DARWINX: EVOLVING AGENT HARNESSES THROUGH NATURAL SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

An LLM agent's capabilities depend on the *harness* that supplies its instructions, tools, memory, and control flow. Improving that harness is a selection problem: an edit that fixes one task can regress another, a discarded variant may contain a complementary capability, and a noisy local gain may not repeat. We introduce DARWINX, a population-based method for evolving harnesses with the model fixed. Its preserve-and-extend contract screens task-level regressions, while independent confirmation determines which variants may steer later search. The archive retains other specialists for recombination, and execution traces and shared memory guide subsequent proposals. Search uses verifier-backed synthetic tasks that reuse benchmark environments; original benchmark tasks remain outside the search loop. On complete official evaluations, the selected harnesses improve five-attempt average success by 7.19 percentage points on Terminal-Bench 2.1 and 9.38 points on DeepSWE. On WebArena-Infinity, the frozen winner achieves 91.19% single-attempt success across 1,260 tasks, versus 75.48% for the composite report baseline, with no static-invalid evolved trajectory flagged by the action audit. These results show that the complete selection procedure can improve a frozen-model agent beyond the synthetic tasks used for search.

1 INTRODUCTION

An LLM agent's behavior depends on its harness as well as its model weights. The harness supplies instructions and skills, exposes tools, stores memory, and controls the action loop (Chen et al., 2026; Zhang et al., 2025). Editing these components can improve the agent while the model remains fixed. Prior work optimizes prompts (Yang et al., 2023; Fernando et al., 2023; Yuksekgonul et al., 2024), skills (Yang et al., 2026), workflows (Hu et al., 2024; Zhang et al., 2024; Zhuge et al., 2024), or agent code (Robeyns et al., 2025; Zhang et al., 2025). Figure 1 previews the population-selection loop developed below.

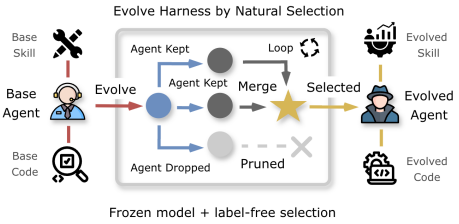


Figure 1: DARWINX's population-selection loop; Figure 2 gives the full procedure.

Despite these different edit spaces, recent systems share a similar inner loop: collect rollouts, diagnose a failure, propose a bounded edit, and gate it on task feedback (Yang et al., 2026; Chen et al., 2026). The unresolved choice is the selection process around that loop. Harness edits have heterogeneous effects: a change that helps one task family can disrupt another, and its value may depend on which other capabilities have already been retained. Evaluating edits independently can therefore miss complementarities or conflicts that emerge only after composition. Keep-best search is path dependent and can discard a useful specialist before its edit meets a complementary one, while noisy evaluation can promote a local gain that does not repeat (Robeyns et al., 2025; Bjarnason et al., 2026; Miller, 2024). The selection rule must consequently manage both uncertainty and candidate diversity rather than simply preserve the highest observed score. Harness improvement therefore requires generating effective edits while deciding which capabilities to retain, combine, and trust across tasks.

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.

Table 1: Positioning among self-improving-agent methods (✓ present, ∼ restricted, ✗ absent). The comparison distinguishes mechanisms for proposing edits from those for evaluating, retaining, and combining them. Appendix Table 9 details the scope and implementation of each mechanism.

	Edits	**Search**		**Selection**		**Signal**
Method	tools & control flow	population archive	cross-lineage merge	bounded regression	noise-aware confirmation	trace & shared memory
Optimizers over one designated artifact						
OPRO/PromptBreeder/TextGrad	✗	∼	∼[†]	✗	✗	✗
ADAS/AFlow/GPTSwarm	∼	✓	✗	✗	∼	✗
SkillOpt	✗	✗	✗	∼	∼	✗
Agents that edit their own scaffold						
SICA	✓	∼	✗	✗	✗	✗
DGM	✓	✓	✗	∼	∼	✗
HarnessX	✓	∼	✗	✓	∼	✗
DARWINX (ours)	✓	✓	✓	✓	✓	✓

[†]Genetic prompt optimizers recombine prompt strings, but not harness variants selected for complementary task coverage.

This tension grows with task diversity. Aggregate scores can hide concentrated regressions, whereas requiring every edit to win everywhere can suppress complementary capabilities. Selection must preserve alternatives without granting them equal influence over later generations.

DARWINX separates *retention* from *steering*. It evolves a population of complete harnesses around a frozen model (Figure 1), retaining alternative lineages instead of forcing every candidate to replace one leader. A preserve-and-extend contract screens task-level regressions, and independent confirmation determines which variants may become parents or the final harness. Other specialists remain available for cross-lineage recombination and as evidence for later proposals. The archive is active genetic material, while the selection rule determines which evidence can redirect search.

Table 1 situates this distinction. DGM grows an archive through single-parent mutation (Zhang et al., 2025), while HarnessX detects regressions and isolates task-family variants (Chen et al., 2026). Isolation limits interference but cannot combine complementary specialists. DARWINX instead uses its archive for both ancestry and screened recombination: specialists may contribute edits without steering search, and merged children are compared with their source variants to detect interference. Failure traces, teacher demonstrations, and passing-versus-failing contrasts guide which edits are proposed; selection separately determines which edits are retained and allowed to steer.

We evaluate the complete procedure in the Terminal-Bench 2.1, DeepSWE, and WebArena-Infinity execution environments while excluding their original tasks from search. Fixed synthetic panels select winners, and disjoint protected panels plus original tasks evaluate the frozen choices (§3). Against matched roots, DARWINX improves Terminal-Bench 2.1 and DeepSWE by 7.19 and 9.38 percentage points. On WebArena-Infinity, it reaches 91.19% pass@1, 15.71 points above the composite report baseline. Task-level, cross-benchmark, matched-budget, and mechanism analyses clarify the scope of these aggregate gains.

Contributions. We formulate harness evolution as population selection that separates retention from steering. We instantiate this principle with bounded-regression gates, independent confirmation, screened cross-lineage merges, and shared execution evidence. Across synthetic search, protected panels, and original benchmarks, we report task-level regressions, narrow baseline margins, and inconclusive mechanism effects alongside the aggregate improvements.

2 DARWINX

DARWINX searches over complete agent harnesses while keeping the acting model and task runtime fixed. A proposal may change prompts, skills, memory, tools, or control flow, spanning both the knowledge exposed to the agent and the code that sequences its actions. Because an edit that repairs one task can disrupt another, each child is compared with its parent at the task level. A genealogy archive retains every scored harness together with its edit delta, taskwise outcomes, trajectories, and distilled lessons. The archive is an active search state: a lineage can be revisited, mined for evidence, or recombined long after it stops leading the search.

