipts for the quarter are scanned in `receipts/`.
- `archive/INV-2023-1215.pdf` was attached by ops for layout reference only - it belongs to a **closed 2023 engagement and is out of scope**. Do not include it in 2024 figures.
- The vendor's rate notice (`correspondence/2024-07-02_vendor_rate_increase_notice.md`) is background only; check it against the contract before relying on it.

Checks

1. Every billed line against the contract terms (rates, day cap, travel policy, VAT).
2. Travel claims against the scanned receipts.
3. Invoice arithmetic (line amounts, subtotals, VAT, totals).
4. Settlement against the payments ledger.

Deliverables - write exactly these three files

1. `reconciliation/line_items.csv`
One row per billed line, exact header:

```
invoice_id,invoice_date,description,category,quantity,unit,unit_price_eur,
    amount_eur_excl_vat,source_file
```

- `category` is one of `senior_consulting`, `data_analysis`, `mileage`, `taxi`, `rail`.
- `unit` is one of `day`, `km`, `trip`.
- `invoice_date` is ISO `yyyy-mm-dd`.
- All amounts are EUR excl. 21% VAT, 2 decimal places, no thousands separators or currency symbols.
- Lines are recorded **as billed** on the invoices (do not yet net out corrections); `source_file` is the workspace-relative path of the source document.

2. `reconciliation/summary.json`
Exact structure:

```
{
  "vendor": "Halden & Verweij Consultancy BV",
  "engagement": "MSA-2024-HV07",
  "quarter": "2024-Q3",
  "invoices": [
    {"invoice_id": "...", "invoice_date": "...", "subtotal_excl_vat": 0.00, "vat":
     0.00, "total_incl_vat": 0.00}
  ],
  "totals": {
    "billed_subtotal_excl_vat": 0.00, "billed_vat": 0.00, "billed_total_incl_vat":
     0.00,
```

Workspace document: finance close-out request (continued)

```
    "paid_total_incl_vat": 0.00,
    "corrected_subtotal_excl_vat": 0.00, "corrected_vat": 0.00, "
     corrected_total_incl_vat": 0.00,
    "credit_due_incl_vat": 0.00
  },
  "days": {"billed_days": 0, "cap_days": 24},
  "corrections": [
    {"invoice_id": "...", "reason": "rate_overbilling", "amount_excl_vat": 0.00}
  ]
}
```

- Money fields are numbers rounded to 2 decimals (round half up).
- `paid_total_incl_vat` comes from the ledger (this vendor only).
- `corrections` uses adjustment codes `rate_overbilling`, `travel_overclaim`, or `day_cap_excess`; `amount_excl_vat` is the positive reduction per code and invoice. Empty list if no corrections.
- `corrected_*` are the amounts that *should* have been billed after applying all corrections. `credit_due_incl_vat` is what we recover (0.00 if none).

3. `reconciliation/findings.md`
Short audit memo to me: engagement summary, each discrepancy with evidence (source document and figures), the net correction, and the recommended action (credit note amount incl. VAT, if any). Keep it under a page if you can.
Thanks - questions before the payment-run sign-off, please.
Marieke

E.2 The Complete Environment Setup

Execution substrate: setup.sh

```
#!/usr/bin/env bash
# Environment provisioning for the docling close-out reconciliation task.
# Idempotent, non-interactive. Installs the docling CLI (skill capability) with
# CPU-only torch and warms the model cache.
set -u

PIP="pip install --no-input"

# 1) CPU-only torch + torchvision wheels (avoids multi-GB CUDA wheels).
$PIP "torch==2.13.0+cpu" "torchvision==0.28.0+cpu" \
  --index-url https://download.pytorch.org/whl/cpu

# 2) docling CLI (pinned; pulls docling-slim / docling-ibm-models / transformers).
$PIP "docling==2.123.1"

# 3) Guard: docling's dependency resolver may have overwritten torchvision with
#    the PyPI (CUDA) build whose ops fail on CPU-only boxes. Restore CPU builds
#    if the import check fails or the local +cpu tag is missing.
python3 - <<'EOF' || REINSTALL=1
import torch, torchvision
assert "+cpu" in torch.__version__ + torchvision.__version__, "non-cpu build"
EOF
if [ "${REINSTALL:-0}" = "1" ]; then
  $PIP --force-reinstall --no-deps "torch==2.13.0+cpu" "torchvision==0.28.0+cpu" \
    --index-url https://download.pytorch.org/whl/cpu
fi

# 4) Warm the HuggingFace/docling model cache with one PDF and one PNG run so the
#    solver's first conversions are fast and do not depend on network reliability.
```

