THE AGENT IS IN THE DETAILS: WHAT MATTERS IN WEB AGENT HARNESS DESIGN?

Anonymous authors
Paper under double-blind review

ABSTRACT

A web agent harness determines what the model sees, what it retains across steps, and how model outputs become browser actions. Harness designs are often introduced inside complete systems with their own models and setups, so reported gains do not show which design to adopt. We build a configurable framework on GenericAgent from AgentLab, implement 30 configurations in six functional categories, and compare each with its setting-specific baseline on WebArena-Lite with Qwen3.5-9B and OpenApps with Qwen3.5-27B. The two highest-scoring configurations in the reported catalog results are multi-turn histories in both settings: full observation history keeps earlier page bodies, whereas observation masking replaces them with placeholders; both retain past model responses. Full history and masking reach success rates of 26.06% and 23.64% against a 15.76% baseline on WebArena-Lite, and 57% and 31% against 19% on OpenApps. On WebArena-Lite, masking trails full history by 2.42 percentage points and uses about a third fewer action-model tokens than the baseline; on OpenApps, it trails by 26 points. Only one tested combination outscores the constituent configurations alone: full history with execution-error feedback on OpenApps. We also evaluate one fixed workflow library, distilled from successful trajectories on the evaluated task set, across five WebArena-Lite model settings. Ordered by baseline success, the settings show declining workflow increments, from +7.88 points with Qwen3.5-4B to -1.82 with GPT-5.6 Sol. These results favor evaluating multi-turn history first, comparing combinations with the constituent configurations alone and the unchanged baseline when selecting by success rate, and evaluating this library on the target model. We will release the framework code and configurations as an extensible platform for web agent harness research, supporting new designs and reuse of existing comparisons.

1 INTRODUCTION

Improving a web agent requires deciding what the model sees, how task progress is retained, and how model outputs become browser actions. As Figure 1 shows, these decisions are made in the harness, also called a scaffold, which is the execution layer between the model and the browser. With the model held fixed, swapping the harness can substantially change measured outcomes (Yao et al., 2026; Zhang et al., 2026c; Zheng et al., 2026; Sen et al., 2026). Web and GUI agent studies have proposed many harness designs, such as trimmed page observations (Kerboua et al., 2025), workflow memory (Wang et al., 2025b), and pre-execution critics (Wanyan et al., 2025).

Many such designs are introduced as parts of complete systems, each reported with its own model and evaluation setup (He et al., 2024; Zheng et al., 2024a; Yang et al., 2025b), so a system-level gain shows neither which design produced it nor what that design would add to another agent. The name of a design also says little about what the model receives: workflow memory can be a library of procedures derived from past trajectories (Wang et al., 2025b) or a small set of hand-written application instructions, and feedback can be a recorded browser error or a model judgment of task progress. The closest studies compare complete model and harness pairings (Yao et al., 2026) or examine particular functional categories on web tasks (Enomoto et al., 2026; Hajimiri et al., 2026). Developers still lack a comparison of gains across functional categories on a common baseline.

