CYBERGRAFT: EVOLVING THE HARNESS DESIGN SPACE THROUGH HIGH-LEVERAGE MECHANISM MINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic harness evolution improves agents without model retraining through trajectory analysis and candidate evaluation. Yet even with accurate root-cause diagnosis, finding effective revisions in a vast design space requires extensive trial and error. Fixed seed harnesses adapt poorly to heterogeneous tasks, unconstrained edits or rigid module recombination can produce low-value candidates, and useful external mechanisms remain difficult to discover and consolidate for reuse. We propose CYBERGRAFT, a framework that continually evolves the harness design space by constructing, reusing, and curating high-leverage mechanism assets. Each asset combines a carefully engineered, reusable mechanism package with a scoped, quantitatively grounded benefit–risk profile. Diagnosis-guided discovery seeks implementations addressing observed needs, while performance-guided discovery investigates strong external agents for alternatives beyond the current diagnosis. Contribution analysis identifies promising core mechanisms, and dependency-guided reconstruction refines their extraction boundaries through validation. Separate comparisons assess source contribution, reconstruction fidelity, and conditional transfer utility. Evidence-guided retrieval supports task-conditioned seed assembly and subsequent candidate generation. Accepted and rejected trials refine packages and profiles, while unmet needs drive further discovery, accumulating both implementations and evidence for future optimization. On Evo-Bench across Search, Office, and General tasks, CYBERGRAFT achieves an Overall score of 53.6 and an AnytimeVal of 54.8, exceeding native evolution by 7.3 and 4.7 points under matched online resource limits with a prebuilt library. Ablations show that continual discovery and adaptive boundaries improve final performance, contribution screening reduces online cost by 27.6% at the same reported performance, and profile-guided reuse raises gate acceptance by 19.8 percentage points.

1 INTRODUCTION

Agent performance depends on the harness that organizes tools, memory, control flow, and recovery around a language model (Yao et al., 2022; Yang et al., 2024; Packer et al., 2023). Improving this execution layer offers a route to stronger agents without retraining the underlying model. Automated optimization has progressed from prompts and workflows to executable agent systems and complete harnesses (Yang et al., 2023; Zhang et al., 2025c;b; Lee et al., 2026). Recent methods use execution trajectories to diagnose weaknesses, propose revisions, and evaluate their effects (Lin et al., 2026; Jiang et al., 2026; Wu et al., 2026).

However, locating a weakness does not determine an effective implementation of its solution. Even an accurate diagnosis can leave candidate generation dependent on repeated implementation and evaluation. Evo-Bench reveals a related empirical challenge: evolutionary gains vary substantially across domains. Search improves strongly and leading evolvers surpass human-engineered harnesses on General tasks, while Office gains remain limited and many runs saturate early (Huang et al., 2026). These observations motivate examining the engineering resources available to search alongside the feedback directing it. Specialized workflows that are difficult to synthesize within a

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.

limited budget may already exist in external implementations, supporting task-adapted initialization and subsequent revisions.

Modular search provides a foundation for reusing such designs. AgentSquare, JIT-Agent, and ModularRSI organize generation, evolution, or composition through predefined functional modules (Shang et al., 2024; Zhang et al., 2026; Wu et al., 2026). Their libraries can expand, and feedback can inform selection. External skill acquisition further demonstrates how repositories and documentation can supply reusable capabilities (Yan et al., 2026; Shen et al., 2026). These advances leave a complementary problem: determining which mechanisms in heterogeneous donors merit extraction, what cooperating components they require, and under which conditions their reuse is beneficial. Standardized interfaces alone do not resolve these questions, while indiscriminate extraction can populate a library with candidates whose value still requires repeated investigation.

This motivates our central question: *Can prior investments in mechanism implementation and validation become measurable advantages for subsequent harness optimization?* Our premise is that useful reuse requires both an executable implementation and evidence about its conditional effects. A retrieval-recovery mechanism, for example, may depend jointly on retry control, source switching, and shared state; its value also depends on failure conditions and execution cost. Retaining the necessary cooperation supports *constructibility*, while recording measured benefits, risks, and applicability supports *predictability*. Selectively accumulating both can change which harness designs are economical to construct and which candidates are sufficiently supported to evaluate.

Such reuse can change how the optimization budget is allocated. Evidence remains reusable within its recorded version, binding, and applicability scope, allowing new trials to focus on changed conditions, adaptations, and compositions. The aim is to concentrate evaluation on unresolved questions and carry forward established findings, making prior validation useful beyond the original experiment.

We propose CYBERGRAFT, a framework that continually evolves the harness design space by constructing, reusing, and curating *high-leverage mechanism assets* (Figure 1). Each asset combines a carefully engineered, reusable mechanism package with a scoped, quantitatively grounded benefit–risk profile. Diagnosis-guided discovery seeks implementations addressing current needs, while performance-guided discovery investigates strong donors for alternatives beyond the current diagnosis. Contribution analysis screens mechanisms for task-relevant effects, and dependency analysis guides reconstruction with boundaries refined through validation. Separate comparisons assess source contribution, reconstruction fidelity, and conditional transfer utility. Evidence-guided retrieval then supports task-conditioned seed assembly and subsequent candidate proposals. Accepted and rejected trials update packages and profiles, while unmet needs direct further discovery, connecting harness improvement with the accumulation of reusable implementations and evidence.

