CONSCRIBE: REPAIRING LEARNED SKILLS WITH RESIDUAL UPDATES FOR CHANGING WEB ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Environment changes can leave learned skills partly useful and partly invalid. Adapting them requires deciding how new behavior should build on their inherited implementation. We introduce ConScribe, a skill-maintenance framework that generates residual updates and composes them with retained components, including those within the edited scope. Execution checks guide acceptance, wider repair, and reconstruction. On 24 public Mind2Web transfers, initial repair raises element-selection accuracy from 67.97% under static reuse to 75.78%. The last candidate outperforms information-aligned, source-preserving dual path by 9.90 percentage points. Author output decreases by 37.57% on 22 measured transfers, and online wall time decreases by 28.83% on 22 complete-record transfers. A controlled comparison across 48 Chromium transitions holds target AUC at 0.867 while source pass rises from 33.3% under local replacement to 100% under residual composition. Fixed-decomposition experiments also identify conditions where reconstruction costs less than repair. Together, these results show that the form of a skill update matters for target recovery, source retention, and adaptation cost, establishing the value of building target behavior on an inherited skill.

1 INTRODUCTION

Large language model agents turn interaction experience into reusable workflows, programs, and callable tools (Wang et al., 2025a;b; Prabhu et al., 2026). These skills improve task success, reduce repeated interaction, and support transfer across tasks and websites (Wang et al., 2025a;b). Yet evolving services, APIs, packages, and configurations can invalidate previously useful skills (Fan et al., 2026). Continuing to use acquired skills therefore requires maintaining them across changes in their operating environment. When part of a skill remains useful, adaptation must connect new requirements to an implementation that already encodes relevant behavior.

A submit action illustrates the choice. A target website may require a new locator while the original locator remains useful on the source website. Local editing can replace the action's implementation; a dual-path design can keep the source skill and generate a complete target version. A third option retains the inherited action and adds the target-specific behavior. These choices can use the same source knowledge but differ in what they preserve and what the skill generator, or *author*, must produce. We study how this choice of update affects recovery, retention, and cost.

ConScribe develops the third option into a skill-adaptation workflow. The author generates a residual relative to the inherited skill; a host-native composer retains existing components, including those inside the edited unit, and connects new behavior through activation conditions. The assembled skill undergoes target and, when required, source checks. Successful candidates become adapted skills; unsuccessful checks guide further edits, wider repair, or reconstruction within budget. This workflow makes the inherited implementation the base of adaptation and treats generation as supplying the update to that base.

Our experiments connect this update choice to both adaptation quality and resource use. On 24 public Mind2Web transfers, initial repair raises element-selection accuracy from 67.97% under static reuse to 75.78%. With source access and source-path retention shared, the last candidate outperforms dual path by 9.90 points. Author output falls by 37.57% on 22 measured transfers, and online

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.

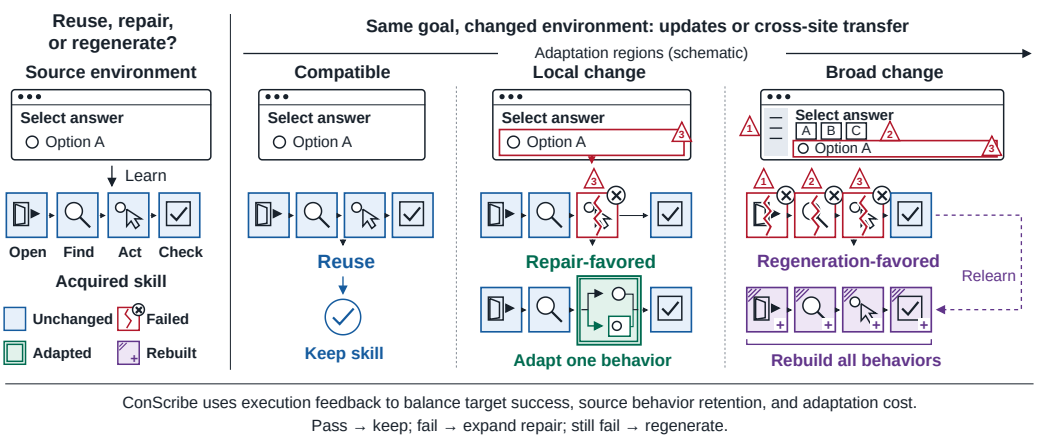

Figure 1: **The skill-adaptation problem under environment change.** Updates or cross-site transfers can leave a skill compatible, locally invalid, or broadly unsupported, favoring reuse, repair, or regeneration. Execution feedback guides adaptation while qualification tests target recovery and source retention.

wall time falls by 28.83% on 22 complete-record transfers, including traversed checks and failed attempts. Controlled studies cover 504 environment pairs and 48 held-out Chromium transitions. In Chromium, local replacement and residual composition share target AUC 0.867, while source pass rises from 33.3% to 100% with composition. Thus the experiments examine both the benefit of retaining implementation within changed units and the value of generating updates when both methods already have old knowledge.

Our contributions are as follows:

1. **Skill maintenance through preservation and composition.** We formulate adaptation after environment change and introduce a workflow that retains inherited implementation, authors relative updates, and checks the composed skill. Bounded repair leads to reconstruction when required.
2. **Update form as a determinant of adaptation outcomes.** Controlled local-edit comparisons identify a source-retention benefit from preserving components inside changed units. Information-aligned dual-path comparisons show improved target recovery and lower measured update costs with source access and source-path retention held fixed.
3. **An empirical account of repair and reconstruction costs.** Fixed-decomposition experiments expose a cost crossover as change support and skill size vary. Model-backed measurements distinguish update generation from the cumulative cost of the executed adaptation path.

2 Related Work

2.1 Agent Skill Acquisition and Reuse

Skills abstract primitive actions (Sutton et al., 1999). LLM agents have progressed from externally specified tools in ReAct and Toolformer (Yao et al., 2023; Schick et al., 2023) to tool synthesis in CREATOR and LATM and reusable collections in CRAFT and TroVE (Qian et al., 2023; Cai et al., 2024; Yuan et al., 2024; Wang et al., 2024b).

Interactive agents further turn experience into persistent skill libraries. Voyager accumulates verified programs during open-ended exploration; Agent Workflow Memory extracts recurring workflows from demonstrations or interaction histories; and Agent Skill Induction constructs, verifies, and reuses programmatic skills online (Wang et al., 2024a; 2025b;a). WALT derives callable tools from functionality provided by websites, while SAGE learns to create and invoke skills over related

tasks (Prabhu et al., 2026; Wang et al., 2026). ConScribe addresses adaptation of these acquired artifacts after their operating environment changes.

2.2 Skill Transfer and Generalization

Skill priors, task-agnostic repertoires, and hierarchical transfer reuse behavior across tasks (Pertsch et al., 2021; 2022; Matthews et al., 2022); related work discovers shared behavioral structure across changing environments (Lee & Seo, 2023). For web agents, Mind2Web evaluates generalization across tasks, websites, and domains, while WebArena provides realistic multi-site tasks with functional execution-based evaluation (Deng et al., 2023; Zhou et al., 2024).

Recent systems directly transfer procedural knowledge across interaction contexts. Agent Workflow Memory transfers induced workflows, and Agent Skill Induction reuses common programs while updating skills incompatible with a new site (Wang et al., 2025b;a). PolySkill separates abstract goals from site-specific implementations to support cross-website composition and continual acquisition, whereas WALT derives deterministic, site-native interfaces (Yu et al., 2026; Prabhu et al., 2026). SAGE similarly exploits skills accumulated over related task chains (Wang et al., 2026).

These methods establish that skills can be retrieved, transferred, and updated. ConScribe asks how much of a target skill must be newly authored when an inherited implementation remains partly useful. It constrains the update to a residual composed with inherited components, and compares this choice against both local replacement and complete target-path generation.

2.3 Self-Improving Agents

Self-Refine, Reflexion, and ExpeL use feedback to revise outputs or accumulate knowledge (Madaan et al., 2023; Shinn et al., 2023; Zhao et al., 2024). AFlow searches over code-represented workflows, ReCreate converts interaction evidence into targeted scaffold modifications, and Mem2Evolve evolves experience together with reusable capabilities (Zhang et al., 2025; Hao et al., 2026; Cheng et al., 2026). SkillTracer localizes failures within composite skill structures and refines the implicated components (Li et al., 2026).

Automated program repair spans semantic analysis, learned fix patterns, and text-to-text patch generation (Monperrus, 2018; Nguyen et al., 2013; Bader et al., 2019; Berabi et al., 2021). ConScribe combines execution-guided proposal search with a composition constraint: inherited components remain represented even inside the edited scope. Target qualification tests the new behavior; source qualification tests whether the composed artifact retains the old capability. This separates the permitted form of an update from its behavioral acceptance.

3 Skill Adaptation across Environment Transitions

3.1 Source Skills and Environment Transitions

A skill-learning host h acquires skill s_s from task distribution $\mathcal{D}_s$ in source environment $\mathcal{E}_s$, using budget B_s, and validates it there:

$$s_s = \mathcal{G}_h(\mathcal{E}_s, \mathcal{D}_s; B_s), \qquad \mathrm{Verify}(s_s; \mathcal{E}_s) = 1. \tag{1}$$

The skill retains its host-native program, workflow, or instruction format.

An environment transition $\delta : \mathcal{E}_s \to \mathcal{E}_t$ preserves task intent while changing execution requirements. A *version transition* changes an existing environment's interface, configuration, or semantics; a *cross-environment transition* moves the capability to a functionally matched environment with a different implementation. Adaptation carries s_s across this transition.

3.2 Adaptation Objective

Given s_s and target evidence $\mathcal{Z}_t$, comprising task inputs, public observations, execution outcomes, and failure feedback collected within budget B, an adaptation operator produces:

$$\hat{s} = \mathcal{A}_h(s_s, \mathcal{Z}_t; B), \qquad \hat{s} \in \mathcal{S}_h \cup \{\perp\}, \tag{2}$$

where $\mathcal{S}_h$ is the host's artifact space and $\perp$ denotes failure to obtain a qualified candidate within budget.

We separately measure *target recovery* on held-out tasks in $\mathcal{E}_t$, *source retention* by the same artifact in $\mathcal{E}_s$, *lifecycle cost* for interaction, generation, editing, and validation, and *edit magnitude* in the host-native representation.

The reference alternative is host-native reconstruction, $s_t^{\text{scratch}} = \mathcal{G}_h(\mathcal{E}_t, \mathcal{D}_t; B)$, which builds a target skill from an empty artifact using the same host and target information.

3.3 Change Locality and the Repair–Reconstruction Continuum

For a fixed host adapter h and segmentation Seg_h, s_s exposes n_h task-relevant units:

$$\text{Seg}_h(s_s) = (u_1, \ldots, u_{n_h}), \qquad K_\delta^h \subseteq \{1, \ldots, n_h\}, \qquad k_h = |K_\delta^h|. \tag{3}$$

Here, K_δ^h is the minimal support of units, dependencies, or insertion sites that must change for target recovery. Its size k_h is the *change support*, and k_h/n_h measures transition locality. Environment construction or structural audit determines this support for analysis; adaptation uses observable $\mathcal{Z}_t$. These quantities depend on the fixed segmentation and are not comparable across segmentations; we write n, k in fixed-host experiments.

Repair retains unaffected units, while reconstruction builds a complete target skill. We hypothesize that repair burden grows primarily with k_h and reconstruction burden with n_h: $k_h = 0$ permits reuse, intermediate k_h permits repair, and reconstruction becomes more competitive as k_h approaches n_h. Section 4 searches this progression using qualification feedback; Appendix C.3 derives its cost condition.

4 ConScribe: Residual-Preserving Skill Adaptation

ConScribe starts from a source-validated skill s_0 and target execution evidence. Each cycle selects an edit scope, proposes a relative update, and qualifies the composed skill. Unsuccessful repair widens the scope before host-native reconstruction (Figure 2).