Execution substrate: setup.sh (continued)

```
WARM="$(mktemp -d)"
python3 - "$WARM" <<'EOF'
import sys
warm = sys.argv[1]
from reportlab.pdfgen import canvas
c = canvas.Canvas(f"{warm}/warm.pdf")
c.drawString(72, 720, "warmup invoice total 1.00")
c.save()
from PIL import Image, ImageDraw
img = Image.new("RGB", (400, 120), "white")
ImageDraw.Draw(img).text((10, 10), "WARMUP RECEIPT TOTAL EUR 1.00", fill="black")
img.save(f"{warm}/warm.png")
EOF
docling "$WARM/warm.pdf" --output "$WARM/out" >/dev/null 2>&1 || true
docling "$WARM/warm.png" --from image --output "$WARM/out" >/dev/null 2>&1 || true
rm -rf "$WARM"

# 5) Self-check (diagnostic lines).
docling --version | head -2
pip freeze 2>/dev/null | grep -Ei '^(docling|torch|torchvision)==' || true
```

E.3 How the Blueprint Connects Requirements and Challenges

The requirements specify the outputs to check. The challenges describe intended obstacles and the behavior that would address them. The six blueprint challenges instantiate combinations of the reusable difficulty patterns in Appendix B. Table 8 summarizes this mapping.

Table 8: Mapping from blueprint challenges in the Docling case study to the instantiated difficulty patterns in Appendix B.

Blueprint challenge	Instantiated difficulty patterns
H1	S01 Capability routing; S06 Parameter binding; S08 Invocation context
H2	U02 Distributed evidence
H3	U01 Target discrimination; U06 Authority resolution; U07 Temporal validity
H4	S15 Precision control; S16 Rule realization
H5	L07 Cross-format agreement; L09 Aggregate agreement; L10 Claim-evidence agreement
H6	S13 Schema mismatches

Blueprint: requirements (R1–R5)

R1. reconciliation/line_items.csv contains exactly the 7 billed lines from the two Q3 2024 invoices with the brief's exact column schema and correct quantities, unit prices, and amounts (excl. VAT); archive invoice excluded
Visible basis. brief schema + invoices
Evaluation. deterministic
Type. objective
Preserve during implementation. Yes
R2. reconciliation/summary.json conforms to the brief's schema with correct as-billed invoice/Q3 totals (per-invoice 4989.00/1047.69/6036.69 and 5165.40/1084.73/6250.13; billed 10154.40/2132.42/12286.82; paid 12286.82) and day counts (26 billed, 24 cap)
Visible basis. brief schema + invoices + ledger
Evaluation. deterministic
Type. objective
Preserve during implementation. Yes
R3. summary.json corrections itemize all three discrepancies with correct ex-VAT amounts (240.00 rate over-billing, 13.00 taxi over-claim, 570.00 day-cap excess) attributed to INV-2024-0930
Visible basis. contract rules + invoices + receipts + brief

Blueprint: requirements (R1–R5) (continued)

Evaluation. deterministic
Type. objective
Preserve during implementation. Yes
R4. summary.json corrected totals and recovery: corrected_subtotal_excl_vat 9331.40, corrected_vat 1959.59, corrected_total_incl_vat 11290.99, credit_due_incl_vat 995.83
Visible basis. brief schema + derived per contract rules
Evaluation. deterministic
Type. objective
Preserve during implementation. Yes
R5. reconciliation/findings.md is an audit-ready memo: identifies all three discrepancies with evidence (invoice lines, contract clauses, receipt amounts), explains why the 2025 rate notice does not apply to Q3 2024, and recommends a credit note of EUR 995.83 incl. VAT
Visible basis. brief + all workspace evidence
Evaluation. llm judge
Type. subjective
Preserve during implementation. Yes