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

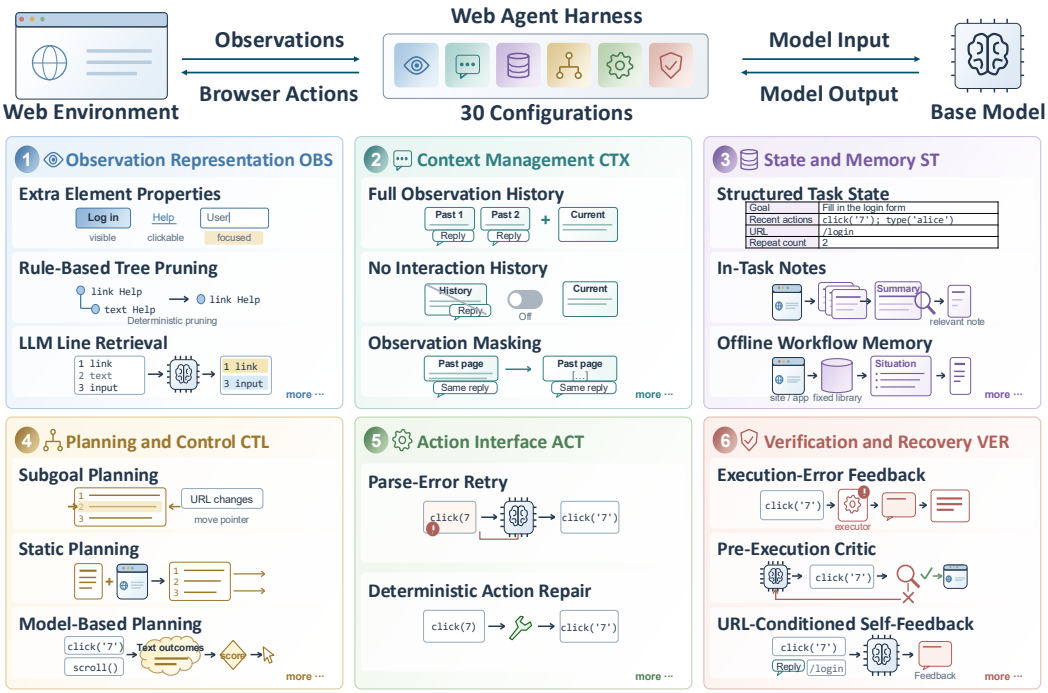


Figure 1. The web agent harness and its six functional categories. The top row shows how the harness connects browser observations and actions with model inputs and outputs. The panels illustrate alternative ways to retain page history, retrieve procedures, select actions, and provide feedback within this loop. Panel title abbreviations correspond to category prefixes in Table A.1. Figure A.1 shows the full catalog.

We build a configurable experimental framework for web agents on GenericAgent in AgentLab (Le Sellier De Chezelles et al., 2025). The framework baseline supplies the task goal, the current accessibility tree, and past actions to the model selecting browser actions. The framework implements 30 executable configurations in six functional categories, including designs drawn from existing agents and framework options, and three content controls that insert irrelevant or uninformative text. We run the full catalog on all 165 WebArena-Lite tasks with Qwen3.5-9B (Zhou et al., 2024b; Liu et al., 2025), and the same catalog on 100 longer-horizon OpenApps composite tasks with Qwen3.5-27B (Ullrich et al., 2025). Within each environment, configurations are compared with the same baseline using the same tasks and action model.

The results answer four adoption questions.

Which configurations to try first. Full observation history ranks first and observation masking second by reported catalog success in both settings, as § 4.1 shows. Full history keeps earlier page bodies within an estimated-token budget, whereas masking replaces them with placeholders; both retain past model responses. Observation history has previously improved performance across most models and settings on WorkArena L1 (Enomoto et al., 2026). Here, the comparison places the two histories alongside configurations from six functional categories in two settings. No observation-representation or action-interface configuration exceeds the baseline in either setting.

How to choose between the two histories. Full history scores 2.42 percentage points above masking on WebArena-Lite, where masking uses about a third fewer action-model tokens than the baseline, and 26 points higher on OpenApps, as § 4.2 reports. Between these histories, full history is the first candidate to evaluate when success comes first; masking is worth considering when action-model tokens are constrained, especially on WebArena-Lite. The choice depends on whether the higher success is worth the additional tokens in the target setting.

When to keep a combination. Selection by success rate calls for comparing a combination with the constituent configurations alone and the unchanged baseline. Only one tested combination in § 4.3 outscores the constituent configurations alone: full history with execution-error feedback on Ope-

nApps, at 63% against 57% for full history alone. On OpenApps, self-feedback added to workflow memory beats memory alone but not self-feedback alone. Improvement over the start is therefore insufficient to identify the higher-scoring option.

Whether the workflow library transfers across models. § 4.4 applies one fixed library distilled from successful trajectories on the evaluated task set to five WebArena-Lite model settings. Ordered by baseline success, these settings show declining workflow increments, from +7.88 points with Qwen3.5-4B to −1.82 with GPT-5.6 Sol. The decision to adopt this library therefore requires comparing it with the unchanged baseline on the target model.

We make three contributions:

1. **An extensible framework and executable catalog.** We implement 30 harness configurations in six functional categories on a framework derived from GenericAgent, supporting component replacement, specified combinations, and model changes. We will release the framework code and configurations as an extensible platform for web agent harness research, enabling comparisons of new designs with the baseline and existing configurations within the same environment.
2. **A comparison across designs, combinations, and model settings.** We evaluate the same full catalog in two settings, compare selected combinations, and assess one fixed workflow library across five WebArena-Lite model settings.
3. **Guidance for harness choices.** We identify the two multi-turn history configurations as the first candidates to evaluate and quantify their success and action-model token trade-off.