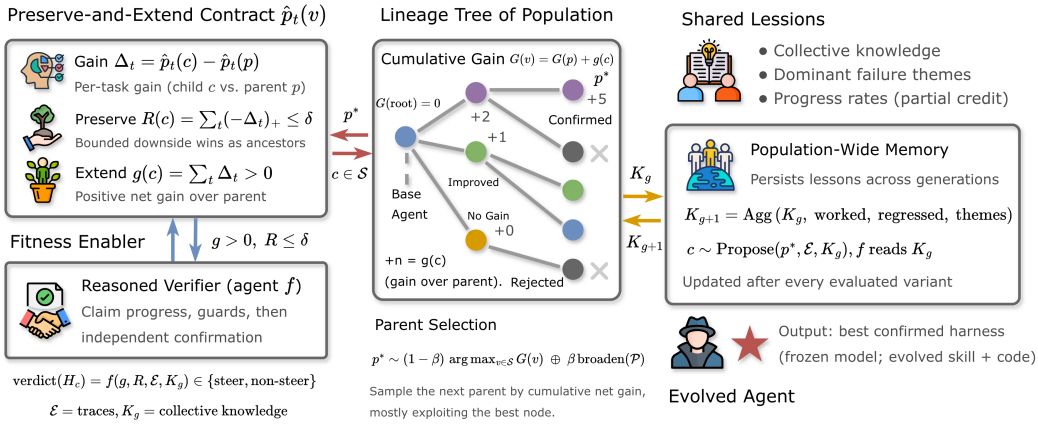


Figure 2: DARWINX's selection loop. A preservation-aware verifier screens edits (left), steering-eligible lineages guide parent selection (middle), and shared memory carries lessons across generations (right). Node gains are schematic.

Figure 2 separates candidate creation, retention, and steering. Task evidence from a selected parent guides a bounded edit, and a local gate compares the child with that parent. Passing the gate admits the child to independent confirmation. Let $\mathcal{S}$ denote the set of confirmed, steering-eligible variants. Only variants in $\mathcal{S}$ can become parents or the final harness; other evaluated variants remain in the archive and contribute evidence to shared memory. Model weights are never updated.

2.1 FITNESS AND THE PRESERVE-AND-EXTEND CONTRACT

For variant v, let $\hat{p}_t(v)$ be its solve rate on task t over k trials (avg@k). For child c and parent p, define $\Delta_t = \hat{p}_t(c) - \hat{p}_t(p)$. The local gate requires positive net gain, $g(c) = \sum_t \Delta_t > 0$, and bounded regression, $R(c) = \sum_t(-\Delta_t)_+ \le \delta$. Net gain rewards additional coverage; the regression budget limits how much measured capability the edit can give back. A reasoned verifier reads the child trajectories $\mathcal{E}$ and shared memory K_g alongside these scores before assigning steering eligibility.

The numerical contract and the reasoned verifier serve different roles. The contract prevents an aggregate improvement from hiding unlimited concentrated losses. The verifier inspects claim progress, guard outcomes, and trajectories to decide whether the measured change reflects a reusable capability. Neither local decision replaces independent confirmation.

Measurement and confirmation. Before steering search, a locally accepted child must pass a preservation probe on inherited solves and higher-fidelity avg@k confirmation. The probe tests whether old capabilities survive, while repeated trials test whether the gain persists beyond the local comparison. Task timeouts count as failures (§3). Separating admission from steering prevents a single lucky rollout from becoming an ancestor whose noise compounds through later edits.

2.2 BRANCH EVOLUTION

At each generation, a branch targets a capability cluster and selects a parent harness. The proposer uses task evidence to make one edit aimed at a fragile or failing task while preserving inherited solves. An additive edit introduces a capability, pruning removes lineage-local material that no longer helps, and consolidation turns accumulated special cases into a reusable mechanism. Every operation faces the same gate; a cleaner harness must still limit regressions.

Each child inherits cumulative lineage gain $G(v) = G(p) + g(c)$. Parent selection usually exploits the steering-eligible variant with highest G and, with probability β, explores another eligible lineage. Because G aggregates local task subsets, it is a search heuristic rather than an official benchmark score. The exploration branch can recover a capability absent from the current leader without letting an unconfirmed specialist control the campaign.

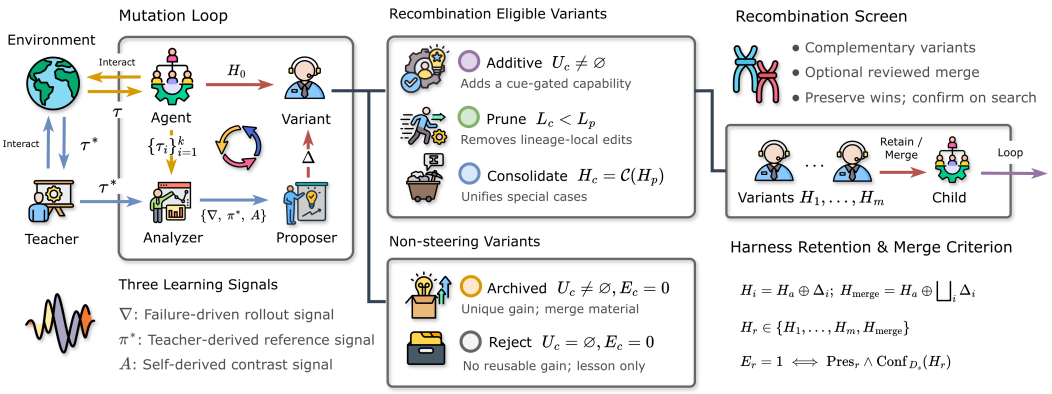


Figure 3: Per-generation operations. Task evidence guides an additive, prune, or consolidate edit (left). The archive retains steering-eligible variants and other specialists (middle). Complementary edits may be merged and screened against their sources (right).

2.3 Population and Recombination

Every scored variant remains in the genealogy archive, but only confirmed variants in $\mathcal{S}$ can become parents or the final harness. For child c with parent p, let $U_c = S(c) \setminus S(p)$ denote its unique solves and $E_c \in \{0, 1\}$ its steering eligibility. A child with $U_c \neq \varnothing$ and $E_c = 0$ remains an *archived specialist*: its edit may support recombination and its traces update shared memory. A child with $U_c = \varnothing$ and $E_c = 0$ contributes failure evidence only. Parallel branches preserve specialized capabilities that a single incumbent might discard.

Archive membership, recombination utility, and steering eligibility remain separate. Archived specialists may contribute distinct capabilities without entering $\mathcal{S}$, while rejected nodes contribute memory only. This preserves rare capabilities and negative evidence without letting unconfirmed variants control parent selection.

Figure 3 connects these archive states to the edit and recombination operators. A merge is useful only if the source gains survive together. For complementary variants $H_i = H_a \oplus \Delta_i$ from a common ancestor H_a, recombination proposes $H_{\text{merge}} = H_a \oplus \bigsqcup_i \Delta_i$. The screen considers $H_r \in \{H_1, \ldots, H_m, H_{\text{merge}}\}$ and assigns retention eligibility only when preservation and search-set confirmation both hold: $E_r = 1 \iff \text{Pres}_r \wedge \text{Conf}_{D_s}(H_r)$. If the merge regresses, its source variants remain available. A retained merge must still clear the same gate and confirmation as any other child.

2.4 Learning Signals and Shared Memory

Selection determines which variants may steer; task evidence determines what the proposer tries next (Figure 3). Failure-derived signals (∇) summarize a failed trajectory τ. Teacher-derived signals (π^*) translate a reference solver's successful trajectory τ^* into a candidate approach when the agent has no successful rollout. Self-derived signals (A) contrast the agent's passing and failing rollouts $\{\tau_i\}_{i=1}^k$ on variable tasks. All three propose edits that face the same selection rule.

The signal matches the evidence available for the target task. Failure-derived diagnosis is the default; a teacher trajectory provides a worked approach when no agent rollout succeeds, while self-contrast isolates behaviors associated with success across mixed outcomes. Proposal evidence thus adapts to the task regime, while retention remains governed by the common gate.