Blueprint challenge H1: multi-format tool use and extraction fidelity

Plausible failure. Weak solver hand-transcribes PDFs/images, loses the bordered table structure, or never OCRs the receipts, so line amounts or the taxi receipt value are wrong/missing
Strong solution behavior. Uses docling per format into a controlled output dir (PDF/DOCX/HTML/image OCR), cross-checks extracted tables against line arithmetic

Blueprint challenge H2: distributed rules across documents

Plausible failure. Applies only the most salient rule (e.g. VAT) and never notices the day cap or taxi cap buried in contract clauses
Strong solution behavior. Extracts the full rule set (rates, 24-day aggregate cap with lowest-rate-first deduction, taxi at-cost capped at 40, VAT 21%) and evaluates every line against every applicable rule

Blueprint challenge H3: target-object identification / scoping

Plausible failure. Includes archive/INV-2023-1215.pdf in the 2024 totals, or treats the vendor's 2025 rate notice as applicable and 'validates' the EUR 450 rate
Strong solution behavior. Scopes to MSA-2024-HV07 Q3 2024 per the brief; reads the notice's effective date (2025-01-01) and the contract's Annex B governing clause, so the 450 rate is flagged as over-billing, not accepted

Blueprint challenge H4: rule realization and numerical precision

Plausible failure. Computes VAT with float drift or inconsistent rounding, mis-orders cap deductions (deducts senior days), or double-counts the taxi cap vs receipt, producing a wrong credit figure
Strong solution behavior. Works in cents/Decimal with round-half-up 2dp; payable taxi = min(52.00 claim, 39.00 receipt, 40.00 cap) = 39.00; 2 excess analyst days x 285.00; credit = 823.00 + 172.83 VAT = 995.83

Blueprint challenge H5: cross-artifact consistency

Plausible failure. Findings.md and summary.json disagree (memo says one credit amount, JSON another) because corrections were computed ad-hoc per artifact
Strong solution behavior. A single consistent correction set flows into CSV totals, JSON corrections/totals, and the memo

Blueprint challenge H6: schema fidelity under an explicit contract

Plausible failure. Invents column names / JSON keys instead of following the brief, breaking downstream audit tooling
Strong solution behavior. Maps the extracted invoice and contract information into the exact CSV and JSON target schemas specified in the finance brief, preserving field meanings and relationships.

E.4 Information Boundaries and Reference Facts

Blueprint: information boundaries

Explicit in the task instruction.

- The overall objective (Q3 2024 close-out reconciliation for vendor Halden & Verweij under MSA-2024-HV07)
- The three required deliverables and their exact root-relative paths (reconciliation/line_items.csv, reconciliation/summary.json, reconciliation/findings.md)
- That the finance brief (correspondence/2024-10-03_finance_closeout_request.md) is the authoritative specification of deliverable schemas, scope, and the checks to perform
- That amounts are EUR, VAT is handled per the contract, and archive/ material is out of scope

Present in the workspace.

- All commercial rules (rates, caps, VAT 21%, cap-deduction ordering, effective-date rule) in contract and rate card, NOT restated in the prompt
- The exact deliverable schemas in the finance brief
- The receipt amounts (39.00 taxi, 88.40 rail) only inside the scanned PNGs
- The 2025 rate-increase notice as a plausibly-confusing but non-applicable document

Discovered through execution.

- That docling is the right tool and which –from/–output flags each format needs
- That the archive invoice shares structure with the in-scope invoices but must be excluded
- That the vendor over-billing stems from early application of the announced 2025 rate

Inferred from the materials.

- payable taxi amount = min(claimed, receipted, cap) from the at-cost + cap rule
- cap excess (2 days) falls on analyst lines per the lowest-rate-first rule

No hidden assumptions.

- No fact requires information outside the workspace; VAT rate, caps, rates, and rounding expectations are all stated in workspace documents
- The 2025 rate notice explicitly states its effective date so its non-applicability is resolvable from context, not guessing

Blueprint: fixed evidence and external access

Search expected. No
Temporal mode. fixed

- All amounts, dates, VAT rate, caps, and rates are fixed workspace facts as of the scenario date 2024-10-03
- Credit note EUR 995.83 incl. VAT is derivable from workspace inputs alone

Evaluation strategy. Fully offline evaluation against fixed reference facts, using deterministic checks for objective criteria and an LLM judge for subjective criterion.