2 Study Design

2.1 Baseline and Configuration Catalog

All configurations modify one baseline agent derived from GenericAgent, the default agent of AgentLab (Le Sellier De Chezelles et al., 2025). This baseline, which § A labels FMT, receives the task goal, its past actions, and the accessibility tree of the current page, abbreviated as AXTree. It keeps no earlier pages in its input and has no planning, memory, or feedback component. Within each environment, every configuration is compared with this baseline on the same tasks and with the same base model, the language model that selects browser actions.

We organize the 30 configurations by the six runtime responsibilities that a survey of agent harnesses identifies (Guo et al., 2026), and name each category by its role in a web agent, as Figure 1 shows. Observation representation determines how the current page is presented to the model, and context management determines which earlier observations, responses, or examples accompany it. State and memory supplies task records, notes, or procedures. Planning and control produces plans or selects among candidate actions. The action interface converts model output into browser actions, and verification and recovery checks actions or their outcomes and handles failures.

Our framework provides replaceable components at the points of each step where these categories act, such as observation rendering, context assembly, and action parsing. Each configuration is defined by the components it replaces and the GenericAgent options it changes. Choosing the base model independently lets the same configuration run across models.

Most configurations follow designs from existing web and GUI agents or options already available in AgentLab and BrowserGym. Examples include accessibility-tree pruning from AgentOccam (Yang et al., 2025b), model-based planning from WebDreamer (Gu et al., 2024), and execution-error feedback from BrowserGym (Le Sellier De Chezelles et al., 2025). Three configurations are content controls: irrelevant notes, an irrelevant plan, and fixed irrelevant feedback. They take the form of notes, a plan, or feedback but carry no task information, and serve as references for components of the same form. Table A.1 gives the operation and design basis of every configuration.

2.2 History, Workflow Memory, and Feedback Configurations

The two history configurations differ in how they retain earlier pages. Full observation history keeps as many old page bodies as an estimated-token budget allows, whereas observation masking

replaces them with placeholders. Both keep past model responses and the current observation, and both use the message template and parser of WebAgent-R1 (Wei et al., 2025); masking also adapts the dynamic context compression of WebAgent-R1. Relative to the baseline, both therefore change message organization and parsing. The intended difference between them is whether old page bodies remain in the input. Table A.1 identifies full history as CTX1 and masking as CTX4.

Offline workflow memory is based on the offline setting of Agent Workflow Memory, abbreviated as AWM (Wang et al., 2025b), and adds procedures for the current website from a library fixed before evaluation. The WebArena library was distilled from successful trajectories on the evaluated task set; § A.2 describes its construction. On OpenApps, the configuration instead uses eight hand-written application workflows, which are excluded from the model comparison. Table A.1 lists it as ST3; we call it workflow memory below.

The two feedback configurations report on the previous step. URL-conditioned self-feedback returns a model judgment of that step based on the previous action, its accompanying response, and the current URL, without independent page text from before or after the action. Execution-error feedback returns the error that the browser recorded for the previous action and involves no model judgment. Table A.1 identifies self-feedback as VER5 and error feedback as VER1.

We compare combinations formed by adding one of these components to a starting configuration, called the start. On WebArena, we add each feedback type to workflow memory, and we add self-feedback to subgoal planning and to in-task notes. On OpenApps, we add each feedback type and workflow memory, one at a time, to each history configuration, and we add each feedback type to workflow memory. Each combination is compared with its start, with the better individual configuration, meaning whichever of its constituent configurations has a higher success rate alone, and with the baseline.

Using the same tasks and base model within each environment gives every catalog entry the same reference point. Content controls probe whether material presented as notes, plans, or feedback supplies useful task information. Applying one fixed workflow library across models then tests whether the same resource remains worth adding after the model changes.

3 Experimental Setup

We evaluate the catalog and follow-up comparisons of § 2 in two environments.

Environments. We use all 165 tasks of WebArena-Lite, the subset of WebArena (Zhou et al., 2024b), introduced by VisualAgentBench (Liu et al., 2025) and adopted by WebRL (Qi et al., 2025). Tasks require information lookup, navigation, and content changes on functional websites; we refer to this task set as WebArena below. On OpenApps (Ullrich et al., 2025), we use all 100 longer-horizon composite tasks from the official task configuration under the default condition. These tasks combine goals across resettable applications.

Models. The catalog and combinations use Qwen3.5-9B as the action model on WebArena (Qwen Team, n.d.c) and Qwen3.5-27B on OpenApps (Qwen Team, n.d.a). To evaluate the fixed WebArena workflow library across models in § 4.4, we also compare the baseline, workflow memory, and self-feedback with Qwen3.5-4B (Qwen Team, n.d.b), Qwen3.5-9B*, Qwen3.5-27B, and GPT-5.6 Sol (OpenAI, n.d.) via Azure; together with 9B, these form five model settings.