Cross-task memory. A failure-mode classifier aggregates recurring outcomes such as setup timeouts or incorrect output. After each evaluated variant, shared memory K_g records what worked, what regressed, and the dominant failure themes: $K_{g+1} = \text{Agg}(K_g, \text{worked}, \text{regressed}, \text{themes})$. These records help the proposer turn recurring problems into cross-task capabilities rather than task-specific patches, while the verifier checks new edits against earlier regressions. Search returns the best confirmed, steering-eligible harness under its fixed metric; protected and downstream evaluations assess that frozen choice without redirecting search (§3).

3 EVALUATION

We run three 30-node DARWINX campaigns on synthetic task pools grounded in TB2.1 (The Terminal-Bench Team, 2026), DeepSWE, and WebArena-Infinity (WAI) (Zhou, 2026). Fixed search panels select each harness, while disjoint protected panels and the original benchmarks evaluate the frozen choice without steering search. Appendix B.1 gives the models, task surfaces, partitions, node scoring, and downstream protocol.

For the matched-budget comparison, DARWINX and controlled adaptations of DGM, ADAS, DGM-H, Meta-Harness, SkillOpt, and GEPA each receive ten proposals under the same evolvee, proposer, panels, and resource accounting. A blind panel reranks search leaders before evaluation on a common original TB2.1 slice. Appendix B.2 defines the adaptations and selection procedure.

We report search-panel scores separately from downstream evidence: paired avg@5 results on original TB2.1 and DeepSWE, WAI pass@1, the common TB2.1-slice score, cross-benchmark transfer, and mechanism contrasts. Appendix B.3 specifies sampling and uncertainty.

4 RESULTS

We organize the results from selection through transfer and attribution. We compare search-panel gains with frozen official outcomes, then analyze task-level gains, population behavior, matched-budget comparisons, WAI transfer and action validity, and mechanism effects.

Research questions. We study five questions. RQ1 tests whether search improves fixed synthetic panels and frozen winners retain gains on official benchmarks. RQ2 examines task-level gains, cross-benchmark transfer, and retained-population behavior. RQ3 compares DARWINX against alternative optimizers under matched budgets. RQ4 measures WAI transfer and audits action validity. RQ5 ablates preservation, memory, consolidation, and quality-diversity retention.

4.1 RQ1: SEARCH IMPROVEMENT

RQ1 separates search effectiveness from downstream validity: whether population search improves fixed synthetic selection panels and whether frozen winners retain those gains on evaluations excluded from search. Table 2 reports the aggregate outcomes, while Figure 4 shows when gains emerge across the 30-node campaigns.

Table 2: Search-panel and official results (%). Reference denotes the root for TB2.1 and DeepSWE and the report baseline for WAI.

	Search panel		Official evaluation	
Campaign	**Ref. → winner**	**Δ**	**Ref. → winner**	**Δ**
TB2.1	53.33 → 59.17	+5.83	72.13 → 79.33	+7.19
DeepSWE	68.06 → 77.78	+9.72	54.34 → 63.72	+9.38
WAI	79.17 → 80.56	+1.39	75.48 → 91.19	+15.71

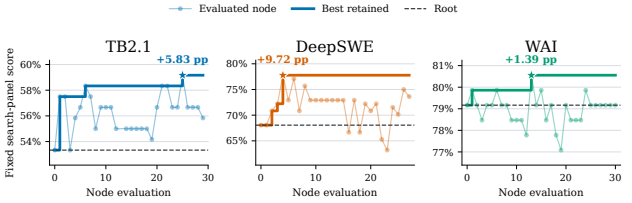


Figure 4: RQ1 search trajectories. Thin curves are eligible nodes, step curves are retained bests, dashed lines are roots, and stars are the frozen winners.

All three campaigns select non-root winners, with search-panel gains of +9.72 points on DeepSWE, +5.83 on TB2.1, and +1.39 on WAI. On the complete official suites at avg@5, TB2.1 improves from 321/445 (72.13%) to 353/445 (79.33%, +7.19 points), and DeepSWE from 307/565 (54.34%) to 360/565 (63.72%, +9.38 points). WAI reaches 91.19% pass1, 15.71 points above the composite report baseline. Protected diagnostics and official outcomes do not rerank candidates.

The trajectories reveal different search regimes. DeepSWE finds its winner after four node evaluations, whereas TB2.1 improves through several smaller steps and reaches its winner at evaluation 25. WAI records retained gains at evaluations 1 and 13. Later proposals do not displace these gains because the archive preserves the best accepted node. Section 4.4 provides the WAI application breakdown and action-validity audit.

4.2 RQ2: Where do gains arise, and do they transfer?

We locate the official gains by root reliability and capability, then test whether the evolved behavior transfers across benchmarks. Every rate in Tables 3 and 4 is recomputed from paired root and evolved trials.

Table 3: Root-reliability success rates (%) on the complete evaluations. Δ is Evolved minus Root (points).

Benchmark	Root band	Tasks	Root	Evolved	Δ
TB2.1	0/5	12	0.00	23.33	+23.33
	1–4/5	27	52.59	76.30	+23.70
	5/5	50	100.00	94.40	−5.60
DeepSWE	0/5	21	0.00	11.43	+11.43
	1–4/5	63	51.43	69.84	+18.41
	5/5	29	100.00	88.28	−11.72

Table 4: TB2.1 success rates (%) by capability cluster. Δ is Evolved minus Root (points).

Capability	Root	Evolved	Δ
SysAdmin/network	72.73	92.73	+20.00
Algorithms/coding	82.86	100.00	+17.14
ML/scientific	65.00	70.83	+5.83
Systems/build	70.91	76.36	+5.45
Security/crypto	62.22	64.44	+2.22
Data/database	91.11	91.11	0.00
Parsing/media	67.27	63.64	−3.64

Most gains come from inconsistent root solves: the 1–4/5 band adds 32 successes on TB2.1 and 58 on DeepSWE, versus 14 and 12 from never-solved tasks. Residual failures fall by 25.8% (32/124) and 20.5% (53/258), respectively.

Recovery remains non-monotonic. The 5/5 bands lose 14 and 17 successes, leaving 45/50 TB2.1 and 19/29 DeepSWE tasks perfect. TB2.1 gains concentrate in system administration/networking (+20.00 points) and algorithms/coding (+17.14); data/database is unchanged and parsing/media falls by 3.64. Improved, unchanged, and regressed counts are 27/52/10 on TB2.1 and 50/41/22 on DeepSWE. Coverage broadens, but preservation remains imperfect.

Figure 5 asks whether these gains encode reusable behavior or target-specific adaptation. Each transferred harness is evaluated on ten stratified target tasks at five trials per task, with the target-native winner as the reference in each direction.

Transfer is positive in both directions. The DeepSWE winner reaches 29/50 (58%) on TB2.1, 12 successes above the native winner. The preselected TB2.1 archive candidate reaches 21/50 (42%) on DeepSWE, four above the native winner. Its single timed-out trial counts as a failure. The +24- and +8-point gains show that the archive contains behavior that transfers across both benchmark distributions.

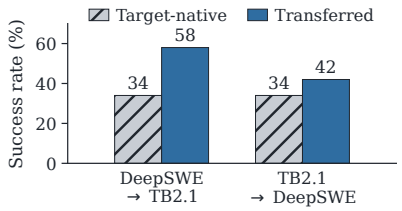


Figure 5: Cross-benchmark transfer over 50 trials per group: 24- and 8-point gains.