4.1 End-to-End Adaptation Workflow

Host-native interface. The adaptation layer requires four host-native operations:

$$\mathcal{I}_h = \big(\text{segment}_h, \text{execute}_h, \text{edit}_h, \text{scratch}_h\big). \tag{4}$$

Here, segment_h exposes the host's editable units, execute_h runs an artifact and returns an execution trace, edit_h proposes a bounded native update from that evidence, and scratch_h constructs from an empty target artifact. Units may be program steps, workflow nodes, browser actions, or instructions. The host provides execution, proposals, ranking, and persistence; ConScribe controls the starting artifact, edit scope, residual composition, and escalation.

Operational policy. Algorithm 1 in Appendix B gives the complete decision procedure. A host provides nested edit scopes $\Omega_0 \subset \cdots \subset \Omega_L$, per-scope proposal limits J_ℓ, and a total budget B. Qualification executes the same candidate on target cases $\mathcal{Q}_t$ and optional source cases $\mathcal{Q}_s$, returning pass indicators (q_t, q_s) and failure feedback. These adaptation-time checks are separate from the final blind evaluation. Acceptance requires $q_t \wedge q_s$; an omitted source check sets $q_s = 1$.

The terminal labels distinguish zero-edit reuse (`KEEP`), qualified residual repair (`REPAIR`), qualified reconstruction (`REBUILD`), and no qualified artifact within budget (`REJECT`).

4.2 Failure-Guided Residual Repair

Execution-grounded localization. Let $s^{(j)}$ be the current working artifact and let $\xi_t^{(j)}$ contain the latest tested candidate's execution trace, verifier outcome, and observations. The working artifact supplies the next proposal's base; the latest candidate supplies its feedback, including after rollback. The host uses this evidence to propose editable support $\Omega^{(j)}$. An execution error supplies a failed

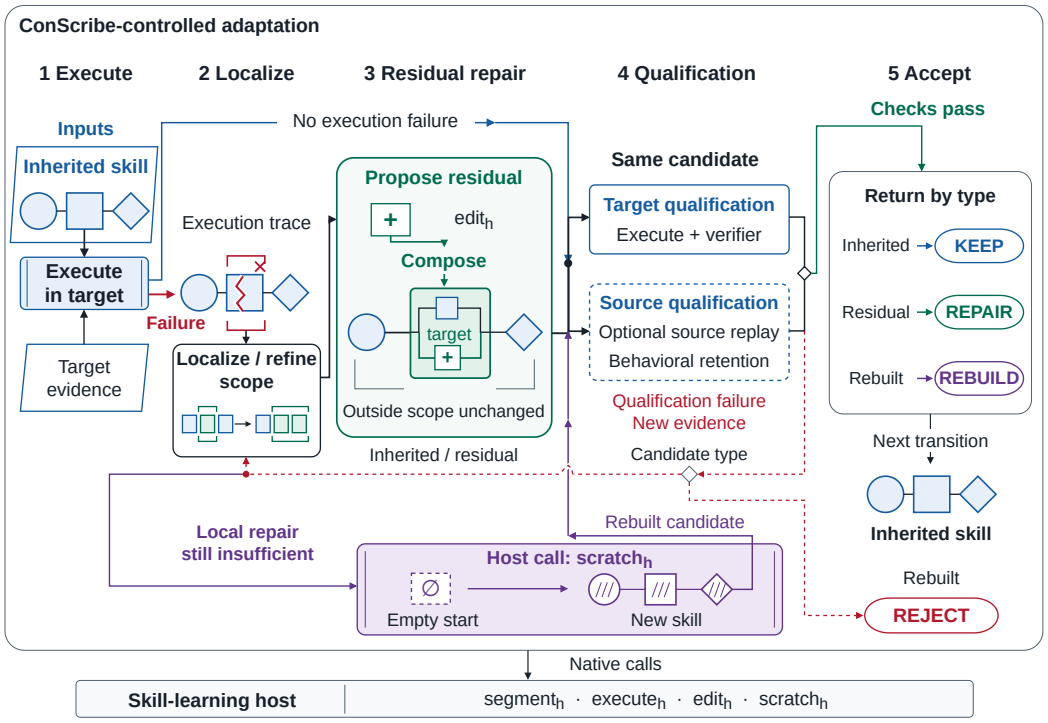

Figure 2: **Overview of the ConScribe adaptation workflow.** Target execution localizes failures for residual composition. The same candidate undergoes target and optional source qualification; failure triggers refinement or reconstruction. Accepted artifacts can be inherited by the next transition. The host supplies native operations; ConScribe controls adaptation.

unit directly; a task-verifier failure supplies an unsuccessful outcome from which the host proposes a repair site. In the browser adapter, a verifier-only failure revisits the last edited unit, or the final source unit when no edit has occurred. Qualification tests the resulting candidate, and widening releases additional units for modification. The hidden change set K_δ^h is reserved for locality analysis.

Residual composition. A residual is a host-native update interpreted relative to a specified base artifact. At iteration j, this base is the working skill $s^{(j)}$: the update specifies additions or revised target rules, and the composer supplies retained implementation from the base. The host proposes $r_t^{(j)}$ over the current support and composes it with that base:

$$r_t^{(j)} = \mathrm{edit}_h\left(s^{(j)}, \xi_t^{(j)}, \Omega^{(j)}\right), \qquad s' = s^{(j)} \oplus_h r_t^{(j)}. \tag{5}$$

The operator $\oplus_h$ preserves the part of the artifact outside the edit support:

$$\Pi_{\overline{\Omega^{(j)}}}(s') = \Pi_{\overline{\Omega^{(j)}}}\left(s^{(j)}\right), \tag{6}$$

where Π projects an artifact onto a set of units. Inside the support, let $\mathcal{B}_i(s)$ denote the stored implementation components of unit i under the host's unit correspondence; a new insertion site has $\mathcal{B}_i(s_0) = \emptyset$. The inherited source components obey

$$\mathcal{B}_i(s_0) \subseteq \mathcal{B}_i(s'), \qquad i \in \Omega^{(j)}. \tag{7}$$

A residual supplies new behavior together with its host-native activation condition: lookup failure, an environment guard, or a continuation condition. Composition retains the inherited components and connects the addition through that condition. The two structural constraints govern candidate construction; qualification governs execution and acceptance. Passing both checks returns the candidate. If the target check fails but the source check passes, the candidate becomes the next working

artifact so subsequent proposals can build on it. A source failure keeps the previous base while retaining the new failure feedback (Algorithm 1).

Appendix C.2 defines storage preservation, execution dependence, and generated content for each representation.

Host-specific composition. Table 1 maps the contract to the evaluated representations. For example, a six-unit contact skill adds a target submit locator while executing the other five units unchanged. Source and target replay test the resulting behavior (Appendices E and F). If an old locator matches the wrong control, lookup-failure activation remains inactive. Task verification then supplies feedback for further permitted edits or reconstruction; compatibility conditions are given in Appendix C.2.

Table 1: Host-specific realizations of the update contract. Search operations refer to the corresponding study protocols.

Representation	Inherited base and residual	Activation	Search progression
Browser units (Chromium)	Retain unit actions/locators; add alternatives or optional execution	Lookup failure or completed task	Local candidates, then global reconstruction
Text skills (Mind2Web)	Retain source instructions; compose target-rule updates	Website identifier	More support examples and revision, then reconstruction
Text skills (SkillRevise)	Retain source fields; attach target constraints	Environment descriptor	Initial update, then one feedback revision
Linear scorer (public index)	Retain w_s; store $\delta = w_t - w_s$	Source: w_s; target: $w_s + \delta$	Fixed evidence-budget sweep

4.3 Validation-Driven Escalation

Target qualification. Each proposed artifact is executed in E_t and passes only if execution terminates cleanly and the task verifier accepts the resulting state. Execution failures localize further mismatches; verifier failures identify behavioral incompleteness despite clean execution. Both inform the next proposal.

Source qualification. When source retention is required, the same repaired artifact is replayed in E_s and source-qualified only if its verifier accepts. A source regression triggers rollback and revision with a conditional branch or wider support. Without source cases, Equation 6 still preserves structure, but source behavior remains uncertified.

From local repair to reconstruction. Widening relaxes the assumption that a correction can be expressed within the current support. In a segmented host, a larger scope permits changes to interacting units; in the text-host protocols, widening supplies additional evidence and revises the target residual. Reconstruction releases the requirement to compose the target implementation from inherited components. Each scope has a proposal limit J_ℓ; exhausting it advances to the next scope. Once repair scopes are exhausted, $\mathrm{scratch}_h$ uses the target evidence and remaining budget. The reconstructed candidate undergoes the same required checks before acceptance. The common budget charges proposals, checks, and failed attempts; exhaustion without a qualified artifact returns `REJECT`. For one bounded repair stage, let c_R be expected proposal and qualification cost, p its acceptance probability, and c_F the expected fallback cost conditional on failure. Trying repair first has expected cost $c_R + (1-p)c_F$. Relative to direct reconstruction costing c_S, it saves cost when $c_R + (1-p)c_F < c_S$, at matched required quality and completion probability. This condition connects update compactness to successful qualification and the price of failed attempts (Appendix C.3). The policy uses qualification outcomes and attempt limits; k_h remains an analysis variable.

Repeated environment evolution. The accepted artifact after version v becomes the inherited skill for the next transition:

$$s_{v+1} = \mathrm{Adapt}_h(s_v, E_v \to E_{v+1}; B_v)\,. \tag{8}$$

`REPAIR` adds a residual while retaining inherited paths. Later transitions can requalify a represented path for `KEEP`, add another residual, or replace the active artifact through `REBUILD`.

5 EXPERIMENTS

We evaluate target recovery and adaptation efficiency, then examine the roles of residual composition and update granularity. Additional hosts test these outcomes across representations, while controlled changes expose the repair–reconstruction cost crossover.

5.1 EXPERIMENTAL SETUP

Tasks and comparisons. Aligned Mind2Web covers 24 functionally matched transfers in 12 source blocks across 27 websites and nine website-dependency clusters (Deng et al., 2023). Static reuse leaves the source skill unchanged; scratch starts from an empty artifact. Aligned dual path and ConScribe share source access, source-path retention, task information, rule specificity, and bilateral qualification opportunities. Further studies use Chromium, SkillRevise, REAL, PolySkill, and SkillEvolBench (Liu et al., 2026; Garg et al., 2025; Yu et al., 2026; Lei et al., 2026). Within each study, methods share the evaluator, execution interface, evidence/budget rules, and model where applicable. Adaptation and blind tests are disjoint. Appendix Tables 6–7 detail the protocols and sampling units.

Measurements. Budget-normalized target AUC integrates accuracy over the evidence-budget range. Source accuracy measures execution outcomes; prediction stability measures unchanged outputs; joint success requires all source and target tests to pass. Author output counts generated tokens; total tokens include input and output; online costs accumulate the executed adaptation path. Aligned Mind2Web weights source blocks equally after averaging paired measurements within blocks; CIs resample website-dependency clusters. Controlled studies use construction clusters; public-index intervals pair five selection seeds. Tables state coverage, and Appendix D.1 specifies cost aggregation and stopping.

5.2 TARGET RECOVERY AND ADAPTATION EFFICIENCY

Recovery from an inherited skill. Aligned Mind2Web evaluates target-site grounding: selecting from the top-50 elements given the operation, value, and action history. Static reuse reaches 67.97% target accuracy; the first residual update (B1) reaches 75.78% (Table 2(a)), a paired gain of 7.81 percentage points.