We evaluate CYBERGRAFT on Evo-Bench across Search, Office, and General tasks. With a prebuilt asset library and matched online resource limits, CYBERGRAFT achieves an Overall score of 53.6 and an AnytimeVal of 54.8, exceeding native evolution by 7.3 and 4.7 points, respectively. Held-out performance improves in every domain, including a 9.3-point gain on Office. Ablations support distinct roles for the proposed components: continual discovery and adaptive boundaries improve final performance, contribution screening reduces online cost by 27.6% at the same reported performance, and quantitative profile reuse raises gate acceptance by 19.8 percentage points. Initial library construction is accounted for separately.

Our contributions are threefold:

- **An asset-based formulation of harness design-space evolution.** We represent accumulated engineering knowledge as reusable mechanism packages paired with scoped, quantitatively grounded benefit–risk profiles. This formulation characterizes how asset accumulation reshapes effective search through two complementary dimensions: *constructibility*, which concerns the cost of implementing candidates, and *predictability*, which concerns the evidence available to anticipate their conditional effects.

- **A framework for selective mechanism acquisition and reuse.** We introduce CYBERGRAFT, which integrates diagnosis-guided and performance-guided discovery with contribution screening and dependency-guided reconstruction. Extraction boundaries are refined through validation, with source contribution, reconstruction fidelity, and transfer utility

assessed separately. The resulting assets support task-conditioned seed assembly and candidate generation, while evaluation feedback and emerging needs guide library curation.

- **An empirical assessment of performance and search efficiency.** We evaluate CYBERGRAFT across Search, Office, and General tasks on Evo-Bench under matched online resource limits, with initial library construction accounted for separately. Beyond improvements in held-out performance and optimization progress, controlled ablations distinguish the performance benefits of continual discovery and adaptive boundaries from the online cost savings associated with contribution screening and quantitative profile reuse.

2 RELATED WORK

2.1 AUTOMATIC HARNESS EVOLUTION

Component-level optimization spans prompts (Zhou et al., 2022; Yang et al., 2023; Pryzant et al., 2023; Opsahl-Ong et al., 2024; Agrawal et al., 2025), skills (Alzubi et al., 2026; Yang et al., 2026; Peng et al., 2026), and memory systems (Xu et al., 2025; Zhang et al., 2025a). Search has expanded to code-defined agents and workflows (Hu et al., 2025; Zhang et al., 2025c;b) and complete harness designs. HarnessX draws inspiration from reinforcement learning in symbolic space: the harness and its accumulated experience correspond to the state, code edits serve as actions, and execution trajectories and verification results provide feedback (Chen et al., 2026c). Across these methods, the evolution workflow can be broadly characterized as **Seed Harness Initialization** $\rightarrow$ **Rollout** $\rightarrow$ **Trajectory Analysis and Attribution** $\rightarrow$ **Candidate Generation** $\rightarrow$ **Gated Update**. Meta-Harness exposes past source code, scores, and traces (Lee et al., 2026); AHE emphasizes component, experience, and decision observability (Lin et al., 2026); and HarnessEvolve uses reference trajectories and quality/performance gates (Jiang et al., 2026). HarnessFix links runtime traces to harness artifacts for targeted repair and regression validation (Chen et al., 2026b). Candidate generation depends not only on feedback but also on how mechanisms are sourced and organized into the search space. CYBERGRAFT complements feedback-driven revision by evolving the mechanism resources available for seed harness assembly and candidate generation.

2.2 AGENT MODULARIZATION AND COMPOSITIONAL SEARCH

Modular representations provide explicit boundaries for search. AgentSquare (Shang et al., 2024) expands its design space by generating and recombining standardized modules for planning, reasoning, tool use, and memory. JIT-Agent uses a four-module protocol and feedback-updated archive for task-specific harness synthesis (Zhang et al., 2026). ModularRSI evolves five functional modules and accumulates an expanding function library for task-aware composition (Wu et al., 2026). HarnessBank maintains a semantic gene bank of high-performing harnesses for recombination and gated screening (Luo et al., 2026). AHE exposes seven semantic component types as editable files (Lin et al., 2026); HarnessX composes typed processors at lifecycle hooks (Chen et al., 2026c); and DeepSeek Harness makes core runtime mechanisms replaceable through Cordis plugins, service dependencies, and lifecycle management (DeepSeek-AI, 2026). CYBERGRAFT uses contribution analysis to select high-impact core components and dependency analysis to define extraction boundaries without fixed functional slots. It reconstructs selected mechanisms and their necessary dependencies as reusable capability packages that can combine skills, tools, and plugins. Behavioral preservation and target utility are assessed separately.