Execution. For the configuration catalogs, Qwen action models are configured with temperature 1.0 and an output cap of 2048 tokens. § A.2 lists the GPT-5.6 Sol settings. Each task permits at most 30 browser actions. Baseline input-fitting targets are 40,000 tokens on WebArena and 24,000 on OpenApps. On WebArena, both history configurations bypass this target; full history estimates prompt length from character counts against a 50,000-token budget and restores old page bodies from newest to oldest. § A.2 explains why this estimate does not bound the final request.

Scoring. We report success rates and differences between configurations in percentage points, denoted pp. WebArena combines programmatic checks with a language-model evaluator configured by default to use GPT-4o-mini for semantic answer judgments; success also requires an explicit stop

*Qwen model names denote non-thinking mode unless marked with an asterisk, which denotes native thinking. In both modes, the baseline prompt requests textual reasoning and one action.

Table 1. Success rates in % for the baseline and the 30 configurations. WA denotes WebArena and OA denotes OpenApps, as § 3 defines. Bold marks values above the column baseline, and each category row gives the number of such configurations out of the category size; colors follow Figure 1. The columns differ in tasks and models and are not an environment contrast. The two history configurations lead in both settings, while most configurations score below baseline.

Configuration	WA	OA
Baseline	*15.76*	*19*
Observation representation	0/5	0/5
Extra element properties	11.52	11
Viewport filtering	10.30	9
Rule-based tree pruning	10.91	15
Default AXTree text	15.15	19
LLM line retrieval	8.48	17
Context management	2/5	4/5
Full observation history	**26.06**	**57**
No interaction history	13.94	1
Recent-observation append	13.94	**20**
Observation masking	**23.64**	**31**
Abstract examples and hints	7.27	**25**
State and memory	2/4	0/4
Structured task state	14.55	17
In-task notes	**16.36**	14
Offline workflow memory	**20.61**	14
Irrelevant notes	14.55	17
Planning and control	2/4	1/4
Subgoal planning	7.27	17
Static planning	**16.36**	18
Model-based planning	**16.97**	11
Irrelevant plan	10.91	**24**
Action interface	0/2	0/2
Parse-error retry	1.21	7
Deterministic action-format repair	13.94	19
Verification and recovery	3/10	2/10
Execution-error feedback	**19.39**	18
Early-stopping rules	13.94	16
No-change retry	14.55	17
Pre-execution critic	14.55	3
URL-conditioned self-feedback	14.55	**20**
Action-effect verification	11.52	17
Repeated-action early stop	**18.18**	13
Parse-error early stop	**16.36**	**20**
Fixed irrelevant feedback	13.94	18
Postcondition check with repair prompts	12.12	14

action. Stored-log inspection can expose evaluation errors that aggregate scores conceal (Kapoor et al., 2026). We review selected WebArena success labels after execution with model-assisted adjudication, described in § A.2. Reported WebArena success rates incorporate these task-level adjudications. OpenApps uses official binary rewards based on application state.

Resource measures. Action-model tokens sum the recorded input and output tokens of action-selection calls per task and do not represent total system cost. WebArena steps include an issued stop action, while OpenApps counts browser actions.

4 Results

The catalog, history comparison, component combinations, and model comparison answer the four adoption questions of § 1. Each setting pairs an environment with a base model and uses the reported scores defined in § 3. Comparisons with baseline are made within each setting.

4.1 Comparing the Full Catalog with Baseline

The two history configurations lead the catalog by reported success in both settings. In Table 1, full observation history and observation masking reach 26.06% and 23.64% on WebArena, against 15.76% for baseline. On OpenApps, they reach 57% and 31%, against 19%. In each setting, 21 of 30 configurations score below baseline, and none of the seven observation-representation and action-interface configurations exceeds it. Smaller gaps do not establish a stable ranking: 16 WebArena configurations lie within four tasks of baseline. One plausible reason for the history lead is that both configurations retain past model responses, which record facts and decisions from earlier pages that the focused page no longer shows in tasks mixing information gathering with changes to persistent state (Zhou et al., 2024b).

Within the action interface, deterministic action-format repair outscores parse-error retry in both settings, 13.94% against 1.21% on WebArena and 19% against 7% on OpenApps, but does not exceed baseline in either. The better of two interface variants therefore need not exceed baseline.