Against source-preserving complete generation, the last residual candidate reaches 77.60% accuracy versus 67.71% for dual path, a **9.90-point gain**; both retain 59.38% source accuracy. The two comparisons establish recovery over the unchanged skill and an advantage over complete target generation with shared source access. Blind scoring covers all last candidates; 23 residual and 24 dual-path candidates qualify for deployment.

Table 2: Aligned Mind2Web adaptation: residual versus dual path. B1 accuracy is absolute; gains and savings are relative to dual path. N counts measured transfers. Savings average paired ratios within and then across source blocks; CIs resample website clusters (Appendix A.2).

(a) Element-selection quality

Measure	Value	95% CI	N
B1 accuracy (%)	75.78	[67.18, 84.09]	24
Last gain (pp)	+9.90	[6.82, 12.28]	24

(b) Resource saving (%)

Measure	Saving	95% CI	N
B1 author output	37.57	[33.00, 41.00]	22
Online total tokens	27.25	[20.57, 30.60]	12
Online wall time	28.83	[25.35, 32.55]	22

Efficiency at generation and along the adaptation path. B1 author output decreases by **37.57%** and author time by **31.17%**, locating a generation benefit before escalation. Online wall time decreases by **28.83%** over 22 complete-record transfers in nine website clusters, including the full adaptation path. Online tokens decrease by 27.25% on 12 complete-usage transfers in four clusters.

5.3 Composition and Update Granularity

Preservation within edited units. With matched edit location and target proposals, the local editor replaces inherited locator/action candidates while residual repair appends alternatives. Across 48 pairs, target AUC, final success, decision route, and edit fraction match, but source pass rises from 33.3% to 100% in post-hoc replay (Table 3); among 32 changed transitions, it rises from 0% to 100%. Composition thus preserves behavior *inside* edited units.

Table 3: Chromium update comparison over 48 transition pairs. Source pass requires both source tests to pass. Local edit and residual repair share failure feedback and the local proposal procedure.

Method	Target AUC	Source pass (%)
Static reuse	0.333	100.0
Scratch	0.475	33.3
Target-only local edit	0.867	33.3
Residual repair	0.867	100.0

Update granularity with source-path retention. Aligned SkillRevise tests two lookup shifts with three replicates and four blind items per environment. Both methods share source skills, retrieved principles, and rule requirements, qualify at B1, and pass all bilateral blind tests. Residual updates reduce output, time, and editing in each shift (Table 4); total author tokens fall 19.2% (Appendix D.1).

Table 4: Aligned SkillRevise: mean per-update authoring over three replicates per transition. Both methods pass all bilateral tests; reductions are relative to dual path, with equal transition weights.

Transition	**Output tokens**		**Author time (s)**		**Edit fraction**	
	Dual path	ConScribe	Dual path	ConScribe	Dual path	ConScribe
Version update	252.3	176.3	13.87	11.72	0.595	0.394
New site	284.0	185.7	13.93	11.19	0.616	0.420
Overall mean	268.2	**181.0**	13.90	**11.46**	0.605	**0.407**
Reduction (%)	**32.5**		**17.6**		**32.8**	

5.4 Adaptation across Hosts and Representations

Across five completed REAL site-family blocks, both methods achieve 7/10 target successes; a sixth fails during source acquisition. ConScribe lowers adaptation cost and edit fraction by 43.2%, with joint success of 2/10 versus 1/10 (Table 5).

Table 5: Host-specific quality and resources. Savings favor ConScribe; quality pairs list ConScribe first and differences subtract the reference. REAL and PolySkill costs cover adaptation; SkillEvolBench costs cover authoring (Appendix D).

Host / reference	Quality comparison	Token saving (%)	Time saving (%)
REAL / dual path	Target: 70% vs. 70%; joint: 20% vs. 10%	8.3 total	11.0
PolySkill / native	Target: -2.1 pp; source: $+4.2$ pp	15.3 total	9.2
SkillEvolBench / scratch	Mean verifier score: -1.32 points	49.1 output	38.0

PolySkill reduces calls by 51.1% and edit fraction by 32.5% across six source blocks; target accuracy decreases while source accuracy increases. SkillEvolBench reduces output and author time in all six executable families, with a lower mean verifier score.

Offline scorers extend coverage to complete eligible Mind2Web and WebLINX candidate indices (8,538 and 4,031 source-plus-target actions) (Deng et al., 2023; Lu et al., 2024). Mind2Web target AUC improves by 15.3% across websites and 16.7% across domains, with positive paired intervals; WebLINX's intervals cross zero. Appendix Table 12 gives all shifts and target-action counts. The original source path retains its predictions.

5.5 Repair Costs across Change Scales

In 504 transitions across 36 generated clusters, local repair improves budget-64 decision success over scratch by 26.6 percentage points (95% CI [15.1, 38.7]). This outcome includes correct rejection of 72 capability-deletion controls alongside 432 reachable tasks. At budget 256, both methods solve all reachable tasks and reject all controls; repair uses 25.1% fewer adaptation steps. Repair thus reduces work even when final outcomes converge.

Chromium crosses $n_h \in \{6, 10\}$ and $k_h \in \{0, 1, 4\}$ in 48 pairs and 16 clusters of same-site updates and cross-site transfers. With fixed decomposition, the deterministic proposer repairs traced failures; scratch builds from target observations and task intent. At $k_h = 4$, repair costs more for $n_h = 6$ (44.0 versus 39.5 interactions), but less for $n_h = 10$ (76.0 versus 81.5): retained skill size shifts the cost balance (Figure 3).

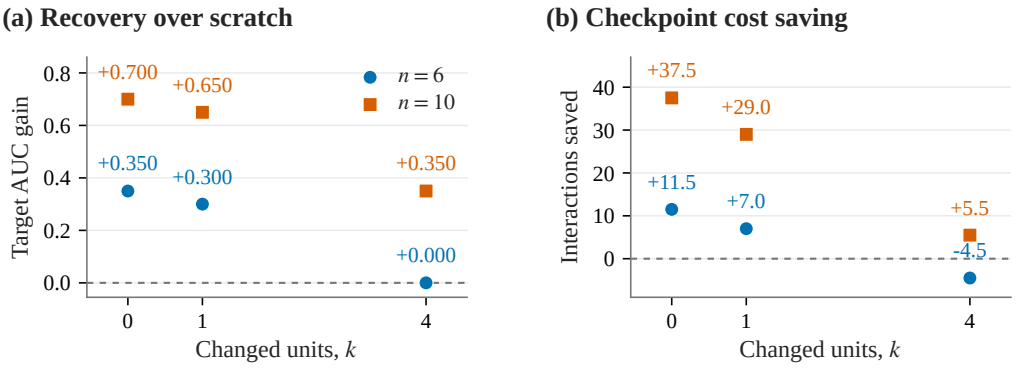

Figure 3: Chromium repair versus scratch: (a) target AUC gain and (b) retrospective checkpoint interactions saved. Points average eight pairs; markers distinguish $n = 6, 10$. Appendix Table 8 gives all conditions.

For recovery burden $Y = 1 - \text{AUC}$, increasing k_h penalizes repair more than scratch ($\Delta_k = 0.09038$, Holm-adjusted $p = 3.05 \times 10^{-5}$); increasing n_h penalizes scratch more than repair ($\Delta_n = 0.08750$, $p = 0.0039$). Overall checkpoint interactions decrease from 60.50 to 46.17 (23.7%). This includes adaptation and route-specific validation through the first held-out target pass or budget exhaustion.

Staged Mind2Web separates proposal and controller costs. Website updates reach 0.438 versus scratch16's 0.417 accuracy, saving 26.0% author time and 41.3% output but using 11.6% more total tokens. With qualification, controller time reaches 1.661× scratch authoring on website transfers and 2.376× on domain transfers; task-transfer quality falls below reference (Appendix Table 21).

6 Conclusion and Future Work

ConScribe carries acquired skills across environment changes by retaining inherited implementation and composing newly generated updates with it. The experiments establish why the update choice matters: composition preserves source behavior at matched target recovery in the controlled local-edit comparison, while aligned Mind2Web improves recovery and measured adaptation time over source-preserving complete generation. SkillRevise achieves the same observed bilateral accuracy with less authoring. These findings support building target capability through updates to existing skills, with qualification guiding continued repair and reconstruction providing a fallback when repair is unsuccessful.

Future work can develop observable estimates of invalidation and update cost for selecting repair scopes before adaptation, and evaluate accumulated residuals over long sequences of environment changes. These directions extend execution-guided adaptation toward continual skill maintenance.

AI USE STATEMENT

In this work, we used generative AI tools to assist with experimental implementation and result analysis, and to help prepare and polish portions of the manuscript. We reviewed all AI-assisted work and take responsibility for the final text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The paper describes the adaptation procedure, comparisons, metrics, and evaluation protocol. Appendices A–F provide task construction and partitions, model and runtime configurations, host-specific implementations, prompts, budgets, qualification rules, cost accounting, per-run results, skill artifacts, and adaptation trajectories.

ETHICS STATEMENT

This work involves no human-subject experiments. The evaluation uses controlled task environments and established web-agent benchmarks. The authors take responsibility for the integrity of the reported experiments and analyses.

REFERENCES

Johannes Bader, Andrew Scott, Michael Pradel, and Satish Chandra. Getafix: Learning to fix bugs automatically. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–27, 2019. doi: 10.1145/3360585.

Berkay Berabi, Jingxuan He, Veselin Raychev, and Martin Vechev. TFix: Learning to fix coding errors with a text-to-text transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 780–791. PMLR, 2021. URL https://proceedings.mlr.press/v139/berabi21a.html.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, pp. 54067–54089, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/ed91353f700d113e5d848c7e04a858b0-Paper-Conference.pdf.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem^{2}Evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 20784–20831. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.952. URL https://aclanthology.org/2026.acl-long.952/.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023. doi: 10.52202/075280-1220.

Linfeng Fan, Yuan Tian, Ziwei Li, and Zhiwu Lu. Skill drift is contract violation: Proactive maintenance for LLM agent skill libraries, 2026. URL https://arxiv.org/abs/2605.10990.

Div Garg, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Prannay Hebbar, Youngchul Joo, Jindong Gu, Charles London, Christian Schroeder de Witt, and Sumeet Motwani. REAL: Benchmarking autonomous agents on deterministic simulations of real websites. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4536. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c63819755591ea972f8570beffca6b1b-Abstract-Datasets_and_Benchmarks_Track.html.

Zhezheng Hao, Hong Wang, Jian Luo, Jianqing Zhang, Yuyan Zhou, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. ReCreate: Reasoning and creating domain agents driven by experience. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp.

31018–31046. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1432. URL https://aclanthology.org/2026.acl-long.1432/.

Sang-Hyun Lee and Seung-Woo Seo. Unsupervised skill discovery for learning shared structures across changing environments. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19185–19199. PMLR, 2023. URL https://proceedings.mlr.press/v202/lee23r.html.

Yingtie Lei, Zhongwei Wan, Jiankun Zhang, Samiul Alam, Zixuan Zhong, Peizhou Huang, Xin Wang, Jingxuan Zhang, Donghao Zhou, Yunta Hsieh, Zhihao Dou, Hui Shen, Yan Xu, Dimitrios Dimitriadis, Tuo Zhang, and Mi Zhang. SkillEvolBench: Benchmarking the evolution from episodic experience to procedural skills. *arXiv preprint arXiv:2605.24117*, 2026. URL https://arxiv.org/abs/2605.24117.

Yuyang Li, Yiran Dou, Jie-Jing Shao, Yueming Lyu, Ivor Tsang, and Haiyan Yin. SkillTracer: Structural failure attribution and refinement of agentic skills in long-horizon web tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. URL https://liyuuuuy.github.io/SkillTracer/.

