SKILLZIP: EXPERIENCE-GATED MACRO ACTIONS FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents routinely rediscover the same multi-step procedures: search a repository, open the relevant implementation and tests, then format, test, and inspect a patch. Existing experience memories preserve such procedures as text that the model must interpret again. We introduce SKILLZIP, a training-free compiler that uses successful-trace support to select parameterized phase templates and admits each executable action only after sandbox replay establishes its observable contract. We evaluate the resulting action interface on 20 temporally held-out software tasks from 10 public repositories with two agent models and four matched conditions. Across 40 model–task pairs, SKILLZIP matches observed no-memory completion at 24/40 (paired 95% CI: -7.5 to $+7.5$ points) while reducing model-mediated tool decisions by **23.1%** (repository-clustered 95% CI: 14.1–31.2% fewer; exact sign-flip $p = .0059$). The pooled effect combines a 40.6% Luna reduction with a neutral 1.1% DeepSeek increase; primary cost is likewise neutral ($+0.4\%$). An oracle exposing the admitted pair is identical after selection, so this comparison isolates the value of the selected executable interface rather than an oracle advantage for admission. It removes **31.2%** of decisions versus an equally sourced textual workflow and **27.6%** versus retrieved action sequences; against retrieval it also reduces API cost by **10.9%** and wall time by **29.1%**. Fresh regeneration reduces decisions by **29.6%**, model cost by **18.4%**, and raises success from 14/20 to 16/20. Across three paired Luna generations, the decision reduction is **35.4%** (two-way repository–generation 95% CI: 19.6–45.0% fewer). Successful experience can therefore become a smaller interface for future decisions, not another block of context.

1 INTRODUCTION

A coding agent receives a request, searches for relevant symbols, opens source and tests, edits a file, runs a focused test, runs the full suite, and checks the resulting diff. On its next task it often makes all of these decisions again. Current experiential-learning methods help by retrieving past trajectories, lessons, or natural-language workflows (Zhao et al., 2023; Wang et al., 2025; Cao et al., 2026; Yan et al., 2026). Yet the retrieved procedure remains text: the model must repeatedly choose, spell, and observe each primitive action. A memory can tell an agent *what sequence worked* while leaving its executable action vocabulary unchanged.

We study a complementary question: *can successful experience change the granularity of the agent's future decisions?* Temporal abstraction has long made sequential decision-making more efficient by packaging repeated primitive behavior into options (Sutton et al., 1999). Tool- using language agents provide the ingredients for learning such abstractions without policy training: their successful traces are executable demonstrations, their tools expose typed arguments, and their environments can replay candidate procedures.

SKILLZIP compiles those ingredients into a small action library. It maps heterogeneous tool events into stable phase evidence, measures recurring productions using a description-length score, and selects supported members of an author-provided tool-semantic template library. The templates make task-dependent arguments explicit as typed slots. Finally, a sandbox replay gate checks each action's observable contract before it is shown to a held-out agent. In our implementation, seven successful source trajectories yield 65 repeated productions and two admitted actions: INSPECT

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.

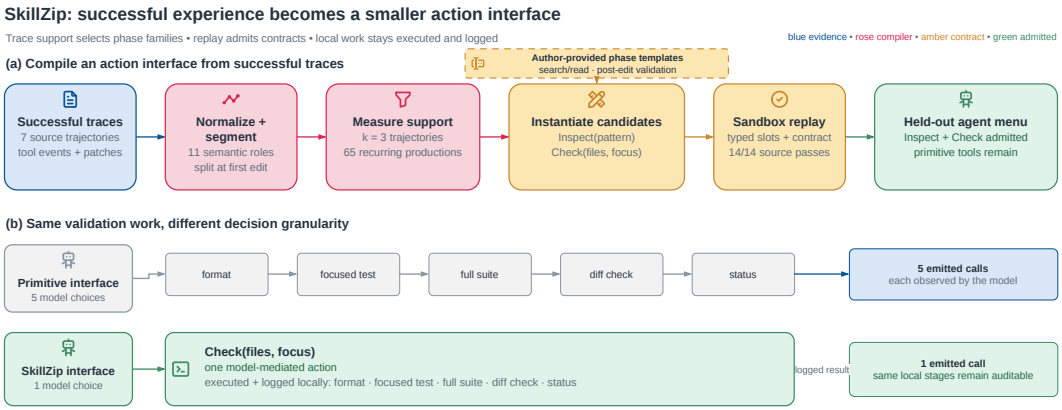


Figure 1: SKILLZIP uses successful behavior to select and check an action interface. Unlike contextual memory, the admitted procedure collapses several model choices into one executable decision.

gathers a relevant implementation/test/build slice, while CHECK formats changed files, executes focused and full tests, checks the diff, and reports status. One invocation of either macro is one model-mediated tool decision; its deterministic local subcommands are still executed and logged, but do not require additional model inference.

