AGENT S4: EVOLUTIONARY COMPUTER USE

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer-use agents are limited by the interfaces humans give them. An agent may type a spreadsheet cell by cell when a single line of code would do, or overwrite a file with a script while the application still holds unsaved edits. Both the interfaces and the knowledge of when to use each are built by hand, which cannot keep pace with new applications, workflows, and models. We introduce *Agent S4*, an autonomous computer-use scientist that studies computer-use agents and improves their interfaces using population-based evolution. A meta-agent examines task agents' trajectories on synthetic training tasks, without success labels, and writes new interfaces and usage knowledge into a repository. Population-based evolution then selects which repositories to develop further by their performance on held-out tasks the meta-agent never sees, favoring changes that generalize beyond the seen tasks. The evolved repositories contain interfaces no one specified, such as a single request that batches dozens of GUI actions, and observations that condense PDFs and archives into compact text. On OSWorld V2, Agent S4 with Qwen3.8-27B reaches 21.3% binary task success, on par with the 20.6% reported for Claude Opus 4.8 with batched actions, at an estimated task-solving cost of USD 1.10 per task versus the reported USD 72.40, approximately 1.5% of the cost. These results suggest that interfaces are an axis of improvement complementary to the model, one that can keep evolving as models and harnesses change.

1 INTRODUCTION

> *We want AI agents that can discover like we can, not which contain what we have discovered.*
>
> — Richard S. Sutton, *The Bitter Lesson*

Computer-use agents (CUAs) carry out tasks by observing and acting on digital environments. Existing systems support a range of interaction modes. GUI agents use screenshots, mouse clicks, and keyboard input. Browser agents interact with rendered pages or the DOM, and software-engineering and terminal agents edit files, execute code, and invoke shell commands (Xie et al., 2024; Zhou et al., 2024; Yang et al., 2024). These modes expose different views of computer state and different operations for changing it. A task may involve several modes as an agent moves between applications, files, and services. In today's agentic frameworks, the interfaces that determine what the agent observes and which operations it can request are typically designed by humans and fixed before task execution.

This practice creates three problems for computer-use agents (CUAs). First, it ties progress to manual labor. Computer use is open-ended, and each new family of tasks or model release exposes new failure modes. A human researcher must read trajectories, diagnose failures, and use them to write new primitives, observation formats, or memory mechanisms, and then release the result as a new agent framework with a new preset interface. This loop does not scale with task diversity. Second, a preset interface often mismatches the task, and current methods compensate with test-time compute instead of a better interface. For example, an agent may complete a shell task by clicking to open a terminal window, typing commands one character at a time, and reading screenshots of a partly occluded terminal for feedback, when a direct shell interface would be faster and more reliable. Additional sampling can raise benchmark scores under such mismatches, but through wasteful and brittle strategies. Third, methods that distill test-time experience into reusable skills or routines inherit the same limit (Agashe et al., 2025a; Wang et al., 2025b). Each skill is a composition of the preset

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

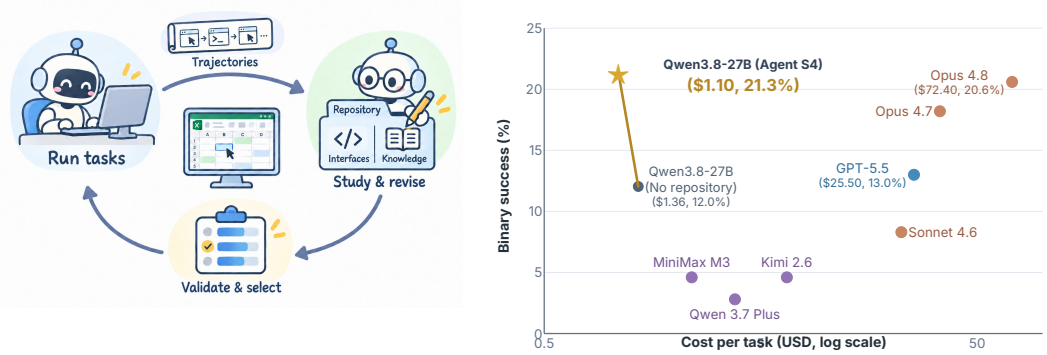


Figure 1: **Agent S4 evolves interfaces and knowledge that make computer use more capable and less costly.** With the evolved repository, Qwen3.8-27B solves 21.3% of OSWorld V2 tasks at about $1.10 per task, compared with 12.0% at $1.36 without a repository. This reaches task success comparable to the 20.6% reported for Claude Opus 4.8 at about 1.5% of its reported cost.

observations and actions. Distillation can reorder what the interface allows, but it cannot extend it. These problems arise because the interface is treated as a design decision that humans make before deployment, and not as an object that can be studied and improved from the agent's own experience.

We argue for an automated process that evolves interfaces from task experience. We introduce **Agent S4**, an autonomous CUA scientist that studies and improves computer-use interfaces through population-based evolution. Its output is a repository of executable interfaces and knowledge about when and how to use them. The task agent's model and harness remain fixed, with the harness governing model invocation, execution, permissions, and task budgets.

The task agent uses candidate repositories to attempt synthetic tasks, producing trajectories without ground-truth success labels. A meta-agent studies these trajectories, forms hypotheses about failures and inefficiencies, and constructs descendants through mutation of one repository or crossover between two. Descendants can add, revise, or remove interfaces and knowledge, and the population retains alternative designs for further development and recombination.

To limit specialization to the tasks that motivated these changes, Agent S4 separates the evidence used for construction from that used for selection. The meta-agent receives training trajectories, while validation instructions, trajectories, diagnostics, and scores remain hidden from it. Held-out validation performance guides parent selection, and a separate test set evaluates the final selected repository.

We evaluate Agent S4 on OSWorld V2 (Yuan et al., 2026) and ScienceBoard (Sun et al., 2025). In the full-set evaluation on 108 OSWorld V2 tasks, Qwen3.8-27B achieves 21.3% binary success with the evolved repository, compared with 19.4% using the seed and 12.0% without a repository. Estimated task-solving cost decreases from approximately $1.36 without a repository to $1.10 per task with the evolved repository. In a separate ScienceBoard experiment, evolution improves held-out mean success from 15.6% to 28.0% for Qwen3.5-397B-A17B. These results support learning reusable interfaces and knowledge from experience as a way to improve computer use while keeping the task agent's model and harness fixed.

2 Problem Formulation

Interface design determines the work required to accomplish a goal. An agent updating a spreadsheet through screenshots must identify visible cells, enter values, and inspect the display to check the result. A structured interface can expose the relevant rows and formulas, accept a bulk edit, and return the changed values. Both interfaces can achieve the same subgoal (Coutaz et al., 1995), but they require different decisions and provide different feedback. Their cost and reliability can also differ. Experience can therefore reveal both which existing interface suits a subgoal and where a new interface would help.

To make interfaces and knowledge available across tasks, we represent them in a persistent repository R. It contains executable interfaces and supporting knowledge such as capability descriptions, reusable workflows, configuration, and notes about when and how to use them.

Let $\mathcal{D}$ be the target distribution of tasks and $\mathcal{P}_A$ a distribution over agents. Let $J(A, R, x)$ denote the expected task score when agent A uses repository R to attempt task x. We seek a repository that maximizes this score across agents and tasks,

$$R^\star \in \arg\max_R \mathbb{E}_{A\sim\mathcal{P}_A,\ x\sim\mathcal{D}}[J(A, R, x)]\,. \tag{1}$$

This objective concerns performance beyond the experience used to construct the repository. We next define the agent, interfaces, and interaction process that determine J.

Computer and task. A computer C has a state $s \in \mathcal{S}$ comprising its files, running applications, displays, services, and reachable network endpoints. A task $x = (g, s_0, V_x, B_x)$ consists of an instruction g, an initial state s_0, an evaluator V_x that scores the outcome, and a budget B_x.

Interface. An *interface* $I = \langle c_I, \mathcal{A}_I, \mathcal{O}_I \rangle$ connects an agent to one or more parts c_I of the computer through a language of requests $\mathcal{A}_I$ and a language of observations $\mathcal{O}_I$. Executing a request can change the state and returns an observation, $(s', o) \sim T_I(\cdot \mid s, a)$. A GUI interface, for example, accepts clicks and keystrokes and returns screenshots. A shell interface accepts commands and returns textual output.

Model, harness, and agent. A *model* M is a language model that generates responses and requests conditioned on its input context. A *harness* H manages model invocation, request execution, task budgets, permissions P_H, and preparation of the model's context, including any context compaction. An *agent* $A = (M, H)$ is the model operating through this harness. The agent initially has access to a base set of interfaces $\mathcal{I}$. The harness executes requests through these interfaces and places the resulting observations in the model's context. For example, the harness can execute a spreadsheet helper under a time limit and permission boundary. The helper defines the requests for reading or editing cell ranges and the values returned to the agent. The harness governs execution, while the interface specifies the available operations and feedback.

Computer use. *Computer use* is the process through which an agent observes and acts on a computer through available interfaces to accomplish a task. The interaction history h_t records the interfaces, requests, and observations from earlier steps. At step t, the agent chooses an interface and a request based on the instruction and this history, with context preparation by H left implicit,

$$(I_t, a_t) \sim M(\cdot \mid g, h_t, \mathcal{I}), \qquad (s_{t+1}, o_t) \sim T_{I_t}(\cdot \mid s_t, a_t). \tag{2}$$

The available interfaces shape which operations the agent can request and what evidence it receives about their effects.

Formally, $R = (\mathcal{I}(R), K(R))$, where $\mathcal{I}(R)$ contains its executable interfaces and $K(R)$ its supporting knowledge. The repository supplies interfaces $\mathcal{I}_R = \mathcal{I} \cup \mathcal{I}(R)$ and knowledge that informs the agent's choices,

$$(I_t, a_t) \sim M(\cdot \mid g, h_t, \mathcal{I}_R, K(R)). \tag{3}$$

Each attempt is subject to the task budget B_x and harness permissions P_H. The score $J(A, R, x)$ averages the evaluator's score over these interactions.

Interface discovery. An agent that can write and execute code can also change how it interacts with the computer. For example, it could write a program that exports a material schedule through Blender's Python API, extracts selected rows from a Calc workbook, or returns supplier prices from a sequence of browser operations. Each program implements an interface with its own requests and observations (Wang et al., 2024; Liang et al., 2023). The agent can decide which operations to combine, what information to return, and how to check the result. Code makes these design choices accessible to the agent within the capabilities permitted by its harness. Task experience can expose a need for such a change when, for example, the agent repeatedly reconstructs a calculation or cannot verify an operation from the feedback available.

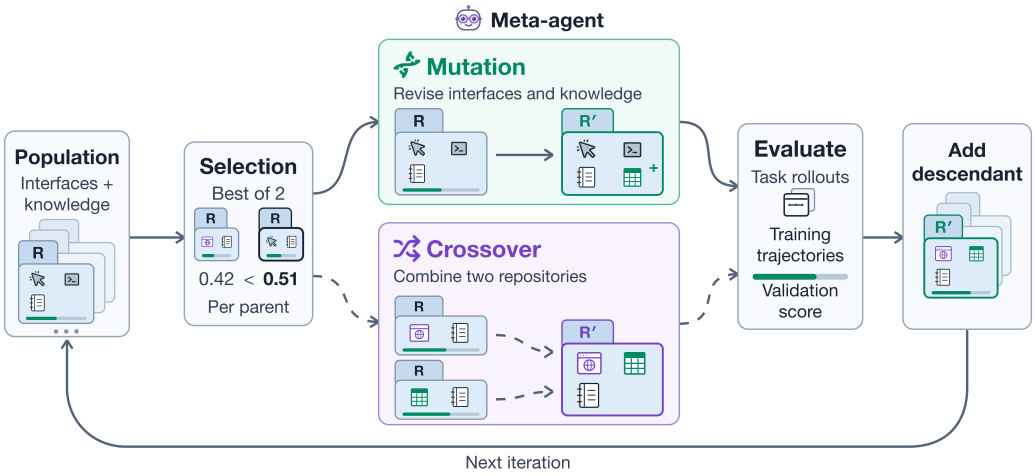


Figure 2: **Agent S4 turns task experience into an evolving repository of interfaces and knowledge.** Starting from parents selected by held-out validation, the meta-agent studies their task trajectories to identify limitations, then revises one repository through mutation or combines ideas from two through crossover. Task agents then use the descendant to produce trajectories on synthetic tasks and a separate score on held-out validation tasks. Each descendant joins the population alongside its parents, allowing earlier designs to be revisited and recombined over successive iterations.