2.3 OPEN-WORLD DISCOVERY AND CONTINUAL ASSET CONSOLIDATION

Reusable experience takes several forms, including reflective memories in Reflexion (Shinn et al., 2023), executable skills in Voyager (Wang et al., 2023), and evolving context playbooks in ACE (Zhang et al., 2025d). OpenSkill, SkillFoundry, and Repo-To-Skill acquire skills from external resources (Yan et al., 2026; Shen et al., 2026; Chen et al., 2026a); ToolRosella converts repositories into standardized tools (Di et al., 2026). AHE brings external exploration into harness initialization by acquiring skills from framework code and public references (Lin et al., 2026). Beyond acquisition, SkillNet organizes reusable capabilities through a skill ontology, relationships, and evaluation, while WikiSkill retains execution knowledge to guide subsequent skill evolution (Liang et al., 2026;

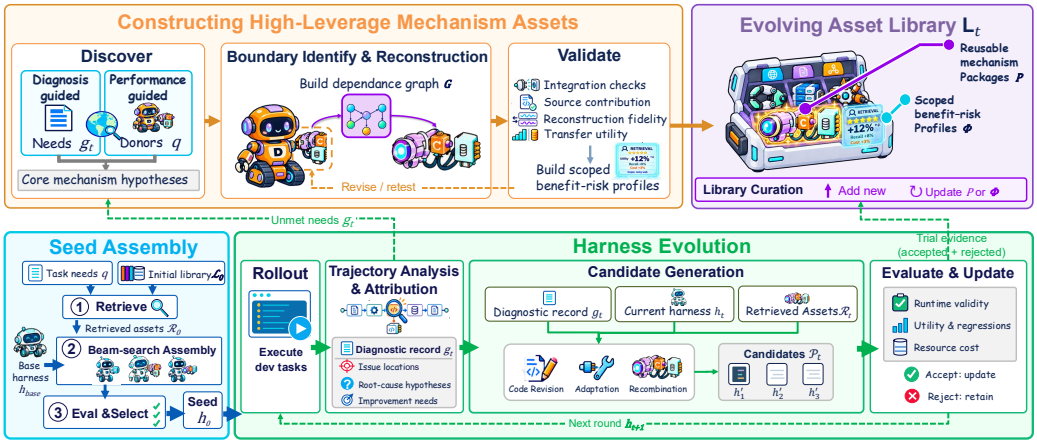


Figure 1: **CyberGraft overview.** CyberGraft evolves the harness design space by constructing, reusing, and curating high-leverage mechanism assets: reusable packages paired with scoped, quantitatively grounded benefit–risk profiles. Evidence-guided retrieval supports task-conditioned seed assembly and trajectory-guided candidate generation. Accepted and rejected trials inform package and profile updates, while unmet needs guide further external discovery. Solid arrows indicate data or asset flow; dashed arrows indicate feedback.

Tang et al., 2026). CyberGraft continually discovers external mechanisms in response to needs arising during harness evolution and selectively consolidates high-leverage assets for future reuse.

3 Problem Formulation

Given a fixed model π_θ, target tasks, and access to external agent implementations, we study the problem of optimizing a harness $h \in \mathcal{H}$ under a limited budget, where $\mathcal{H}$ denotes the space of executable harness designs. The objective is to improve task performance while continually evolving the harness design space through a library $\mathcal{L}_t$ of reusable mechanisms and accumulated evidence, supporting more efficient search across successive optimization runs.

3.1 Harnesses, Donors, and Reusable Assets

A harness h defines the agent's components and execution workflow around model π_θ, producing trajectories $\tau \sim \pi_\theta(\cdot \mid x; h)$ for task x. A donor d is an external agent implementation at a pinned repository revision. A mechanism is a concrete strategy implemented by one or more cooperating components. To support reuse, CyberGraft represents a selected mechanism as an asset combining an executable package P with a scoped benefit–risk profile Φ:

$$a = (P, \Phi, \rho, \mathcal{V}), \tag{1}$$

where ρ records provenance and versions, and $\mathcal{V}$ stores evaluation records.

Mechanism packages. A package contains the selected mechanism and the components required to preserve its behavior:

$$P = (m, G, \kappa, \delta, I), \tag{2}$$

where m describes the intended behavior and scope, G specifies the extraction boundary and internal dependencies, κ defines the runtime contract, δ declares external dependencies, and I contains the implementation and host bindings. A package may encapsulate cooperating components or combine existing assets; a composition configuration specifies member asset versions, parameter settings, interface bindings, and ordering constraints.

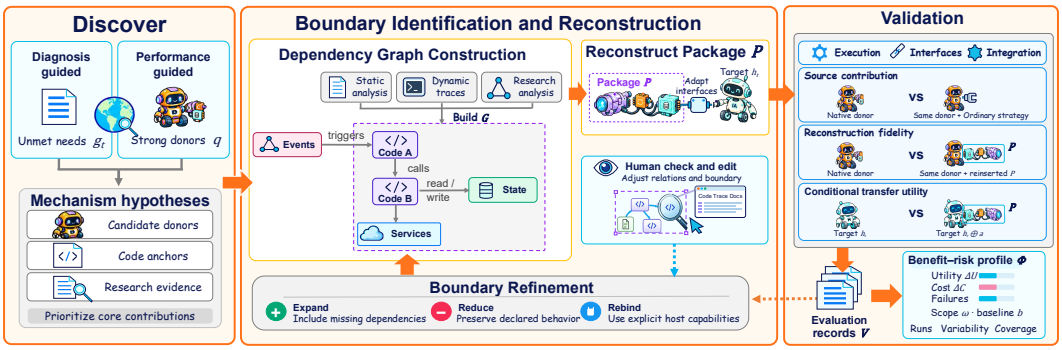


Figure 2: **Constructing high-leverage mechanism assets.** Diagnosis-guided and performance-guided discovery propose core mechanism hypotheses. Dependency graphs guide package reconstruction, while validation feedback refines extraction boundaries through expansion, reduction, and rebinding. Separate comparisons assess source contribution, reconstruction fidelity, and conditional transfer utility. Evaluation records preserve these comparisons, and comparable target-harness evaluations inform scoped benefit–risk profiles.