4.3 RQ3: Matched-budget baseline comparison

RQ3 compares eight methods initialized from the same root under a matched ten-proposal budget. We evaluate each selected harness on the same 20-task official TB2.1 slice at avg@5 and report search and downstream outcomes in Table 5.

All seven optimizers beat the 77/100 root. DarwinX leads the official slice at 84/100, followed by DGM and GEPA at 83/100; the remaining methods score 79–82/100. Search rank does not determine official order: DGM-H leads the panel (49/80) but reaches 80/100 officially, whereas DarwinX reaches 44/80 at proposal 2 and ranks first downstream. Although both methods first attain their best search scores at proposal 2, their official outcomes differ by four successes. The top five official scores span only 81–84, so the lead remains narrow. Appendix Figure 7 shows the full trajectories.

Table 5: Matched-budget results: best search score and first attainment on 16 tasks (80 outcomes), and official TB2.1 avg@5 on 20 tasks (100 outcomes); Δ is the point gain over root.

	Search panel		Official slice	
Method	Best	First	Succ.	Δ root
DarwinX	44/80	2	**84/100**	+7
DGM	41/80	3	83/100	+6
GEPA	43/80	3	83/100	+6
Meta-Harness	46/80	10	82/100	+5
ADAS	47/80	10	81/100	+4
DGM-H	**49/80**	2	80/100	+3
SkillOpt	45/80	4	79/100	+2
Root	39/80	–	77/100	0

Note: Adaptations are defined in Appendix Table 12.

Table 6: Official WAI pass@1 by application (%). Contextual columns reproduce the prior frozen snapshot; current columns use GPT-5.6-Sol Medium with OpenCode.

	Contextual external agents			GPT-5.6-Sol Medium + OpenCode		
Application	**Kimi-K2.5 vision**	**Qwen-3.5-Plus vision**	**Gemini 3 Flash + Browser Use**	**OpenCode base**	**DarwinX evolved**	**Δ**
Elation clinical records	50.0	54.2	81.7	86.67	98.33	+11.67
Elation prescriptions	23.3	41.7	80.8	80.00	97.50	+17.50
GitLab plan and track	39.3	37.1	63.6	53.57	92.86	+39.29
Gmail	70.0	56.7	75.0	83.33	90.00	+6.67
Gmail accounts and contacts	40.0	33.3	61.7	78.33	90.00	+11.67
Handshake career exploration	50.0	50.5	50.5	84.00	94.00	+10.00
Linear account settings	54.2	65.8	73.3	93.33	85.83	−7.50
PayPal wallet	70.7	71.4	88.6	97.86	88.57	−9.29
Superhuman general	15.0	25.8	50.0	78.33	75.83	−2.50
Xero invoicing	52.5	55.8	80.8	17.50	96.67	+79.17
Overall	43.3	48.3	69.3	75.48	**91.19**	**+15.71**

4.4 RQ4: WebArena-Infinity transfer and action audit

WebArena-Infinity (WAI) tests whether a harness selected on WAI-derived synthetic tasks transfers to the official interactive benchmark. This separates the synthetic signal used for selection from downstream performance across application-specific workflows. The frozen winner, node `e41c94ee`, improves the search panel from 79.17% to 80.56%. We evaluate it once on all 1,260 tasks from ten applications using GPT-5.6-Sol at medium reasoning and deterministic task verifiers; six bounded agent timeouts count as failures.

Official-task performance. The winner solves 1,149/1,260 tasks (91.19% pass@1), compared with the report baseline of 951/1,260 (75.48%), a gain of 15.71 points. Table 6 also reports the prior external-system snapshot for context; the current base and evolved columns use the same OpenCode and GPT-5.6-Sol configuration.

Table 7 and Figure 6 summarize the corresponding action-validity audit, while the application breakdown shows how the overall gain is distributed across heterogeneous workflows. Seven of the ten applications improve, led by Xero (+79.17), GitLab (+39.29), and Elation prescriptions (+17.50). PayPal (−9.29), Linear (−7.50), and Superhuman (−2.50) regress, showing that the aggregate gain still conceals application-level interference.

Table 7: WAI validity audit (%). E/P/R: evaluation-plane, exploit/privilege, and raw-state.

Metric	**Baseline**	**DarwinX**	**Δ**
Raw pass@1	75.48	91.19	+15.71
Audit-clean pass@1	74.84	91.19	+16.35
Logs audited	91.83	100.00	+8.17
Static invalid	9/1,157	0/1,260	−9
Verifier-passing invalid	8	0	−8
Invalid E/P/R	7/1/1	0/0/0	−9

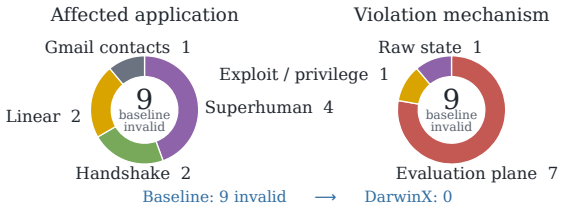


Figure 6: Static-invalid composition. Donuts partition the nine baseline violations by application and mechanism; the frozen winner has none.

Action-validity audit. We audit the base and evolved runs with the WAI anti-cheat auditor. A deterministic adapter preserves logged tool names and inputs while mapping JavaScript payloads from the two Playwright tools into the auditor's `browser_execute` field.

The audit covers all 951 credited base successes, 1,157 available base logs overall, and all 1,260 evolved trajectories. It finds nine static-invalid base trajectories: seven evaluation-plane accesses, one exploit or privilege escalation, and one raw-state mutation. Eight passed the task verifier; the evolved run contains none. Reclassifying invalid trajectories as failures reduces the base to 943/1,260 (74.84%) while leaving DarwinX at 1,149/1,260 (91.19%), increasing the gap to 16.35 points.

The violations span four applications, led by four in Superhuman, and are dominated by evaluation-plane access. The evolved harness therefore improves raw pass@1 while eliminating the verifier-passing shortcuts detected in the base run.

4.5 RQ5: How do DarwinX mechanisms shape evolution?

RQ5 combines a three-seed TB2.1 stability check with four DeepSWE-derived mechanism comparisons. Guard and memory use matched removals; consolidation and quality-diversity (QD) retention use activation-aware comparisons. All ablation arms share the frozen OpenCode evolvee and task pool, and each selection is evaluated on 24 blind tasks at avg@3, yielding 72 outcomes per arm. Table 8 reports both analyses; Appendix E.1 gives the protocols, paired uncertainty, and selected-edit cases.

Table 8: Seed stability and ablations. (a) Best search-panel score and 14-task TB2.1 blind avg@5. (b) Blind passes/72; Δ = enabled−control.

(a) Cross-seed stability

Run	Nodes	Search	TB2.1 eval
Seed 1	11	55.00%	48/70
Seed 2	11	52.50%	49/70
Seed 3	11	57.50%	48/70

(b) Mechanism ablations

Mechanism	Enabled	Control	Δ pp
Guard	48/72	44/72	+5.56
Memory	48/72	47/72	+1.39
Consolidation	50/72	50/72	0.00
QD retention	49/72	50/72	−1.39

Panel (a) shows that search-panel ordering does not transfer directly to blind evaluation. Panel scores span 52.50–57.50%, yet the three selected harnesses finish within one blind success: Seed 3 leads search but ties Seed 1 at 48/70, whereas Seed 2 ranks last in search and first on the blind panel at 49/70.