3 Agent S4

Agent S4 learns a repository of interfaces and knowledge through population-based evolution, keeping the task agent's model M and harness H fixed. Each iteration selects parents from a population, constructs a descendant through mutation or crossover, evaluates it, and adds it to the population while retaining its parents (Figure 2). Training experience guides the repository changes, and held-out validation guides subsequent selection. This optimizes Equation 1 with an agent distribution concentrated on A during training. Algorithm 1 gives the procedure.

3.1 Population and selection

The population contains repository variants, each with its own interfaces and knowledge. We initialize $\mathcal{R}_0 = \{R_0\}$ from a seed repository and evaluate it to obtain training trajectories and a validation score. These records are retained with each candidate as the population grows.

At each step, an operator schedule chooses mutation or crossover. Mutation requires one parent, and crossover requires two distinct parents. For each parent, binary tournament selection draws two eligible variants uniformly and chooses the one with the higher validation score. A sole eligible candidate is selected directly. For crossover, the first selected parent is excluded from the second selection. The selected repositories and their saved training trajectories provide the starting point for the next descendant.

3.2 Mutation and crossover

The *meta-agent* $A_{\text{meta}} = (M_{\text{meta}}, H_{\text{meta}})$ constructs descendants using a fixed model and harness. Its computer C_{meta} holds the selected parent repositories, their training trajectories, and a lineage notebook in its filesystem. It studies the recorded behavior and observations to diagnose apparent failures and inefficiencies and form hypotheses about useful changes. For example, repeated cell-by-cell edits can motivate a bulk spreadsheet interface, while poor use of an existing interface can motivate revised usage knowledge. It implements these hypotheses through file, shell, and code interfaces using one of two operators.

Mutation. Starting from one parent R_p, the meta-agent adds, revises, or removes interfaces and knowledge. Changes can affect operations, returned observations, and reusable workflows, including capabilities inherited from the seed.

Algorithm 1 Agent S4 evolves repositories by selecting parents, constructing descendants from task experience, and returning evaluated descendants to the population.

Require: Seed repository R_0, task agent A, meta-agent A_{meta}, training tasks $\mathcal{T}_{\text{train}}$, validation tasks $\mathcal{T}_{\text{val}}$, operator schedule σ, evolution budget N

$\mathcal{R}_0 \leftarrow \{R_0\}$
Run A with R_0 on $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{val}}$
for $k = 1, \ldots, N$ **do**
  $z \leftarrow \sigma(k)$ ▷ MUTATION or CROSSOVER
  Select one or two parents from $\mathcal{R}_{k-1}$ by validation tournament
  $C_{\text{meta}} \leftarrow$ the parents, their training trajectories, and lineage notebook
  $R' \leftarrow$ descendant constructed by A_{meta} on C_{meta} using z
  Freeze R' and run A with it on $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{val}}$
  $\mathcal{R}_k \leftarrow \mathcal{R}_{k-1} \cup \{R'\}$
  Record parent links and apply population control, if configured
end for
Select $\hat{R}$ from $\mathcal{R}_N$ by validation score
return $\hat{R}$, $\mathcal{R}_N$, and the lineage record

Crossover. Given two distinct parents R_a and R_b, the meta-agent constructs one coherent descendant by choosing between overlapping interfaces, porting complementary capabilities, or combining their ideas into new code and knowledge. The descendant need not contain the union of its parents.

Both operators produce a single descendant repository R'. The fixed base access $\mathcal{I}$ remains unchanged as the repository's interfaces $\mathcal{I}(R)$ and knowledge $K(R)$ evolve.

3.3 EVALUATION AND POPULATION UPDATE

After construction, we freeze R' and run the fixed task agent A on synthetic training tasks $\mathcal{T}_{\text{train}}$ and disjoint validation tasks $\mathcal{T}_{\text{val}}$. The repository is available in the task agent's filesystem, and a fixed prompt identifies its location and explains how to discover its capabilities. The agent can consult its documentation and source code and invoke the interfaces it selects through the harness's existing code-execution tool (Appendix A.2). Each rollout $\tau \sim \text{Rollout}(A, R', C, x)$ starts from task x's initial state within its budget and records the instruction, interfaces, requests, and observations. The repository remains fixed during execution. Temporary programs written by the agent do not persist across tasks.

Training experience and validation score. Training trajectories have no ground-truth success labels. We save them as evidence for future repository changes and reuse them whenever the descendant becomes a parent. Validation rollouts provide a separate selection score for each candidate R,

$$F_{\text{val}}(R) = \frac{1}{|\mathcal{T}_{\text{val}}|} \sum_{x \in \mathcal{T}_{\text{val}}} V_x(\text{Rollout}(A, R, C, x)) \,. \tag{4}$$

Asymmetric evaluation. Only training trajectories are exposed to the meta-agent. Validation tasks, trajectories, diagnostics, and scores remain inaccessible to it. Validation therefore influences construction through parent selection, without supplying direct feedback for repository edits.

Adding the descendant. We add R' to the population alongside its parents regardless of its validation score. This retains alternative designs for further development and recombination. Parent links form a directed acyclic graph, with one parent per mutation and two per crossover. Optional pruning removes parent eligibility while preserving lineage records. The updated population supplies candidates for the next iteration. After N steps, we select a member of $\mathcal{R}_N$ by validation score and freeze it for evaluation on $\mathcal{T}_{\text{test}}$, disjoint from training and validation.

4 EXPERIMENTAL SETUP

Benchmarks. Our primary benchmark is OSWorld V2 (Yuan et al., 2026), with ScienceBoard used to evaluate generalization to scientific workflows. Detailed benchmark descriptions are provided in Appendix A. The 30 synthetic training tasks and their construction procedure are described in Appendix D.

Model and harness. For OSWorld V2, the task agent, meta-agent, and GUI grounding use Qwen3.8-27B. We use a modified Prime Agent harness that executes code in a persistent IPython kernel. The model uses Medium reasoning, a 131,072-token context window, and a 32,000-token output limit, with automatic context compaction enabled. The exact harness version, sampling parameters, and compaction settings are provided in Appendix A. Implementation constraints and builder prompts are given in Appendix C.

Baselines. We compare the task agent without a repository, with the seed repository, and with the evolved repository, alongside the Qwen CUA baseline.[1] Qwen3.8-27B results are from our own evaluations. Results for other models in Figure 1 and Table 1 are taken from Table 3 of the OSWorld 2.0 paper (Yuan et al., 2026), including its reported per-task costs. We use the batched-action results for Claude Opus 4.8, Claude Opus 4.7, and GPT-5.5, and the single-action results for Claude Sonnet 4.6, MiniMax M3, Kimi 2.6, and Qwen 3.7-Plus. These reference runs use release `v2026.06.24`, a 500-step budget, and the reasoning settings reported in that paper.

Repository evolution. For OSWorld V2, starting from a single seed repository, we maintain one population and construct 15 descendants, one per construction step. Evolution follows an alternating schedule of mutation and crossover, starting with mutation. Each mutation uses one parent, while each crossover combines two distinct parents, selected from the eligible population by binary tournament selection based on mean partial validation score. After constructing and evaluating the descendant at step 10, we remove the five lowest-scoring variants from the eligible parent pool before selecting parents for step 11. This elimination does not consume a construction step.

For each construction step, the meta-agent is limited to 300 agent steps and a wall-clock time of 2 hours. We select the repository with the highest binary validation success rate and freeze it for full evaluation.

Evaluation and cost. For OSWorld V2, the task agent has a budget of 300 steps per task. During evolution, each task rollout has a two-hour wall-clock limit. Full evaluation retains the step budget without a per-task wall-clock limit. We report binary success and partial score, together with task-solving tokens and cost. Costs are estimated from task-solving model-token usage using Parasail's input and output rates, assuming an 80% input-cache hit rate. These estimates exclude repository construction and runtime infrastructure. The rates and remaining runtime settings are provided in Appendix A.

5 RESULTS AND ANALYSIS

5.1 MAIN RESULTS

Table 1 reports results on the full OSWorld V2 benchmark. We compare the same Qwen3.8-27B task agent without a repository, with the seed repository, and with the evolved repository to assess the contributions of repository use and subsequent evolution. We also include the Qwen CUA baseline and frontier-model results from the OSWorld V2 leaderboard to place the open-source agent's performance in a broader context. Binary success and partial score capture complete task execution and intermediate progress, while mean token usage and model cost per task characterize the inference expenditure required to achieve this performance.

Providing the seed repository improves both task completion and partial score, and evolution further improves both metrics. In particular, binary success rises from 19.44% with the seed to 21.30% after

[1] We use the Qwen CUA implementation provided by OSWorld V2: `https://github.com/xlang-ai/OSWorld-V2/tree/main/mm_agents/qwen`.

Table 1: **Repository evolution improves Qwen3.8-27B beyond its starting interfaces.** On the full 108-task OSWorld V2 benchmark, the seed repository raises binary success from 12.04% to 19.44%, and evolution further increases it to 21.30% while also improving partial score. These gains retain the seed repository's low estimated task-solving cost of about $1.10 per task, below the cost of using no repository.

Method	Binary ↑	Partial ↑	Input tokens ↓	Output tokens ↓	Cost ($) ↓
Base Agent	12.04	37.23	10.8M	**186K**	∼1.359
+ Agent S4 Seed Repository	19.44	44.90	7.65M	197K	∼1.106
+ Agent S4 Evolved Repository	**21.30**	**46.90**	**7.61M**	197K	∼**1.102**
Qwen CUA Harness	5.56	20.69	7.45M	48.6K	∼0.763
Claude Opus 4.8 (batched actions)	20.60	54.80	/	/	∼72.4
GPT-5.5 (batched actions)	13.00	49.50	/	/	∼25.5
Qwen 3.7-Plus (single action)	2.80	21.50	/	/	∼3.8

evolution. This progression suggests that reusable interfaces and knowledge provide a useful starting point that can be improved through experience. The evolved agent also outperforms the Qwen CUA baseline on both metrics. Using the open-source Qwen3.8-27B model, Agent S4 achieves binary success comparable to Claude Opus 4.8 and higher than GPT-5.5 on the OSWorld V2 leaderboard. Learned repositories thus enable competitive task completion with an open-source model. The evolved agent achieves this performance at approximately $1.10 per task, about 1.5% of Claude Opus 4.8's reported cost and 4.3% of GPT-5.5's. The resulting gains therefore come with lower inference expenditure, making interface learning a promising way to improve the performance and cost of an open-source agent together.

Figure 3 tracks the best validation scores observed across 15 repository variants. Individual candidates fluctuate in performance, while the best observed binary success and mean partial score reach 26.67% and 47.86%, respectively. Figure 4 in Appendix B.1 shows the full parent lineage and the validation scores of the seed and each descendant. Appendix B.3 examines how a reusable CDP interface reduces browser setup and interaction overhead in a client photo-delivery task. Appendix B.2 complements this case with a comparison of tool-call counts across 108 tasks.

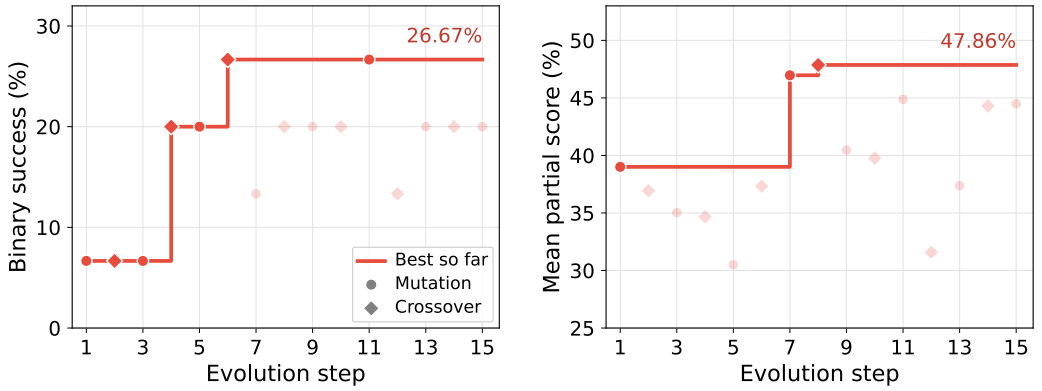


Figure 3: **Evolution discovers stronger repositories without improving at every step.** Across 15 descendants evaluated on 15 OSWorld V2 validation tasks, the best binary success reaches 26.67% and the best partial score reaches 47.86%. Different repositories achieve these peaks, and later revisions can perform worse, making earlier candidates valuable starting points for continued evolution.

5.2 Case Studies

We examine two interfaces constructed during exploratory evolution. These cases show how task experience motivated new operations and how the meta-agent tested and revised their implementations.