Yuxuan Liu, Zhaochen Su, Lingyun Xie, Yuhao Zhang, Qing Zong, Jiahe Guo, Zhongwei Xie, Yiyan Ji, Yauwai Yim, Hongyu Luo, Xiyu Ren, Chenyu Ruan, Haoran Li, and Yangqiu Song. SkillRevise: Improving llm-authored agent skills via trace-conditioned skill revision, 2026. URL https://arxiv.org/abs/2606.01139.

Xing Han Lu, Zdeněk Kasner, and Siva Reddy. WebLINX: Real-world website navigation with multi-turn dialogue. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33007–33056, 2024. URL https://proceedings.mlr.press/v235/lu24e.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019.

Michael Matthews, Mikayel Samvelyan, Jack Parker-Holder, Edward Grefenstette, and Tim Rocktäschel. Hierarchical kickstarting for skill transfer in reinforcement learning. In *Proceedings of the 1st Conference on Lifelong Learning Agents*, volume 199 of *Proceedings of Machine Learning Research*, pp. 856–874. PMLR, 2022. URL https://proceedings.mlr.press/v199/matthews22a.html.

Martin Monperrus. Automatic software repair: A bibliography. *ACM Computing Surveys*, 51(1): 1–24, 2018. doi: 10.1145/3105906.

Hoang Duong Thien Nguyen, Dawei Qi, Abhik Roychoudhury, and Satish Chandra. SemFix: Program repair via semantic analysis. In *Proceedings of the 35th International Conference on Software Engineering*, pp. 772–781. IEEE, 2013. doi: 10.1109/ICSE.2013.6606623.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2021. URL https://proceedings.mlr.press/v155/pertsch21a.html.

Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J. Lim. Demonstration-guided reinforcement learning with learned skills. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 729–739. PMLR, 2022. URL https://proceedings.mlr.press/v164/pertsch22a.html.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, Silvio Savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *International Conference on Learning Representations*, pp. 55589–55607, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5b175f9e93873e3a10a6ce43dbb82e05-Paper-Conference.pdf.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6922–6939. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.462. URL https://aclanthology.org/2023.findings-emnlp.462/.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. URL https://mlanthology.org/tmlr/2024/wang2024tmlr-voyager/.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 1529–1550. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.69. URL https://aclanthology.org/2026.acl-long.69/.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51177–51191. PMLR, 2024b. URL https://proceedings.mlr.press/v235/wang24az.html.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

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

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. PolySkill: Learning generalizable skills through polymorphic abstraction for continual learning. In *International Conference on Learning Representations*, pp. 140298–140326, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e350897ed832f9cad6f2e223e7acad6d-Paper-Conference.pdf.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *International Conference on Learning Representations*, pp. 40097–40125, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/af31604708f3e44b4de9fdfa6dcaa9d1-Paper-Conference.pdf.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, pp. 34040–34077, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5492ecbce4439401798dcd2c90be94cd-Paper-Conference.pdf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

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 *International Conference on Learning Representations*, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

A APPENDIX

The evaluation separates controlled mechanism tests, public-data component studies, and model-backed host adaptation. Table 7 identifies their sampling units. A model replicate, an action prediction, and a new environment transition describe different sources of variation; we keep their counts separate throughout the supplementary results.

Table 6: Evaluation protocols and their roles. Each study compares methods within its own protocol; source checks and returned artifacts are specified separately from blind scoring.

Study	Evidence role / evaluated object	Checks and adaptation path
Aligned Mind2Web	Public-task adaptation; element selection	Bilateral qualification through reconstruction; last candidates scored with qualification status.
REAL / PolySkill	Browser rollouts / library-based element selection	Target-driven revision or reconstruction; source performance evaluated afterward.
Chromium	Composition and cost crossover; executable units	Failed-unit feedback; post-hoc source replay and checkpoint analysis.
Aligned SkillRevise	Update granularity; text lookup rules	Bilateral qualification; all runs stop at the initial update.
Staged Mind2Web	Routing costs; selected candidates	Local and widened checks; scratch fallback scored without final online qualification.
Public-index	Coverage; linear candidate scores	Fixed evidence budgets; source predictions evaluated separately.

Table 7: Evaluation units and execution settings. Counts describe the reported records separately by study and sampling unit. All listed LLM comparisons use the same model and low reasoning effort within a study.

Study	Units and coverage	Execution
Controlled	504 transitions; 36 base-site/length clusters; 432 reachable, 72 deletion controls	Deterministic simulator
Chromium	48 transitions; two task families, two transfer types; 16 clusters for the k_h contrast	Browser; deterministic proposer
Public index	Mind2Web: 8,538 actions, 1,330 task groups; WebLINX: 4,031 actions, 534 demonstration groups	Linear candidate scorer
SkillEvolBench	Six families; two verifier-scored tasks each	GPT-5.4
Mind2Web controller	Three source-disjoint replicates, three splits; nine routed transitions	GPT-5.6-sol
SkillRevise (X4)	Two controlled shifts, three model replicates; 18 aligned runs; earlier 18-run protocol reported separately	GPT-5.6-luna
Mind2Web aligned	12 source blocks, 24 transitions, 27 websites; nine website-dependency clusters	GPT-5.5
REAL	Five completed site families, two transitions each; 30 method cells	GPT-5.5
PolySkill	Six source blocks, two shifts each; 24 method cells	GPT-5.5

Model and runtime configuration. The model identifiers are `gpt-5.4` for SkillEvolBench, `gpt-5.6-sol` for staged Mind2Web, `gpt-5.6-luna` for X4, and `gpt-5.5` for REAL, PolySkill, and aligned Mind2Web. Each comparison fixes its identifier and uses low reasoning effort. Launchers disable web search and do not set a sampling temperature, top-p, or generation seed; the seeds below control data selection, not deterministic LLM output. X4 disables fallback to another model. Its host LLM timeout is 600 seconds and its batched execution timeout is 300 seconds, with one model call per evaluated split. The shared isolated-call launcher used by the public text-skill studies and SkillEvolBench has a 900-second timeout.

The Chromium study runs under Python 3.12.3 on Linux/WSL2. It uses Playwright's default browser with a 1,500-ms action timeout. REAL uses the recorded Playwright headless-shell build 1161 and a twelve-step rollout limit. LLM inference is remote; reported wall time is observed client-side latency rather than local accelerator time. The source-aware SkillRevise host is revision

25569d8; REAL and PolySkill use revisions 97a245d and fff8807, respectively. Host-specific attempt limits and qualification thresholds are specified in Appendix B.

A.1 Controlled and Browser Transitions

Controlled construction. The sealed controlled test crosses 12 base-site seeds (3101–3112), three skill lengths (8, 12, 16), seven transition families, and two transfer types. The families are no-impact changes, surface-token changes, path rebinding, control substitution, new required steps, workflow reordering, and capability deletion. Transfer types are same-site version updates and functionally similar new sites. Two host representations instantiate these transitions; they do not double the environment count. Each transition supplies two target calibration tasks, four target test tasks, and four source-retention tasks. Decision success scores recovery on reachable transitions and correct rejection on capability deletions. Budgets are $\{0, 16, 32, 64, 128, 256\}$ environment transitions. The test analysis resamples the 36 base-site/length clusters with 10,000 bootstrap draws (seed 20260902). The 72 deletion controls remain in this decision-success denominator; reachable-target recovery uses 432 instead.

Chromium construction and separation. The held-out grid crosses contact creation and ticket updating, same-site updates and functionally matched new-site transfers, $n_h \in \{6, 10\}$, $k_h \in \{0, 1, 4\}$, and two mutation seeds. This yields 48 transitions, five methods, and nine saved budgets $\{0, 1, 2, 3, 4, 5, 6, 8, 10\}$. Each saved candidate receives two held-out target tests and two held-out source tests. The run contains 2,160 method–transition–budget cells and 10,288 browser episodes. Adaptation first runs against a calibration-only server. That server is closed before the test server is created; held-out outcomes cannot alter proposals or calibration stopping.

For target recovery a_j at budget b_j, we compute

$$\mathrm{AUC} = \frac{1}{b_J - b_0} \sum_{j=1}^{J} (b_j - b_{j-1}) \frac{a_j + a_{j-1}}{2}. \tag{9}$$

The changed-support contrast uses 16 clusters fixing task family, transfer type, n_h, and mutation seed. The skill-size contrast uses eight strata fixing family, transfer type, and seed. Exact sign-flip tests and Holm correction yield the reported interaction tests. Here n_h and k_h refer to the fixed browser-unit representation and serve as offline analysis axes; the updater receives execution feedback.

Table 8 reports the cell means plotted in Figure 3.

Table 8: Chromium repair–reconstruction comparison, averaging eight pairs per row. Positive differences favor repair. Interaction savings use the retrospective checkpoint cost defined in Appendix D.1.

n_h	k_h/n_h	Δ Target AUC	Interactions saved
6	0.000	+0.350	+11.5
6	0.167	+0.300	+7.0
6	0.667	0.000	−4.5
10	0.000	+0.700	+37.5
10	0.100	+0.650	+29.0
10	0.400	+0.350	+5.5

A.2 Public Candidate Indices and Model-Backed Tasks

Public-index component study. Mind2Web uses its task split as the source distribution and its website and domain splits as targets. WebLINX uses its validation split as source and its web and geography splits as targets. For each of five seeds (20260904–20260908), the partitioner selects 128 support actions per split and reserves their entire task or demonstration groups from evaluation. All remaining indexed actions are evaluated; the counts in Table 7 include both support and evaluation partitions. A missing gold candidate prevents using an action for support but counts as an incorrect prediction at evaluation. Evidence budgets are $\{0, 1, 2, 4, 8, 16, 32, 64, 128\}$, using nested prefixes of

the same support selection and three training passes. Mind2Web reports task-group-macro accuracy; WebLINX reports action-micro accuracy. Both use Equation 9, normalized by 128. The paired intervals quantify selection-seed variation on these fixed archives.

Table 9: Actual public-index partitions: ranges across the five selection seeds. Entire support groups are withheld from evaluation, so the support/evaluation action counts need not sum to the indexed archive size.

Dataset	Split	Support actions	Support groups	Eval. actions	Eval. groups
Mind2Web	test_task	128	15–18	1,719–1,736	228–231
Mind2Web	test_website	128	19–22	1,082–1,100	154–157
Mind2Web	test_domain	128	24–27	5,282–5,292	881–884
WebLINX	validation	128	11–20	706–721	77–86
WebLINX	test_web	128	20–28	693–701	124–132
WebLINX	test_geo	128	14–20	2,218–2,222	265–271

SkillEvolBench families. The six-family component study uses safe refactoring, response validation, key alignment, cross-format migration, citation verification, and thread-action extraction. For each family, an acquired artifact is adapted once and then applied to two fixed tasks with the official verifier, shared evidence, and fixed task order. No-skill execution, static reuse, scratch, whole revision, and residual repair are evaluated. The score is the verifier's normalized task score, while source retention here means exact inherited-content retention, not source execution accuracy. Table 20 reports both task scores and the separate authoring costs.

Staged Mind2Web controller. The source skill is authored from eight source examples. Each target split provides eight initial support examples, four qualification-A examples, four qualification-B examples, and sixteen blind target examples. Sixteen blind source examples are reused across the three target splits of a replicate. Source groups are disjoint across the three replicates; adaptation and blind groups are separated globally by the sampling procedure (seed 20260907). Nine routed evaluations produce 144 target and 144 source prediction records, with repeated source items across splits. The broader saved artifact study contains 2,016 blind prediction cells. These are prediction counts, not additional independent transitions.

SkillRevise source-aware comparison. X4 uses two controlled opaque-payload lookup shifts (Table 11). A context contains distractor slots and an environment code; the executable rule selects the payload to return. Each shift has two labeled target training items, two target validation items, two source validation items, and four blind items per side. Three model replicates reuse these shifts and rotate strategy order. Each method produces 24 blind predictions per side across two environment changes. Joint pass is evaluated per run and requires all eight blind answers to be correct. The aligned E1 study repeats this matrix with common task-specific information and rule requirements. All eighteen runs stop at B1; their 144 blind predictions correspond to sixteen distinct items repeated across methods and model runs. The earlier X4 study is retained separately in Table 15.