Panel (b) assigns distinct observed roles to the four mechanisms. The guard has the largest positive point estimate (48/72 versus 44/72), with all four lost successes on the primary panel and a tie on the extension panel. Memory shifts coverage, producing six more primary outcomes and five fewer extension outcomes for a one-pass aggregate gain. Consolidation selects a different descendant but leaves both continuations at 50/72. QD exposes two eligible candidates instead of one but selects the same seed harness; its one-pass difference therefore reflects reevaluation while the archive retains an additional alternative.

Overall, the guard best protects blind performance, memory redirects coverage, consolidation changes the continuation path without affecting aggregate score, and QD preserves diversity without displacing the incumbent. Appendix E.1 reports paired intervals; Appendix E.1.1 traces how selected edits combine loop-level control with reusable verification.

5 Discussion and Limitations

DarwinX separates search from evaluation: search panels guide proposals, while protected panels and original benchmarks assess the selected harness. Synthetic tasks are checked against reference solutions, untouched environments, and common errors. Because mechanisms interact, ablations characterize their joint effects; matched budgets and a frozen acting model control comparisons. Extending the archive to new models requires re-evaluating retained harnesses.

6 Related Work

DarwinX lies at the intersection of three lines of work: optimizing a designated agent artifact, evolving the executable scaffold around a model, and selecting candidates under noisy evaluation. We organize related work by these mechanisms; Appendix A compares the closest systems.

From prompts to harnesses. Prior work optimizes a designated artifact within a fixed outer optimizer. Prompt optimizers search instructions (Yang et al., 2023; Fernando et al., 2023; Yuksekgonul et al., 2024; Guo et al., 2024; Zhou et al., 2022); reflective methods reuse feedback or distill trajectories (Agrawal et al., 2025; Shinn et al., 2023; Madaan et al., 2023; Ni et al., 2026); and DSPy systems compile prompts, demonstrations, or textual parameters against data (Khattab et al., 2023; Cheng et al., 2024; Ou et al., 2025). Workflow methods search graphs of fixed components (Zhang et al., 2024; Zhuge et al., 2024; Hong et al., 2024), while skill agents build reusable libraries (Yang et al., 2026; Wang et al., 2023; Alzubi et al., 2026). HarnessX exposes typed prompts, tools, and control flow (Chen et al., 2026). DarwinX instead searches the repository harness, including skills, prompts, memory, tools, control flow, and bounded source changes, under shared preservation and selection rules.

Self-evolving agents. A second family edits an agent's executable scaffold. With a fixed meta-optimizer, ADAS generates agent programs (Hu et al., 2024), HarnessX evolves a typed harness through a staged pipeline (Chen et al., 2026), and Agentic Harness Engineering uses execution observability to evolve coding harnesses (Lin et al., 2026). Self-modifying systems also allow scaffold changes to affect later modification attempts (Zhang et al., 2025). This direction descends from the Success-Story Algorithm and Gödel machines (Schmidhuber et al., 1997; Schmidhuber, 2003; 2007) and includes STOP, Gödel Agent, SICA, and Live-SWE-agent (Zelikman et al., 2023; Yin et al., 2024; Robeyns et al., 2025; Xia et al., 2025). Coding-agent interfaces and code-generation benchmarks provide adjacent improvement targets without making self-modification the object (Yang et al., 2024; Chen et al., 2021).

DGM retains an open-ended archive and samples parents stochastically (Zhang et al., 2025); HGM, HyperAgents, and RQGM add lineage-aware scoring, broader domains, and co-evolved evaluators (Wang et al., 2025; Zhang et al., 2026; Iacob et al., 2026). A neighboring line evolves task-solution programs (Lehman et al., 2023; Romera-Paredes et al., 2024; Novikov et al., 2025), building on quality-diversity and open-ended search (Lehman & Stanley, 2011; Mouret & Clune, 2015; Wang et al., 2019; Fontaine et al., 2020). These methods provide much of the archive machinery used in agent evolution, but their usual fitness signals do not directly resolve conflicting task-level effects under stochastic evaluation. DARWINX therefore gates which descendants may steer search while retaining non-steering specialists for later recombination.

Selection under noisy evaluation. Agent scores vary across runs even at zero temperature (Bjarnason et al., 2026; Miller, 2024), motivating variance-aware reporting and consistency checks (Madaan et al., 2024; Yao et al., 2024) and echoing reproducibility work in reinforcement learning and architecture search (Agarwal et al., 2021; Li & Talwalkar, 2020). Classical bandit and racing methods eliminate candidates once evidence is sufficient (Auer et al., 2002; Audibert et al., 2010; Maron & Moore, 1993; Birattari et al., 2002), while successive-halving schedules allocate evaluation adaptively (Karnin et al., 2013; Jamieson & Talwalkar, 2016; Li et al., 2018; Jin & Branke, 2005). Self-evolving agents adopt parts of this machinery: DGM expands its evaluated task subset with confidence (Zhang et al., 2025), HGM uses posterior-guided selection (Wang et al., 2025), SkillOpt gates edits on held-out performance (Yang et al., 2026), and HarnessX combines per-edit checks with variant isolation (Chen et al., 2026). Related work studies reward manipulation (Skalse et al., 2022; Iacob et al., 2026) or improves model behavior rather than the harness (Huang et al., 2023). DARWINX separates these decisions: preservation probes and higher-fidelity confirmation determine steering eligibility, the fixed search panel selects the final harness, and protected and official evaluations occur only after selection is frozen.

Agent evaluation environments. Harness optimization requires task surfaces that separate search time selection from final measurement. Terminal-Bench evaluates broad terminal operation, and TerminalWorld adds a held-out terminal-task split (The Terminal-Bench Team, 2026; Chu et al., 2026); SWE-bench evaluates repository-level issue resolution (Jimenez et al., 2024); and browser-agent evaluation extends from WebArena to the larger verifiable environments of WebArena-Infinity (Zhou et al., 2024; Zhou, 2026). Our protocol implements this separation through task-disjoint synthetic search and protected panels grounded in these settings. It then freezes the selected harness before evaluation on the original benchmarks, preventing benchmark feedback from steering search and testing whether improvements transfer beyond synthetic tasks.

7 CONCLUSION

DARWINX improves a frozen-model agent by evolving its harness under a preserve-and-extend contract. A population archive retains lineages and reusable lessons, while merging, pruning, and consolidation manage specialization. Across terminal, repository, and browser tasks, frozen winners improve downstream performance, especially where the root is unreliable. A fixed model need not imply a fixed agent: evolving its tools, memory, skills, and control flow can accumulate capability without changing model weights. More broadly, archives could preserve alternative harnesses and evidence across changing tools and tasks, allowing useful variants to be re-evaluated rather than rediscovered. Appendix F develops this vision.

ETHICS STATEMENT

Agent self-improvement can increase both useful capability and the ability to exploit flawed evaluation surfaces. We therefore isolate protected evaluation tasks, scan candidate diffs for verifier tampering and answer hard-coding, count agent failures rather than silently discarding them, and retain complete genealogy and evaluation records. The method changes an agent harness, not model weights, but evolved artifacts should still be reviewed before deployment because benchmark verification is not a substitute for a deployment safety assessment.