The synthetic tasks used to generate these trajectories did not have ground-truth success labels, and the repository variants changed multiple components. The cases therefore provide evidence about interface construction and repair rather than the isolated effect of either interface on task success. We also analyze execution efficiency and present an additional browser-interface case study in Appendices B.2 and B.3 to illustrate how evolved helpers can reduce the overhead of task execution.

Process-isolated LibreOffice automation. The seed repository could inspect the text and package structure of Office files, but it did not provide operations for editing documents or controlling LibreOffice. After observing failures in document tasks, evolution added an interface for launching a separate LibreOffice instance and using UNO, LibreOffice's programming interface for opening, editing, and exporting documents. It executed UNO operations in client subprocesses and added helpers for managing background jobs and inspecting or converting ODF documents. Isolating these calls prevented a failed Office operation from destroying the task agent's persistent Python kernel. The task agent later used the new LibreOffice and document helpers across both training and validation trajectories, showing that the interface was discoverable and reusable beyond the interaction that motivated it.

Later branches expanded this design with interfaces for loading, saving, and converting Office files and for inspecting files, archives, and rendered PDFs. The builder also discovered that an inherited connector was using the wrong object after establishing a UNO connection. It repaired the connector by retrieving LibreOffice's service manager, the object used to create and control document components. The sequence shows evolution producing a useful abstraction, observing how agents used it, and repairing a failure exposed by that use.

Batched GUI interaction and visual checks. This interface lets the task agent submit an ordered sequence of desktop actions in one call. A batch can combine mouse clicks and drags, text entry, keypresses, scrolling, window focus, short waits, and screenshots. The interface executes the sequence in order and returns how many actions completed. If one action fails, it stops at that point and identifies the failing action, allowing the agent to correct the problem and continue from a known state.

A companion operation compares screenshots taken before and after an action or batch. It reports whether the screen changed and the fraction of pixels that differ. Together, the two operations let the agent carry out a predictable multi-step procedure without spending a separate model turn on every click, while still checking whether the application responded. For example, the agent can focus a window, open a menu, choose an item, wait for a dialog, enter a filename, confirm the dialog, and compare the resulting screen through one structured interface call.

5.3 GENERALIZATION

Table 2: **Repository evolution also improves computer use in scientific applications.** With Qwen3.5-397B-A17B, evolution raises mean success on 94 held-out ScienceBoard tasks from 15.6% with the seed repository to 28.0%, exceeding Agent S3's 18.4%. It also improves consistency across repeated attempts. Each task has three 10-minute attempts, and Pass^3 measures the percentage of tasks solved on all three.

Repository / agent	Mean success (%)	Pass^3 (%)
Agent S3	18.4	10.6
Seed repository	15.6	8.5
Evolved repository	**28.0**	**18.1**

We also evaluate Agent S4 on ScienceBoard (Sun et al., 2025), which spans scientific applications including algebra, astronomy, and geographic information systems. We use 50 tasks directly from the benchmark to collect trajectories for repository evolution, 25 validation tasks for repository selection, and 94 held-out test tasks, with no overlap between these sets. The task agent uses Qwen3.5-397B-A17B in the open-source OpenCode harness, while Claude Opus 4.7 in Claude Code serves as the meta-agent. The task model and harness remain fixed as the repository of interfaces and knowledge evolves. We evaluate the seed repository, evolved repository, and Agent S3 (Gonzalez-Pumariega et al., 2025) with the same Qwen model over three attempts per test task, each with a 10-minute wall-clock budget.

The evolved repository raises Qwen3.5's mean success from 15.6% with the seed to 28.0%, exceeding Agent S3's 18.4% (Table 2). We use a 10 minute budget for all three settings. These results show the general benefits of evolving better interfaces.

6 Related Work

Computer-use agents and interaction design. Computer-use systems differ in how they expose observations and actions. Agent S and Agent S2 organize planning, perception, and action through engineered agent components (Agashe et al., 2025a;b), while UI-TARS learns policies for GUI interaction (Qin et al., 2025; Wang et al., 2025a). CoAct-1 combines GUI control with agents that write and execute Python or Bash (Song et al., 2025). Agent S3 and its behavior judge improve task completion by selecting among multiple rollouts (Gonzalez-Pumariega et al., 2025). These approaches motivate both richer interaction capabilities and better use of task-solving compute. Agent S4 studies how experience across tasks can improve the interfaces themselves. It retains the resulting operations and usage knowledge in a repository that can be developed beyond the task that motivated a change.

Skills, memory, and tool construction. Reflexion stores textual feedback for later attempts (Shinn et al., 2023), and Agent Workflow Memory induces reusable procedures from experience (Wang et al., 2025b). Reuse can also take executable form. LLMs as Tool Makers separates tool construction from subsequent tool use (Cai et al., 2024), and SkillWeaver discovers and practices web interactions to build APIs that other agents can reuse (Zheng et al., 2025). CUA-Skill instead provides an engineered library of parameterized computer-use skills and composition graphs (Chen et al., 2026). These methods establish that agents can benefit from persistent knowledge and executable tools. Our search operates over complete repository versions containing interfaces, observation routines, configuration, and usage knowledge. Mutation and crossover can change these components together, with selection based on their combined effect on task-agent performance.

Automated agent design and evolutionary search. ADAS searches for agent designs expressed in code (Hu et al., 2025), and the Darwin Gödel Machine modifies its own coding agent while maintaining an archive of alternatives (Zhang et al., 2025). Self-Harness studies execution traces, proposes harness changes, and validates them through regression tests (Zhang et al., 2026). AlphaEvolve and ShinkaEvolve combine model-generated code changes with evolutionary search and executable evaluation (Novikov et al., 2025; Lange et al., 2025), while GEPA uses trajectory reflection to evolve prompts and combine complementary improvements (Agrawal et al., 2025). Validation-based control of overfitting also predates language-model agents, including evolutionary curtailment in genetic programming (Fitzgerald & Ryan, 2011). Agent S4 evolves repositories of interfaces and knowledge through population-based search with asymmetric access to evaluation evidence. The meta-agent studies synthetic-task trajectories without ground-truth success labels and implements changes to application-specific operations, capability descriptions, and reusable workflows. The fixed harness retains control of model invocation, execution, permissions, and budgets. Keeping capabilities in the repository makes them available for revision, selection, recombination, and reuse with compatible agents and harnesses. Validation tasks, trajectories, diagnostics, and scores remain with the controller and are hidden from the meta-agent. Held-out performance determines which variants become parents, applying selection pressure beyond the tasks used to construct the software.

7 Conclusion

We presented Agent S4, a method for learning reusable interfaces and knowledge from computer-use experience. On 108 OSWorld V2 tasks, the learned interfaces and knowledge raise Qwen3.8-27B's success rate from 12.0% to 21.3% while reducing estimated task-solving cost from \$1.36 to \$1.10 per task. The success rate is comparable to Claude Opus 4.8's reported 20.6%, at about 1.5% of its reported cost of \$72.40 per task. On 94 held-out ScienceBoard tasks, evolution raises Qwen3.5-397B-A17B's mean success rate from 15.6% to 28.0%. These findings support treating interface design as a learning problem and a source of progress complementary to advances in the underlying models.

AI USE STATEMENT

We used generative AI tools to polish the writing of this manuscript, including improving grammar, clarity, and readability. The authors take full responsibility for the final content of the paper, including all claims, analyses, and results.

We also used AI to generate synthetic tasks as a part of this method the details of which are provided in Appendix D.

ETHICS STATEMENT

To the best of our knowledge, this work complies with the ICLR Code of Ethics. Our experiments use existing publicly available benchmarks and do not involve new data collection from human participants.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: an open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=lIVRgt4nLv.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. *CoRR*, abs/2504.00906, 2025b. doi: 10.48550/ARXIV.2504.00906. URL https://doi.org/10.48550/arXiv.2504.00906.

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. *CoRR*, abs/2507.19457, 2025. doi: 10.48550/ARXIV.2507.19457. URL https://doi.org/10.48550/arXiv.2507.19457.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=qV83K9d5WB.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Suzhen Zheng, Leon Xu, Hao Fan, Pashmina Cameron, Justin Wagle, and Kazuhito Koishida. Cua-skill: Develop skills for computer using agent. *CoRR*, abs/2601.21123, 2026. doi: 10.48550/ARXIV.2601.21123. URL https://doi.org/10.48550/arXiv.2601.21123.

Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M. Young. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In Knut Nordby, Per H. Helmersen, David J. Gilmore, and Svein A. Arnesen (eds.), *Human-Computer Interaction, INTERACT '95, IFIP TC13 Interantional Conference on Human-Computer Interaction, 27-29 June 1995, Lillehammer, Norway*, IFIP Conference Proceedings, pp. 115–120. Chapman & Hall, 1995.

Jeannie Fitzgerald and Conor Ryan. Validation sets for evolutionary curtailment with improved generalisation. In Geuk Lee, Daniel Howard, and Dominik Ślęzak (eds.), *Convergence and hybrid information technology*, pp. 282–289, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24082-9.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. The unreasonable effectiveness of scaling agents for computer use. *CoRR*, abs/2510.02250, 2025. doi: 10.48550/ARXIV.2510.02250. URL https://doi.org/10.48550/arXiv.2510.02250.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *CoRR*, abs/2509.19349, 2025. doi: 10.48550/ARXIV.2509.19349. URL https://doi.org/10.48550/arXiv.2509.19349.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 9493–9500. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *CoRR*, abs/2506.13131, 2025. doi: 10.48550/ARXIV.2506.13131. URL https://doi.org/10.48550/arXiv.2506.13131.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Silvio Savarese, Zeyuan Chen, Jieyu Zhao, Ran Xu, and Caiming Xiong. Coact-1: Computer-using agents with coding as actions. *CoRR*, abs/2508.03923, 2025. doi: 10.48550/ARXIV.2508.03923. URL https://doi.org/10.48550/arXiv.2508.03923.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, Lingpeng Kong, and Zhiyong Wu. Scienceboard: Evaluating multimodal autonomous agents in realistic scientific workflows. *CoRR*, abs/2505.19897, 2025. doi: 10.48550/ARXIV.2505.19897. URL https://doi.org/10.48550/arXiv.2505.19897.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents, 2024. URL https://arxiv.org/abs/2402.01030.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

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, 2024.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lu, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks, 2026. URL https://arxiv.org/abs/2606.29537.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *CoRR*, abs/2606.09498, 2026. doi: 10.48550/ARXIV.2606.09498. URL https://doi.org/10.48550/arXiv.2606.09498.

Jenny Zhang, Shengran Hu, Cong Lu, Robert T. Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *CoRR*, abs/2505.22954, 2025. doi: 10.48550/ARXIV.2505.22954. URL https://doi.org/10.48550/arXiv.2505.22954.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *CoRR*, abs/2504.07079, 2025. doi: 10.48550/ARXIV.2504.07079. URL https://doi.org/10.48550/arXiv.2504.07079.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Experimental Design

A.1 Benchmark Detail

OSWorld V2. OSWorld V2 (Yuan et al., 2026) evaluates computer-use agents on 108 long-horizon workflows spanning everyday and professional activities in real desktop and web applications. Tasks require agents to combine information across sources, track application state, and adapt to changes during execution. The benchmark reports both binary task completion and partial scores, measuring whether agents can complete an entire workflow and how much of its requirements they satisfy. We use it as our primary benchmark for evaluating repository evolution on complex computer-use tasks.

ScienceBoard. ScienceBoard (Sun et al., 2025) evaluates autonomous agents in realistic scientific workflows involving professional software. It contains 169 human-curated tasks across scientific domains such as biochemistry, astronomy, and geoinformatics. These tasks combine scientific reasoning with software operation through graphical and command-line interfaces, testing whether agents can translate domain-specific instructions into successful execution in scientific environments. We use ScienceBoard to evaluate whether repository evolution extends to specialized scientific applications.

A.2 Settings

Harness. For our OSWorld V2 repository-evolution experiments, we use a modified harness based on Prime Agent version `0.8.0-beta.546.1.a9b5d88`,[2] an IPython-backed coding agent. The harness prompts are reproduced in Appendix C. The model issues tool calls containing Python code, which the harness executes in a persistent IPython kernel and returns the execution outputs to the model for the next step. Variables, imports, and function definitions persist across calls, allowing the agent to reuse intermediate results and invoke repository interfaces throughout a task. Prime Agent's native threshold-based context compaction is enabled within the 131,072-token context window. The harness reserves 36,096 tokens for the 32,000-token output budget and a 4,096-token overhead margin. Compaction is triggered when the last observed token usage plus the estimated trailing messages exceeds 94,976 tokens. It summarizes older conversation history while retaining the most recent 20,000 tokens verbatim. Prime Agent also automatically rescales oversized images to fit within 2000×2000 pixels while preserving aspect ratio, with further compression or downscaling as needed to keep the base64-encoded payload below 4.5 MiB.