Blueprint: complete reference facts


```
{
  "inv1_subtotal_excl_vat": 4989.0,
```

Blueprint: complete reference facts (continued)


```
  "inv1_vat": 1047.69,
  "inv1_total_incl_vat": 6036.69,
  "inv2_subtotal_excl_vat": 5165.4,
  "inv2_vat": 1084.73,
  "inv2_total_incl_vat": 6250.13,
  "billed_subtotal_excl_vat": 10154.4,
  "billed_vat": 2132.42,
  "billed_total_incl_vat": 12286.82,
  "paid_total_incl_vat": 12286.82,
  "billed_days": 26,
  "cap_days": 24,
  "excess_days": 2,
  "correction_rate_overbilling_excl_vat": 240.0,
  "correction_taxi_overclaim_excl_vat": 13.0,
  "correction_day_cap_excl_vat": 570.0,
  "corrections_total_excl_vat": 823.0,
  "corrections_vat": 172.83,
  "corrected_subtotal_excl_vat": 9331.4,
  "corrected_vat": 1959.59,
  "corrected_total_incl_vat": 11290.99,
  "credit_due_incl_vat": 995.83,
  "line_item_count": 7
}
```


Blueprint: derivations of the reference outcomes

INV-2024-0731 totals 4989.00/1047.69/6036.69. 9x420.00 + 4x285.00 + 300x0.23 = 3780+1140+69 = 4989.00; VAT 4989.00x0.21 = 1047.69 exact; total 6036.69
Validation. calculation + docling extraction cross-check
INV-2024-0930 billed totals 5165.40/1084.73/6250.13. 8x450.00 + 5x285.00 + 52.00 + 88.40 = 5165.40; VAT 1084.734 -> 1084.73 (2dp half-up as printed); total 6250.13
Validation. calculation + docling extraction cross-check
rate over-billing 240.00. Annex B senior rate 420.00 governs Q3 2024; 450.00 applies only from 2025-01-01; the invoice bills 8 senior days at 450.00; 8x(450-420)=240.00
Validation. calculation
taxi over-claim 13.00. travel reimbursed at cost with receipt, taxi capped at 40.00 per trip; receipt = 39.00; payable = min(52.00, 39.00, 40.00) = 39.00; over-claim 13.00
Validation. OCR extraction + calculation
day-cap correction 570.00, excess 2 days. Q3 billed days 9+4+8+5 = 26 vs cap 24; excess 2 days deducted lowest-rate-first from the final invoice of the quarter -> 2 analyst days x 285.00 = 570.00
Validation. calculation
credit_due_incl_vat 995.83 and corrected totals. corrections 823.00 excl VAT; VAT on corrections 172.83; corrected 9331.40/1959.59/11290.99; both invoices fully paid (12286.82); credit = 12286.82 - 11290.99 = 995.83
Validation. calculation + ledger cross-check

Blueprint: invariants and critical requirements

- line_items.csv has exactly 7 data rows; every amount equals quantity x unit_price to the cent
- summary.json internal consistency: invoice totals sum to billed totals; corrected + corrections = billed (excl VAT); credit_due = paid - corrected_total
- No line references archive/INV-2023-1215.pdf and the 2025 rate is not treated as applicable to Q3 2024

Requirements carrying the central challenges: R3 and R4.
Critical failure requirements: R3 and R4.

E.5 Complete Evaluation Criteria

Rubric: line items csv

Criterion. reconciliation/line_items.csv follows the brief's exact 9-column schema and contains exactly the 7 billed Q3 2024 lines with correct invoice dates, quantities, units, unit prices and ex-VAT amounts, excluding the out-of-scope 2023 archive invoice.
Weight. 0.25
Evaluation method. deterministic

Rubric: summary billed totals

Criterion. reconciliation/summary.json conforms to the brief's schema: both invoices with their as-billed subtotal/VAT/total (4,989.00/1,047.69/6,036.69 and 5,165.40/1,084.73/6,250.13), quarterly billed totals (10,154.40 / 2,132.42 / 12,286.82), ledger-paid total 12,286.82, and day counts 26 billed vs 24 cap.
Weight. 0.15
Evaluation method. deterministic