Aligned public Mind2Web extension. The E3 protocol predefines twelve source blocks and two target transitions per block: ten same-domain cross-website blocks and two cross-domain blocks, drawn from original Mind2Web records. Source and target support, qualification, and blind tasks are group-disjoint. Each source supplies eight support, four qualification, and sixteen blind actions; each target supplies eight initial support actions, three four-action qualification sets, and sixteen blind actions. Widening uses twelve support examples and reconstruction uses sixteen. Both methods retain the source path and use the same public website identifier for dispatch. Both receive the same source information and environment-rule requirements; the target update is either a complete target skill or a residual.

All twelve blocks, twenty-four transitions, and forty-eight method cells are complete. They cover 1,200 original actions in 208 disjoint task groups across 27 websites, forming nine connected website-dependency clusters. Each method produces 384 final target predictions. The 38 target actions without a gold element in the candidate index remain in the denominator and count as errors. Source equality follows from the shared source path. The protocol qualifies candidates on source

and target cases, including the final reconstruction stage. Target qualification requires three correct answers out of four; source qualification requires accuracy at least equal to source prevalidation. It preserves the final candidate and qualification status when the budget ends. Blind outcomes never select the returned candidate. At the final checkpoint, 23 residual candidates and all 24 dual-path candidates qualify; all candidates remain in the blind evaluation. Both methods stop at B1/B2/B3 on 16/7/1 transitions. These counts describe the terminal update stage. Online attempts count individual calls along each transition's adaptation path, including authoring, diagnosis, and qualification; tokens and wall time sum their reported usage and measured durations, respectively.

Table 10: Completed aligned public Mind2Web study: residual versus dual path. Accuracy rows are absolute; gains name their comparator, other differences are residual minus dual, and savings are $100(1 - C_R/C_D)$. Transition estimates are averaged within blocks and then equally across blocks. Cost estimates require two measured targets per block. Counts are measured transitions/blocks/website clusters. Reported intervals use 2,000 paired cluster-bootstrap draws (seed 20260908). Quality intervals sample website clusters with replacement, retain all constituent source blocks, and use the 2.5th and 97.5th percentiles of the equally weighted block means.

Measure	Estimate	95% CI	Counts
Static-reuse target accuracy (%)	67.97	[58.41, 77.46]	24/12/9
B1 target accuracy (%)	75.78	[67.18, 84.09]	24/12/9
B1 gain over static reuse (pp)	+7.81	[3.13, 12.50]	24/12/9
B1 author output saving (%)	37.57	[33.00, 41.00]	22/11/8
B1 author total-token saving (%)	33.89	[18.48, 43.59]	22/11/8
B1 author wall saving (%)	31.17	[25.59, 36.35]	24/12/9
Last-candidate target accuracy difference (pp)	+9.90	[6.82, 12.28]	24/12/9
Final source accuracy difference (pp)	0.00	[0.00, 0.00]	24/12/9
Final edit-fraction difference	−0.098	[−0.118, −0.082]	24/12/9
Online total-token saving (%)	27.25	[20.57, 30.60]	12/6/4
Online wall saving (%)	28.83	[25.35, 32.55]	22/11/9
Online attempt saving (%)	23.94	[20.92, 27.72]	24/12/9

Table 10 reports coverage separately for each metric. Complete-cost estimates require both target transitions in a source block to have complete measurements for that metric. The ledger contains 270 physical attempts and 256 successful responses; fourteen attempts lack token usage, and one also lacks wall time. Partial costs remain recorded; unknown usage is excluded from complete-cost ratios, not filled with zero. Each deployment includes source qualification and its target B0 check; source acquisition and blind scoring are separate. Savings, including the 23.94% attempt saving, average transition-level paired percentages within and across source blocks. Aggregate cost ratios are reported separately for aligned SkillRevise (Appendix D.1). Twenty-two transitions match or exceed dual path's final blind target accuracy.

Adaptation quality by transfer type. The last-candidate accuracy advantage is 10.63 percentage points for same-domain cross-website transfers (20 transitions, ten source blocks) and 6.25 points for cross-domain transfers (four transitions, two source blocks).

Table 11: The two SkillRevise lookup transitions. Slot values vary across items; the environment-to-slot mapping is the skill being adapted.

Shift	Source rule	Target rule
Version	`IV-A17`: token after `[M7]`	`IV-Q42`: token after `[P2]`
New site	`N4X`: `slot-r7`	`Q9Z`: `cell-d8`

REAL and PolySkill. The five completed REAL families are DashDish, GoMail, GoCalendar, UDriver, and NetworkIn, with two target transitions per family. The sixth planned family, StayNB, fails source acquisition before comparative adaptation begins; it is not counted among the ten completed transitions. Each method receives at most three target rollouts of twelve steps each. Source execution follows final target adaptation. PolySkill uses six source blocks from the Mind2Web candidate index (seed 20260906). Each block has four source induction examples and two target shifts, each with four initial examples, two qualification-A examples, two qualification-B examples, and four blind target examples. The same four blind source examples are evaluated for both shifts. Its

raw field `joint_success` is $M = \min(a_{\text{target}}, a_{\text{source}})$, not a binary joint pass or the frequency of paired correct answers. We label it M in Table 17 and report the two accuracies separately.

A.3 Public-Index Transfer Results

Table 12 evaluates offline action-prediction scorers on complete eligible target partitions over nine evidence budgets. Support and evaluation groups are disjoint, and missing positive candidates count as errors. Mind2Web uses 1,330 source-plus-target task groups and WebLINX uses 534 demonstration groups. The intervals describe paired variation across five selection seeds; residuals preserve the original source path and its predictions.

Table 12: Public-index adaptation over nine evidence budgets. Gain is $100(\text{AUC}_R/\text{AUC}_S - 1)$ before rounding; bold denotes a positive paired AUC-difference interval across five selection seeds. Actions count target examples.

Dataset	Shift	Actions	Target AUC		Gain (%)	ΔAUC [95% CI]
			Scratch	Repair		
Mind2Web	Website	1,237	0.117	0.135	**+15.3**	+0.018 [0.007, 0.028]
Mind2Web	Domain	5,428	0.105	0.123	**+16.7**	+0.018 [0.003, 0.032]
WebLINX	Website	829	0.284	0.281	−0.8	−0.002 [−0.037, 0.032]
WebLINX	Geography	2,351	0.268	0.290	+8.0	+0.022 [−0.037, 0.080]

B Host-Specific Implementations

Algorithm 1 specifies the adaptation decision procedure. The host-specific protocols below describe its experimental instantiations, including proposal operators, stopping rules, and where validation enters the update loop.

Budget accounting in the decision procedure. B and B_{used} use the host's declared budget unit. Every BUDGETED operation receives the remaining allowance $b = B - B_{\text{used}}$, charges its consumed cost c to $B_{\text{used}} \leftarrow B_{\text{used}} + c$, and returns its result with a completion flag d. The host enforces $0 \le c \le b$ using the operation's call, step, token, or time allowance. An operation that cannot complete within that allowance returns $d = 0$; zero remaining allowance starts no call. Qualification charges its constituent executions and checks, including failed attempts. An incomplete qualification does not authorize acceptance. The ledger covers the initial check, edits, qualifications, and reconstruction.

B.1 Executable Browser Units

A browser skill is an ordered list of units. Each unit records an action, an accessible role/name locator, an optional argument binding, and an ordered list of alternative locators. The executor returns the failing unit and the visible controls. Repair focuses on that unit; when absent, it uses a previously touched unit and then the final inherited unit. The deterministic proposer filters visible, enabled controls for action compatibility and ranks alternatives by lexical overlap, action match, and deterministic name/DOM-order tie breaks. Rejected candidates are remembered and not proposed again.

Ordinary local editing may replace a locator. Residual repair retains the source locator first and adds a target alternative; an optional-unit edit is also available. If task verification already succeeds despite an execution failure, the optional edit takes priority. At most four local candidates are proposed over a trajectory. Exhausting this limit or the eligible local candidates invokes the global proposer; two consecutive empty-control observations trigger rejection. This browser instantiation uses local-to-global escalation.

Calibration stops after an execution succeeds and its target verifier passes; later budget snapshots repeat that state. Source replay measures retained behavior in the subsequent held-out measurement phase.

Algorithm 1 ConScribe adaptation in host h

Require: Source skill s_0, evidence $\mathcal{Z}_t$, qualification sets $\mathcal{Q}_t$, $\mathcal{Q}_s$, scopes $\{\Omega_\ell, J_\ell\}_{\ell=0}^{L}$, budget B

$B_{\text{used}} \leftarrow 0$
$((q_t, q_s, \xi), d) \leftarrow \textsc{Budgeted}(\text{Qualify}(s_0, \mathcal{Q}_t, \mathcal{Q}_s))$
if $\neg d$ **then**
  return $(\texttt{REJECT}, \bot)$
end if
if $q_t \wedge q_s$ **then**
  return $(\texttt{KEEP}, s_0)$
end if
$s \leftarrow s_0$; $\Omega_0 \leftarrow \textsc{Localize}(\xi)$
for $\ell = 0, \ldots, L$ **do**
  for $j = 1, \ldots, J_\ell$ while $B_{\text{used}} < B$ **do**
    $(r, d) \leftarrow \textsc{Budgeted}(\text{edit}_h(s, \xi, \Omega_\ell))$
    if $\neg d$ **then**
      return $(\texttt{REJECT}, \bot)$
    end if
    $s' \leftarrow s \oplus_h r$
    $((q_t, q_s, \xi), d) \leftarrow \textsc{Budgeted}(\text{Qualify}(s', \mathcal{Q}_t, \mathcal{Q}_s))$
    if $\neg d$ **then**
      return $(\texttt{REJECT}, \bot)$
    end if
    if $q_t \wedge q_s$ **then**
      return $(\texttt{REPAIR}, s')$
    end if
    if q_s **then**
      $s \leftarrow s'$
    end if
  end for
  if $B_{\text{used}} \geq B$ **then**
    return $(\texttt{REJECT}, \bot)$
  end if
  if $\ell < L$ **then**
    $\Omega_{\ell+1} \leftarrow \textsc{Widen}(\Omega_\ell, \xi)$
  end if
end for
$(\tilde{s}, d) \leftarrow \textsc{Budgeted}(\text{scratch}_h(\mathcal{Z}_t))$
if $\neg d$ **then**
  return $(\texttt{REJECT}, \bot)$
end if
$((q_t, q_s, \xi), d) \leftarrow \textsc{Budgeted}(\text{Qualify}(\tilde{s}, \mathcal{Q}_t, \mathcal{Q}_s))$
if $d \wedge q_t \wedge q_s$ **then**
  return $(\texttt{REBUILD}, \tilde{s})$
else
  return $(\texttt{REJECT}, \bot)$
end if

The implementation selects source-preserving local proposals from the browser host's candidate set. A source-first fallback places the target alternative after the inherited locator sequence, which the executor tries in order.