Benefit–risk profiles. For execution conditions ω and a comparison baseline b, the profile summarizes:

$$\Phi(a;\omega,b) = \Big(\underbrace{\widehat{\Delta U}}_{\text{utility change}} , \underbrace{\widehat{\Delta C}}_{\text{cost change}} , \underbrace{\mathcal{F}}_{\text{failure summary}} , \underbrace{(n, s_{\Delta U}, \mathcal{T})}_{\text{evaluation summary}} \Big). \tag{3}$$

Here ω specifies task, host, and execution conditions. The estimates $\widehat{\Delta U}$ and $\widehat{\Delta C}$ quantify utility and execution-cost changes relative to b; $\mathcal{F}$ summarizes failure modes, triggering conditions, and observed frequencies. The evaluation summary records the number of valid trials n, the sample standard deviation of utility changes $s_{\Delta U}$, and tested triggering scenarios $\mathcal{T}$. Profiles summarize comparable controlled evaluations in target harnesses, while $\mathcal{V}$ preserves the corresponding trial conditions, asset versions, baselines, and observed outcomes.

3.2 An Evolving Harness Design Space

High-leverage mechanism assets in $\mathcal{L}_t$ combine carefully engineered compositions of interdependent atomic mechanisms with reliable, quantitatively grounded benefit–risk profiles. As these assets accumulate and are refined, they change the cost of constructing designs in $\mathcal{H}$ and the evidence available to anticipate their effects. We characterize this evolution of the effective design space along two dimensions: (i) Constructibility: constructing complex, high-quality candidates at lower cost through mechanism reuse; and (ii) Predictability: forming evidence-backed expectations of conditional mechanism effects before new trials, reducing repeated validation costs.

4 CyberGraft

4.1 Overview

CyberGraft couples harness evolution with asset construction, reuse, and curation (Figure 1). Task requirements q guide seed assembly; execution trajectories yield diagnostic records g_t containing issue locations, cause hypotheses, and improvement needs. *Construction* (Section 4.2) discovers core mechanisms, reconstructs their necessary dependencies, and establishes scoped profiles. *Reuse* (Section 4.3) retrieves assets for seed assembly and candidate proposals. *Curation* (Section 4.4) incorporates trial outcomes into packages and profiles and directs further discovery. This loop evolves the effective design space through reusable implementations and accumulated evidence.

4.2 CONSTRUCTING HIGH-LEVERAGE MECHANISM ASSETS

CYBERGRAFT constructs the assets defined in Section 3.1 through discovery, dependency-guided reconstruction, and validation (Figure 2).

Discover. Inspired by how engineers seek alternative solutions in high-performing systems, CYBERGRAFT pursues two parallel discovery routes. *Diagnosis-guided discovery* searches for external mechanisms addressing unmet needs identified in g_t. *Performance-guided discovery* investigates donors relevant to task requirements q, seeking mechanisms beyond the current diagnosis. Together, these routes reduce dependence on potentially inaccurate diagnoses or overly narrow hypotheses. Research prioritizes mechanisms with plausible task-relevant contributions over indiscriminate extraction. Its output comprises candidate donors, mechanism localization hypotheses with code anchors, and supporting research materials for subsequent dependency analysis.

Boundary identification and reconstruction. Module boundaries need not follow a system's processing decomposition (Parnas, 1972). Here, extraction follows necessary cooperation rather than rigid partitions based on file layout or fixed functional slots.

Dependency graph construction. Starting from discovery records, CYBERGRAFT constructs a dependency graph using a code property graph representation (Yamaguchi et al., 2014) enriched with mechanism semantics. Nodes represent code units at adjustable granularity, state, events, and services; edges capture calls, state access, event triggering and ordering, and service dependencies. Static analysis establishes code relationships, dynamic traces reveal runtime interactions, and research materials and model analysis supply design intent and framework semantics. Relations retain their supporting sources, enabling visual exploration and human revision of dependencies and boundary constraints.

Reconstruction and boundary refinement. Selected components are reconstructed as a package P with interfaces adapted to the target harness. Validation feedback guides *expansion* to include missing dependencies, *reduction* through behavior-preserving group removal, and *rebinding* to host-provided capabilities. Every necessary dependency must be packaged or explicitly provided by the host. Implementation faults trigger repair; unsupported mechanism effects prompt revision of the mechanism or applicability hypothesis. Refinement stops when the declared reconstruction criteria are met or the acquisition budget is exhausted.

Validation. Validation checks package execution, compliance with declared interfaces and dependencies, and integration with the target harness. Controlled comparisons assess (i) *source contribution*: the native donor versus an ordinary-strategy replacement preserving basic functionality; (ii) *reconstruction fidelity*: the native donor versus the donor with the reconstructed package reinserted, following the principle of validating individual transformations (Pnueli et al., 1998); and (iii) *conditional transfer utility*: matched target harnesses with and without the asset.

Trigger conditions, metrics, and acceptance tolerances are predefined. Development cases guide repair and regression checks; independent confirmation cases withheld from refinement assess the final package. All comparisons are recorded in $\mathcal{V}$, while comparable target-harness evaluations are summarized in Φ as defined in Section 3.1. Packages meeting the declared criteria are passed to library curation with their profiles.