The distinction from contextual memory is experimentally consequential. We construct 10 source tasks and 20 later held-out tasks from 10 public Go repositories. Every benchmark parent passes its existing tests, while developer tests from the future fix fail on that parent. We then run a fully crossed matrix of two models and four conditions—no memory, a textual workflow distilled from the source traces, a deterministically retrieved successful action sequence, and SKILLZIP—for 160 trajectories. SKILLZIP is invoked in every one of its 40 primary runs. It matches no-memory pooled success (24/40) while lowering mean model-mediated decisions from 21.2 to 16.3, a 23.1% reduction. This pooled effect combines a 40.6% Luna reduction with a neutral 1.1% DeepSeek increase, and primary API cost is neutral ($+0.4\%$). The reduction is larger against textual workflow memory (31.2%) and remains 27.6% against retrieved action sequences, where it additionally saves 10.9% API cost and 29.1% wall time. Fresh regeneration on the alphabetically first held-out task in each repository reproduces the effect: 29.6% fewer decisions, 18.4% lower model cost, and 16/20 rather than 14/20 successful patches. A third Luna generation yields the same direction: across three paired generations, decisions fall 35.4% even after resampling both repositories and generations. DeepSeek varies more, from 4.4% to 27.0% savings across two draws, while the pooled two-generation effect remains 16.4%.

Our contributions are:

- **An experience-gated action interface.** We introduce a template-backed compiler that combines trace-derived phase support, typed macro instantiation, descriptive action-grammar analysis, and replay-gated admission without policy training.

- **A temporally separated evaluation.** We release a reproducible suite with 10 source tasks, 20 held-out tasks, 230 main-study trajectories, complete traces, patches, hidden-test evaluation, and repository- and generation-aware inference.

- **Evidence that granularity matters.** Across matched agents, executable procedures remove 23.1–31.2% of model-mediated decisions relative to three contextual controls; repeated generations and practical cost/time savings establish that the effect extends beyond counting.

2 FROM EXPERIENCE TO AN ACTION LIBRARY

2.1 PROBLEM FORMULATION

Let a tool-using agent produce a trajectory $\tau = ((z_1, a_1, o_1), \ldots, (z_T, a_T, o_T))$, where z_t is a model state, $a_t = (u_t, x_t)$ invokes tool u_t with arguments x_t, and o_t is the observation. A *model-mediated decision* is one externally emitted tool call a_t. Local operations that a tool executes internally are measured separately. Given successful source trajectories $\mathcal{D}_{\text{src}}$, our objective is to produce an action library $\mathcal{M}$ that reduces future model decisions on disjoint tasks while preserving task completion.

A macro $m = (\pi, \theta, C)$ contains a primitive program π, typed parameters θ, and an observable contract C. Calling $m(\theta)$ is atomic from the policy's perspective but transparent to the evaluator: its primitive subcommands, outputs, runtime, and exit state are logged. This is temporal abstraction at the agent–environment interface, not parallel execution or hidden compute.

2.2 MEASURE RECURRING PRODUCTIONS

Tool names and arguments vary even when their roles agree. We map each event to one of 11 semantic tokens: PLAN, GLOB, GREP, READ, EDIT, FORMAT, FOCUSEDTEST, FULLTEST, DIFF, STATUS, and SHELL. This retains action order while abstracting paths, search strings, and test names.

For every contiguous production p of length $2 \leq |p| \leq 8$, let $n(p)$ be its total occurrence count and $s(p)$ the number of distinct successful trajectories that contain it. For descriptive grammar analysis, we retain $s(p) \geq 3$ and rank productions by the simple description-length savings

$$G(p) = (|p| - 1)n(p) - |p|. \tag{1}$$

The first term counts primitive symbols removed by reuse; the second charges once for defining the production. This grammar-mining view follows the principle that useful sequence structure should compress its demonstrations (Nevill-Manning & Witten, 1997). It corroborates both repeated local subsequences and longer phase signatures; it analyzes the selected families but does not synthesize their executable bodies.

2.3 SELECT AND INSTANTIATE TYPED ACTION TEMPLATES

Overlapping productions often describe one operational phase. The reference compiler starts from a small author-provided library of two repository-agent templates. Each template declares required event roles, an executable body, typed slots, and an observable contract. The search/read template is supported when a pre-edit phase contains content/file search plus reads; the post-edit template is supported when test and repository-state operations co-occur. A family must appear in at least three distinct successful trajectories. Source traces therefore decide whether a template earns instantiation and supply its support/coverage evidence; the template supplies its body and slot types.

Template-backed compilation. Input: successful traces $\mathcal{D}_{src}$, template library $\mathcal{T}$, minimum support $k = 3$. (1) Normalize and segment each trace around the first edit. (2) Measure template-role matches and enumerate the descriptive grammar in Eq. 1. (3) For each $q \in \mathcal{T}$, discard q if fewer than k traces contain its required roles; otherwise instantiate its declared typed slots and executable body. (4) Replay the candidate against every successful source workspace and add it to $\mathcal{M}$ only if all declared contracts pass. Return the admitted library $\mathcal{M}$ and grammar audit.

Table 1 shows the library induced in our study. Seven successful source traces contain 77 pre-edit search/read calls and 45 post-edit shell calls. The induced INSPECT parameterizes the search regex and include glob; CHECK parameterizes changed paths and an optional focused-test regex. The template library supplies safe deterministic operators such as "read matching files" and "run full tests," while source support and replay decide admission. This separation makes compilation lightweight and auditable without asking another language model to synthesize shell code.