Listing 1: Browser-host selection of local residual proposals.

```
def _choose_preserving_local(
    self, feedback: AdaptationFeedback
) -> bool:
    self._rejected.add(_artifact_key(self.current.artifact))
    focus = feedback.execution.failed_unit
    if focus is None and self.current.touched_units:
        focus = self.current.touched_units[0]
    if focus is None and self.source.units:
        focus = self.source.units[-1].unit_id
    proposals = self.proposer.local_neighbors(
        self.current.artifact,
        _latest_observation(feedback.execution),
        focus,
    )
    allowed = {"source_first_fallback", "make_optional"}
    preserving = [
        proposal for proposal in proposals
        if proposal.kind in allowed
    ]
    if (
        feedback.verifier_success
        and not feedback.execution.success
    ):
        preserving.sort(
            key=lambda proposal: (
                proposal.kind != "make_optional"
            )
        )
    else:
        preserving.sort(
            key=lambda proposal: (
                proposal.kind != "source_first_fallback"
            )
        )
    for proposal in preserving:
        if _artifact_key(proposal.artifact) in self._rejected:
            continue
        self.current = proposal
        self._decision = MethodDecision(
            "repair", proposal.artifact
        )
        return True
    return False
```

B.2 Residuals in Candidate Scorers and Text Skills

Candidate scorer. The public-index experiment uses a linear candidate scorer with lexical overlap, action/tag compatibility, and token features. A mistake-driven update increases gold-candidate feature weights and decreases predicted candidate weights. Scratch starts with zero learned weights. Whole revision updates source weights w_s into w_t; residual repair stores $\delta = w_t - w_s$, applying $w_s + \delta$ to target cases and w_s to source cases. Thus target predictions agree between whole revision and residual repair by construction. This comparison isolates update storage and source prediction stability.

Mind2Web controller. One local text residual is authored from eight target examples and qualified on four fresh examples. Three correct answers are required. On failure, one widened residual uses twelve examples and qualification feedback, followed by a second four-example check. A second failure selects scratch with sixteen examples. This last candidate proceeds to blind evaluation without a third online qualification call. Source accuracy is measured blindly, not used by these target qualification gates. Widening increases evidence and rewrites the target residual; it is not an intervention on a measured browser-unit count k_h.

In the aligned public Mind2Web study, the deployed target text is built by composing the inherited skill with the authored residual at B1 and B2. At B3, the target path uses the reconstructed skill.

Listing 2: Target-skill composition in the aligned Mind2Web runner.

```
def compose(source, update, arm, stage):
    if arm == "conscribe" and stage < 3:
        return (
            source.rstrip()
            + "\n\n## Target-conditioned residual\n\n"
            + update.strip()
            + "\n"
        )
    return update
```

SkillRevise. All three source-aware arms evaluate both environments at B0 and B1, and, if needed, after one further revision at B2. Selection stops at the first jointly qualified checkpoint; otherwise it returns the final candidate for blind scoring. Native whole revision modifies one skill. Dual-path regeneration keeps the source skill and authors a separate target skill. ConScribe anchors the source fields and authors a target-conditioned constraint, replacing its previous residual on revision. The host supplies diagnosis, revision, and BM25 principle retrieval (at most four principles). Dual path and residual composition receive the environment identifier for dispatch. For aligned E1, each arm receives the same task-based BM25 retrieval and shared action-map requirements. The source/target context is common, and all updates use the same Markdown schema with distinct structural contracts. B0 bilateral execution precedes diagnosis and authoring; B1 bilateral execution determines whether one further diagnosis/revision at B2 is needed. All observed runs stop at B1.

REAL. After initial target authoring, a failed rollout supplies feedback for a revision. A second failed rollout triggers a further feedback revision for the baselines and target reconstruction for ConScribe, which retains the source path. The third rollout ends the target attempt sequence. Source replay evaluates the resulting artifact afterward. This study uses target-based stopping and a separately measured source objective, whereas X4 uses bilateral stopping.

PolySkill. Native adaptation induces abstract and concrete skill entries from individual examples. ConScribe authors a target repair entry while retaining the source library. Both methods check qualification A and then B if needed. With two examples per check and the 0.75 threshold, both answers must be correct. Native adaptation incorporates each failed check's examples into additional induction; ConScribe widens its residual after A and reconstructs the target entry after B. Source evaluation is blind. Both arms retain the source library, isolating update granularity and resource use.

C Prompt and Baseline Specifications

The comparison distinguishes empty-start construction, full-artifact revision, and residual output. Empty-start authors generate a target artifact without copying source implementation; dual path stores the original separately. Whole revision receives the inherited artifact. Residual authors receive that artifact plus target evidence and return an update that is composed with the retained source. Validation data and final evaluation data remain separate under the protocols above.

The original aligned SkillRevise study includes three arms; Table 4 focuses on residual updates versus dual-path regeneration. All arms receive the same source artifact, labeled target evidence,

task-based retrieved principles, and requirements to preserve supported environment-conditioned rules. Whole revision writes a complete joint skill, dual path writes a complete target skill alongside the source, and ConScribe writes a target-conditioned residual. All use the host's trace serialization and diagnosis parser, with the same diagnosis and revision opportunities and bilateral stopping rule. The saved-prompt audit checks common context and requirements across all eighteen runs.

Author input and output contracts. The aligned SkillRevise prompt concatenates the task instruction, complete source skill, task-retrieved principles, common requirements, current candidate, bilateral diagnosis, and an arm-specific structural instruction. Both arms return the same Markdown sections: purpose, applicability, procedure, and constraints. Dual path is instructed to output a complete target-environment skill; the residual author outputs the changes needed under the target guard. Composition retains the source text and attaches the residual under that guard. Thus output granularity is specified by the method, while input length also depends on the rendered candidate and diagnosis. Total author cost sums these input and output components.

X4 (Table 15) compares complete authoring configurations: native authoring requests generic task-family procedures, while residual authoring requests an environment-conditioned rule. Aligned E1 holds task information and rule requirements fixed to compare update structure. Appendix E illustrates an X4 rule.

C.1 Aligned Mind2Web Author Prompt

Both arms received the following task instruction. The workspace supplied `SOURCE_SKILL.md`, `SOURCE_SUPPORT.md`, `TARGET_SUPPORT.md`, and the initial and source qualification diagnostics. Later stages also supplied the previous update, composed target skill, and qualification feedback.

Listing 3: Shared aligned Mind2Web author instruction.

```
Infer and preserve reusable environment-conditioned rules
    supported by the evidence,
including concrete selectors, attributes, semantic roles and
    operation constraints when supported.
Do not replace supported rules with vague discovery advice. Do not
     memorize case IDs, option IDs,
or individual answer values. SOURCE_SKILL.md and SOURCE_SUPPORT.md
     are common prior information.
TARGET_SUPPORT.md contains exactly the currently acquired target
    evidence. CURRENT_UPDATE.md,
CURRENT_DEPLOYED.md and FEEDBACK.json, when present, contain your
    own previous candidate and its
observed qualification feedback. Diagnose the mismatch from these
    same materials before updating.
The source path remains unchanged; the target path must restore
    target behavior. Public website
identity selects the path. No blind examples or outcomes are
    available during adaptation.
Gold=null in acquired feedback means the correct element is
    missing from the supplied top-50
candidates; do not invent an option or memorize the missing case.
    Treat it as retrieval failure.
Write only UPDATE.md in this workspace, with Purpose, Environment
    condition, Procedure and
Constraints sections. Do not access the web, other paths, other
    agents or external services.
```

The instruction was followed by `Source website:` and `Target website:` with their public site names, then `Stage B1; acquired target examples: 8.` at the initial update. At B2 and B3, the corresponding counts were 12 and 16. Each call appended exactly one of these authoring instructions:

Dual path, B1–B3: Write a complete target skill, including all target procedures needed for deployment.

ConScribe B1: Write only the target-conditioned residual that updates the inherited source skill. The runner appends it to the unchanged source skill for the target path.

ConScribe B2: Write one replacement target-conditioned residual, broadening the previous residual as supported by the expanded evidence. It replaces, rather than stacks on, the previous residual.

ConScribe B3: Reconstruct a complete target skill as fallback using all currently acquired evidence. It replaces the target residual path; the source path remains unchanged.

C.2 Composition, Activation, and Repair Scope

Representable corrections. For a fixed working artifact s and support Ω, let $\mathcal{R}_h(s, \Omega)$ contain the host's permitted residuals, including their activation conditions. Define $\mathcal{C}_h(s, \Omega) = \{s \oplus_h r : r \in \mathcal{R}_h(s, \Omega)\}$. A qualified repair exists in this family exactly when some member satisfies the required target and source checks. Finding that member additionally depends on the proposer and the available budget. Local replacement changes implementation inside Ω freely; residual composition imposes Equation 7 there. Dual path preserves the original artifact and requests a complete target implementation. These definitions separate the composition constraint from access to source information.

Storage, execution, and generation. These are three distinct properties of an adapted artifact. Storage preservation concerns which inherited components remain represented, as in Equation 7. Execution reuse concerns which of those components the target computation invokes or reads. For executable skills, a component participates through a control or data dependency on the target result, rather than through its presence in an inactive source branch. An execution trace can establish invocation; whether a particular component is necessary for success additionally depends on the effect of removing or replacing it while holding the other components fixed.

Generation size concerns the newly authored update in a fixed host format. A base-relative update leaves retained implementation to the composer; a self-contained target implementation supplies that content anew. For example, a locator update can name one alternative while the composed browser skill executes the inherited surrounding actions. A target guard that dispatches to an entirely new procedure provides source storage without target-execution reuse of that source procedure. Thus component inclusion, execution reuse, and update size are separate comparison axes.

Text-host residuals provide environment-conditioned instructions alongside the inherited skill. Their authoring contract and deployed text establish which content is supplied and generated; behavioral dependence on inherited instructions is a counterfactual question about execution with those instructions changed or removed. The aligned comparisons measure adaptation quality, author output, and edit magnitude under shared source access. The browser mechanism comparison tests source behavior under local replacement versus retaining the inherited alternatives. Together these comparisons distinguish the update contract from its behavioral and cost consequences.

Activation and compatibility. A missing locator and a semantically obsolete locator require different activation conditions. Lookup-failure activation can select an alternative in the first case. In the second, the successful old match prevents that alternative from running; recovery requires a condition that selects the new behavior on the conflicting target states, or reconstruction. Consider two contexts requiring different branch choices but identical under all inputs visible to the host's dispatcher. Any deterministic dispatcher must choose the same branch in both; a randomized dispatcher has the same branch distribution. Reliable separation therefore requires a distinguishing observation or execution history. Environment identifiers provide such information in the text-host protocols; the browser fallback uses locator availability. Task verification evaluates the result of the selected behavior and feeds unsuccessful candidates back to the proposal search.

Sufficient condition for source preservation. Consider executable dispatch in a fixed environment with the same initial state, exogenous inputs, and random draws. Suppose the composer leaves the inherited path intact, every new branch is inactive on source executions, and dispatch adds no

side effects. The composed and inherited artifacts then have identical source traces. At the first step they select the same inherited action; equal transitions give the same next state. Induction repeats this argument at every step. This establishes behavioral preservation for isolated executable paths. Text-host guards are instructions interpreted by the model, whose behavioral retention is checked through source replay.

Scope expansion. If widening retains the smaller scope's permitted edits, then for fixed s, $\Omega_a \subseteq \Omega_b$ implies $\mathcal{C}_h(s, \Omega_a) \subseteq \mathcal{C}_h(s, \Omega_b)$. Every smaller-scope residual remains admissible, establishing the inclusion. Consequently, the best achievable target score subject to the same source constraint cannot decrease when such a feasible family expands. This is a property of the candidate family; the finite-budget proposer selects the candidates that qualification actually evaluates. In staged text adaptation, additional evidence also changes proposal generation. Its outcomes therefore depend jointly on evidence, generated candidates, and qualification.

C.3 Cost of Repair Before Reconstruction

One-stage identity. Fix an adaptation state and one cost unit (tokens, time, or interactions). Let a bounded repair stage include all its proposals and qualification calls, with expected cost c_R and acceptance probability p. Conditional on its failure, let fallback, including its qualification, have expected cost c_F. The law of total expectation gives

$$\mathbb{E}[C_{R\to F}] = c_R + (1-p)c_F, \qquad G = c_S - c_R - (1-p)c_F, \tag{10}$$

where c_S is the expected cost of immediate reconstruction under the same accounting. Correlation between repair cost and acceptance is allowed: c_R is the unconditional stage expectation, while c_F is conditional on failure; when $p = 1$, the fallback contribution is zero. For a comparison at equal required quality and completion probability, $G > 0$ favors trying repair first. In the special case $c_F = c_S > 0$, this becomes $c_R < pc_S$: successful early repairs must avoid enough reconstruction work to pay for all repair attempts. Failed-case selection or reused evidence can change c_F, so the general expression retains it separately.