Table 2. Success, action-model tokens, and task outcomes for the baseline and two history configurations. Measures use the same task records. Tokens are in K/task and do not represent total system cost; WA counts steps, OA actions. Bold marks the higher-scoring history. Both histories solve 27 WA tasks and 25 OA tasks; neither solves 110 WA tasks or 37 OA tasks. Full history achieves higher success than masking at the cost of more action-model tokens.

	Success rate, %			Tokens, K/task			Steps or actions/task			Solved by one only	
Setting	Base	Full	Mask	Base	Full	Mask	Base	Full	Mask	Full	Mask
WebArena	15.76	**26.06**	23.64	62.25	150.03	41.51	8.83	8.10	10.04	16	12
OpenApps	19	**57**	31	75.74	395.86	76.85	26.79	19.74	24.31	32	6

On OpenApps, content controls match or exceed several configurations with the same function. Irrelevant notes reach 17%, above 14% for in-task notes, and the irrelevant plan reaches 24%, above all three planning configurations; fixed irrelevant feedback ties execution-error feedback at 18%. On WebArena, the controls tie or exceed some configurations of their categories but score below the highest one and below baseline. Abstract examples and hints reach 25% on OpenApps, 1 pp above the irrelevant plan, so this comparison does not establish that the intended mechanism produced the gain. Such controls provide another comparator for proposed planning and memory modules.

Some configurations fall on opposite sides of baseline in the two settings. Offline workflow memory reaches 20.61% on WebArena; the OpenApps configuration of that name uses eight hand-written workflows, a separate resource, and reaches 14%. Abstract examples and hints show the opposite pattern, 7.27% on WebArena and 25% on OpenApps. The leading reported scores make the two history configurations the first candidates to evaluate in either setting. The reversals among other configurations favor comparing each remaining candidate with the baseline of its target setting.

4.2 Choosing Between the Two History Configurations

Masking scores slightly below full history on WebArena and far below it on OpenApps. On WebArena, masking scores 7.88 pp above baseline and uses about a third fewer action-model tokens per task, 41.51K against 62.25K, as Table 2 shows. Full history scores 2.42 pp higher than masking but uses 150.03K tokens per task, 3.61 times as many.

On OpenApps, full history reaches 57% and masking 31%. One plausible reading of the 26 pp gap is that retained page bodies carry information across the application switches that these tasks require, as § 3 describes. Full history uses 395.86K tokens per task, 5.15 times as many as masking. Unlike on WebArena, masking uses slightly more tokens than baseline, 76.85K against 75.74K.

Fewer tokens do not imply fewer browser actions. On WebArena, masking takes 10.04 steps per task, against 8.83 for baseline and 8.10 for full history. On OpenApps, full history and masking take 19.74 and 24.31 actions per task, both below 26.79 for baseline. The choice therefore depends on whether browser actions or action-model tokens constrain deployment. Retaining more page text enlarges the input to later decisions, whereas revisiting a page spends another browser action. The measured trade-off makes these two resource demands visible: a shorter prompt can accompany a longer interaction. For developers facing a browser-action limit, action-model token savings alone are therefore an incomplete selection criterion.

Appending recent page text to the baseline input does not reproduce these scores. In Table 1, recent-observation append reaches 13.94% on WebArena and 20% on OpenApps, below both history configurations, and 19 of the 165 WebArena tasks were rejected for input length before inference. No interaction history, which also removes action history, reaches 13.94% on WebArena and 1% on OpenApps, below baseline in both. Unlike no interaction history, masking retains past model responses, which record earlier decisions but not page details that the model did not mention. Relative to baseline, both history configurations also change message organization and parsing, as § 2.2 specifies, so their differences from baseline cannot be attributed to retained history alone.

Each configuration solves tasks that the other misses. On WebArena, full history and masking have 16 and 12 exclusive successes, tasks that only one of the two solves. On OpenApps, they share 25 successes and have 32 and 6 exclusive successes. The recorded outcomes therefore show both

Table 3. Success rates in % after adding one component to a start. On OpenApps, offline workflow memory uses eight hand-written application workflows. The last row gives the full-history minus masking difference in each OpenApps column. Bold marks the higher-scoring history configuration; ‡ marks the workflow-memory configuration itself. Only full history with execution-error feedback on OpenApps outperforms each configuration used alone.