2.4 REPLAY-GATED ADMISSION

Compression alone can package an unreliable sequence. We therefore replay each candidate on every successful source workspace before exposing it to a held-out agent. INSPECT must return

Table 1: The induced action library. Coverage counts primitive calls in the seven successful source trajectories; one macro invocation becomes one model-mediated decision.

Action	Typed execution	Trace support	Calls covered
INSPECT	`pattern: regex`, `include: glob`; search, return source/test context and build metadata	7/7	77
CHECK	`files: path[]`, `focus: regex?`; format, focused/full test, diff check, status	7/7	45

Table 2: Evaluation scale. The primary matrix crosses every held-out task, model, and contextual condition. Repeats use one mechanically preselected task per repository.

Stage	Repositories	Tasks	Models × conditions	Trajectories
Source collection	10	10	1×1	10
Primary held-out	10	20	2×4	160
Fresh repeat	10	10	2×2	40
Third generation	10	10	1×2	20
Formal total	10	30 unique		**230**

at least one implementation file, one test file, and build metadata while leaving the workspace tree digest unchanged. For CHECK, we apply the source agent's saved patch in a clean parent, pass the changed paths to the macro, and require every applicable stage—formatting, full tests, diff check, and working-tree status—to be present, finish without timeout, and exit successfully. All seven source replays satisfy both contracts. The admission process never reads held-out tasks or future developer implementations. This establishes compatibility with the declared contract; end-to-end held-out results determine whether the action is useful.

3 EXPERIMENTAL DESIGN

3.1 FRESH TEMPORAL TASK SUITE

We build a fresh suite from 10 public Go repositories covering retry logic, environment decoding, formatting, structured logging, map decoding, merging, schema conversion, SQL binding, query construction, and WebSockets. For each repository we select one earlier *source* change and two later *held-out* changes. A task begins at the parent of its target commit. Existing tests pass at that parent; developer tests extracted from the target commit fail until the missing behavior is implemented. Source commits precede held-out target commits, and agents cannot inspect history, remotes, online patches, target implementations, or developer commits. Appendix A lists every task and immutable revision.

Ten Luna source trajectories cost $0.42; seven solve their developer tests and enter compilation. The other three remain in the release but contribute neither productions nor action definitions. This produces a useful transfer slice: held-out tasks from three repositories have no successful source trajectory in the compiler input.

3.2 AGENTS AND MATCHED CONDITIONS

We use GPT-5.6 Luna and DeepSeek V4 Flash through the same OpenCode runner, workspace adapter, primitive tools, task prompt, six-minute limit, retry policy, and test evaluator. Sampling follows each hosted model's production setting. Run metadata records the exact provider identifier, prompt, timestamps, status, token accounting, and cost. The four primary conditions are:

- **No memory:** ordinary primitive tools and the task request.
- **Text workflow:** two concise instructions distilled from the same successful source traces: inspect matching implementation/tests/build metadata; after editing, format, test, diff, and check status. The agent executes every operation with primitive tools.
- **Retrieved actions:** a compact successful source action sequence: up to 32 calls, with command and basename cues retained and edit contents redacted. Retrieval uses the same repository when available (14/20 tasks), otherwise frozen prompt-cosine similarity over the seven successful sources; ties are lexical and no target outcome enters selection.
- **SKILLZIP:** the same procedural content exposed as INSPECT and CHECK commands, including their typed arguments and contracts. Primitive tools remain available.

Thus the central comparison changes execution granularity, not access to source experience. All conditions receive identical task text and ordinary tools; SKILLZIP adds two locally executed tool entries. We preserve timeouts, empty outputs, failed patches, and unused tool opportunities.

3.3 METRICS AND INFERENCE

Completion requires that the agent patch apply cleanly to a fresh parent and pass the full repository suite after future developer tests are overlaid. The primary efficiency metric is the number of *model-mediated tool decisions*, counted directly from emitted tool-use events. This includes each macro call once and does not count deterministic commands internal to the macro as additional model decisions. A single model step can emit more than one tool call, so calls measure interface choices rather than inference round trips. We separately report model steps, wall time, timeout count, API cost, and deterministic local stages executed inside macros.

Comparisons are paired by model and task. Confidence intervals resample the 10 repositories with replacement, retaining both tasks, models, and conditions within a sampled repository (20,000 bootstrap draws). We additionally report a two-sided exact sign-flip test over repository-level mean paired differences. This treats repositories, rather than nested calls, as independent units.

4 RESULTS

4.1 RQ1: EXECUTABLE EXPERIENCE REMOVES MODEL DECISIONS

SKILLZIP reduces the pooled mean from 21.2 to 16.3 model-mediated tool decisions, or **23.1%** (95% CI: 14.1–31.2% fewer; exact repository sign-flip $p = .0059$), while observed completion is 24/40 in both conditions (paired difference 0.0 points; 95% CI: -7.5 to $+7.5$). Table 3 gives the full matrix. The action library is actively used: every SKILLZIP run calls INSPECT, 38/40 call CHECK, and the 40 runs emit 114 macro calls in total.

The primary pooled step and cost effects are reported separately: steps fall 9.1% (95% CI: -19.5 to $+3.4\%$), while cost changes by $+0.4\%$ (95% CI: -7.2 to $+8.9\%$). Thus the no-memory contrast resolves interface-level compression; the stronger contextual controls determine whether that compression translates into resources.