Repository access. For OSWorld V2, each candidate repository is installed at `/home/user/shadow_interface/`. A fixed task-agent prompt supplies this path, directs the agent to inspect the toolkit, and encourages its use as the primary approach. The agent invokes `python3 -m shadow_interface` through the harness's IPython tool. The `capabilities` subcommand lists interfaces with signatures and short descriptions; `doc <module>.<function>` displays full documentation, and `call <module>.<function> [args ...]` invokes an interface. The prompt also directs the agent to read the source code when the documentation is insufficient. These files expose the repository's interfaces and usage knowledge without changing the harness's tool set.

Model Service. We access Qwen3.8-27B through OpenRouter using the Parasail provider. We set temperature to 1, top-p to 0.9, and thinking effort to Medium, with a 131,072-token context window and a 32,000-token output limit. The same model is used for the task agent, GUI grounding, and the interface builder. We estimate task-solving costs using Parasail's rates of \$0.24 per million uncached input tokens, \$2.20 per million output tokens, and \$0.05 per million cached input tokens. The cost estimates assume an 80% input-cache hit rate. The Qwen3.8-27B cost estimates in Table 1 use these rates. These estimates cover task-solving model calls; repository-evolution and runtime infrastructure costs are excluded.

Cloud infrastructure. For OSWorld V2, each task episode runs in a freshly initialized virtual machine on AWS using the same benchmark release image, with a screen resolution of 1920×1080

[2] `https://github.com/PrimeIntellect-ai/prime-agent/releases`

pixels. The virtual machine is terminated after the episode. Model inference is accessed through the API described above.

Runtime Setting. All tests and experiments use a budget of 300 agent steps per task, with a timeout of 720 seconds for each API call. During evolution, each task episode has a wall-clock limit of two hours. Full evaluation has no per-task wall-clock limit, while retaining the API-call timeout and 300-step budget. Episodes affected by infrastructure failures during evolution may be rerun once.

B Experimental Analysis

B.1 Evolution Lineage

Figure 4 shows the repository lineage for the 15-step OSWorld V2 run. Starting from the seed, construction alternates between mutation of one parent and crossover of two distinct parents. Binary tournaments based on mean partial validation score select parents from the eligible population; after step 10, the five lowest-scoring variants lose parent eligibility. The resulting graph branches from earlier repositories and combines branches through crossover, allowing prior variants to contribute beyond the next construction step. Validation performance does not improve monotonically: step 8 has the highest partial score, whereas steps 6 and 11 share the highest binary success rate. The lineage thus records both alternative development paths and differences between the two evaluation metrics.

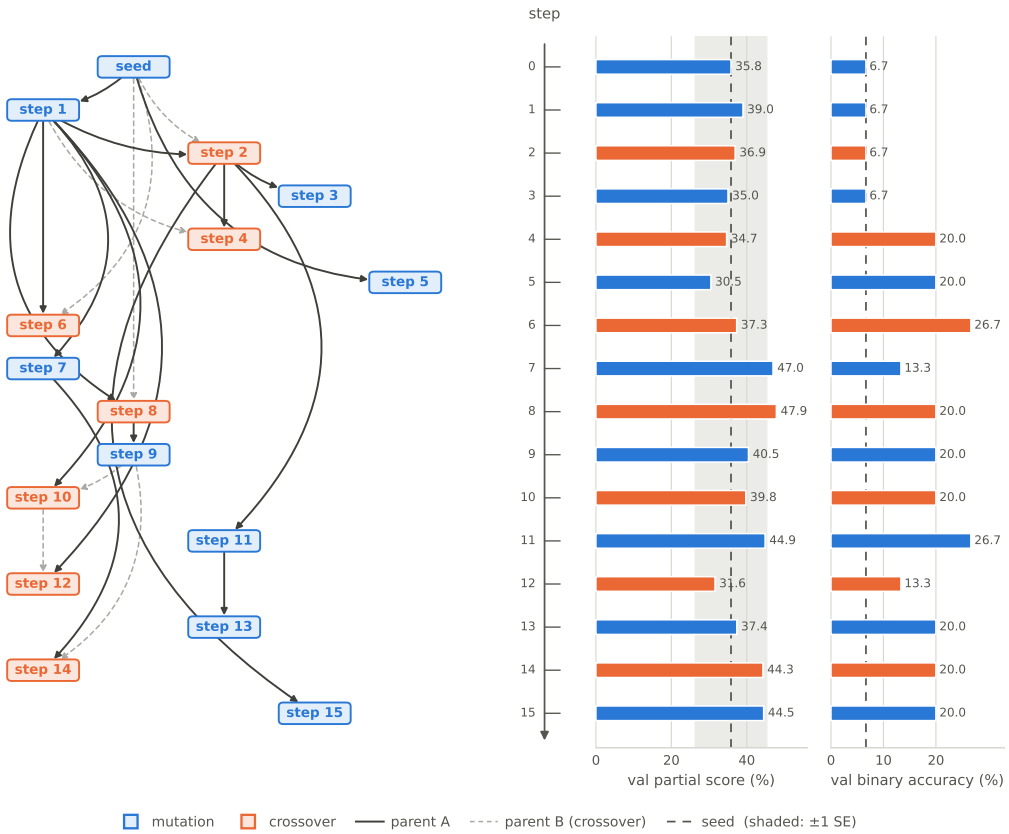


Figure 4: **Useful repositories can emerge by returning to an earlier branch.** Mutation and crossover build 15 descendants from a shared seed while preserving multiple designs for further development. Step 11 revisits the repository from step 2 and reaches 26.7% binary success on 15 validation tasks, tying the best in the run. The highest partial score occurs on a different branch at step 8, so the strongest repository depends on which aspect of task performance is measured.

B.2 Execution Efficiency

Table 3 compares the seed and evolved repository across 108 OSWorld V2 tasks. On each repository's full-credit tasks, the evolved agent uses 7.1% fewer tool calls on average while receiving full credit on more tasks. Together with the CDP case in Appendix B.3, this suggests a useful role for helper functions: reusable operations can reduce repeated setup and combine multiple low-level interactions within a single tool call. Such savings can free steps for task-specific reasoning and execution.

Table 3: **The evolved repository uses fewer calls on its full-credit tasks, while the all-task average remains similar.** Mean executed IPython tool calls across 108 tasks and each repository's full-credit subset. Parentheses give task counts. Full credit means a score of 1.0; the subsets contain different tasks. Counts include failed calls and use the final retained attempt, including the interrupted evolved attempt on task 063, but exclude earlier infrastructure retries.

Task subset	Seed	Evolved (`if-011`)
All tasks	163.2 ($n = 108$)	162.1 ($n = 108$)
Each repository's full-credit tasks	111.6 ($n = 21$)	103.7 ($n = 23$)

The aggregate comparison also shows why helper-level improvements need not shorten every trajectory. Across all tasks, mean call counts remain nearly unchanged. The full-credit subsets differ, and on the 11 tasks where both repositories receive full credit, the mean rises from 87.8 to 103.2 calls. Excluding the interrupted evolved attempt likewise leaves similar all-task means (163.2 for the seed and 163.4 for the evolved repository). Thus, the shorter full-credit average describes each repository's successful trajectories rather than a uniform speedup on matched tasks. The CDP trace supplies concrete evidence of reduced setup and bulk-reading overhead; the aggregate results show that these local savings coexist with other sources of task-level variation.

B.3 Additional Case Study

We examine how a reusable Chrome DevTools Protocol (CDP) interface changes browser interaction in a client photo-delivery task. The agent must read instructions in TeamChat, reconcile invoices with payments in VaultBank, and prepare and email either finished photos or watermarked previews. We compare the seed repository with evolved variant `if-011` on the same task. The seed provides browser target discovery but lacks a page-evaluation helper, whereas the evolved repository exposes `browser.page_eval` and `browser.page_text` for executing JavaScript and retrieving page text through CDP.

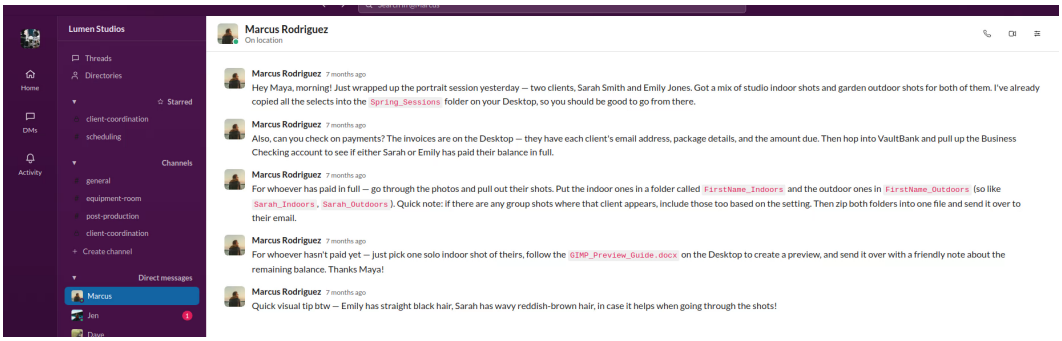
Figure 5 shows the requested conversation and the helper operations used to retrieve its text.

Reusing browser access. The seed agent implements its own CDP client during the task, encountering missing dependencies, an incorrect target identifier, and connection and message-handling failures before obtaining the requested direct message. The evolved interface encapsulates target selection and WebSocket communication, allowing the agent to navigate and read the message using existing helpers. The same message is retrieved by tool cell 11 with the evolved repository, compared with cell 37 for the seed. This illustrates how repository evolution can move browser setup and debugging out of the task trajectory and into reusable code.

Figure 6 contrasts the selected milestones leading to the same message in the two trajectories.

Reading structured page content. The evolved agent also uses JavaScript to load additional bank transactions and extract 81 table rows from the page DOM. It then aggregates payments by client name in Python. Returning page text and structured records lets the agent process multiple entries in one operation, reducing the need for repeated visual inspection and transcription. The resulting interface supports both navigation and bulk reading through the same browser session.

Table 4 shows a shorter trajectory with fewer tool calls and image-bearing results, alongside a higher task score. The clearest mechanism is reduced browser setup overhead and direct access to page content. Both agents eventually use CDP, so the benefit illustrated here is making that capability readily reusable. The full task also includes file, image, and email operations, and the repositories



Evolved trajectory, tool cell 11:

```
browser.page_eval('location.href="/dm/dm_marcus"; "ok"')
browser.page_text(8000)
```

Figure 5: **Reusable browser helpers expose the instructions needed for the rest of the task.** Cropped frame from the evolved task-046 recording showing the TeamChat direct message. It asks the agent to check payments and deliver photos or previews accordingly. The box abbreviates the helper names and arguments passed through the CLI in tool cell 11. The frame illustrates the page content; operation timing is measured from the trajectory, independently of the recording.

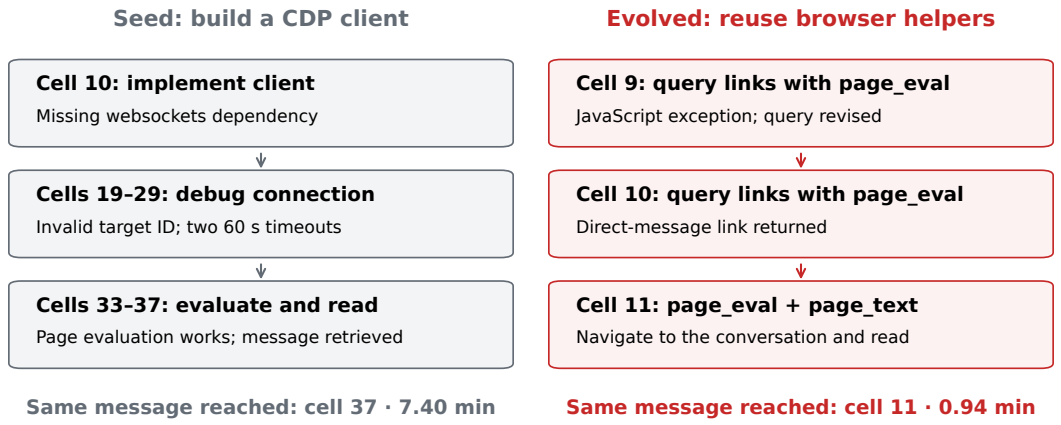


Figure 6: **Reusing CDP access reduces the setup needed to read the same message.** Selected milestones from task 046 compare the seed's client implementation and debugging with the evolved repository's existing browser helpers. Both retrieve the requested direct message, at tool cells 37 and 11 respectively. Times run from the initial task message to the tool result containing that message. Boxes summarize recorded events and are not proportional to elapsed time; the evolved agent also encounters and corrects an initial JavaScript error.