	No addition	+ URL-conditioned self-feedback	+ Execution-error feedback	+ Offline workflow memory
WebArena				
Baseline	15.76	14.55	19.39	20.61‡
Subgoal planning	7.27	12.12	–	–
In-task notes	16.36	15.15	–	–
Offline workflow memory	20.61	13.94	19.39	–
OpenApps				
Full observation history	**57**	**45**	**63**	**54**
Observation masking	31	18	30	19
Offline workflow memory	14	19	16	–
Full − masking, pp	+26	+27	+33	+35

gains and losses under a change of history configuration. A deployment may depend on a particular application workflow, so an aggregate improvement can still hide a regression on a task that matters to that deployment. Comparing which tasks succeed remains useful even when one configuration has a clearly higher overall score. Together with the resource measures, these results favor evaluating full history first when task completion is the priority, most clearly on OpenApps, and masking on WebArena when the action-model token budget binds.

4.3 Adding Components to Selected Configurations

Only one tested combination has a higher reported success rate than the better individual configuration. With the individual scores in Table 1, Table 3 compares each combination with its start, the better of its constituent configurations used alone, and baseline; differences from the no-addition column describe final configurations and do not isolate the addition itself. The one exception is full history with execution-error feedback on OpenApps, at 63% against 57% for full history alone. On WebArena, workflow memory alone reaches 20.61%, against 19.39% with error feedback and 13.94% with self-feedback. With self-feedback, subgoal planning reaches 12.12%, above planning alone at 7.27% but below self-feedback alone at 14.55% and baseline. In-task notes with self-feedback reach 15.15%, below baseline and the 16.36% of notes alone.

On OpenApps, workflow memory reaches 19% with self-feedback and 16% with error feedback, against 14% alone. Each combination scores below the corresponding feedback alone, at 20% and 18% in Table 1, and neither exceeds the 19% baseline. Judged only against the 14% start, both would appear to be gains.

Full history stays ahead of masking under every OpenApps addition, by 27, 33, and 35 pp with self-feedback, error feedback, and workflow memory. The order holds even where both score below their no-addition values: with self-feedback, full history reaches 45% and masking 18%, and with workflow memory, 54% and 19%. With error feedback, the scores are 63% for full history and 30% for masking; masking alone is 1 pp higher.

The two feedback types receive different evidence. Self-feedback judges the previous action from that action, its accompanying response, and the current URL, without independent page text, although a requested update can succeed or fail without navigation. Error feedback reports failed browser operations but not progress on the remaining goals, which OpenApps scores through application state (Ullrich et al., 2025). One plausible reading of the 63% combination is that the error identifies the failed operation while retained pages supply task context.

For an agent already using full history on OpenApps, error feedback is the addition to evaluate first: it is the only addition in Table 3 that scores above full history alone. No feedback type scores higher at every start: on OpenApps, error feedback scores higher with full history, and self-feedback

Table 4. Success rates in % for the baseline, workflow memory, and self-feedback in five WebArena model settings, ordered by baseline success rather than model size; the asterisk marks native thinking mode. The workflow increment is workflow memory minus baseline in pp. Workflow increments decline as baseline success rises, becoming negative in the setting with the highest baseline.

Model setting	Success rate, %			Workflow increment, pp
	Baseline	Workflow memory	Self-feedback	
Qwen3.5-4B	12.12	20.00	13.33	+7.88
Qwen3.5-9B	15.76	20.61	14.55	+4.85
Qwen3.5-9B*	22.42	24.85	27.27	+2.42
Qwen3.5-27B	27.27	28.48	27.27	+1.21
GPT-5.6 Sol, API	52.73	50.91	52.73	−1.82

with workflow memory. These results favor judging each combination against its start, the better individual configuration, and baseline, and choosing the feedback type on the target start.

4.4 Applying the Fixed Workflow Library Across Model Settings

Across the five model settings, reported workflow increments decline as baseline success rises. Table 4 applies the fixed workflow library unchanged in five WebArena model settings, ordered by baseline success. The library was distilled from successful trajectories on the evaluated task set, so the table does not test transfer to unseen tasks. The workflow increment, workflow-memory success minus baseline, decreases from +7.88 pp with Qwen3.5-4B to −1.82 pp with GPT-5.6 Sol, the lowest and highest baselines. One plausible explanation is that higher-baseline models already recover more of the procedures that the library supplies.

A smaller increment does not mean lower final success. Qwen3.5-4B shows the largest increment but reaches 20.00% with workflow memory, whereas GPT-5.6 Sol reaches 50.91% despite its negative increment; final success with workflow memory rises with baseline across all five model settings.