Multiple stages and widening. For stages $1, \ldots, M$, including fallback, let A_ℓ denote reaching stage ℓ, and let c_ℓ be its expected cost conditional on being reached. Setting unreachable contributions to zero,

$$\mathbb{E}[C] = \sum_{\ell=1}^{M} \Pr(A_\ell) c_\ell. \tag{11}$$

This follows by expressing total cost as the sum of reached-stage costs and applying linearity of expectation; it requires no independence between stages. At a reached state, an additional widened stage with cost c_W, acceptance probability p_W, and subsequent conditional fallback cost $c_{F,W}$ has cost advantage $c_S - c_W - (1-p_W)c_{F,W}$ over reconstructing immediately. Previous costs are sunk at this decision. Attempt limits cap the number of proposals spent at a scope; their execution and verification costs enter c_W. Acceptance frequencies and stage costs are the quantities needed to assess this tradeoff. Algorithm 1 implements bounded attempts and qualification-driven escalation. The stage-cost identities describe the resulting cost tradeoff.

D Cost Accounting and Detailed Results

D.1 Checkpoint Costs and Executed Route Costs

Chromium checkpoint diagnostic. For a completed calibration trajectory, let b^* be the first saved checkpoint passing both held-out target tests, or the final budget when none passes. The reported interaction diagnostic is

$$C_{\text{diag}} = C_{\text{adapt}}(b^*) + C_{\text{target test}}(b^*) + \mathbf{1}[\text{target pass}] C_{\text{source check}}(b^*). \tag{12}$$

The adaptation scalar counts target calibration episodes, source probes, and page actions; a test scalar counts verifier calls and page actions. Keep/repair checks the selected artifact on source tasks; the rebuild route charges source validation of the separately frozen original skill. The reported source-pass metric nevertheless executes the adapted artifact, including for scratch. These are distinct

measurements. Tests of other saved checkpoints do not enter C_{diag}. The quantity is retrospectively selected from held-out measurements, not the cost of an online policy performing all these checks and stopping at their first success. The 23.7% reduction and the $(n_h, k_h) = (6, 4)$ cost crossover use this diagnostic.

LLM cost boundaries. Token totals are input plus output tokens; cached input is included, not subtracted, and reasoning-output tokens are not added a second time. Wall time sums measured calls along the specified path. X4 includes diagnosis, authoring, revisions, and target/source qualification from B0 through the selected or final checkpoint. REAL includes authoring and attempted target rollouts; source replay is reported separately. PolySkill includes induction or patch calls and the qualification calls actually traversed. Shared source acquisition and final blind scoring are outside these adaptation totals. Unsuccessful adaptation runs remain in the corresponding averages.

Aggregation of cost savings. Let $C_{R,i}$ and $C_{D,i} > 0$ be paired repair and comparator costs in the same unit. SkillRevise (X3, X4, and aligned E1), REAL, PolySkill, and the six-family SkillEvolBench summary use $100(1 - \sum_i C_{R,i} / \sum_i C_{D,i})$, the reduction in aggregate cost. For equal numbers of paired runs this equals the ratio-of-means reduction. E2 averages the paired percentages $100(1 - C_{R,i}/C_{D,i})$, with equal weight for its three routes within each of three blocks. E3 first averages these paired percentages over eligible transitions within each represented source block, then weights the represented blocks equally. Its complete-cost estimates use the eligibility rules in Appendix A.2. Staged-controller cost ratios follow the matched-quality authoring reference below. All calculations use unrounded costs; input-plus-output tokens and output tokens remain separate metrics.

For staged Mind2Web (E2), local, widened, and fallback routes contain two, four, and five calls, respectively. Other saved experimental candidates are not charged to an untraversed route. AC@MTS compares to the cheapest measured scratch authoring option (eight or sixteen support examples) whose mean target accuracy meets the scratch16 reference; it does not interpolate between checkpoints. This is an authoring reference, not a scratch controller with symmetric qualification calls. The domain $4.024\times$ value evaluates the full five-call path across saved replicates; $2.376\times$ is the actual routed mean, which reaches fallback in one replicate. Table 25 gives that replicate's actual, rather than averaged, path.

Completion and recovery records. All 504 controlled transitions enter decision success, including deletion controls. The completed Chromium grid includes every planned cell. REAL reports five completed blocks out of six planned; StayNB's source-acquisition failure precedes arm comparison. PolySkill completes all six planned blocks. The Mind2Web screen records 139 model attempts and 138 valid calls, including recovery of one malformed blind-scoring response; blind scoring is outside adaptation cost for all arms. X4 reports all eighteen cells from the `luna_formal02` rerun following correction of a scorer-format mismatch. For SkillEvolBench refactoring, a corrected scratch prompt names the refactoring family and replaces that arm's authoring call and two evaluations. Its scores are 87.50 and 100.00, with 56.9 seconds and 1,980 output tokens; other arms retain their original records.

Aligned SkillRevise accounting. E1 separates authoring from complete online execution. Table 4 reports means over three authoring calls per transition and method. Across both transitions, author input/output totals over six author calls are 23,204/1,609 for dual path and 18,973/1,086 for ConScribe. Total author tokens decrease from 24,813 to 20,059, a 19.2% reduction relative to dual path. The 4,754-token difference comprises 4,231 input tokens and 523 output tokens. Output tokens measure generated update length; input tokens measure the model's received context. Their sum measures author-call token use. Normalized edit fraction uses character matching between source text and the complete deployed representation, including dual-path routing.

Table 13 compares the two source-preserving methods: observed online tokens decrease from 966,640 to 768,450 (20.5%), and wall time from 1,552.66 to 1,486.10 seconds (4.3%). These online totals include validation and failed attempts, whereas the authoring totals above isolate skill-update generation.

The B1 protocol specifies eight calls per deployment, including two shared no-skill calls, for every arm. The original three-arm ledger contains 161 physical calls and counts shared reference calls

Table 13: Aligned E1 online ledger, summed over six runs per arm. Each deployment includes the same no-skill reference measurement, diagnosis, authoring, bilateral checks, and failed attempts; final blind scoring is separate. Observed tokens sum reported usage; unreported usage makes these totals lower bounds. Wall time includes all measured attempts.

Method	Calls	Observed total tokens	Wall (s)	Timeouts
Dual path	49	966,640	1,552.66	1
ConScribe	51	768,450	1,486.10	2

once; the independent-deployment totals in Table 13 allocate those reference costs to each arm. All 144 blind predictions were recomputed from saved outputs. The earlier X4 costs use their original accounting and are reported separately below.

D.2 Per-Run and Per-Transition Results

Tables 15–19 expose the units behind the main means. X4 repeats fixed shifts; REAL groups transitions by site; PolySkill groups them by source block; Mind2Web groups its splits within source-disjoint replicates. Costs and accuracy are reported together, retaining failed outcomes and resource trade-offs. Table 14 first reports aligned E1, which supplies the main SkillRevise comparison.

Table 14: Aligned E1: eighteen completed runs. All stop at B1 and have target EM, source EM, and joint pass equal to 1. Edit is normalized deployed-text change; Calls counts online calls. Aggregate online costs appear in Table 13.

(a) Version

Rep.	Method	Edit	Calls
0	Whole	0.713	8
0	Dual	0.566	8
0	ConScribe	0.360	10
1	Whole	0.850	8
1	Dual	0.606	8
1	ConScribe	0.386	8
2	Whole	0.678	8
2	Dual	0.612	8
2	ConScribe	0.434	8

(b) New site

Rep.	Method	Edit	Calls
0	Whole	0.631	9
0	Dual	0.606	9
0	ConScribe	0.410	9
1	Whole	0.714	8
1	Dual	0.602	8
1	ConScribe	0.440	8
2	Whole	0.594	8
2	Dual	0.640	8
2	ConScribe	0.409	8

Table 15: Earlier X4 source-aware comparison with different authoring objectives, not the aligned E1 main result: all 18 completed runs. T and S are blind exact-match fractions (four items per side per run); J is all-tests joint pass. Costs include adaptation and qualification, including unsuccessful runs; K denotes thousands of total tokens.

Rep.	Shift	Method	T	S	J	Calls	Tokens (K)	Wall (s)
0	New site	ConScribe	1.00	1.00	1	6	103.8	104.7
0	New site	Dual	0.00	1.00	0	9	158.4	174.9
0	New site	Whole	0.00	0.00	0	10	172.8	207.9
0	Version	ConScribe	1.00	1.00	1	6	103.7	107.0
0	Version	Dual	0.00	1.00	0	9	158.5	178.5
0	Version	Whole	0.00	0.00	0	10	172.8	221.4
1	New site	ConScribe	1.00	1.00	1	6	104.1	107.4
1	New site	Dual	1.00	1.00	1	9	159.4	181.3
1	New site	Whole	0.00	1.00	0	10	187.5	211.4
1	Version	ConScribe	1.00	1.00	1	6	103.8	104.4
1	Version	Dual	1.00	1.00	1	9	158.3	177.1
1	Version	Whole	0.00	0.00	0	10	172.5	217.0

Rep.	Shift	Method	T	S	J	Calls	Tokens (K)	Wall (s)
2	New site	ConScribe	1.00	1.00	1	10	169.0	173.2
2	New site	Dual	0.00	1.00	0	9	170.2	186.9
2	New site	Whole	0.00	0.00	0	10	173.4	204.5
2	Version	ConScribe	1.00	1.00	1	6	103.6	108.9
2	Version	Dual	0.00	1.00	0	9	158.7	178.6
2	Version	Whole	0.00	0.50	0	10	183.8	223.1

Table 16: REAL: all completed method–transition cells. T, S, and J are binary target, source, and joint outcomes. Adaptation tokens and wall time include target rollouts and failed attempts; the final column separately reports source evaluation time, which is not included in the adaptation column.

Transition	Method	T	S	J	Calls	Tokens (K)	Adapt. (s)	Source (s)
dashdish-17	ConScribe	1	1	1	13	1,313.2	577.0	455.6
dashdish-17	Dual	1	0	0	39	5,346.5	2,126.9	438.5
dashdish-17	Whole	1	1	1	36	4,672.5	1,927.4	545.9
dashdish-26	ConScribe	1	0	0	10	1,226.8	447.0	485.6
dashdish-26	Dual	1	0	0	11	1,662.3	590.1	543.7
dashdish-26	Whole	1	1	1	11	1,131.3	493.4	501.1
gocalendar-13	ConScribe	0	1	0	31	4,807.5	1,584.1	291.5
gocalendar-13	Dual	0	1	0	39	6,196.8	1,931.9	275.2
gocalendar-13	Whole	0	1	0	37	6,386.7	1,949.1	295.3
gocalendar-23	ConScribe	1	1	1	19	2,994.5	903.6	263.0
gocalendar-23	Dual	1	1	1	4	767.4	212.7	305.0
gocalendar-23	Whole	1	1	1	8	1,151.7	388.4	213.5
gomail-1	ConScribe	0	1	0	39	8,811.8	2,821.9	542.5
gomail-1	Dual	0	1	0	39	7,841.5	2,519.4	344.8
gomail-1	Whole	0	0	0	39	9,933.7	3,103.4	579.4
gomail-23	ConScribe	0	0	0	34	5,636.1	1,474.9	497.1
gomail-23	Dual	0	0	0	35	5,206.4	1,506.2	486.2
gomail-23	Whole	0	0	0	34	5,068.3	1,634.5	483.7
networkin-10	ConScribe	1	0	0	13	1,103.1	438.8	393.3
networkin-10	Dual	1	0	0	13	1,269.8	452.8	431.3
networkin-10	Whole	1	0	0	11	1,094.8	406.0	351.7
networkin-27	ConScribe	1	0	0	21	2,263.6	873.1	475.0
networkin-27	Dual	1	0	0	10	999.3	403.6	353.5
networkin-27	Whole	1	0	0	12	1,156.4	512.9	325.0
udriver-20	ConScribe	1	0	0	9	754.9	294.2	449.2
udriver-20	Dual	1	0	0	18	1,562.0	596.5	424.7
udriver-20	Whole	1	0	0	9	850.5	337.2	487.6
udriver-9	ConScribe	1	0	0	13	1,160.8	413.9	335.7
udriver-9	Dual	1	0	0	20	1,938.3	697.5	433.0
udriver-9	Whole	1	0	0	20	1,939.8	734.0	380.6

Table 17: PolySkill: all 24 completed cells, nested within six source blocks. T and S each average four blind items. M is $\min(T, S)$, the field called `joint_success` in the raw export; it is not an all-tests joint pass. Source examples are shared across the two shifts within a block. Aggregate costs are reported in Table 18.