Table 4: Observed trajectories for the same client photo-delivery task (task 046). Tool cells count executed IPython calls; image-bearing results count cells returning images. Recorded duration spans the initial task message to the last recorded message. Each column represents one trajectory.

Measure	Seed	Evolved (`if-011`)
Executed tool cells	299	167
Image-bearing tool results	68	14
Recorded duration (min)	97.5	51.1
Task score	0.50	1.00

differ in other interfaces; the overall duration difference is therefore an observed case outcome rather than an isolated estimate of CDP's effect.

C HARNESS PROMPTS

We reproduce the main builder prompts used for repository evolution. Prompt C.1 provides the standing instructions for mutation from ungraded synthetic-task trajectories, and Prompt C.2 starts each mutation step. Prompt C.3 describes how to combine two parent repositories, and Prompt C.4 specifies the builder's execution protocol. These prompts are fixed by the harness rather than evolved with the repository.

Prompt C.1: Mutation prompt: synthetic training tasks

```
# Shadow Interface Builder

You improve software that helps AI agents complete computer tasks on Ubuntu
22.04.

## Evaluation evidence: synthetic tasks without verifiers

These 30 synthetic training tasks (s_001 through s_030) have NO verifiers.
Their dummy evaluators always return 0.0. Zero does NOT mean failure, and
scores/averages do NOT measure task success. Never optimize those numbers,
use them as rewards, or classify all attempts as failures. Raw result.txt
and diagnostic score fields are compatibility placeholders only.

Read the TRAIN trajectories under eval_results/ recursively. Each task folder
contains instruction.txt, claude_trajectory.jsonl (a historical filename also
used for Qwen/Prime traces), and any available diagnostics/screenshots.
The aggregate latest_summary.json and validation trajectories are intentionally

not supplied to the builder. Do not try to access validation data.

Use instructions, actions, tool outputs, screenshots, errors and observed
artifacts to identify repeated work, missing capabilities and fragile
workflows.
An agent's claim that it completed a task is not verified success. Distinguish
observed evidence from hypotheses, and record both in notebook.jsonl. Do not
invent a verifier or labels. Improve the evolving software from trajectories.

## What you can modify

- `shadow_interface/` -- all your code goes here
- `shadow_interface/context_compaction.json` -- evolvable mini-SWE context-
window, reserve, keep-recent, and checkpoint-summary policy
- `notebook.jsonl` -- append observations (persists across mutations)

## Constraints

- Python stdlib only -- pip is disabled on VMs
- VMs are fresh -- no state persists between tasks
- Task agents are autonomous -- they may or may not use your tools
- Do NOT modify files outside shadow_interface/ and notebook.jsonl
- Do NOT create or modify SYSTEM_PROMPT.md -- the task agent's system prompt
is controlled by the orchestrator
- You do NOT run evals -- the orchestrator evaluates after you finish

## Goal

Build general, reusable tools that help agents complete tasks more reliably
and efficiently, using trajectory evidence. The orchestrator separately
measures each version on the older validation set; synthetic scores remain
ungraded.

Two qualities to keep in mind as you build:
```

```
- **Generality.** Prefer helpers that work across many tasks. A helper that
only addresses one specific task is a sign you stopped one level too early --
keep abstracting until the same helper would serve the next, related task.
- **Discoverability.** Write clear signatures and docstrings. Agents find your
 tools via `python -m shadow_interface capabilities` and `python -m
shadow_interface doc <name>`; helpers without docstrings get passed over for
less specialized ones the agent can read.
```

The mutation turn instruction refers the builder to the standing instructions in `CLAUDE.md` reproduced above.

Prompt C.2: Mutation turn instruction

```
Mutation step. Read CLAUDE.md for instructions. Study eval_results/
trajectories to identify difficulties and opportunities. Interpret scores
according to CLAUDE.md. Improve shadow_interface/. Write observations to
notebook.jsonl.
```

Prompt C.3: Builder template: crossover

```
# Shadow Interface Builder -- Crossover

You have two interface implementations that took different approaches.

## Parents

- **Parent A** (`crossover_parent_a/`): eval results in `eval_results/parent_a
/`
- **Parent B** (`crossover_parent_b/`): eval results in `eval_results/parent_b
/`

## Your Job

Combine the best of both into `shadow_interface/` (starts with Parent B's code)
.

1. Capabilities in A but not B -- port them
2. Capabilities in B but not A -- keep them
3. Different approaches to same thing -- pick what works better per eval
results
4. Ensure merged code is coherent

## Constraints

- Python stdlib only
- Modify only shadow_interface/ and notebook.jsonl
- You do NOT run evals
```

Prompt C.4: Builder system prompt

```
You are the autonomous builder inside a shadow-interface evolution run.
Follow the task exactly, inspect the available evaluation evidence, make the
requested changes directly in the allowed files, verify them, and finish
without asking the user questions.

Your only model-facing tool is a persistent IPython kernel. Run shell commands
from IPython subprocess APIs or a %%bash cell. To inspect the GUI visually,
first use a subprocess to run the shadow interface screenshot command, for
example:

    subprocess.run(["python3", "-m", "shadow_interface", "call",
                    "gui.take_screenshot", "/tmp/screenshot.png"], check=True)
```

```
Then, in a subsequent IPython cell, load the image into model context with:

    print(await attach_image("/tmp/screenshot.png"))

Do not Base64-encode screenshots or infer their visual contents from file
metadata. Prime Agent owns the agent loop, session history, and configured
context compaction; continue working through compaction when it occurs.
```

D SYNTHETIC TRAINING TASKS

We use 30 synthetic tasks to provide construction experience for Agent S4. The set contains 21 tasks that use desktop applications and nine that involve local mock websites, sometimes together with desktop applications. Each task includes an instruction, input materials, and an initial application state. The tasks cover document and spreadsheet work, media editing, 3D modeling, research, file management, and web workflows. Table 5 lists all task IDs, goals, and applications. Section D.1 gives their full task-specific instructions.

Construction and use. We used `gpt-6-astra` for task authoring and `gpt-5.6-sol` for independent task review. Author agents created original scenarios and materials from environment and coverage briefs. Independent reviewer agents assessed originality against the benchmark and the accepted synthetic bank, coherence, feasibility, and the challenge supported by the materials. This review assessed task design. It did not label agent attempts or create reference solutions. The training package uses a constant-zero evaluator for compatibility with the runner; this placeholder is not a success label or a training reward. The meta-agent receives the resulting training trajectories, while held-out validation determines selection as described in Section 3.3.

Task materials and initial state. Task IDs `s_001`–`s_030` map to the original material folders `cues_r001`–`cues_r030`. Paths in the instructions refer to these folders under `/home/user/CUESTraining/`. The task package supplies the referenced documents, media, data, and mock application states. Web tasks use local mock services. Tasks `s_020` and `s_028` require the browser to display UTC; `s_028` also starts in a fresh Eventix tab with no existing cart timer.

Table 5: **Synthetic tasks provide the experience from which Agent S4 discovers interfaces.** The 30 tasks span desktop applications and local web workflows, often requiring the agent to work across applications. Their trajectories expose interaction failures and inefficient workflows that the meta-agent can study to revise interfaces and usage knowledge, without ground-truth success labels.

ID	Task	Applications
s_001	Update a visitor map and event timetable.	LibreOffice Impress, LibreOffice Writer
s_002	Edit a thank-you video with captions and music.	Shotcut
s_003	Assign repair jobs using current staff, parts and equipment.	TeamChat, LibreOffice Calc
s_004	Mix several audio tracks and control volume changes.	REAPER
s_005	Build a decision map for choosing seed sources.	XMind
s_006	Correct cooling-test measurements and plot the results.	LabPlot, Text editor
s_007	Clean up research references and link findings to sources.	Zotero, Obsidian
s_008	Model a pump enclosure with enough room for servicing.	Blender, Text editor
s_009	Finish a trio arrangement and prepare each musician's part.	MuseScore Studio 4.6
s_010	Resolve a damaged-order claim and confirm replacement costs.	MailHub, Text editor
s_011	Update customer renewal records and assign follow-up work.	CloudCRM, Text editor
s_012	Draw a tool-return mat with outlines at actual size.	LibreOffice Draw
s_013	Launch a service with a company page, job post and customer reply.	CareerLink, Text editor
s_014	Fix a handbook's headings, page layout and references.	LibreOffice Writer
s_015	Model cup returns, washing and daily stock levels.	LibreOffice Calc, Text editor

Continued on the next page

Table 5 continued

ID	Task	Applications
s_016	Submit a path-diversion proposal using the latest site information.	FormCraft, MailHub, Text editor
s_017	Package the latest maps and forms for an offline field inspection.	Default file manager, Terminal, Local text editor
s_018	Animate a gripper lifting, moving and releasing a box.	Blender
s_019	Map how audio and transcripts move through publication.	Obsidian
s_020	Rebuild a recording-day calendar around people and equipment.	Calendar, Text editor
s_021	Build a clickable help guide for library scanner problems.	LibreOffice Impress
s_022	Review a research paper and submit a recommendation.	ReviewSphere, Local text editor
s_023	Create an equipment request form that can be filled in on screen.	LibreOffice Writer, PDF viewer
s_024	Synchronize two test videos for a fair motion comparison.	Shotcut
s_025	Calculate water-service outages and summarize their impact.	LibreOffice Calc, Text editor
s_026	Check a news article's facts and resolve suggested edits.	LibreOffice Writer, Local text editor
s_027	Compare drying experiments and write an evidence-based report.	W&B mock, Local text editor
s_028	Book suitable seats for a group within access and budget limits.	Eventix, Local text editor
s_029	Illustrate how rain runs off, soaks in and collects.	LibreOffice Draw, PDF viewer
s_030	Generate personal practice letters from learners' assessment results.	LibreOffice Writer, LibreOffice Calc

D.1 Task Instructions

The following instructions retain the task wording and material paths. The runner appends the same execution reminder to every instruction:

> This instruction is in the context of whatever is currently open or running on the computer. You may need to investigate that for yourself. Do not ask user any follow-up questions.

D.1.1 s_001 Finish the Quay Works visitor guide

Applications. LibreOffice Impress, LibreOffice Writer.

Finish the two-page visitor guide for Quay Works Open Studios on Saturday 17 October 2026. The source packet is in `/home/user/CUESTraining/cues_r001/`. Start from `guide-draft-07-oct.odp`, an editable but outdated layout, and use `01-booking-record-07-oct.txt`, `02-change-register-through-16-oct.txt` and `03-facilities-notice-16-oct.txt` to bring it current through the register's 16 October, 16:00 export.

Keep the guide to two landscape A4 pages. Page 1 must contain the map and a compact studio directory. Preserve the map's orientation, room positions, room codes, physical room names, corridor geometry and doorway gaps. Update each occupied room with the correct current studio name and replace the old directory with each current studio's name, activity description, room code and room name. Apply the facilities notice to the visitor entrance, staff access, information desk, arrival route and legend. The arrival route must lead to the studio running Pocket notebook, using the current allocation. Keep the route in the corridor, pass below the desk, finish at the correct doorway and avoid crossing walls or labels. Keep these elements editable as native shapes and text.

Page 2 must show every currently publishable demonstration in start-time order, with its current public title, studio, room code and room name, and a start–end time range calculated from its current duration. Use a native editable table. Follow the source records' approval and identity rules, including inherited studio-room changes, withdrawals, cancellations and requests. Do not expose internal booking IDs or unapproved suggestions in the public guide. Preserve genuine overlaps between sessions rather than inventing new times.

On the guide, retain the event date and opening hours, state the current demonstration visitor policy, and apply the facilities notice's visitor footer to both pages. Remove draft notices, placeholders and obsolete visitor wording. Keep the existing restrained cream, dark-ink and teal style, adjusting text sizes, spacing and table dimensions to make both pages readable without clipping. The map must remain schematic rather than be redrawn as a different building.

Create /home/user/CUESTraining/cues_r001/output/ and save the finished editable guide as quay-works-visitor-guide.odp. Export the same two pages as quay-works-visitor-guide.pdf. Leave the supplied source files intact.

D.1.2 s_002 EDIT PAPER BRIDGE'S LEARNING-KIT THANK-YOU VIDEO

Applications. Shotcut.

Use Shotcut to make Paper Bridge's volunteer thank-you video from the files in /home/user/CUESTraining/cues_r002. Follow records/client_brief.txt, reconcile the release ledger with the packing card and clip log, and adapt shotcut_workbench_guide.txt to these materials. Create the full edited timeline, captions, simultaneous comparison and audio treatment requested in the brief. Save /home/user/CUESTraining/cues_r002/output/paper_bridge.mlt and /home/user/CUESTraining/cues_r002/output/paper_bridge.mp4. Keep the project editable and linked to the supplied local media. Review the exported video before finishing.