The choice between workflow memory and self-feedback also depends on the model setting. Workflow memory scores above self-feedback with Qwen3.5-4B, Qwen3.5-9B, and Qwen3.5-27B. With Qwen3.5-9B*, self-feedback reaches 27.27% and workflow memory 24.85%, a four-task difference. With GPT-5.6 Sol, neither addition exceeds the 52.73% baseline. These rows favor comparing both candidate additions with baseline on the target model and choosing by the resulting score.

5 Related Work

Agent Harnesses. Harnesses coordinate agent reasoning, actions, feedback, and tool use (Yao et al., 2023; Shinn et al., 2023; Yang et al., 2024; Wang et al., 2025a), with coupled runtime responsibilities (Guo et al., 2026). With the model fixed, harness changes substantially alter measured outcomes (Yao et al., 2026; Zhang et al., 2026c; Zheng et al., 2026; Sen et al., 2026), consistent with evidence that implementation and evaluation details shape reported results (Henderson et al., 2018; Engstrom et al., 2020; Kapoor et al., 2024; Zhu et al., 2025b; Kapoor et al., 2026). Other work automatically searches for better agent systems and harnesses (Hu et al., 2025; Zhang et al., 2025a;b; Lin et al., 2026; Zhang et al., 2026a). These studies mainly compare complete harnesses. We compare configurations with a baseline built from GenericAgent within each environment.

Web Agents. Benchmarks evaluate natural-language tasks on websites and applications (Deng et al., 2023; Zhou et al., 2024b; Koh et al., 2024; Drouin et al., 2024; Le Sellier De Chezelles et al., 2025; Ullrich et al., 2025), including WebArena-Lite (Liu et al., 2025). Systems use specialized observation, grounding, or action designs (He et al., 2024; Zheng et al., 2024a; Yang et al., 2025b), search or model-based action lookahead (Zhou et al., 2024a; Gu et al., 2024; Zhang et al., 2025c), or open models trained with online reinforcement learning (Qi et al., 2025; Wei et al., 2025). Other studies audit evaluation or diagnose failures beyond task success (Xue et al., 2025; Lù et al., 2025; Aghzal et al., 2026).

Harness Components. Component studies in web and other AI agents cover observation representation (Enomoto et al., 2026; Kerboua et al., 2025; Chi et al., 2024), context management (Lindenbauer et al., 2025), planning and control (Shlomov et al., 2024; Erdogan et al., 2025; Zambrano et al., 2026), state and memory (Zheng et al., 2024b; Zhao et al., 2024; Wang et al., 2025b; Hajimiri et al., 2026; Ye et al., 2026), action interfaces and granularity (Yang et al., 2024; 2025b; Wang et al., 2026), and verification and recovery, from critics and self-correction (Shinn et al., 2023; Wu et al., 2025; Wanyan et al., 2025; Zhang et al., 2026d; Lu et al., 2026) to step-level process reward models for reasoning and web navigation (Lightman et al., 2024; Wang et al., 2024; Zhang et al., 2025d; Chae et al., 2025; Zhang et al., 2026b). Studies of attribution across several components use question answering, math, or general assistant tasks (Liu, 2026; Zhu et al., 2025a). The closest web studies are narrower: they compare observation representations and histories on WorkArena L1 (Enomoto et al., 2026), or weigh memory, workflow, and skill modules against a budget-matched actor on WebArena and WorkArena L1 (Hajimiri et al., 2026). We evaluate all 30 configurations across six categories in two web environments, further compare history implementations by success and token use, and test whether designs above baseline retain gains under additions and across models, as reported in § 4.1 to § 4.4.

6 Conclusion

We compared 30 web agent harness configurations with a common baseline within each setting: WebArena-Lite with Qwen3.5-9B and OpenApps with Qwen3.5-27B. The two multi-turn history configurations have the highest reported catalog success rates in both settings, with full observation history first. Observation masking uses fewer action-model tokens than full history but trails it in success rate by 2.42 pp on WebArena and 26 pp on OpenApps. The results therefore favor evaluating full history first when success is the priority and considering masking when action-model tokens are constrained, especially on WebArena. Only one tested combination outperforms each constituent configuration alone in success rate, so a combination should be judged against the stronger individual configuration and the baseline, not only the configuration it extends. Our fixed workflow library, distilled from successful trajectories on the evaluated WebArena-Lite tasks, yields smaller increments in settings with higher baseline success and a negative increment with GPT-5.6 Sol, so it should be evaluated on the target model. Together, these comparisons make harness adoption an explicit measurement problem: what information a design preserves, what resources it consumes, and whether its gain survives comparison with the alternatives. They provide a basis for developing web agents whose added complexity delivers measurable value. We will release the framework code and configurations to support comparisons of new designs, combinations, and models using the same configuration definitions and measurement procedures.