The equally sourced textual workflow reaches 23/40 completion and takes 23.7 decisions on average. Exposing the matched procedure as actions rather than instructions therefore reduces decisions by **31.2%** (95% CI: 25.5–36.8% fewer, $p = .0020$) and raises completion by one task. It also reduces mean model steps by 13.2% and total model cost by 10.9% relative to the textual condition. Successful experience becomes useful at execution time when the agent can invoke its repeated phase directly.

Retrieved action-sequence context is a stronger contextual baseline: it reaches 22/40 completion, averages 22.53 decisions, costs $1.498, and takes 168.9 seconds on average. SKILLZIP reaches 24/40 at 16.30 decisions, $1.334, and 119.7 seconds. The paired reductions are **27.6%** in decisions (95% CI: 20.9–34.9%, $p = .0020$), **10.9%** in API cost (95% CI: 2.6–16.5%, $p = .0313$), and **29.1%** in wall time (95% CI: 18.0–38.7%, $p = .0059$). Executable granularity therefore produces practical savings relative to both forms of experience context. Two retrieved-action runs end in non-timeout provider errors after 12 and 13 emitted calls. On the 38 complete runner pairs, completion is $22/38$ for both conditions, while the efficiency effects strengthen to **29.6% fewer** decisions, **13.0% lower**

Table 3: Primary 160-trajectory matrix. Calls are model-mediated tool decisions; wall time is the median, while paired effects in text use means. Cost is total API cost over 20 tasks per model or 40 pooled pairs. Bold marks the best efficiency at equal or higher success within each block.

Model	Condition	Success	Calls↓	Steps↓	Cost ($)↓	Median s
Luna	No memory	13/20	24.65	15.95	0.810	75.0
	Text workflow	13/20	27.05	15.60	0.904	76.0
	Retrieved actions	13/20	24.35	15.15	0.921	87.0
	SKILLZIP	**14/20**	**14.65**	**12.55**	**0.756**	**60.5**
DeepSeek	No memory	**11/20**	**17.75**	**17.75**	**0.519**	159.0
	Text workflow	10/20	20.35	19.70	0.593	189.5
	Retrieved actions	9/20	20.70	18.00	0.577	232.0
	SKILLZIP	10/20	17.95	18.10	0.578	**158.5**
Pooled	No memory	**24/40**	21.20	16.85	1.328	108.5
	Text workflow	23/40	23.70	17.65	1.498	92.0
	Retrieved actions	22/40	22.53	16.58	1.498	136.5
	SKILLZIP	**24/40**	**16.30**	**15.33**	1.334	**80.0**

Table 4: Fresh repeated generation on one frozen task per repository. Totals make the action-level savings visible; percentages compare pooled paired means.

Model	Condition	Success	Total calls↓	Cost ($)↓	Mean wall s
Luna	No memory	7/10	219	0.430	78.7
	SKILLZIP	**8/10**	**148**	**0.367**	**71.8**
DeepSeek	No memory	7/10	230	0.343	**191.7**
	SKILLZIP	**8/10**	**168**	**0.263**	204.3
Pooled	No memory	14/20	449	0.773	**135.2**
	SKILLZIP	**16/20**	**316**	**0.630**	138.1

cost, and **29.7% lower** wall time. The intention-to-treat and complete-pair analyses therefore agree on the resource result; the latter attributes no completion advantage to either interface.

4.2 RQ2: THE REDUCTION REPLICATES UNDER FRESH GENERATION

We mechanically select the alphabetically first held-out task per repository before inspecting the repeat and regenerate those same 10 tasks, producing 40 new trajectories: two models, no memory versus SKILLZIP. This rule excludes the primary Luna task that flips in favor of SKILLZIP. The result is stronger, not merely directionally similar. Table 4 shows 449 decisions for no memory and 316 for SKILLZIP, a **29.6% reduction** (repository-clustered 95% CI: 20.0–39.7% fewer). Model cost falls by **18.4%** (95% CI: 7.2–29.3% fewer), while success rises from 14/20 to 16/20.

Both model families show compression in this generation: Luna falls from 219 to 148 calls (32.4%, 95% CI: 17.5–43.7% fewer) and DeepSeek from 230 to 168 (27.0%, 95% CI: 5.2–43.1% fewer; exact sign-flip $p = .0859$); each model gains one successful task. Wall time is statistically neutral because three SKILLZIP DeepSeek runs reach the fixed six-minute limit, including successful runs. The repeat demonstrates that the pooled effect persists across a fresh generation draw.

4.3 RQ3: REPEATED GENERATIONS PRESERVE THE DECISION REDUCTION

The primary aggregate combines a decisive Luna reduction with a neutral DeepSeek draw. Luna uses 10.0 fewer calls per task (40.6%, 95% CI: 35.3–45.1% fewer) and improves from 13/20 to 14/20. DeepSeek changes from 17.75 to 17.95 calls and from 11/20 to 10/20; neither difference is resolved by the interval. The task transitions are correspondingly stable: one Luna win, one DeepSeek loss, and 38 unchanged model–task outcomes.