Rubric: corrections breakdown

Criterion. summary.json corrections itemize all three contract violations with correct ex-VAT amounts, attributed to INV-2024-0930: rate_overbilling 240.00 (8 senior days at 450 vs the Annex B rate 420 that governs 2024), travel_overclaim 13.00 (52.00 taxi claim vs 39.00 receipt under the 40.00 cap), day_cap_excess 570.00 (26 vs 24 billed days, 2 analyst days at 285.00 deducted lowest-rate-first).
Weight. 0.20
Evaluation method. deterministic

Rubric: corrected totals credit

Criterion. summary.json reports the corrected position: corrected_subtotal_excl_vat 9,331.40, corrected_vat 1,959.59, corrected_total_incl_vat 11,290.99, and credit_due_incl_vat 995.83 (both invoices were paid in full, so this is the recoverable amount).
Weight. 0.15
Evaluation method. deterministic

Rubric: findings memo

Criterion. reconciliation/findings.md is an evidence-grounded audit memo to the finance manager covering all discrepancies and the recommended recovery.
Weight. 0.25
Evaluation method. llm judge
Judge instruction. Score from 0 to 1 the memo reconciliation/findings.md against these criteria. Ground truth (do not reveal beyond judging): (1) INV-2024-0930 bills 8 senior consultant days at EUR 450.00/day although Annex B of MSA-2024-HV07 (governing all 2024 work) sets EUR 420.00/day; the vendor's rate notice raises rates only from 2025-01-01, so the over-billing is 8 x 30 = EUR 240.00 excl. VAT. (2) The 12 Sep taxi is claimed at EUR 52.00 but the receipt (receipts/2024-09-12_taxi_citycab.png) shows EUR 39.00 and the contract caps taxi at EUR 40.00/trip and reimburses at receipted cost, so EUR 13.00 is over-claimed; the EUR 88.40 rail claim matches its receipt. (3) Q3 billed days total 26 (13 July + 13 September) vs the contract's aggregate cap of 24 per quarter; the 2 excess days are deducted from the lowest day-rate lines (data analyst, EUR 285.00) on the final invoice of the quarter, i.e. EUR 570.00 excl. VAT. Total corrections EUR 823.00 excl. VAT; with 21% VAT the credit note due is EUR 995.83 incl. VAT; corrected totals 9,331.40 / 1,959.59 / 11,290.99; both invoices were already paid in full (12,286.82), so the amount must be recovered by credit note. Award roughly: 0.35 for identifying and correctly quantifying all three discrepancies (0.12-ish each; a discrepancy counts only if its cause AND amount are right), 0.25 for the correct net correction and credit-note recommendation (EUR 995.83 incl. VAT) explicitly tied to the fact that both invoices were fully paid, 0.15 for correctly explaining why the 2025 rate notice does not legitimize the 450 rate (effective 2025-01-01; Annex B governs 2024), 0.15 for evidence grounding (memo cites the specific source documents/figures: invoice lines, contract clauses or Annex B, receipt values, ledger payments) without fabricated content, 0.10 for clarity and audit-readiness (addressed to the finance manager, concise, internally consistent figures). Deduct for any figure that contradicts the ground truth

Rubric: findings memo (continued)

(e.g. a different credit amount), for inventing findings not supported by the documents, or if the memo conflicts with reconciliation/summary.json when that file is also available. An otherwise strong memo in different wording or structure must still score highly.
Files available to the judge. reconciliation/findings.md, reconciliation/summary.json, reconciliation/line_items.csv

Rubric: passing condition

Original evaluator pass threshold. 0.7
A passing solution converts the mixed-format close-out package into the exact-schema line-item CSV and summary JSON, finds and correctly quantifies all three contract violations (240.00 + 13.00 + 570.00 excl. VAT), derives the corrected totals and the EUR 995.83 credit note, and explains them in an evidence-grounded audit memo.

E.6 How Execution Evidence Informed Two Hardening Revisions

Consistency validation. Before using rollout evidence for hardening, the task conditions and rubric results were checked against the blueprint. The workspace contained the required evidence, the evaluator criteria were consistent with the reference facts, and no evaluator or environment inconsistency affecting the diagnosis was identified. The validated execution evidence was then used for diagnosis.