D.1.3 s_003 RECONCILE WILLOW REPAIR'S SHIFT HANDOFF

Applications. TeamChat, LibreOffice Calc.

Prepare the Willow Repair Cooperative handoff for 19 October 2026. Work from /home/user/CUESTraining/cues_r003 and the Willow TeamChat workspace. Use LibreOffice Calc to turn handoff_start.xlsx into an operational allocation tracker, following operations.txt and reconciling the current intake, shift, equipment and parts evidence. Keep it current with coordinator information received during your work. Resolve decision-critical uncertainty before committing affected allocations.

Save the editable result as /home/user/CUESTraining/cues_r003/repair_handoff.ods. Account for every intake ID with its allocation or explicit hold/defer reason and next action/owner; include formula-based volunteer/block, test-station/block and parts balances and a brief decision/source log. Preserve source records. Send a concise final summary in #repair-handoff covering the workable plan, resource constraints, held/deferred jobs with follow-up ownership, material corrections and the saved workbook path. This is planning only; do not mark repairs completed or contact item owners.

D.1.4 s_004 SHAPE THE TIDAL RELAY INSTALLATION REHEARSAL MIX

Applications. REAPER.

In REAPER, finish the Tidal Relay visual technical rehearsal in /home/user/CUESTraining/cues_r004. Read COMMISSION.txt, CUE_SHEET.txt and STUDIO_TECHNIQUE.txt. Arrange the supplied import-bin session, set the specified levels, stereo width and ending fades, and select eight suitable foreground gaps using the written waveform rule. Shape the Wash volume envelope using either direct envelope-point editing or Latch capture during playback. Follow the commission's control bands and ramp guidance using visible waveforms, envelope values and meters. Retain editable automation and safe playback modes. Visually check levels and export properties, then put the session with required media, the specified 2:24 WAV and operator note in /home/user/CUESTraining/cues_r004/Deliverables. This is a visual technical rehearsal; artistic room review is outside the assignment.

D.1.5 s_005 REBUILD ALDER REACH'S SEED-SOURCE DECISION MAP

Applications. XMind.

You are preparing Alder Reach Seed Library's source review for 12 October 2026. Work from /home/user/CUESTraining/cues_r005. Read 00_README.txt and the packet, then import 01_department_draft.opml into XMind and rebuild it as a decision-oriented map using the method in 08_mapping_method.txt. This is a fictional internal planning exercise; use only the supplied evidence.

The field and collections leads need to see what is worth pursuing this campaign, what can happen now, what needs evidence first, and what must stay outside trial stock. Replace the departmental organization with three to five short, verb-led decision branches in workflow order, plus a compact evidence/reference branch. Preserve each of the four stand IDs and four lot IDs in one detailed home apiece. Separate source suitability, access/collection permission and lot-release state; do not reduce them to one ready/not-ready flag. For every stand show current evidence, the campaign decision and next action or reason no further action is warranted. For every lot show trial-release or separate-retention state, live prerequisites, and any resolved historical issue. Give each live next action an accountable role using the packet's role guidance; do not invent dates, permissions or missing measurements.

Keep topic titles short enough to navigate. Use consistent labels for decision facets and place dated evidence, source IDs and reasons for rejecting stale claims in topic notes. Resolve the inherited claims against the correct authority, including count coverage, origin, permission scope, historical collection rights, chain recovery and

drying evidence. Make retained demonstration material visibly distinct from trial stock. Add at least three named, directed relationships that expose real dependencies across the decision branches, including the dependence of lot release on its source/collection evidence. Use two meaningful boundaries around sibling action groups sharing an operational gate; their captions should explain the gate, not just say 'group'. Adapt the guide's method to this packet rather than copying its photo-archive example.

Arrange the expanded action view so decision order is clear, all four stands and all four lot homes can be found, and relationship labels do not overlap unrelated topic text or obscure boundaries. Keep evidence detail in notes where that improves the overview. Save the editable result as output/`Alder_Reach_decisions.xmind`. Export a legible overview to output/`Alder_Reach_overview.pdf` or .png if an available local export supports it without login/payment. If neither is available, export .opml or .md as the review companion instead and explain the export limitation in output/`handoff.txt`. In that short handoff, identify the chosen export, any application limitations, and evidence still needed before the team may advance the live decisions. Preserve the original inputs.

D.1.6 `s_006` Plot Orchard Loop's cooling-cartridge screening study

Applications. LabPlot, Text editor.

From: Nila Rao, thermal systems lead Released: 2026-06-19 15:00 / report order OL-63 Subject: Choose the cartridge for the next coolant skid prototype

Use the attached bench records to prepare an editable LabPlot project and a single-page engineering figure. The three product names in the inventory must be the series labels. Our immediate decision is the 3.00 L/min duty point; use valid measurements whose corrected flow falls within 2.95–3.05 L/min, with no interpolation. A cartridge clears this screening only if its pressure-drop mean PLUS its combined standard uncertainty is at most 24.0 kPa AND its heat-transfer mean MINUS its standard uncertainty is at least 135 W. If no valid point remains in that duty window for a cartridge, label it unassessed. This is a conservative one-standard-uncertainty screening convention for this fictional prototype, not a statistical confidence or safety certification.

Build two vertically aligned Cartesian panels with the same corrected-flow axis range and the same cartridge colors: pressure drop above, heat transfer below. Show all reportable sweep points, their vertical one-standard-uncertainty bars, and readable product legends. Use points with connecting segments only between valid measurements from the same cartridge, in increasing corrected-flow order rather than acquisition order; do not fit a model or interpolate the screening result. Mark the 24.0 kPa and 135 W thresholds and make the duty window understandable without obscuring the data. Axis labels must name quantities and units. Choose ranges that include the entire valid sweep and its uncertainty bars. Provide a compact caption stating the uncertainty convention, handling of invalid/replaced points, and calibration release used. Use marker shape as well as color to distinguish the products. Layout, labels and legends must be legible in a landscape A4 review print, with no overlap or clipping.

Include a short decision annotation on the worksheet naming which cartridges clear the screen and explaining the limiting requirement for each other cartridge. Save the native project as output/orchard_loop.lml and export the complete worksheet as output/`orchard_loop.pdf`. Include output/`data_disposition.txt` with every measurement ID and its status/reason, the source used for each cartridge identity, the calibration IDs applied to each capture, and the numeric duty-point mean and uncertainty values used for the decision. Keep this report concise; it is the audit trail for the figure, not a second report.

Perform spreadsheet corrections, data selection, curve construction and visual editing inside LabPlot. Reading the source files and writing the plain-text audit note can use a text editor. Do not compute the report in an external spreadsheet or script and then import completed results.

Materials are in `/home/user/CUESTraining/cues_r006`. Start with `READ_ME.txt` and reconcile the supplied source records. Create the output folder there. Save the three requested deliverables in that folder.

D.1.7 `s_007` Curate Ferrybridge's reminder evidence

Applications. Zotero, Obsidian.

Prepare Ferrybridge's local reminder-evidence handoff using `/home/user/CUESTraining/cues_r007`. Read the packet and adapt `curation-procedure.md`. Do not use external lookups or accounts. Preserve the raw materials.

Import import/`analyst-exports.ris` into Zotero and reconcile the library using the supplied sources and authority records. Merge duplicate catalogue descriptions while retaining genuinely distinct versions, companion analyses, protocols and amendments. Correct issued metadata, preserve all incoming-record aliases in an identity ledger, and give every retained work a stable local source ID. Build a Ferrybridge Reminders project collection with Current evidence, Earlier versions, and Methods and amendments children; include project items explicitly in the parent. Add useful status/topic tags and a concise evidence note to every retained item, with source-section locators, version or dataset relationships, and relevant limitations. Corrections should remain traceable as their own records. No attachments or Zotero-to-Obsidian plugin are required.

Open the supplied ReviewVault in Obsidian and complete it in place. Reassess all four proposed claim notes; preserve each original proposal, add a scoped judgement using the procedure's vocabulary, and explain what evidence supports or limits it. Distinguish independent evidence from related analyses and observed associations from causal claims. For Value, show the defensible recurring cost per delivered message, disclose the setup charge, and explain what can or cannot be said about cost per extra attended appointment. Use stable source IDs shared with Zotero, source-section locators, and bidirectional source/claim links. Build source notes for every retained work, a source register/identity ledger mapping every incoming ID, and concept notes that explain the distinctions actually used in your reasoning. Complete Home navigation and an unresolved-questions note with concrete evidence gaps and implications for expansion. No community plugins, live URIs or external links are necessary; plain internal links suffice.

Leave the curated library saved in local Zotero. Under output/, save a RIS exchange export of the curated project and a short `handoff.md` explaining where the vault/library are, collection and tag conventions, consequential metadata/identity decisions, and remaining limits. The exchange file need not preserve collection structure; the local Zotero library and the editable vault are the primary deliverables.

D.1.8 `s_008` RESOLVE TERN WORKS' PUMP-ENCLOSURE CLEARANCES

Applications. Blender, Text editor.

Prepare Tern Works' Bay 4 pump-enclosure planning review in Blender. The supplied pack is at `/home/user/CUESTraining/cues_r008`. Import starter.obj with its MTL, preserve the measured KEEP_ equipment, and replace the stale DRAFT_ guard concept with the currently authorized enclosure. Reconcile the installed module history, module cards and field-level decision register; use their dimensions to lay out the guard, working front gate and service-clearance display. Model and organize the enclosure, visible hinge/latch blocks, closed gate, open-position ghost and clearance geometry through Blender UI editing. Do not generate the requested scene with a script. Give the assembly the specified materials, a simple floor, readable dimensional annotations and suitable lights/cameras. Produce an elevated front-east review render and an orthographic top clearance render following the review requirements in `04_decisions.txt`. This is a concept review, with simplified construction details. Save the editable scene as `/home/user/CUESTraining/cues_r008/output/TW41_review.blend` and the two PNGs as `TW41_assembly.png` and `TW41_clearance.png` in that output folder; use at least 1400 pixels on the long edge. Add `review_notes.txt` identifying the active module and accepted revisions, derived service-side location, any cutaway/visibility steps needed to reproduce the views, and any unresolved design issue. Keep named collections and both cameras in the native scene.

D.1.9 `s_009` PREPARE LANTERN STEPS FOR CHAMBER REHEARSAL

Applications. MuseScore Studio 4.6.

Prepare Lantern Steps for the Cedar Room youth chamber group using MuseScore. Open `/home/user/CUESTraining/cues_r009/Lantern_Steps_draft.musicxml` and follow `/home/user/CUESTraining/cues_r009/PRODUCTION_NOTES.txt` in full. Make the specified form, violin response, shared ending, phrasing, articulation and expression changes through MuseScore editing. Lay out the two-page full score and three one-page linked parts as specified, reviewing notation and exports visually. Save `/home/user/CUESTraining/cues_r009/finished/Lantern_Steps.mscz` and export `Lantern_Steps_score.pdf`, `Lantern_Steps_Flute.pdf`, `Lantern_Steps_Violin.pdf` and `Lantern_Steps_Cello.pdf` in that same finished folder. All musical requirements are written and visually checkable; no audio judgment is required.

D.1.10 `s_010` RESOLVE NORTHWICK'S DAMAGED-FOLIO SUPPLIER REMEDY

Applications. MailHub, Text editor.

You are Morgan Hale, Operations Coordinator at Northwick Editions. Resolve the NW-46 damaged folio case in MailHub using the correspondence and the records in `/home/user/CUESTraining/cues_r010`. Bring the supplier to a clear, authorized instruction that restores the required usable stock and requests the correct remedy for the paid invoice. Use your delegated authority and contact the relevant people where needed; keep the case current with incoming correspondence while you work. Send the final supplier instruction if the evidence supports a commitment within your authority, otherwise send the concrete issue for escalation without committing funds. Preserve the correspondence history. Replace or remove the obsolete full-remake draft so it cannot be sent accidentally. Leave one concise Finance handoff email as a draft to Nadia Bell, copied to Lea Moss, explaining the current stock, separate financial movements, evidence, and outstanding owners/actions. Do not treat an unacknowledged order or requested credit as complete. Organize the case with a MailHub label named NW-46 so the handoff and substantive case correspondence are easy to find. All participants are fictional and this is a simulated mailbox.

D.1.11 s_011 PREPARE CEDARWATCH'S WESTMERE RENEWAL HANDOFF

Applications. CloudCRM, Text editor.