REPRODUCIBILITY STATEMENT

Appendix C specifies the models, harness revision, search budgets, task partitions, sampling rates, selection rules, and baseline implementations. The supplementary artifact will contain configuration files, panel manifests, genealogy databases, generated figure sources, and scripts that reproduce the reported tables from raw evaluation records.

AI USE STATEMENT

Generative AI systems are part of the studied system: a frozen coding model acts as the evolvee, and a separate coding agent proposes harness changes. Generative AI tools also assisted with research execution by implementing and editing research code and generating the synthetic coding-task datasets used for search and evaluation. They were used to search for and identify candidate related work, draft sections of the paper, and polish the writing. They were not used to originate the central research idea or to formulate or prove mathematical claims. The authors reviewed the AI-assisted code, synthetic datasets, sources, citations, and manuscript and take responsibility for the final paper and artifacts.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory (COLT)*, pp. 41–53. Omnipress, 2010.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 11–18. Morgan Kaufmann, 2002.

Bjarni Haukur Bjarnason, André Silva, and Martin Monperrus. On randomness in agentic evals. *arXiv preprint arXiv:2602.07150*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A composable, adaptive, and evolvable agent harness foundry. *arXiv preprint arXiv:2606.14249*, 2026.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 71596–71642, 2024.

Zhaoyang Chu, Jiarui Hu, Xingyu Jiang, Pengyu Zou, Han Li, Chao Peng, Peter O'Hearn, Earl T. Barr, Mark Harman, Federica Sarro, and He Ye. TerminalWorld: Benchmarking agents on real-world terminal tasks. *arXiv preprint arXiv:2605.22535*, 2026.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 94–102, 2020.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, 2023.

Alex Iacob, Andrej Jovanović, William F Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, et al. The red queen Gödel machine: Co-evolving agents and their evaluators. *arXiv preprint arXiv:2606.26294*, 2026.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pp. 240–248. PMLR, 2016.

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.

Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pp. 1238–1246. PMLR, 2013.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer, 2023.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377. PMLR, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023.

Lovish Madaan, Aaditya K Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems*, volume 6, 1993.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, et al. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025. doi: 10.1016/j.aiopen.2025.11.004.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Jürgen Schmidhuber. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs/0309048*, 2003.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial General Intelligence*, pp. 199–226. Springer, 2007.

Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28(1): 105–130, 1997.

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.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9460–9471, 2022.

The Terminal-Bench Team. Terminal-bench 2.1. `https://www.tbench.ai/news/terminal-bench-2-1`, 2026. Version 2.1.

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.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Wenyi Wang, Piotr Piękos, Nanbo Li, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. *arXiv preprint arXiv:2510.21614*, 2025.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-SWE-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

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.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursive self-improvement. *arXiv preprint arXiv:2410.04444*, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

Shuyan Zhou. WebArena-Infinity: Generating browser environments with verifiable tasks at scale. `https://webarena.dev/webarena-infinity/`, March 2026. Accessed July 2026.

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*, 2024.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024.

A Detailed Method Positioning

Table 9 grounds the closest rows of the summary in Table 1. These systems overlap in what they edit, but use different search and retention rules when candidate effects conflict across tasks.

Table 9: Mechanism-level positioning of the closest harness-optimization systems. Descriptions follow the cited method reports; Appendix Table 12 separately defines our controlled adaptations.

Method	Evolved artifact	Search and retention	Cross-task handling
ADAS (Hu et al., 2024)	Agent/workflow code.	Archive-conditioned generation of new agent designs.	Candidate accuracy is evaluated on the target task distribution.
SkillOpt (Yang et al., 2026)	External skill instructions.	Iterative skill mutation with validation-gated retention.	Training and validation splits control promotion within the target domain.
DGM (Zhang et al., 2025)	Agent source code.	Growing archive with stochastic single-parent mutation.	Task-subset evaluation admits diverse descendants to the archive.
HarnessX (Chen et al., 2026)	Typed prompts, tools, and control flow.	Staged harness evolution with task-family variants.	A seesaw test detects regressions and isolates conflicting variants.
DarwinX	Skills, prompts, memory, tools, control flow, and bounded source changes.	Population archive with gain-weighted parent selection and optional cross-lineage merge.	Independent confirmation and a preservation probe gate which variants may steer search; non-steering specialists remain available for recombination.

B Evaluation Protocol

B.1 Campaigns and task surfaces

The frozen evolvee is OpenCode 1.18.16 with GPT-5.6-Sol at medium reasoning; Codex with GPT-5.6-Sol at xhigh reasoning proposes edits from task traces. All harnesses share this evolvee, native model transport, and task runtime. The verifier-backed pools contain synthetic terminal, repository, and WAI-structured coding tasks that reuse benchmark environments but exclude original goals, verifiers, tests, and solutions. Original benchmark tasks never influence proposals, search scores, or campaign memory.

Each campaign uses a fixed search panel of 40 TB2.1-derived tasks or 48 DeepSWE- or WAI-derived tasks, a disjoint 28-task protected panel, and the remaining tasks for proposals and off-target guards. Proposal tasks motivate edits, guards detect regressions, and the search panel ranks the 30 non-root nodes at $\mathrm{avg@3}$. The selected node and root are evaluated on the protected panel at $\mathrm{avg@5}$; frozen TB2.1 and DeepSWE harnesses are then compared on their complete original suites at $\mathrm{avg@5}$, while the WAI winner is evaluated once on all 1,260 tasks at pass@1 against the composite report baseline. Protected and original tasks cannot rerank nodes or guide later proposals. Appendices C.1 and C detail the partitions, operator schedule, and failure handling.

B.2 Matched-budget method comparison

The lightweight TB2.1 study assigns ten proposals each to DarwinX and controlled adaptations of DGM, ADAS, DGM-H, Meta-Harness, SkillOpt, and GEPA. All arms share the evolvee, proposer

model, split, transport, anti-cheat checks, and resource accounting; Appendix Table 12 specifies the adaptations. Candidates search on 16 tasks at avg@5. The root and two search leaders are re-ranked on a blind 14-task panel at avg@3, and each selected output is evaluated on the same 20-task original TB2.1 slice at avg@5. The controlled adaptations preserve each method's defining outer-loop mechanism, while the common evolvee, panels, and budget isolate search and selection behavior rather than reproduce every upstream system end to end.

B.3 Metrics and uncertainty

We report search-panel gains separately from downstream evidence. Downstream measures include paired avg@5 differences on original TB2.1 and DeepSWE, WAI pass@1 against the composite report baseline, and the common TB2.1-slice score. Cross-benchmark tests use ten stratified tasks with five trials each. Official TB2.1 and DeepSWE comparisons use paired 95% task-bootstrap intervals that preserve within-task trials. Blind mechanism contrasts use paired task-bootstrap intervals (Appendix E.1). The nine-application WAI analysis instead uses the matched OpenCode base for applications with complete runs.

C Reproducibility Details

The synthetic pools are central to the search protocol, so we first explain how their task content is kept separate from the source benchmarks while their execution environments remain informative. We then record the frozen campaign configuration, node-state semantics, and restart and concurrency behavior. Together, these details determine what may steer evolution, what is evaluated only after selection, and how concurrent workers share the search budget.

C.1 Synthetic task-pool construction

The synthetic tasks are task-disjoint from the benchmarks while retaining their execution environments. For TB2.1, synthesis reuses an isolated terminal container and its installed tools and resource limits; for DeepSWE, it reuses a repository image with its language, dependency, build, and test stack; and for WAI, it reuses the application deployment and code structure. The source task's instruction, goal, verifier, tests, and reference solution are withheld. Synthetic-task success measures improvement on the constructed distribution; original-benchmark evaluation measures downstream transfer.