4.3 REUSING ASSETS FOR HARNESS EVOLUTION

Asset reuse supports seed assembly and candidate generation. Both use gate $\mathsf{Gate}(h' \mid h)$ to require valid execution, execution cost within the limit, and no degradation beyond prespecified tolerances in protected metrics relative to h. Harness candidates are ordered by measured development utility, with ties favoring lower execution cost.

Evidence-guided retrieval. Given a request ξ and harness h, retrieval uses the domain–subdomain index and functional tags to locate behaviorally relevant assets across domains. It retains assets that can be integrated into h and checks their profiles for transfer evidence applicable to the current request and target harness. Here $\xi = q$ for seed assembly and $\xi = (q, g_t)$ during evolution. Assets

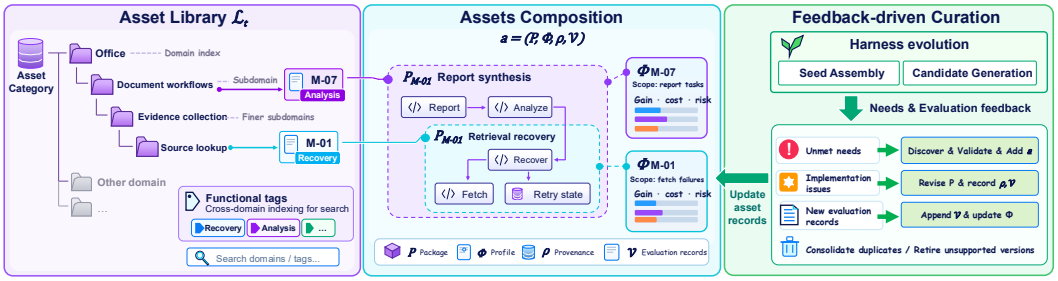


Figure 3: **Asset organization and curation.** Domain–subdomain indices and functional tags organize assets for retrieval. Assets at different composition levels retain their own scopes and benefit–risk profiles. Curation updates packages and profiles and manages asset admission and retirement.

with comparable transfer evidence are ranked by

$$s(a \mid \xi, h) = \widehat{\Delta U}_a - \beta s_{\Delta U,a} - \lambda \widehat{c}_{\text{adapt}}(a, h), \tag{4}$$

where $\widehat{\Delta U}_a$ and $s_{\Delta U,a}$ are the profile's utility-change estimate and dispersion under comparable task, host, and baseline conditions; $\widehat{c}_{\text{adapt}}$ estimates adaptation effort, and $\beta, \lambda \geq 0$ control the penalties. Composed assets are scored using their own profiles. Retrieval returns up to K assets, reserving up to K_{p} slots for relevance-ranked assets lacking comparable transfer evidence. Unmet requirements trigger Discover (Section 4.2); validated additions reenter retrieval.

Task-conditioned seed assembly. Given q, $\mathcal{L}_0$, and budget B_{seed}, beam search starts from the feasible base harness h_{base}. Each parent is expanded with retrieved assets, considering at most J configuration, binding, and ordering proposals per asset. Deduplicated candidates undergo execution checks and development-task evaluation. Each round retains up to W executable candidates with the highest measured development utility. All evaluated candidates, including the base, are archived; intermediate candidates need not outperform the base. All attempts consume the budget, and search stops at exhaustion or when no untested expansion remains. The seed h_0 is the highest-ranked archived candidate passing $\mathsf{Gate}(\cdot \mid h_{\text{base}})$.

Candidate generation and gated update. Diagnostics g_t and retrieved assets guide replacements, adaptations, recombinations, and direct code edits to h_t. Evidence is reused within its recorded version, binding, and applicability scope; affected behavioral and utility claims receive fresh checks. After development-task evaluation, the highest-ranked candidate passing $\mathsf{Gate}(\cdot \mid h_t)$ becomes h_{t+1} if it outranks h_t; otherwise $h_{t+1} = h_t$. Both accepted and rejected trials are archived and inform library curation.

4.4 Curating the Asset Library

Asset organization. The library $\mathcal{L}_t$ stores independently retrievable assets (Figure 3). Assets may incorporate other assets while retaining their own behavioral scopes and benefit–risk profiles. A domain–subdomain index and functional tags support browsing and retrieval. Classification follows each asset's own role and scope, so an asset and its constituent assets may belong to entirely different categories.

Feedback-driven curation. Curation centers on asset admission, package revision, and profile updates. Unmet requirements drive external discovery, with new assets admitted after validation. Implementation or adaptation issues trigger package revision and validation of new versions. New valid evaluation records update conditional benefit–risk profiles used in subsequent retrieval (Section 4.3). Duplicate records may be consolidated, and versions that no longer meet reuse requirements are retired from active retrieval, while preserving their histories.

Table 1: **Overall effectiveness on Evo-Bench.** Both evolving methods use GPT-5.6 Sol. † denotes officially reported single-run results; CYBERGRAFT reports means over three independent runs. Artificial Harness uses the listed systems for Search, Office, and General, respectively.