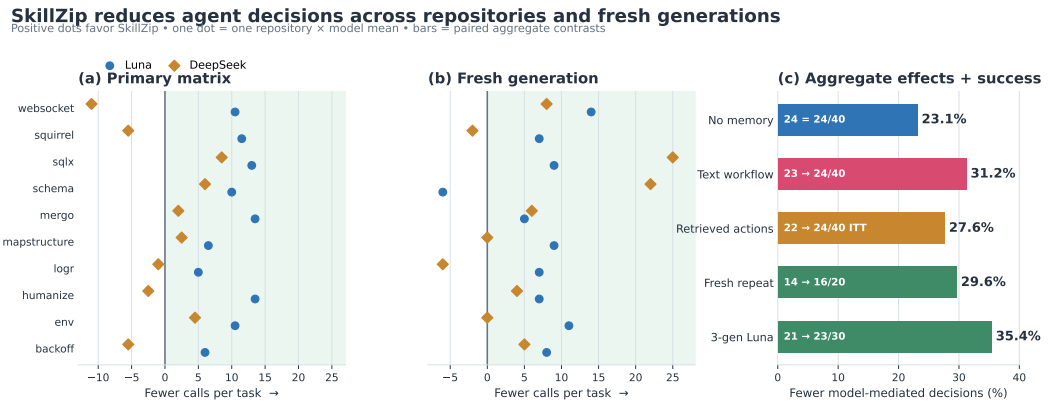


Figure 2: The primary and repeated generations agree that executable actions reduce model-mediated decisions while preserving the observed quality distribution. On 38 complete runner pairs, retrieval and SKILLZIP each solve 22 tasks: 21 successes are shared, with one exchanged in each direction.

The same 10 mechanically selected tasks make generation variance visible. Luna changes from 239 to 141 calls in generation 1 (41.0% fewer), 219 to 148 in generation 2 (32.4%), and 203 to 138 in generation 3 (32.0%). Success is respectively $8 \to 8$, $7 \to 8$, and $6 \to 7$. Pooling all 60 Luna trajectories gives 661 versus 427 decisions, or **35.4% fewer**; a two-way bootstrap that independently resamples 10 repositories and three generations gives 19.6–45.0% fewer. The success difference is $+6.7$ points with a $[0, +20]$ point interval. This directly incorporates stochastic generation into the uncertainty estimate instead of treating one trajectory per cell as repeated evidence.

One generation-3 no-memory run exits at the provider runner before emitting a call. Keeping it as an observed failure and zero calls is conservative for decision savings. A complete-pair analysis over the other 29 repository–generation cells gives **38.6% fewer** decisions and $21/29 \to 22/29$ completion; counting the missing no-memory outcome as successful still gives $22/30 \to 23/30$. The repeated efficiency and quality directions therefore do not depend on this runner event.

DeepSeek has two generations on the same cells: 204 to 195 decisions (4.4% fewer) and 230 to 168 (27.0% fewer). Pooling both gives 434 to 363 (16.4% fewer) and moves success from 15/20 to 16/20. The direction survives for both policies, while the model-specific magnitude is more variable for DeepSeek. We report policy and generation heterogeneity as measured behavior rather than assigning a post-hoc cognitive mechanism to one draw.

The transfer slice further supports an action-level rather than instance-level explanation. In the three repositories with no successful source trajectory, SKILLZIP reaches 7/12 pooled completions versus 6/12 for no memory while reducing mean calls from 18.25 to 15.00 (17.8%). The macros therefore remain useful where no solved example from that repository entered compilation; they carry a cross-repository procedure, not a saved task answer. This exploratory slice is Luna-led (21.67 to 14.50 calls; $3/6 \to 4/6$); DeepSeek changes from 14.83 to 15.50 calls with completion fixed at $3/6$.

5 ANALYSIS

5.1 WHAT SOURCE SUPPORT AND GRAMMAR ANALYSIS REVEAL

The seven successful source traces contain 65 productions with support in at least three traces. The highest-saving rules are repeated read blocks; longer productions combine search, source/test reading, and repository inspection. Post-edit traces vary more in ordering but consistently contain format/test/diff/status operations. The descriptive grammar corroborates the two template families and exposes their internal recurrence without claiming to synthesize their bodies.

We audit whether source support responds to candidate semantics rather than admitting every author-provided action. A follow-up audit expands the library to eight repository actions. The fixed $k = 3$ gate admits the deployed inspection and validation families (support 7 each) and rejects six (supports

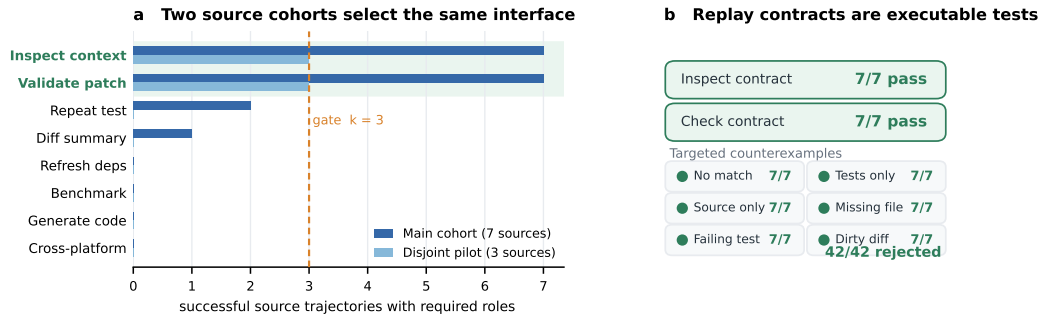


Figure 3: The admission and replay mechanism is reproducible across disjoint source cohorts. Both cohorts select the same two supported phase families from eight candidates (left); every positive contract passes and all six targeted mutation types are rejected in every main-source workspace (right). The selection audit is follow-up evidence, not a held-out oracle comparison.