Table 10 summarizes the frozen pool and evaluation surfaces for each task distribution.

Table 10: Frozen synthetic task surfaces. Search and protected panels are disjoint and do not supply proposal claims, guards, or campaign memory.

Distribution	Pool	Search	Protected
TB2.1-derived	134	40	28
DeepSWE-derived	170	48	28
WAI-derived	257	48	28

Environment alignment still provides a useful search signal. New tasks exercise the same operational substrate that the harness must navigate, including shell tools, file and data formats, repository structure, package conventions, and application code. Improvements in planning, tool use, debugging, and recovery can therefore be selected without exposing benchmark answers. Protected synthetic panels and original-benchmark evaluations then measure generalization after the search-selected harness has been frozen.

We construct and admit tasks in four stages:

1. **Recover environment context.** The synthesizer receives the container or application deployment context, resource limits, and available codebase. An audit ledger records every source read and excludes benchmark goals, verifiers, tests, and solutions.
2. **Author a new task triple.** For a sampled capability archetype, an authoring model creates a new instruction, verifier, and reference solution grounded in the recovered environment. These three artifacts are generated together so the task is executable and mechanically checkable.
3. **Run admission controls.** The reference solution must pass, the untouched environment must fail, and a plausible but vacuous or otherwise incorrect output must also fail. Prove-

Table 11: Shared configuration of the three full campaigns. Values are frozen before protected evaluation.

Component	Setting
Evolvee	OpenCode 1.18.16, fixed revision
Evolvee model	GPT-5.6-Sol, medium reasoning
Proposer	Codex, GPT-5.6-Sol, xhigh reasoning
Node budget	30 completed non-root nodes
Inner edit loop	at most 4 analyze–implement–review iterations
Claim / guard count	4 claimed tasks / 2 off-target guards
Node score	fixed search panel, avg@3
Final selection	best eligible search-panel node; later panels do not rerank
Protected score	28 tasks, avg@5
Downstream score	original TB2.1 / DeepSWE, root vs. winner at avg@5
Merge cadence	attempt every 6 completed nodes
Prune	enabled, rate 0.15 after minimum lineage depth
Consolidate	enabled; depth-adaptive, forced after 4 additive accepts
Parent depth exploration	probability 0.4
Archive	keep-all records plus archive-only lossy specialists
Anti-cheat	evaluator-tampering and answer-hard-coding scan

nance and canary scans reject tasks that contain source-benchmark markers or violate the permitted read surface.

4. **Screen and freeze partitions.** The unevolved agent is run repeatedly to estimate task difficulty. We retain both variance-band tasks, which expose measurable improvement headroom, and zero-pass walls for exploration, balance capability themes, and freeze disjoint claim, search, and protected subsets before evolution begins.

Original benchmark tasks remain outside synthesis and evolution. When a native downstream evaluation is available, it is run only after the selected harness has been frozen and cannot feed proposals, alter campaign memory, or rerank the winner.

Search panels are selected before evolution from replicated root measurements to provide capability-cluster coverage and an informative mixture of difficulty levels. The panel-construction seed and complete task lists are included in the artifact.

C.2 Full campaign configuration

The evolvee may add cue-gated skills under its mounted configuration root, register plugins through documented hooks, or make bounded core-code changes. Global prompt paths are forbidden and reverted before evaluation. The shared core churn budget is 40 changed lines for an existing core surface and 250 lines for a new core file. Root, evolved, and baseline harnesses share a 150-turn limit and the same task runtime. The campaign requires complete task and sample coverage for every panel evaluation, with two panel-repair attempts and two transient-infrastructure retries. Agent timeouts and malformed outputs count as failures; task-budget timeouts are not absorbed as infrastructure failures, and transient faults cannot yield positive reward.

C.3 Selection states

Every node has one persisted terminal status. *Completed* nodes cleared the search gate and may steer search. *Archived* nodes retain a real diff and task deltas but are ineligible as the deployed incumbent; they may contribute to recombination. *No-change* nodes produced no accepted candidate or returned to their parent. *Rejected* nodes violate an acceptance condition, and *failed* nodes terminate because the pipeline or evaluation did not complete. Only terminal scored records count as completed search steps.

C.4 Restart and concurrency semantics

Multiple workers share one SQLite genealogy with atomic task claims. The campaign-wide node count, rather than a per-process counter, meters the 30-node budget and six-node merge cadence.

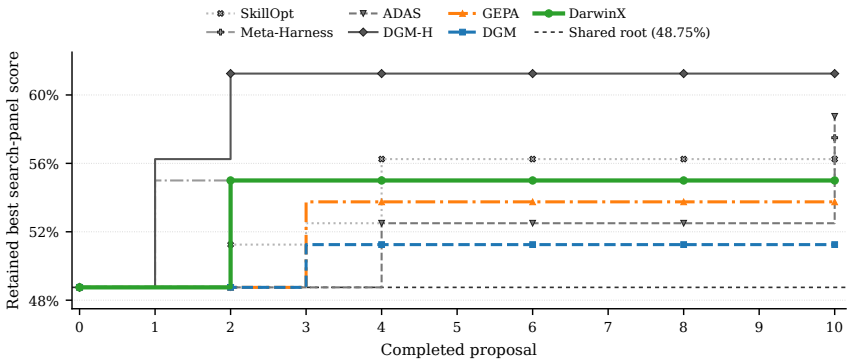


Figure 7: Best-so-far search-panel score after each completed proposal for seven matched-budget arms (shared 39/80 root; ten proposals; 16 synthetic TB2.1 tasks at avg@5). DGM-H peaks at 49/80 on this panel but records 80/100 on the official slice, compared with DARWINX's 84/100, illustrating the search-to-official rank reversal.

Restarts read the existing count and cannot grant a fresh budget. A final-panel record is reused only if its task set, sample count, metric, and coverage match the frozen protocol exactly.

D BASELINE ADAPTATIONS

The matched-budget comparison holds the evolvee, proposer transport, evaluation panels, and proposal count fixed while retaining each method's defining outer-loop mechanism. Table 12 summarizes these adaptations before the complete search trajectories.

Table 12: Defining mechanism retained by each matched-budget adaptation and the main departure from an exact upstream reproduction.

Method	Retained mechanism	Controlled adaptation
DGM	Score/child-count archive sampler; one target; open mutation; keep-all archive	Common proposer, OpenCode evolvee, and panel evaluator
ADAS	Root-based independent designs; source/score archive; novelty and correctness reflection	Common coding-task interface and fixed node budget
DGM-H	Editable task harness, meta-improvement procedure, and persistent hyper-state	Hosted OpenCode workspace and common evaluator
Meta-Harness	Root-based candidates with logical long-lived proposer history	One hosted candidate per node and bounded artifact surface
SkillOpt	One-skill mutation, bounded meta-memory, strict validation promotion	Fixed blind panel substitutes for the original dataset split
GEPA	Prompt-only mutation, per-example frontier sampling, same-batch strict acceptance	Common proposer transport and coding-task prompts

All arms retain method-specific mutation restrictions even when DARWINX's generalization gate is disabled. These controlled mechanism adaptations share one evolvee, proposer, evaluator, and set of anti-cheat checks.