Method	Search	Office	General	Overall	AnytimeVal
CodeAct†	11.7	38.4	48.4	29.7	–
Artificial Harness† (MiroFlow + Stirrup + Claw-Eval)	46.7	43.9	56.3	47.5	–
Evo-Bench evolution†	44.5	41.6	59.4	46.3	50.1
CYBERGRAFT	51.8	50.9	62.5	53.6	54.8

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks and metrics. We use Evo-Bench's 160 development and 448 held-out tasks (Huang et al., 2026): BrowseComp and HLE for Search (Wei et al., 2025; Center for AI Safety et al., 2026), GDPval and APEX-Agents for Office (Patwardhan et al., 2025; Vidgen et al., 2026), and Claw-Eval for General (Ye et al., 2026). Development tasks guide optimization; the selected harness is frozen before test evaluation. Following Evo-Bench, we use Pass@1 for BrowseComp, HLE, and APEX-Agents, rubric-mean scores for GDPval, and Pass^3 for Claw-Eval (success in all three trials). Overall aggregates domain scores with weights 2:2:1. AnytimeVal averages best-so-far development scores over the iteration budget, carrying the best score forward after early termination.

Implementation details. We use DeepSeek-V4-Flash for task execution, GPT-5.6 Sol for evolution, and Qwen3.7-Plus for model-based evaluation. Each optimization run is limited to 1,000 evolver steps, 20 formal validation evaluations, and 48 hours. We report means over three independent optimization runs. The main setting uses a prebuilt library of 45 validated mechanism assets, constructed by screening 23 candidate donor designs and implementations without held-out test feedback. Online costs include seed assembly, retrieval, subsequent discovery and reconstruction, and candidate evaluation, including failed attempts. Initial library construction and final test execution are excluded from online cost.

5.2 MAIN RESULTS

Table 1 compares CYBERGRAFT with Evo-Bench's reported CodeAct baseline (Wang et al., 2024), domain-specific Artificial Harnesses—MiroFlow, Stirrup, and Claw-Eval (MiroMind, 2025; Artificial Analysis, 2026; Ye et al., 2026)—and native evolution (Huang et al., 2026). Both evolving methods use GPT-5.6 Sol and matched online limits; CYBERGRAFT additionally starts with a prebuilt library.

Held-out task performance. CYBERGRAFT reaches an Overall score of 53.6, exceeding native evolution by 7.3 points and Artificial Harness by 6.1. Gains over native evolution span Search (+7.3), Office (+9.3), and General (+3.1). Native evolution trails Artificial Harness on Search and Office, whereas CYBERGRAFT exceeds both baselines in every domain. These results support asset-guided optimization beyond the development tasks, given the prebuilt library.

Optimization progress. AnytimeVal rises from 50.1 to 54.8, indicating better average best-so-far development performance within the iteration budget. This metric reflects both initialization and subsequent revisions; it does not isolate seed quality or search speed. The following ablations examine the components underlying these gains.

5.3 ABLATION STUDIES

We evaluate four ablations: (1) **w/o Continual Discovery** restricts external donors to the initial pool while retaining native mechanism generation; (2) **w/ Fixed Boundaries** replaces adap-

Table 2: **Ablations of CYBERGRAFT.** Values are means over three independent runs. Online cost includes failed attempts and excludes initial library construction and final test evaluation.

Variant	Overall↑	AnytimeVal↑	Acceptance (%)	Online cost ($)
CYBERGRAFT	53.6	54.8	82.3	691
w/o Continual Discovery	49.1	53.5	67.1	623
w/ Fixed Boundaries	52.0	53.7	74.4	662
w/o Contribution Analysis	53.6	54.8	82.3	954
w/o Profile-Guided Reuse	53.6	54.1	62.5	739

tive extraction boundaries with predefined functional modules; (3) **w/o Contribution Analysis** removes contribution-based screening when selecting mechanisms for asset construction; and (4) **w/o Profile-Guided Reuse** removes quantitative benefit–risk evidence from retrieval ranking and proposal generation while retaining functional descriptions, applicability conditions, and compatibility constraints.

All variants use the same development tasks, online resource limits, and acceptance criteria. The fixed-boundary and contribution-analysis variants rebuild their initial libraries from the same candidate sources under a matched construction budget; the other variants share the full method's initial library. Table 2 reports Overall, AnytimeVal, gate acceptance, and online optimization cost. Acceptance is the fraction of candidates entering the evaluation pipeline that pass the common gates, including construction and compatibility failures in the denominator; passing these gates does not necessarily imply an improvement over the current harness.

Continual discovery and adaptive boundaries. Restricting donors to the initial pool reduces Overall by 4.5 and acceptance by 15.2 percentage points. With the initial library unchanged, this supports the value of external acquisition during evolution. Fixed boundaries reduce Overall by 1.6 and acceptance by 7.9 points, supporting extraction based on mechanism-specific cooperation. Both variants cost less but achieve lower final performance, indicating a performance–cost tradeoff.

Contribution-based screening. Removing contribution analysis leaves reported Overall, AnytimeVal, and acceptance unchanged, but raises online cost from $691 to $954. Screening thus reduces online expenditure by 27.6% at the same reported performance. This is consistent with lower downstream search overhead.

Quantitative profile reuse. Removing profile guidance preserves Overall but lowers AnytimeVal by 0.7 and acceptance by 19.8 percentage points, while raising cost from $691 to $739. Because functional descriptions and compatibility constraints remain available, this comparison supports the additional decision value of quantitative evidence: higher gate acceptance and better optimization progress with lower online expenditure.