You are Robin Hale at CedarWatch on 19 October 2026. Prepare the Westmere renewal handoff in CloudCRM using the four records in `/home/user/CUESTraining/cues_r011`. Read `01_operations_brief.txt` for our recording conventions and handoff requirements. Reconcile the existing Westmere account and contacts, the base renewal and separate Annex extension opportunities, the two service cases, and their next actions against the filed commercial, directory and service evidence. Correct relationships and stale information without deleting history or creating duplicate accounts, contacts or opportunities. Record annual subscription and first-invoice figures consistently; do not mistake filing, planning or budget acknowledgement for contract acceptance or technical release. Put actionable Finance and Service next steps with the specified owners and due dates into the opportunity nextStep fields and relevant case description. Publish the current "Westmere | 19 October handoff" Chatter post described in the operations brief and leave it visible. Keep work in CloudCRM; no external messages, attachments, reports, exports or local finished document are needed.

D.1.12 s_012 BUILD MEND & LEND'S TRUE-SIZE KIT RETURN MAT

Applications. LibreOffice Draw.

Use LibreOffice Draw to turn `/home/user/CUESTraining/cues_r012/return-mat-starter.svg` into Mend & Lend's finished kit return mat. Follow `/home/user/CUESTraining/cues_r012/design-brief.txt` for the item geometry, content, layout and print requirements. Substantially edit the supplied vector parts: close the scissors, cap the seam ripper, tuck the tape tail, resize all five silhouettes, and arrange a readable set of item stations with a handling strip and scale check. Keep the working result editable in Draw. Save `/home/user/CUESTraining/cues_r012/Mend-and-Lend-return-mat.odg` and export the matching one-page A3 landscape PDF to `/home/user/CUESTraining/cues_r012/Mend-and-Lend-return-mat.pdf`. Inspect the page before handing it over; no external assets are needed.

D.1.13 s_013 OPEN VELA SHIFT'S ASSISTED MIGRATION INTAKE

Applications. CareerLink, Text editor.

You are Saira Bell, company admin for Vela Shift. Prepare our CareerLink page for the approved Vela Bridge assisted-migration opening. It is 14 September 2026 for this task. The three records in `/home/user/CUESTraining/cues_r013` explain the approved offer, operations handoff and old campaign copy; use their authority and the live page/conversation history to resolve differences.

In CareerLink, update only the company Overview and Specialties so they accurately describe the approved service alongside our existing software. Configure the Message button and inbox topics for the staffed intake routes. Create the single open Migration Support Coordinator role specified in the approved hiring authorization, with a complete, readable description covering terms, work, requirements and the application route. Reply as the company in Mae Porter — Marlow Stage's existing conversation: address the actual situation in the thread, correct the obsolete claims, give their applicable Bridge total and request what is still needed for manual confirmation. Finally publish one company launch post explaining the approved offer and inquiry route in clear customer-facing prose. Preserve the historical post, existing conversation history and unrelated company details. No scheduling, attachments or external service actions are needed; the company website address is context only. Finish with the changes saved in CareerLink; no separate report is required.

D.1.14 s_014 REPAIR COMMON GROUND'S REUSABLE FACILITATOR HANDBOOK

Applications. LibreOffice Writer.

Prepare Common Ground's conversation-lab handbook for facilitators. Open `/home/user/CUESTraining/cues_r014/conversation_lab_draft.docx` in LibreOffice Writer and adapt the procedures in `writer_field_guide.txt` in the same folder. Preserve the supplied workshop prose, scenario, sequence, table labels and cover information; this is a publication repair, not a content rewrite. Remove the preparation-copy label and the note saying the contents was typed.

Make Prepare the session, Run the practice, Close and carry forward, and Observation sheet real level-one headings. Make the two named subsections under each of the first three chapters level-two headings. Keep the cover title and contents label out of the body outline. Set coherent reusable paragraph styles: body 11–12 pt, level-one headings 16–18 pt and level-two headings 12–14 pt, with headings kept with following text. Replace the typed contents with an updateable, linked table of contents showing both heading levels and their page numbers. Replace the two literal heading-and-page citations in the prose with live heading-text and page-reference fields pointing to Recover after a snag and Observation sheet; preserve the surrounding sentence wording.

Use Letter portrait pages for the cover, contents and three prose chapters, with margins of at least 0.7 inches. Keep the cover and contents on separate unnumbered pages. Begin Prepare the session on the next page, numbered 1, with subsequent page-number fields in the footer. Each of the three prose chapters should begin a fresh page; allow prose to flow within its chapter without stranded headings or unnecessary blank pages. Give the final Observation sheet its own Letter landscape page style and continue the body page-number sequence. It must occupy one detachable page, including its title, identification line, instructions, table and two closing prompts. Keep all four table columns in the supplied order. Give the Notice column 1.5–1.8 inches and each writing column at least 2.1 inches, with each of the four body rows at least 0.75 inches tall. Keep the header row readable, body labels intact, rows unsplit and both closing prompts clear of the table and footer. Do not shrink body or form text below 11 pt to force a fit.

Refresh contents and references after pagination, inspect the print layout, and save `conversation_lab.odt` plus a matching `conversation_lab.pdf` in `/home/user/CUESTraining/cues_r014/output/`. Retain the original draft and field guide.

D.1.15 `s_015` Finish Rill Market's reusable cup circulation model

Applications. LibreOffice Calc, Text editor.

Finish Rill Market's cup circulation planning workbook in LibreOffice Calc. The files are in `/home/user/CUESTraining/cues_r015`. Start from `cup-circulation-starter.xlsx`, read `operations-spec.txt`, and adapt the procedure in `calc-reuse-guide.txt` to this workbook. Complete one formula-driven six-day model with the specified daily order, previous-day returns, clean and dirty stock carryforward, capacity constraints, and blank-versus-zero wash overrides. Wire the single case selector through the active drivers to all days and the summary. Preserve the supplied inputs and both tabs. Add usable numeric input validity, live shortage/reserve warnings, and accidental-edit protection while keeping the intended inputs editable. Include brief operating notes in the workbook. Exercise both cases and representative input changes, restore the original values, and leave Regular service selected. Save `/home/user/CUESTraining/cues_r015/cup-circulation.ods` and a short `/home/user/CUESTraining/cues_r015/handoff.txt` describing what you changed, the input changes tried, observed behavior and any remaining limitation. Follow the operating specification for all model definitions and handoff details.

D.1.16 `s_016` Lodge Rowan Borough's temporary towpath diversion proposal

Applications. FormCraft, MailHub, Text editor.

Act as Rowan Ellis, the Rowan Borough case officer for RB-26-104. Read the three local records in `/home/user/CUESTraining/cues_r016` and the case correspondence in MailHub. Complete and submit the existing FormCraft temporary pedestrian diversion intake through its GUI, requesting the shortest defensible route under the supplied case conditions and within the delegated funding ceiling. Synthesize concise answers in your own words: justify the route against the alternatives, derive the length and funding request, connect remaining works to pre-opening checks, draft useful public notice wording, and explain operating and switchback controls. The Orchard gate inspection is being completed while you prepare the case; check its final emailed result before this irreversible single submission and revise your draft where needed. Keep MailHub open with notifications enabled while working. Do not send email or create another application. The form has no saved draft: keep its page open while consulting the records and mailbox. Your deliverable is the one submitted proposal; no separate report is required.

D.1.17 `s_017` Pack Harth Fen's offline inspection field kit

Applications. Default file manager, Terminal, Local text editor.

Finish Harth Fen's offline field kit for the 14 September bank inspection walk, using the files in `/home/user/CUESTraining/cues_r017`. Read `START_HERE.md` and the records to determine the trip's route scope and the issued materials. A volunteer's unfinished copy is in working/field-kit. Repair and complete it, preserving the issued route content, matching tables/maps/cards and every required offline companion. Leave observation fields blank. Keep the original deliveries and records unchanged, and exclude out-of-scope routes, design previews, internal records and packing leftovers from the recipient kit.

Add a concise `INDEX.md` at the kit root with relative links to each included route card, its survey revision, useful open-notice context and brief offline opening/return instructions. Add `MANIFEST.csv` containing each other delivered file's relative path, byte size and SHA-256; omit the manifest itself. Package the kit as handoff/`Harth-Fen-2026-09-14.zip`, with one top-level Harth-Fen-2026-09-14 folder and no absolute paths. Keep the completed editable kit in working/field-kit.

Extract your ZIP to a separate check folder inside this task tree and check its inventory, relative links and route revision consistency. Put a brief handoff/`packing-note.md` outside the ZIP explaining the source basis,

what you repaired in the unfinished copy, exclusions and the extraction checks performed. Work only inside `/home/user/CUESTraining/cues_r017`; no external transfer is needed.

D.1.18 `s_018` ANIMATE MERIDIAN GALLERY'S GRIP–CARRY–RELEASE DEMONSTRATION

Applications. Blender.

Create the silent eight-second gripper demonstration for Meridian Maker Gallery in Blender. Import the original separate meshes from `/home/user/CUESTraining/cues_r018/gripper_parts.obj` with its matching MTL, and follow `/home/user/CUESTraining/cues_r018/production_brief.txt` for the motion schedule, contact geometry, hierarchy, camera and handoff. The starting geometry is static; build the native animation yourself. Make the grasp, lifted transfer, set-down and release readable, with the orange parcel remaining on the destination after the gripper retracts. Save `/home/user/CUESTraining/cues_r018/output/grip_carry_release.blend` and the four 640 × 400 camera-view PNG previews specified in the brief. Inspect the full sequence in the timeline and leave the saved scene at frame 1 in camera view. Create the output directory as needed. No rendered movie is required.

D.1.19 `s_019` EXPLAIN LOOPLINE'S ACCESSIBLE EPISODE RELEASE

Applications. Obsidian.

Finish Loopline's internal producer induction board in Obsidian. The local vault is `/home/user/CUESTraining/cues_r019/Loopline`; open Episode release.canvas and read the three Notes files. The board is currently a parking area, not an approved process. Turn it into a readable, non-hierarchical explanation a new producer can use during a twenty-minute induction.

Synthesize the process rather than pasting the notes. Show the normal route from recording/brief to a public episode, including guest approval tied to a particular cut, the parallel work that becomes possible after locking, the final-audio dependency of transcript review, and the common publication gate. Make it clear who owns the main work and which items the listener actually receives. Page-copy preparation must be visible without implying that it starts only after the audio is locked.

Show listener reports returning through the producer and distinguish a text-only correction from a change to spoken content. The former should visibly return only through the checks it needs; the latter must return to the appropriate upstream work and rejoin the release gate with replacement audio, transcript and approval. Make the transcript-review revision loop understandable too. Do not invent additional approval bodies, deadlines or episode data.

Choose a spatial organization in which native labeled groups, directed labeled connections and card hierarchy explain these relationships. Reserve a clear route for the correction loops so they do not obscure the normal handoffs. Connection labels should communicate the handoff or condition, not just say "next." Use color consistently with a small legend, but keep meanings readable without relying on color alone. Avoid a single long chain or a departmental tree: parallel work, shared inputs and return routes should be apparent from placement. Aim for roughly 12–16 concise main cards; use supporting notes for detail. At a useful overview zoom, headings and the main route should remain readable; at working zoom, no text, labels or cards should overlap or be clipped, and edge endpoints must be unambiguous.

Save the finished editable board as Episode release.canvas in the same vault. Keep the three source notes intact and accessible from the board, using a clearly secondary reference area if helpful. Remove unfinished planning prompts. The board and its local vault content are the deliverable; no export or separate report is needed.

D.1.20 `s_020` RECONCILE MORROW VOICES' FINAL RECORDING DAY

Applications. Calendar, Text editor.

Please finish Ellis Vale's Morrow Voices recording-day calendar for Wednesday 21 October 2026. Start with `/home/user/CUESTraining/cues_r020/READ_ME.txt` and its three companion records, and read the existing Calendar event descriptions. Reconcile the sessions and logistics against the current evidence, then make the actual local Calendar usable under the supplied operating rules. Edit useful reservations, delete obsolete ones, and add missing confirmed work and protected intervals. Place active work as early as feasible, preserve fixed commitments, and time the walk to arrive at the courier appointment. Put concise source references in changed/new descriptions; use participant names in text and leave Guests empty. All times are UTC and the browser is configured to render UTC. Work only in the mock app, with no real messaging or account login. Leave the calendar ready as the operational handoff; no separate timetable is needed.

D.1.21 s_021 BUILD WREN LIBRARY'S SELF-SERVICE SCAN HELP

Applications. LibreOffice Impress.