$2, 1, 0, 0, 0, 0$); the pair is stable in all seven leave-one-source-out folds, while an empty source set selects none. The earlier pilot supplies a disjoint source-cohort replication: its three successful traces independently support the same pair and no alternative (Figure 3). An oracle exposing that pair is behaviorally identical to SKILLZIP after admission. Thus these audits establish reproducible support-based admission; the primary matrix measures the resulting author-specified interface, not a causal advantage over that oracle.

Replay is independently falsifiable. In each clean source workspace we target six contract clauses: no inspection match, test-only context, source-only context, a missing declared Go path, an injected failing test, and a whitespace-corrupted diff. The responsible stages reject all **42/42** mutations, while the actions pass all **14/14** positive macro–workspace contracts, explicitly checking both context types, build metadata, tree immutability, stage presence, exits, and timeouts.

This small library is a strength of the representation. A flat trajectory pool grows with every task; a textual document must keep describing familiar behavior. SKILLZIP pays once for two typed actions and lets task-specific arguments be supplied at invocation. Across primary runs, agents use INSPECT 43 times and CHECK 71 times. All 40 runs inspect at least once, and 38 check at least once, demonstrating both broad applicability and policy control over reuse.

5.2 A CONCRETE TRANSITION

On the held-out raw-JSON logging task, Luna with no memory fails after 18 primitive decisions. With SKILLZIP it calls INSPECT to retrieve the relevant renderer and tests, edits the conversion path, and calls CHECK; it passes the developer suite in 14 decisions. This is the sole primary success transition in favor of SKILLZIP. Conversely, one DeepSeek WebSocket run loses a previous success while using more calls. The remaining 38 outcomes are unchanged while call count falls in all 20 Luna tasks. The headline therefore reflects broad decision compression rather than a small set of quality flips.

5.3 ACCOUNTING BOUNDARY

The primary metric measures tool-interface choices emitted by the language model; one model step can emit several such choices. It does not claim that internal commands disappear. CHECK, for example, still runs formatting, tests, a diff check, and status locally. We preserve these commands and their outputs in the trace and separately report wall time and cost. Across the 114 primary macro calls, the runtime executes 617 deterministic local subprocess stages (5.41 per macro on average), all without another model choice. This boundary explains why a 23.1% reduction in tool decisions yields a cost-neutral no-memory draw: model context, output length, and caching also determine cost. The textual-workflow agent can batch primitive operations, yet still makes 31.2% more model decisions; shell segments and macro-local stages are not compute-equivalent units. Against retrieved actions, SKILLZIP reduces cost by 10.9% and time by 29.1%; the fresh repeat reduces cost by 18.4%.

Adding local stages to decisions gives 1,269 for SKILLZIP versus 848 for no memory, making the interface trade explicit: deterministic work moves behind the action rather than vanishing. The reduction persists after removing all planning-only `todowrite` calls from the frozen primary traces: the no-memory contrast changes from 23.1% to **24.7%** fewer decisions (repository-clustered 95% CI: 15.3–33.6% fewer, $p = .0078$); reductions versus textual workflow and retrieved actions remain 32.4% and 26.2% (Appendix Table 8).

6 RELATED WORK

Experience and procedural memory. Reflexion and ExpeL reuse verbal experience (Shinn et al., 2023; Zhao et al., 2023); Agent Workflow Memory, ReMe, and Tree-of-Experience organize retrieved procedures (Wang et al., 2025; Cao et al., 2026; Deng et al., 2026). Concurrent SkillGLoW execution-gates textual skill families (Yan et al., 2026). SKILLZIP changes the representation and target: recurring phases become typed actions whose removed decisions are measured against equally sourced textual workflows and action sequences.

Tool learning and action interfaces. Toolformer, ToolLLM, and EASYTOOL improve use or description of existing tools (Schick et al., 2023; Qin et al., 2024; Yuan et al., 2025); CREATOR and LLMs as Tool Makers synthesize task-local tools (Qian et al., 2023; Cai et al., 2024), and CodeAct uses code as the action language (Wang et al., 2024). GUI-explorer mines primitive-action guidance (Xie et al., 2025), while Voyager stores embodied programs (Wang et al., 2023). SKILLZIP contributes a trace-supported, replay-checked interface evaluated with three contextual controls and explicit decision/resource accounting.

Software agents and temporal abstraction. SWE-bench and SWE-agent establish executable repair (Jimenez et al., 2024; Yang et al., 2024); our smaller public revisions enable a fully crossed audit. The actions instantiate options (Sutton et al., 1999) at the agent interface: traces supply phase support, arguments, and replay environments for selecting checked temporal abstractions.

7 DEMONSTRATED SCOPE AND EXTENSIONS

The study establishes checked executable experience for Go repository tools, two hosted policies, and a deployed library of two broad author-provided action templates. The primary no-memory contrast identifies decision compression; both contextual baselines and the fresh repeat also establish resource savings. Developer tests supply strong executable oracles for this task scope. Template synthesis, domain-specific contracts, larger libraries, action composition, and semantic equivalence oracles define the next extensions. Within the demonstrated scope, checked executable experience consistently reduces future policy decisions.