Block	Shift	Method	T	S	M	Calls
b00	Domain	ConScribe	0.50	0.25	0.25	2
b00	Domain	Native	0.25	0.25	0.25	5
b00	Website	ConScribe	0.00	0.25	0.00	5
b00	Website	Native	0.25	0.25	0.25	10

Block	Shift	Method	T	S	M	Calls
b01	Domain	ConScribe	0.50	0.50	0.50	2
b01	Domain	Native	0.50	0.75	0.50	5
b01	Website	ConScribe	0.25	0.50	0.25	5
b01	Website	Native	0.25	0.50	0.25	10
b02	Domain	ConScribe	0.25	0.50	0.25	2
b02	Domain	Native	0.25	0.50	0.25	5
b02	Website	ConScribe	0.75	0.50	0.50	2
b02	Website	Native	0.75	0.25	0.25	5
b03	Domain	ConScribe	0.00	0.75	0.00	4
b03	Domain	Native	0.00	0.50	0.00	10
b03	Website	ConScribe	0.25	0.50	0.25	5
b03	Website	Native	0.50	0.75	0.50	5
b04	Domain	ConScribe	0.50	0.75	0.50	5
b04	Domain	Native	0.50	0.75	0.50	10
b04	Website	ConScribe	0.50	1.00	0.50	5
b04	Website	Native	0.75	0.50	0.50	10
b05	Domain	ConScribe	0.25	0.50	0.25	2
b05	Domain	Native	0.25	0.50	0.25	5
b05	Website	ConScribe	0.75	0.50	0.50	5
b05	Website	Native	0.50	0.50	0.50	10

Table 18: PolySkill aggregate cost comparison across six source blocks and two shifts per block. Reductions are computed from the aggregate totals, relative to native adaptation.

Method	Total tokens	Wall (s)
Native	5,565,500	3,045.10
ConScribe	4,714,500	2,765.76
Reduction (%)	15.3	9.2

Table 19: Mind2Web: all nine pre-evaluation route decisions. T/S are controller blind accuracies (16 items per side per route); Scratch T uses sixteen support examples. Q_A is local qualification accuracy; Q_B is shown only when the actual route executes widening. Route costs include all traversed proposal and qualification calls.

Rep.	Shift	Route	Q_A	Q_B	T	S	Scratch T	Calls	Wall (s)
r0	Task	Local	3/4	–	0.625	0.500	0.688	2	146.8
r0	Website	Local	3/4	–	0.438	0.500	0.500	2	124.0
r0	Domain	Rebuild	2/4	2/4	0.375	0.688	0.375	5	428.8
r1	Task	Local	3/4	–	0.250	0.375	0.438	2	137.2
r1	Website	Widen	2/4	3/4	0.500	0.500	0.438	4	243.6
r1	Domain	Local	4/4	–	0.562	0.438	0.500	2	117.7
r2	Domain	Local	3/4	–	0.438	0.500	0.500	2	156.2
r2	Task	Rebuild	0/4	2/4	0.375	0.438	0.375	5	432.0
r2	Website	Local	3/4	–	0.312	0.438	0.312	2	141.2

Host comparison protocols. X3 compares residual adaptation with target-only scratch: mean wall time is 152.98 versus 197.35 seconds (22.5% lower), and mean edit fraction is approximately 0.227 versus 0.917 (75.3% lower). X4 uses source-aware dual path and reports 117.6 versus 179.5 seconds and 114.7K versus 160.6K tokens (Table 15). Aligned E1 matches authoring information and rule requirements across methods (Table 4). Each protocol retains its own model, stopping rule, and cost estimates.

D.3 Staged Adaptation Cost Diagnostic

Across three group-disjoint replicates and three Mind2Web splits, local and widened candidates receive target qualification. Two failed checks trigger scratch16 without final online qualification;

Table 20: SkillEvolBench: six English task families. Scores are means of the two verifier-scored tasks, expressed on a 0–100 scale. Time and output tokens measure the single authoring call per family and method, not task execution. S denotes scratch and R residual repair.

Family	Score S	Score R	Wall S	Wall R	Output S	Output R
Refactoring	93.75	87.50	56.9	35.8	1,980	1,055
Response validation	95.84	95.84	40.5	23.9	1,376	633
Key alignment	78.33	80.83	65.4	33.9	2,228	947
Format migration	100.00	100.00	43.6	35.3	1,620	1,070
Citation verification	87.50	83.34	50.0	34.7	1,709	876
Thread actions	65.00	65.00	77.2	43.2	2,727	1,339

blind scoring follows routing. The cost reference is the cheapest measured scratch authoring option attaining scratch16 target quality. Table 21 reports cumulative controller wall time against this authoring reference; the executed-route breakdown appears in Table 19.

Table 21: Staged Mind2Web cost diagnostic: target accuracy and cumulative controller wall time relative to the scratch authoring reference. Controller cost includes authoring and qualification; scratch cost is authoring only. Both exclude blind evaluation. *Unmatched* denotes failure to reach the target reference.

Shift	**Target accuracy**		**Controller wall / scratch**
	Controller	Scratch16	
Task	0.417	0.500	Unmatched
Website	0.417	0.417	$1.661\times$
Domain	0.458	0.458	$2.376\times$

D.4 Support Similarity in the Model-Backed Routes

E2 reuses all nine staged Mind2Web routes from three source-disjoint sampling blocks; it adds no model runs. Let S_s and S_t be the sets of structural signatures from eight source and eight initial target support examples. We express source-relative structural similarity as $\rho = 1 - |S_t \setminus S_s|/|S_t \cup S_s|$, the complement of support novelty. Larger values indicate a smaller proportion of target-specific signatures in the union. The measure is computed before considering candidate outcomes and describes support examples rather than skill-unit change support k_h/n_h.

Table 22: E2 proposal-stage comparisons. Each source block and each of its three routes have equal weight. Quality differences are local minus scratch in percentage points; savings are means of paired percentages. Total tokens include input and output for authoring, not controller validation.

Comparison	Target Δ	Source Δ	Token saving (%)	Wall saving (%)
Local8 versus scratch8	-2.08	-6.94	1.37	27.17
Local8 versus scratch16	-2.78	-4.17	22.16	35.83

At equal eight-example evidence, the descriptive Pearson correlation between similarity ρ and local target advantage is $+0.094$, with within-block signs $-, -, +$. Its correlation with author-time saving is -0.733, negative in all three blocks: higher support similarity is associated with smaller author-time savings in these routes. These are descriptive support-level associations; adaptation decisions use execution feedback.

E Skill Artifacts Before and After Adaptation

E.1 A Locator Fallback That Retains the Original Action

Consider contact-creation transition `p_e4d6d0ba4cb307847919`. The inherited skill has six executable units. Its first target execution completes five page actions, then fails at `submit`: the accessible locator `Save Contact` matches no control. The observed target instead exposes `Create`

`Contact`. The saved repair changes only the submit unit, retaining the original locator before the new alternative (Table 23). The argument bindings and all other units remain unchanged.

Table 23: The six-unit contact-creation skill before and after one residual update. The arrow inside the submit cell denotes ordered locator fallback, not two sequential clicks. The other five unit objects are unchanged.

Unit	Inherited locator	Adapted locator(s)
gate_1_check	Confirm workspace gate 1	Unchanged
gate_1_continue	Continue gate 1	Unchanged
open	New Contact	Unchanged
primary	Contact name	Unchanged
secondary	Channel	Unchanged
submit	Save Contact	Save Contact → Create Contact

The executor tries the alternatives in order and executes the matching action. In the source environment, the retained locator remains usable; in the changed target, its absence activates the new alternative. Replacing `Save Contact` outright would remove that source alternative. This is the concrete within-unit difference between residual composition and destructive local replacement. Source replay verifies the retained behavior.

E.2 An Environment-Conditioned Rule in SkillRevise

In X4 replicate 0's version-update skill, the inherited procedure reads:

> When `Interface-Code: IV-A17`, return the token after `[M7]` and ignore `[P2]`.

The saved adapted artifact retains that procedure and adds the following constraint (verbatim, including the repeated "When"):

> When `Interface-Code: IV-Q42` is present, When Interface-Code is IV-Q42, return the token after [P2] in `<answer>...</answer>`; otherwise do not apply this residual.

The output convention, original guard, and source mapping remain in the same skill. This example shows what the text residual contains: a target-conditioned mapping rather than a rewritten generic procedure. Its independent validation and blind outcomes appear in the corresponding row of Table 15.

F Adaptation Trajectories

F.1 Local Repair and Subsequent Replay

For the contact case, calibration episode 1 produces the missing-locator trace and one edit. Episode 2 executes the repaired artifact successfully and passes the task verifier, after which calibration is absorbing. The saved budget-1 candidate already passes its later held-out target and source tests (Table 24). This distinction explains why a retrospective checkpoint can precede the successful calibration execution that stops the actual trajectory.

Table 24: Saved checkpoints for the contact-creation case. Target/source held-out tests are performed after adaptation and do not guide its updates. The first passing saved target checkpoint is budget 1; calibration execution itself becomes absorbing after episode 2.

Budget	Target correct	Source correct	Adapt. episodes	Adapt. actions	Edited units
0	0/2	2/2	0	0	0
1	2/2	2/2	1	5	1
2	2/2	2/2	2	11	1

F.2 Failed Qualification and Reconstruction

The Mind2Web cross-domain route in replicate r0 demonstrates the complete fallback sequence. The local and widened candidates each answer two of four qualification cases correctly, below the three-answer threshold. The controller therefore returns the sixteen-example scratch candidate. Table 25 accounts for every traversed call, totaling 428.8 seconds. Its final target blind accuracy is 0.375, equal to the scratch16 candidate selected by this route. Earlier proposals contribute to the cost even though they are not deployed.

Table 25: An actual cross-domain fallback trajectory (replicate r0). The acceptance threshold is three correct answers out of four. Costs accumulate only along the executed route; final blind evaluation is separate.

Step	Operation	Outcome	Wall (s)	Cum. (s)	Cum. tokens (K)
1	Local proposal (8 examples)	–	68.6	68.6	278.0
2	Qualification A	2/4: continue	75.8	144.4	588.1
3	Widened proposal (12 examples)	–	71.2	215.6	812.5
4	Qualification B	2/4: rebuild	74.3	289.9	1,030.0
5	Scratch fallback (16 examples)	Return fallback	138.9	428.8	1,846.4

These cases were selected after inspecting the records to illustrate one local fallback edit and one actually traversed reconstruction route. Table 19 reports the complete set of routes.