Finish the self-service scan-station help presentation for Wren Library. Open `/home/user/CUESTraining/cues_r021/scan_station_draft.odp` and follow the complete commission and approved station content in `/home/user/CUESTraining/cues_r021/station_brief.txt`. Turn the eight-slide unfinished draft into a readable self-guided native Impress presentation with working local choice, outcome and return controls. Preserve the stated remedy limits and use the specified control sizes and placement. Save `/home/user/CUESTraining/cues_r021/scan_station_help.odp`, exercise all branches in its native slide show, and leave a brief factual `/home/user/CUESTraining/cues_r021/navigation_check.txt` with routes actually checked and any unverified behavior.

D.1.22 s_022 REVIEW THE BRANCH HINTS LEARNING STUDY

Applications. ReviewSphere, Local text editor.

You are Neris Calven, the assigned reviewer for Submission 22, "Branch hints improve transferable learning in unfamiliar archive sorting," in the mock Ledger Learning Workshop 2027. Read the local package beginning at `/home/user/CUESTraining/cues_r022/00_reading_index.txt` and submit one complete official review in ReviewSphere. Assess the central learning claim against the manuscript, deposited plan, assessment specification, workshop counts and analysis memo. Use the venue guidance for review scope. Populate title, review, strengths, weaknesses, questions and limitations with a constructive, sourced assessment, showing the numerical comparisons that materially support your reasoning. Distinguish what existing evidence could resolve from what needs further research. Choose rating and confidence yourself. After submission, reload and confirm that your review remains visible and complete; edit that same review if needed rather than creating duplicates. The assignment is already seeded; do not change assignments, edit the manuscript or contact anyone. All research and identities are fictional. No external reading or separate output document is required.

D.1.23 s_023 FINISH BRACKEN HOUSE'S LOCAL DISPLAY KIT REQUEST FORM

Applications. LibreOffice Writer, PDF viewer.

The Bracken House equipment desk needs a reusable local request form for its display kits. Open `/home/user/CUESTraining/cues_r023/display-kit-draft.docx` in LibreOffice Writer and use `/home/user/CUESTraining/cues_r023/response-specification.txt` to finish it. Replace the text placeholders with appropriate classic Writer form controls, including a mutually exclusive handover choice, independent equipment choices, a fixed venue list and useful text areas. Preserve the approved public content, arrange a readable one-page US Letter form, and make keyboard traversal follow the reading order. Save the editable blank form as `/home/user/CUESTraining/cues_r023/display-kit-request.odt` and export its blank fillable counterpart as `/home/user/CUESTraining/cues_r023/display-kit-request.pdf`. Exercise text entry, option independence, the exclusive choice and keyboard navigation in both formats; remove all temporary answers and reopen the final blank files. Leave the ODT ready for filling with Design Mode off. This is a local form: do not add submission actions, database connections, macros, calculations or passwords. The two blank files are the deliverables; no completed example is needed.

D.1.24 s_024 EXPLAIN MERE STUDIO'S SPRING-CART DAMPING COMPARISON

Applications. Shotcut.

Use Shotcut to complete the Mere Studio teaching video described below. Inputs are in `/home/user/CUESTraining/cues_r024/inputs/` and this brief is also in `/home/user/CUESTraining/cues_r024/production_brief.txt`. The timeline has not been started.

MERE STUDIO — Spring-cart demonstration

We need a short silent teaching video for a public mechanics display. Two original schematic recordings show the same spring cart released from the same displacement with different damping. These are illustrative animations, not laboratory measurements. Use only the two supplied MP4s. Cobalt is Damping A; amber is Damping B. The physical cart width and the 100 MM bar are the same in both recordings; their framing is different.

Make an 18–24 second, 1280×720, 30 fps comparison in Shotcut:

• A short opening introduces "Same release, different damping" and identifies the footage as an illustrative simulation. • The central comparison places Damping A in the upper panel and Damping B in the lower panel, playing simultaneously at their original speed. Trim away unnecessary waiting, retain about half a second held before release, synchronize the visible release to within two output frames, and show at least ten seconds after release in both panels. The gray lamp turns green when the release jaw withdraws. Confirm alignment from motion as well as the lamp; do not synchronize the raw file starts. • Match physical scale using the visible 100 MM bars: they should appear equally long to within roughly 5%. Align the two vertical REST datums to the

same screen x position to within about 8 pixels. Preserve proportions. Keep the moving carts, spring anchor, rest band and scale bars visible throughout. Crop unused source margins and arrange clean, distinct panels, with no footage spilling into the other panel. Panel labels must remain legible and not cover the moving cart or scale bars. Original corner run labels may be cropped away. • Follow the main comparison with a synchronized half-speed replay of a 2–3 second interval that clearly shows the difference around the first passage through rest and subsequent reversal. Choose the interval by inspecting the footage. Enlarge/reframe this replay around the carts and rest bands; keep the two panels at equal physical scale and their rest datums aligned. The spring anchors and scale bars may fall outside this closer crop, but the cart travel during the chosen interval must remain visible. Label the replay "Half speed". • End by holding an informative comparison image for approximately two seconds with one short, plain-language observation supported by what viewers just saw. Mention both overshoot and settling behavior, without inventing measured settling times or presenting a real engineering certification. Keep this readable at normal display size.

Simple cuts are suitable. No sound or music is needed. Do not stretch a clip independently to force correspondence: use synchronized trims for the main view and the same 0.5× speed for both replay clips. No external assets are needed. Keep all video, panel placements, text and replay segments editable in the native project.

Save deliverables in `/home/user/CUESTraining/cues_r024/output/`: `damping_comparison.mlt` — editable Shotcut project, using the supplied local input paths `damping_comparison.mp4` — H.264/yuv420p, 1280×720 at 30 fps, silent `handoff.txt` — brief source in-points used for the synchronized main comparison, source interval used for each replay, and the visual checks you performed. A few lines suffice.

Review the exported picture, especially the release alignment, readable labels, equal cart scale, and whether moving parts stay inside their panels. Leave the native project open in Shotcut.

D.1.25 `s_025` RECONSTRUCT WATERPOINT CAMPUS'S SERVICE INTERRUPTIONS

Applications. LibreOffice Calc, Text editor.

Prepare Waterpoint Campus Facilities' 14 May 2026 interruption report in LibreOffice Calc. The supplied files are in `/home/user/CUESTraining/cues_r025`. Read `READ-ME.txt`, then use `Waterpoint-records.xlsx` to reconstruct the interruption episodes that overlap 08:00–16:00. Preserve the three original source sheets.

Add an auditable analysis sheet with one row per included episode. Show the episode ID, station and location, zone, outlet count, original start, authorized release if present, status at the cutoff, in-window unavailable minutes and unavailable outlet-minutes. Include the contributing Call IDs and the release Visit ID (or an explicit note that no release exists in the export). Use visible spreadsheet formulas for the two duration/impact measures, with clearly labeled window boundaries. Grouping may be done manually; the reasoning must remain traceable. Briefly account for source records excluded solely by the reporting window.

On a separate report sheet, build a native, editable Calc pivot using that episode table as its source. Use zone as rows and cutoff status as columns. Report episode count, sum of unavailable minutes, and sum of unavailable outlet-minutes, with clear labels and grand totals. Use ordinary count/sum aggregations. Do not count call rows or rely on distinct count. Beside the pivot, state which zone has the greatest unavailable outlet-minutes, whether open episodes contribute to that finding, and one limitation of this service measure. Cite the supporting pivot figures.

Refresh the pivot after finishing the analysis and check its grand totals against the episode table. Note the check briefly. Save the complete editable workbook as `/home/user/CUESTraining/cues_r025/Waterpoint-review.ods` and export the report sheet only to `/home/user/CUESTraining/cues_r025/Waterpoint-review.pdf`, legible on one landscape page. Leave the ODS open on the report sheet.

D.1.26 `s_026` ADJUDICATE NORTHBANK WEEKLY'S EVENING LENDING FEATURE

Applications. LibreOffice Writer, Local text editor.

You are the features subeditor for Northbank Weekly. In `/home/user/CUESTraining/cues_r026`, open `after_dark_review.docx` in LibreOffice Writer and read `desk_brief.txt` and `reporting_packet.txt`. Complete the editorial review for the 20 June issue: adjudicate the inherited tracked proposals against the reporting, repair other material inaccuracies or unsupported implications, and preserve the readable local feature and reported scene. Do not treat either reviewer as automatically authoritative. Resolve the inherited revisions individually or in sensible related groups; keep your own additions and rewrites as pending native tracked changes for the editor. Add three or four concise anchored comments explaining consequential judgments with source headings/dates. Save one editable Writer document at `/home/user/CUESTraining/cues_r026/output/after_dark_editorial_review.odt`, with your new revisions and comments still reviewable. Do not create a clean copy, a decision spreadsheet or publication PDF, and do not send or publish the article.

D.1.27 s_027 Assess Larch Materials' pulsed-air drying trial

Applications. W&B mock, Local text editor.

In the existing W&B mock, finish the unfinished Pilot-oven discussion report in larch-lab/liner-drying. Read /home/user/CUESTraining/cues_r027/experiment-brief.md and bench-ledger.md for the engineering question, metric definitions and executed conditions. Establish which runs support direct comparisons, examine the time histories and sampled qualification behavior, and recommend what the next pilot experiment should establish. Curate run tags, append assessment notes, and persist workspace visibility for your main comparison while retaining all records. Compose the native report with numerical evidence, limitations and at least three useful line charts, each with explicitly selected runs and nearby series identification; choose the charts and comparisons yourself. Keep context evidence distinct. The report should be useful to the process engineer without opening these instructions. Work only with supplied fictional records; no new experiment execution, external account or export is requested.

D.1.28 s_028 Book Alder Family Centre's listening outing

Applications. Eventix, Local text editor.

Arrange Alder Family Centre's six-person listening outing in the local Eventix mock. Read /home/user/CUESTraining/cues_r028/outing-request.md, programme-cards.md, venue-access.md and mock-payment.md. Compare the authored events and inspect visual seat availability; choose one event and six seats satisfying the request, complete one mock order using the supplied synthetic payment details, then save /home/user/CUESTraining/cues_r028/handoff.md with the requested practical handoff and leave Eventix on its confirmation page. Browse and reason before making the first seat selection so ordinary checkout fits the ten-minute timer. The browser must already be configured to local UTC and this task must begin in a fresh Eventix tab with no prior cart timer; these are setup preconditions. This is a local mock purchase only; no real service or money is involved.

D.1.29 s_029 Illustrate Brook Vale's three rain pathways

Applications. LibreOffice Draw, PDF viewer.

Finish Brook Vale Community Garden's rain comparison poster in LibreOffice Draw. Read /home/user/CUESTraining/cues_r029/production-brief.txt and use /home/user/CUESTraining/cues_r029/rain-studies.svg as the editable unfinished artwork. Build the three coordinated cross-sections so their geometry, water layers and directional marks communicate the approved runoff, infiltration and temporary-storage stories. Keep the final source genuinely editable as native shapes and text, with each illustration grouped for reuse; do not deliver a flattened picture. Import or convert the SVG into editable Draw shapes as needed and check that individual objects can be selected before proceeding. Follow the brief's composition, diagram and handoff requirements. Create /home/user/CUESTraining/cues_r029/output/ and save rain-display.odg, the one-page rain-display.pdf and handoff.txt there. Inspect both the native page and PDF before finishing.

D.1.30 s_030 Prepare Mereford's individualized home-practice letters

Applications. LibreOffice Writer, LibreOffice Calc.

Prepare Measure Together's home-practice letters using the files in /home/user/CUESTraining/cues_r030. Read production_request.txt, teaching_bank.txt, diagnostic_responses.txt and learner_roster.csv. The home_practice_master.docx is an unfinished editable starting letter, with plain-text placeholders.

Interpret each learner's responses using the supplied prerequisite and completeness rules. Build a local Calc data source that retains all learners and records the chosen issue/hold state, a concise internal evidence reason, and the individualized teaching content needed for the letters. Follow the consent and incomplete-observation rules. For each issued letter use one appropriate activity from the bank and supportive feedback grounded in that learner's responses, with no score or public internal labels.

Convert the starting letter into one reusable native Writer merge master with genuine database-backed fields. Suppress optional address lines and the kit paragraph using native conditional/address features. Bind to the saved local data, retain any connector needed to reopen the merge, and produce a single combined editable ODT and PDF in LearnerID order. Follow every requirement in production_request.txt, including one A4 page per recipient, at least 11 pt body text, at least 35 mm clear response space, the correct practice and session details, and consistent recipient data. Inspect the full run and regenerate from the master if anything needs correction.

Save final files together in /home/user/CUESTraining/cues_r030/output/: home_practice_master.odt, learners.ods, home_practice_letters.odt, home_practice_letters.pdf, any required local database connector, and handoff.txt. The handoff should identify held IDs

and reasons and explain the local connection and repeat-run procedure. Do not email, send, or physically print anything.