8 CONCLUSION

SKILLZIP turns successful experience into evidence for a replay-checked action interface. On 20 temporal tasks, its two actions remove 23.1% of decisions at matching observed no-memory completion and 27.6–31.2% versus two contextual controls. Fresh generation yields 29.6% fewer decisions and 18.4% lower cost; three paired Luna generations retain 35.4% savings. Successful traces can select the actions with which the next agent reasons.

REFERENCES

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, 2024.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of ACL*, pp. 16803–16822, 2026. doi: 10.18653/v1/2026.findings-acl.829.

Zihao Deng, Yining Zhu, Leiming Wang, Junbo Wang, and Jingfei Lu. Tree-of-experience: Hierarchical experience management for self-evolving agents. *arXiv preprint arXiv:2608.09044*, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Craig G. Nevill-Manning and Ian H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997. doi: 10.1613/jair.374.

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 EMNLP*, pp. 6922–6939, 2023. doi: 10.18653/v1/2023.findings-emnlp.462.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *International Conference on Machine Learning*, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Learning Representations*, 2025.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. GUI-explorer: Autonomous exploration and mining of transition-aware knowledge for GUI agent. In *Proceedings of ACL*, pp. 5650–5667, 2025. doi: 10.18653/v1/2025.acl-long.282.

Ao Yan, Xin Zhang, Jiawei Du, and Joey Tianyi Zhou. SkillGLoW: Procedural-family skill consolidation for self-improving agents on long-horizon task streams. *arXiv preprint arXiv:2609.02217*, 2026.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. EASYTOOL: Enhancing LLM-based agents with concise tool instruction. In *Proceedings of NAACL-HLT*, pp. 951–972, 2025. doi: 10.18653/v1/2025.naacl-long.44.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

A Task and Split Details

Table 5: The 20 temporally held-out tasks. Each row names one public repository and the two later behaviors evaluated from their parent revisions. Full 40-character hashes are in the released `configs/main_tasks_frozen.json`.

Repository	Held-out task A	Held-out task B
backoff	context cause	RetryAfter cause
env	complex default reference	uppercase field name
humanize	BigComma immutability	minimum byte digits
logr	raw JSON message	omitted first JSON field
mapstructure	text-hook nil safety	empty hook composition
mergo	map value assignment	empty-map overwrite
schema	nil embedded indirection	default-value tags
sqlx	reset destination slice	named trailing space
squirrel	remove columns	conjunction placeholders
websocket	temporary read errors	close handler

Temporal construction. For task j, the agent sees an archive of parent revision r_j^-. Evaluation overlays developer test files from target revision r_j^+ and runs `go test ./...`. Candidate validation requires (i) the parent suite passes, (ii) the overlaid developer tests fail on the unmodified parent, and (iii) r_j^- contains neither the target implementation nor a later source task. Before any agent evaluation, 22/30 initial candidates pass all oracle checks and eight are rejected. Ten replacement candidates—including two temporal-split replacements—all pass; the final 30 frozen tasks each map to an exact valid parent/target record. Selection uses no agent outcome.

B Reproducibility and Harness

Each condition starts from an isolated archive and a new local Git repository. The prompt asks for the behavioral change and focused tests, then instructs the agent to run appropriate tests and leave the complete patch in the workspace. It prohibits Git history, remotes, external source trees, network downloads, and developer commits. The runner records JSONL tool events, step-level token and cost fields, stderr, exit status, wall time, the final binary patch, model ID, condition, source parent, and the exact prompt. A six-minute process timeout is common to all conditions. Evaluation occurs in a separate temporary directory created from the parent archive.

For retrieved-action context, a frozen lexical rule chooses among the seven successful sources. It prefers the same repository when available and otherwise maximizes cosine similarity between lowercased prompt-token count vectors, with lexical task ID as the tie breaker. The full ranking for every target is released; no held-out result is available to the retriever. The rendered context contains at most 32 model-emitted calls, preserves tool order, basenames, search patterns, and bounded shell commands, and redacts edit contents and all tool observations.

Macro execution. INSPECT uses ripgrep to locate content matches, selects bounded implementation and test files, emits matching context, and includes Go module or Make metadata. It is read-only. CHECK formats declared Go paths, optionally runs a focused test regex, always runs the full suite, performs `git diff --check`, and reports `git status --short`. Return codes, stdout, stderr, and timeouts are exposed. Agents retain access to all primitive tools and can decline or repeat either macro.

The follow-up selection audit expands the candidate library to eight templates. Main-cohort supports are $7, 7, 2, 1, 0, 0, 0, 0$: two pass $k = 3$ and replay, six fail support, and the admitted pair is stable in all seven leave-one-source-out folds. A disjoint pilot source cohort has supports $3, 3, 0, 0, 0, 0, 0, 0$ and selects the same pair. Replay-contract negative tests reject 42/42 targeted violations across the seven main source workspaces. These audits are in `template_selection_audit.json`, `induction_replication_audit.json`, and `replay_mutation_audit.json`.

Table 6: Author-provided template constants and data-derived admission fields. These values are frozen before held-out execution.