6 CONCLUSION

CYBERGRAFT evolves the effective harness design space by accumulating reusable mechanism packages and scoped quantitative evidence. Selective discovery, contribution screening, and dependency-guided reconstruction support asset-guided seed assembly and candidate generation. On Evo-Bench, this approach improves held-out performance and optimization progress; ablations distinguish performance gains from discovery and adaptive boundaries from online cost savings through screening and profile reuse. Measuring how initial construction costs amortize across repeated task families remains an important direction.

AI ASSISTANCE

Large language models assisted with research discussions, literature discovery and organization, manuscript editing and translation, figure draft preparation. The authors retain responsibility for the scientific claims, references, implementations, and reported results.

REFERENCES

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

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. URL https://arxiv.org/abs/2603.02766.

Artificial Analysis. Stirrup, 2026. URL https://github.com/ArtificialAnalysis/Stirrup. Software repository. Accessed September 26, 2026.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4. URL https://www.nature.com/articles/s41586-025-09962-4.

Jianlyu Chen, Yuyang Hu, Hongjin Qian, Jiawei Liu, Wenqing Wei, Xiaolong Chen, Defu Lian, Zhicheng Dou, Chaozhuo Li, Qiwei Ye, and Zheng Liu. Repo-To-Skill: Distilling GitHub repositories into AI4AI skills. *arXiv preprint arXiv:2609.02749*, 2026a. URL https://arxiv.org/abs/2609.02749.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws. *arXiv preprint arXiv:2606.06324*, 2026b. URL https://arxiv.org/abs/2606.06324.

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*, 2026c. URL https://arxiv.org/abs/2606.14249.

DeepSeek-AI. DeepSeek Harness, 2026. URL https://github.com/deepseek-ai/deepseek-harness. Software repository. Accessed September 2026.

Shimin Di, Xujie Yuan, Hanghui Guo, Chaoqian Ouyang, Yongxu Liu, Ling Yue, Zhangze Chen, Libin Zheng, Jia Zhu, Shaowu Pan, Jian Yin, Yong Rui, and Min-Ling Zhang. ToolRosella: Translating code repositories into standardized tools for scientific agents. *arXiv preprint arXiv:2603.09290*, 2026. URL https://arxiv.org/abs/2603.09290.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Lisheng Huang, Chen Yang, Hao Zhou, Huatong Song, Zongchao Chen, Ran Le, Yang Song, Wayne Xin Zhao, and Tao Zhang. Evo-Bench: Can language models improve agent harness? *arXiv preprint arXiv:2608.09096*, 2026. URL https://arxiv.org/abs/2608.09096.

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. HarnessEvolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 2026. URL https://arxiv.org/abs/2609.00829.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026. URL https://arxiv.org/abs/2603.28052.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Yida Xue, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, Gang

Yu, Haiwen Hong, Longtao Huang, Hui Xue, Chenxi Wang, Yijun Wang, Zifei Shan, Xi Chen, Zhaopeng Tu, Feiyu Xiong, Xin Xie, Peng Zhang, Zhengke Gui, Lei Liang, Jun Zhou, Chiyu Wu, Jin Shang, Yu Gong, Junyu Lin, Changliang Xu, Hongjie Deng, Wen Zhang, Keyan Ding, Qiang Zhang, Fei Huang, Ningyu Zhang, Jeff Z. Pan, Guilin Qi, Haofen Wang, and Huajun Chen. SkillNet: Create, evaluate, and connect AI skills. *arXiv preprint arXiv:2603.04448*, 2026. URL https://arxiv.org/abs/2603.04448.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026. URL https://arxiv.org/abs/2604.25850.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. HarnessBank: Semantic gene-bank search with gated verification for agent-harness self-evolution. *arXiv preprint arXiv:2607.13683*, 2026. URL https://arxiv.org/abs/2607.13683.

MiroMind. MiroFlow, 2025. URL https://github.com/MiroMindAI/MiroFlow. Software repository. Accessed September 26, 2026.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024. URL https://arxiv.org/abs/2406.11695.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972. doi: 10.1145/361598.361623.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025. URL https://arxiv.org/abs/2510.04374.

Siran Peng, Cuiyu Yang, Tianyu Fu, Tianshuo Zhang, Haoyuan Zhang, Weisong Zhao, Anyang Su, Minghui Wu, Huiying Li, Xiangyu Zhu, Chenxu Zhao, and Zhen Lei. Self-supervised skill optimization. *arXiv preprint arXiv:2607.28777*, 2026. URL https://arxiv.org/abs/2607.28777.

Amir Pnueli, Michael Siegel, and Eli Singerman. Translation validation. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 151–166, 1998. doi: 10.1007/BFb0054170.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "Gradient Descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023. URL https://arxiv.org/abs/2305.03495.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024. URL https://arxiv.org/abs/2410.06153.

Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. SkillFoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources. *arXiv preprint arXiv:2604.03964*, 2026. URL https://arxiv.org/abs/2604.03964.

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, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Liyan Tang, Cyrus Rashtchian, Chun-Sung Ferng, Andrew Tomkins, Da-Cheng Juan, and Tu Vu. WikiSkill: Compiling agent experience into persistent knowledge for skill evolution. *arXiv preprint arXiv:2608.27454*, 2026. URL https://arxiv.org/abs/2608.27454.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents. *arXiv preprint arXiv:2601.14242*, 2026. URL https://arxiv.org/abs/2601.14242.