First hardening: extend a workflow the solver already completed

Observed execution. The original rollout received full credit on all five rubric items. The diagnosis agent read that result and inspected execution records showing document conversion, deliverable writes, and numerical self-checks. The complete self-check command was:

```
cd /app && python3 -c "
import json, csv
# validate json
d=json.load(open('reconciliation/summary.json'))
t=d['totals']
print('invoices sum subtotal:', round(sum(i['subtotal_excl_vat'] for i in d['invoices
    ']),2))
print('invoices sum vat:', round(sum(i['vat'] for i in d['invoices']),2))
print('invoices sum total:', round(sum(i['total_incl_vat'] for i in d['invoices']),2))
print('billed totals consistent:', t['billed_subtotal_excl_vat']==10154.40 and t['
    billed_vat']==2132.42 and t['billed_total_incl_vat']==12286.82)
# corrected
sub=4989.00+4342.40
vat=1047.69+round(4342.40*0.21,2)
tot=6036.69+round(4342.40+4342.40*0.21,2)
print('corrected sub/vat/tot',sub,vat,tot)
print('credit',12286.82-round(tot,2))
print('corrections sum',sum(c['amount_excl_vat'] for c in d['corrections']))
# csv rows
r=list(csv.DictReader(open('reconciliation/line_items.csv')))
print('csv rows',len(r))
print('csv subtotal',round(sum(float(x['amount_eur_excl_vat']) for x in r),2))
# verify each invoice line total
for x in r:
    assert abs(float(x['amount_eur_excl_vat'])-round(float(x['quantity'])*float(x['
    unit_price_eur']),2))<0.01
print('line arithmetics OK')
# days
print('days',sum(int(x['quantity']) for x in r if x['category'] in ('senior_consulting
    ','data_analysis')))
"
```

First hardening: extend a workflow the solver already completed (continued)

Design response. The diagnosis agent judged the current task insufficiently challenging. New requirements make previously straightforward checks consequential:

- Put the principal-consultant rate only in the signed DOCX amendment: the HTML rate card alone is insufficient.
- Rebill one rail journey across months: individual invoice checks must become a cross-invoice comparison.
- Add another engagement of the same vendor, with its own ledger payment: scope must be checked in both invoices and payments.
- Print EUR 1,435 for $5 \times 285 =$ EUR $1,425$: preserve the original billed line and separately record the EUR 10 correction.
- Partially pay an invoice: distinguish billed amounts from cash already paid.

This is hardening informed by a *successful* rollout. The new duplicate-claim and partial-payment challenges were proposed extensions.

Second hardening: make document extraction robustness consequential

Observed execution. A later archived rollout shows four aborted OCR conversions caused by an incorrectly split output path. The command passed `/tmp/` and `receipts_out` as separate arguments after `--output`, causing each conversion to return `Aborted.` The solver nevertheless recovered by directly reading the PNG receipts with the image-reading tool and obtained the required receipt amounts. This recovery was valid, but it also exposed a brittle document-processing path: the intended OCR/conversion procedure failed, and the rollout succeeded only because the relevant evidence happened to remain directly readable through an alternative interface.
Design response. To make this weakness consequential, the hardened task places required evidence in a text-less scanned PDF whose content must be extracted reliably as part of the document-processing workflow. The solver must correctly invoke the conversion/OCR procedure, preserve the extracted structure, and use the resulting evidence in the reconciliation. This revision directly targets the brittle invocation behavior observed in the rollout. Additional revisions increase the amount and interaction of extracted evidence, including date-dependent rates, an unreceipted claim, and a day-cap deduction spanning two rate tiers.

E.7 How the Task Changes Across Hardening

What changed across the task versions

	Original	Hardening 1	Hardening 2
Workspace files	10	14	21
Input formats	6	6	6
Valid invoices	2	3	4
Billed rows	7	14	20
Designed discrepancies	3	7	12
Partially paid invoices	0	1	2

The comparison shows that task complexity increases across hardening stages through more records, interacting discrepancies, and additional settlement dependencies.