Component	Author-provided template	Data-derived / replay-gated
INSPECT	max 4 source + 3 test files; ± 8 context lines; 8k chars/file; Go/Make metadata	search/read support 7/7; patterns from source calls; read-only digest 7/7
CHECK	format declared Go files; optional focus; full suite; diff check; status; 240s/stage	validation support 7/7; changed paths from saved patches; contract pass 7/7
Compiler	two semantic families; typed slots and bodies	minimum support $k = 3$; 65-rule grammar audit

C STATISTICAL PROCEDURE

Let d_r be the mean paired condition difference over all held-out tasks and selected models in repository r. The point estimate is the mean over all model–task pairs. For the cluster bootstrap we sample 10 repositories with replacement 20,000 times and recompute the paired mean and relative change, carrying both tasks and all relevant model conditions together. Percentile 2.5 and 97.5 quantiles form the reported interval. The exact sign-flip test enumerates all 2^{10} sign assignments to the repository-level differences and reports the fraction whose absolute mean is at least the observed absolute mean. We call this a sign-flip randomization test, not a test over 40 independent trajectories. For primary completion, the pooled paired estimate is 0.0 percentage points with a repository-clustered interval of $[-7.5, +7.5]$. In the repeat, the call-count sign-flip result is $p = .0020$ pooled; the model-specific call intervals are reported in Section 4. For the three-generation Luna analysis, a two-way paired bootstrap independently resamples the 10 repositories and three generation indices, preserving both conditions within every sampled cell.

Table 7: Primary paired effects for SKILLZIP. Relative intervals are repository-clustered 95% bootstrap intervals; p is the exact repository sign-flip result.

Model	Contrast	Decision change	Relative 95% CI	p
Luna	no memory → SKILLZIP	-40.6%	$[-45.1, -35.3]$	.0020
DeepSeek	no memory → SKILLZIP	$+1.1\%$	$[-16.3, +24.9]$	.9336
Pooled	no memory → SKILLZIP	-23.1%	$[-31.2, -14.1]$	.0059
Pooled	text workflow → SKILLZIP	-31.2%	$[-36.8, -25.5]$	.0020
Pooled	retrieved actions → SKILLZIP	-27.6%	$[-34.9, -20.9]$	.0020

D ADDITIONAL ACCOUNTING

The experimental model spend is $9.106929: $0.422269 for 10 main source trajectories, $4.159230 for the original 120 primary runs, $1.497552 for the 40-run retrieval arm, $1.403386 for the 40-run repeat, $0.883107 for the third Luna generation, and $0.741386 for the separate four-repository pilot that triggered expansion. Two already-running retrieval calls brought the final ledger $0.106929 above the $9 local stop; no further experimental call followed. Pilot repositories and tasks do not enter the headline. Premium reviewer models are tracked separately. All analysis, task validation, grammar mining, replay, hidden-test execution, bootstrapping, and figure generation run locally.

Macro internals are retained rather than credited as vanished work. Parsing saved tool outputs gives 617 local subprocess stages inside 114 primary macro calls (mean 5.41) and 280 stages inside 48 repeat calls (mean 5.83). These counts exclude metadata files read directly by Python. They measure local execution, not model-mediated interface choices, and accompany the reported wall-time and cost results in `data/derived/macro_internal_accounting.json`.

E ARTIFACT MAP

The submission package contains the complete anonymous manuscript, immutable task configuration, source and held-out traces, patches, evaluation JSON, derived CSV/JSON

Table 8: Primary decision-count sensitivity. The robustness metric excludes planning-only `todowrite` events but retains every environment-facing tool call; the final column is the SKILLZIP reduction relative to each comparator.

Condition	All calls	Planning-only	Filtered calls	Reduction
No memory	848	70	778	24.7%
Text workflow	948	81	867	32.4%
Retrieved actions	901	107	794	26.2%
SKILLZIP	652	66	586	—

Table 9: Action adoption in the primary SKILLZIP condition. An adopted action appears at least once in that run.

Model	Runs	Any macro	INSPECT calls	CHECK calls	Total calls
Luna	20	20	20	28	48
DeepSeek	20	20	23	43	66
Pooled	40	40	43	71	114

tables, compiler/runtime code, statistical scripts, prompts, cost ledger, claim ledger, and checksums. The key result tables are `data/derived/main_task_results.csv` and `data/derived/repeat_task_results.csv`; three-generation inference is in `three_generation_summary.json`; selection and replay audits are `template_selection_audit.json`, `induction_replication_audit.json`, and `replay_mutation_audit.json`. No task, model output, or score from an earlier paper is included. Complete-pair checks for non-timeout runner errors are in `data/derived/runner_sensitivity.json`; raw-trace count sensitivity is in `data/derived/decision_count_sensitivity.json`.

F AI USE STATEMENT

Language-model agents are the object of study and were used to execute the frozen software-engineering tasks. Language-model assistance also supported literature retrieval and discovery, research ideation and execution, implementation, drafting, and editorial revision. All experimental outcomes are produced by executable tests or deterministic analysis scripts, and all cited relationships and reported numbers are checked against the saved artifacts by the authors. The anonymous code, configurations, trajectories, and derived results are available at `https://anonymous.4open.science/r/skillzip-iclr2027/`.