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. doi: 10.48550/arXiv.2305.16291. URL https://arxiv.org/abs/2305.16291.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 50208–50232, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

Siwei Wu, Jincheng Ren, Yizhi Li, Haau-Sing Li, Chengran Yang, Yuxuan Zhang, Weicheng Gu, Jian Yang, Riza Batista-Navarro, Chuanyi Zhang, Xianglong Liu, Ming Zhou, Bryan Dai, and Chenghua Lin. ModularRSI: Modular and generalizable recursive harness self-improvement. *arXiv preprint arXiv:2609.14857*, 2026. URL https://arxiv.org/abs/2609.14857.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. Modeling and discovering vulnerabilities with code property graphs. In *2014 IEEE Symposium on Security and Privacy*, pp. 590–604, 2014. doi: 10.1109/SP.2014.44. URL https://fabianyamaguchi.com/files/2014-ieeesp.pdf.

Zhiling Yan, Dingjie Song, Hanrong Zhang, Wei Liang, Yuxuan Zhang, Yutong Dai, Lifang He, Philip S. Yu, Ran Xu, Xiang Li, and Lichao Sun. OpenSkill: Open-world self-evolution for LLM agents. *arXiv preprint arXiv:2606.06741*, 2026. URL https://arxiv.org/abs/2606.06741.

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. URL https://arxiv.org/abs/2309.03409.

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*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

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. URL https://arxiv.org/abs/2605.23904.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022. doi: 10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629.

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

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025a. URL https://arxiv.org/abs/2512.18746.

Guibin Zhang, Leo Lu, Fangzhou Xie, Kang Zhu, Junhao Wang, Zhifei Xie, Zhaochen Yu, Zihang Liu, Zhongxiang Sun, Qiankun Li, Yue Liao, Heng Chang, Xiaobin Hu, Qibing Ren, Wangchunshu Zhou, Chuanrui Hu, Yafeng Deng, and Shuicheng Yan. JIT-Agent: Scaling harness intelligence via just-in-time harness evolution. *arXiv preprint arXiv:2608.25593*, 2026. URL https://arxiv.org/abs/2608.25593.

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*, 2025b. URL https://arxiv.org/abs/2505.22954.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=z5uVAKwmjf.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*, 2025d. doi: 10.48550/arXiv.2510.04618. URL https://arxiv.org/abs/2510.04618.

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. URL https://arxiv.org/abs/2211.01910.

A Mechanism Validation Protocol

Mechanism-level validation separates source contribution, reconstruction fidelity, and conditional transfer utility. The comparison set comprises the native donor, a valid ordinary-strategy replacement, reconstructed reinsertion, an isolated package, the common target harness, and the target with the package integrated. Source revisions, activation conditions, declared dependencies, behavioral observables, and acceptance criteria define the scope of each comparison.

Three mechanism families illustrate this protocol: retrieval and recovery in MiroFlow (A) (MiroMind, 2025), artifact-delivery lifecycle management in Stirrup (B) (Artificial Analysis, 2026), and Todo–compaction coordination in Claw-Eval (C) (Ye et al., 2026). For C, the behavioral question is whether current Todo state reaches the next model-visible history through the selected compaction path. The replacement control preserves ordinary Todo functionality and summarization. Test conditions cover absent and repeated compaction, state updates before compaction, the next model request, empty Todo state, and error or cancellation paths. Negative mutations target omitted injection, stale reads, incorrect ordering, and hidden donor-state access.

For A, comparisons separate retrieval organization from differences in search or archive services. For B, they distinguish declared-file existence and cross-environment delivery from artifact content quality. Fidelity is assessed against behavior present in the donor. Trigger-enriched diagnostic cases and natural benchmark tasks remain separate evaluation strata.

Composition analysis considers individual mechanisms, pairwise combinations, and the joint combination, with whole-agent routing and delegation as additional comparison settings. Effects are evaluated jointly with execution cost, state conflicts, and resource contention. Development probes guide refinement, while independent confirmation cases assess the resulting packages. Positive task gains are not required for a valid comparison; null effects, failures, and uncertainty remain part of the evidence record.

B Runtime Contracts and Evidence Records

An asset's runtime contract specifies its inputs and outputs, state ownership, lifecycle requirements, execution effects, failure handling, external services, and permissions. The package records component boundaries, required event ordering, host bindings, source revision, license, and implementation versions.

Each reconstruction or adaptation record links the parent version, mechanism hypothesis, supporting evidence, boundary operation, validity assessment, regression checks, and cost. Source contribution, fidelity, isolation, target utility, and composition effects are maintained as distinct claims. Unevaluated cases are distinguished from invalid comparisons, and adapters expose required dependencies explicitly.

C Reproducibility Records

The experiment record consists of source and benchmark revisions, task identities and splits, model endpoints and reasoning settings, underlying tool services, host configuration, evaluator version, discovery access policy, resource limits, acceptance thresholds, repetitions, and statistical units. Acquisition records retain source queries, unsuccessful imports, rejected candidates, manual interventions, and service calls. Source-derived information and task-derived feedback are tracked separately to support contamination checks.

Cost accounting distinguishes initial library construction, online seed assembly and evolution, and final test execution. Library inventories distinguish candidate mechanisms, admitted assets, versions, and mechanism families. Transfer and composition records retain the task, host, configuration, baseline, and participating asset versions so that outcomes are aggregated only under comparable conditions.