D.1 MATCHED-BUDGET SEARCH TRAJECTORIES

The trajectories separate search behavior from downstream performance. DARWINX reaches its retained 55.00% score after two proposals, while DGM reaches 51.25% after three. DGM-H attains the highest search-panel maximum (61.25%) but falls below five other evolved arms on the official slice, where it records 80/100 and DARWINX records 84/100. The task sets and difficulty distributions differ, so these percentages are not points on one learning curve. The rank reversal instead shows that stronger optimization of the synthetic panel need not produce stronger benchmark generalization.

E Result Provenance and Extended Analyses

All reported results are extracted from task-level evaluation records rather than rendered figures. Extraction requires one frozen task manifest per evaluation, complete task and sample coverage, disjoint search and protected surfaces, a unique search-selected commit, and exclusion of superseded runs. TB2.1 and DeepSWE confidence intervals resample tasks while preserving the five within-task trials. Infrastructure and setup failures are replaced according to the frozen correction policy, whereas bounded agent timeouts remain failures. The final official ledgers contain 445 TB2.1 outcomes, 565 DeepSWE outcomes, and 1,260 WAI outcomes. None of these evaluations changes the winner selected on the synthetic search panel.

E.1 Mechanism-ablation protocol and uncertainty

The seed-stability study repeats the lightweight TB2.1 search three times from the shared root and freezes each run's selected harness before evaluation. Figure 8 shows the paths to those selections.

Each harness is evaluated on the same 14-task blind panel with five samples per task. All three searches evaluate the shared root and ten scored proposals; their blind scores span one success out of 70 (Table 8). Seed 2 improves on the first evaluated proposal to 52.50%, Seed 1 reaches 55.00% at evaluated node 3, and Seed 3 rises in stages to 57.50% at node 5. No later evaluated node exceeds these retained maxima. The searches therefore follow distinct optimization paths while selecting harnesses with nearly identical blind outcomes.

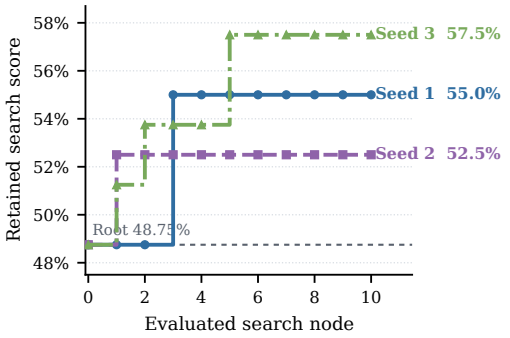


Figure 8: Three-seed TB2.1 search trajectories on the shared 16-task panel at avg@5. Curves track the best eligible score from the 48.75% root.

The mechanism study uses two matched designs on the DeepSWE-derived synthetic pool. The original five-arm study runs four search nodes from one immutable root and removes one mechanism per arm. Its complete control activates the preservation guard and campaign memory, so those two removal contrasts directly measure executed mechanisms. Consolidation and QD retention require activation-aware follow-ups: consolidation compares equal two-node continuations rooted before and after a completed consolidation, while QD compares archive-enabled and incumbent-only selection from the same historical pair. Search tasks and the 24 blind tasks are disjoint, and selection is frozen before the three blind trials per task.

Table 13: Mechanism estimates underlying Table 8. Blind passes report the mechanism-enabled arm and its matched control, each with 72 outcomes. Intervals use paired task bootstrap over the 24 blind tasks.

Mechanism	Enabled arm / control	Blind passes	Δ pp	95% CI
Preservation guard	Complete / no guard	48/72 vs. 44/72	+5.56	[−6.94, 18.06]
Campaign memory	Complete / no memory	48/72 vs. 47/72	+1.39	[−13.89, 16.67]
Consolidation	Post- / pre-consolidation root	50/72 vs. 50/72	0.00	[−11.11, 11.11]
QD retention	Archive / incumbent-only	49/72 vs. 50/72	−1.39	[−11.11, 8.33]

For the guard, the complete system records 25/36 on the primary panel versus 21/36 without the guard and ties 23/36 on the extension panel. Campaign memory changes the panel profile in both directions: 25/36 versus 19/36 on the primary panel and 23/36 versus 28/36 on the extension panel. The consolidation arms select different commits, establishing a changed continuation path despite their equal aggregate score. The two QD arms select the same seed commit; the archive-enabled search has two eligible candidates and the incumbent-only search has one. The observed QD score difference therefore comes from repeated blind evaluation of the shared selection, while the search result records the archive's effect on candidate retention.

All four paired intervals include zero. The score estimates therefore describe these matched short-horizon runs, while the path and coverage observations identify how each mechanism changed the executed search.

E.1.1 SELECTED-EDIT CASE STUDIES

The selected winners combine a loop-level control edit with a compact reusable verification module. These capabilities are implemented as routed TypeScript modules rather than standalone cue-gated `SKILL.md` files.

TB2.1: acceptance is an explicit verdict. The selected TB2.1 edit routes the post-mutation decision in `mutation-verification.ts` through a new `acceptance-verdict.ts` module. The loop now stops only on a complete verdict; otherwise it distinguishes unavailable evidence from targeted recovery. The module combines acceptance clauses with checks that occur after the final mutation and directs recovery to one of three concrete failures: an invalid ledger, uncovered clauses, or unusable evidence. This separates the control decision from the reusable capability that evaluates evidence.

DeepSWE: completion carries unresolved evidence forward. The selected DeepSWE edit changes `completion-audit.ts` to delegate mutation detection and failure tracking to a new `completion-evidence.ts` module. The audit now recognizes workspace changes even when the transcript contains no patch part, retains unresolved tool failures until a matching successful rerun, and inserts that bounded evidence into the final audit. Here again, the core loop schedules the check while a reusable module defines the verification behavior. The two cases show how benchmark-specific selection can produce different inherited controls: clause-complete acceptance for TB2.1 and persistent completion evidence for DeepSWE.

E.2 WAI ACTION-SURFACE INVENTORY

The trajectory audit in Section 4.4 classifies task validity. A separate inventory checks the complete evolved run's available action surface and logged tool inputs (Table 14). All 40,783 calls use one of 16 structured Playwright tools. The inventory finds no non-Playwright or code-capable tool call and no input matching the recorded evaluation-plane, privileged-host, or raw-state patterns.

Table 14: Complete structured action surface for the evolved WAI run. Every logged call belongs to one of these Playwright tools; the scan is separate from the trajectory-level validity classifier in the main text.

Playwright tool	Calls	Playwright tool	Calls
`browser_click`	14,826	`browser_snapshot`	19,273
`browser_drag`	45	`browser_take_screenshot`	200
`browser_drop`	4	`browser_fill_form`	475
`browser_find`	2,072	`browser_hover`	139
`browser_navigate`	1,430	`browser_navigate_back`	203
`browser_press_key`	587	`browser_resize`	85
`browser_select_option`	55	`browser_tabs`	22
`browser_type`	1,277	`browser_wait_for`	90
		Total	40,783

F VISION: CONTINUAL HARNESS EVOLUTION

DARWINX frames the model as one component of a longer-lived, evolving agent system. A population archive preserves alternative harnesses, evidence, and reusable lessons across changes in models, tools, and tasks, enabling re-evaluation rather than rediscovery. Future systems could connect selection to task generation, evaluator improvement, and deployment feedback while separating proposal from acceptance. Human operators would set preservation constraints and protected evaluations, while automated search explores within them. This supports continual improvement that remains inspectable, reversible, and evidence-bound.