AI Use Statement

The evaluated agents use the language models listed in § 3. WebArena tasks requiring semantic judgments are also scored by a model-based evaluator. Beyond these methodological uses, we used the generative AI tools OpenAI Codex and Anthropic Claude to organize run records into analysis tables. Synthetic data, mathematical propositions, and proofs are not applicable to this paper. We also used these tools to correct grammar, spelling, and word choice, clarify unclear or ambiguous sentences, and shorten verbose prose. We reviewed all AI-assisted work and take responsibility for the final content of this paper.

References

Mohamed Aghzal, Gregory J. Stein, and Ziyu Yao. Why Do LLM-based Web Agents Fail? A Hierarchical Planning Perspective. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32157–32180, 2026. URL https://aclanthology.org/2026.acl-long.1483/.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Han, Taeyoon Kwon, Minju

Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for Reinforcing Web Agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/5bac36213824c6e0b0a173df8e2276a6-Abstract-Conference.html.

Wayne Chi, Ameet Talwalkar, and Chris Donahue. The Impact of Element Ordering on LM Agent Performance. *arXiv preprint arXiv:2409.12089*, 2024. URL https://arxiv.org/abs/2409.12089v3.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1etN1rtPB. arXiv title: Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO.

Masafumi Enomoto, Ryoma Obara, Haochen Zhang, and Masafumi Oyamada. Read More, Think More: Revisiting Observation Reduction for Web Agents. *arXiv preprint arXiv:2604.01535*, 2026. URL https://arxiv.org/abs/2604.01535v1.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15419–15462. PMLR, 2025. URL https://proceedings.mlr.press/v267/erdogan25a.html.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024. URL https://arxiv.org/abs/2411.06559.

Jianyuan Guo, Zhiwei Hao, Chengcheng Wang, Cheng Fan, Tingzhang Luo, Hongguang Li, Ying Gao, Hefei Mei, Jiankun Peng, Rongjian Xu, Minjing Dong, Han Wu, Mengyu Zheng, Kai Han, Shiqi Wang, Chang Xu, and Yunhe Wang. From Question Answering to Task Completion: A Survey on Agent System and Harness Design. *arXiv preprint arXiv:2606.20683*, 2026. doi: 10.48550/arXiv.2606.20683. URL https://arxiv.org/abs/2606.20683v1. Version 1.

Sina Hajimiri, Masih Aminbeidokhti, Jose Dolz, Ismail Ben Ayed, Issam H. Laradji, Spandana Gella, and Nicolas Gontier. Are Online Skill and Memory Modules Always Worth Their Tokens? A Budget-Constrained Study of Web Agents. *arXiv preprint arXiv:2606.15017*, 2026. URL https://arxiv.org/abs/2606.15017v2.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, 2024. URL https://aclanthology.org/2024.acl-long.371/.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

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.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter. *arXiv preprint arXiv:2407.01502*, 2024. URL https://arxiv.org/abs/2407.01502.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a0928f924a344aaebbb7f6cd8d56e34c-Abstract-Conference.html.

Imene Kerboua, Sahar Omidi Shayegan, Megh Thakkar, Xing Han Lù, Léo Boisvert, Massimo Caccia, Jérémy Espinas, Alexandre Aussem, Véronique Eglin, and Alexandre Lacoste. FocusAgent: Simple Yet Effective Ways of Trimming the Large Context of Web Agents. *arXiv preprint arXiv:2510.03204*, 2025. URL https://arxiv.org/abs/2510.03204.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024. URL https://aclanthology.org/2024.acl-long.50/.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The BrowserGym Ecosystem for Web Agent Research. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=5298fKGmv3.

Aiden Yiliu Li, Xinyue Hao, Shilong Liu, and Mengdi Wang. Avenir-Web: Human-Experience-Imitating Multimodal Web Agents with Mixture of Grounding Experts. *arXiv preprint arXiv:2602.02468*, 2026. URL https://arxiv.org/abs/2602.02468.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

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.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The Complexity Trap: Simple Observation Masking Is as Efficient as LLM Summarization for Agent Context Management. *arXiv preprint arXiv:2508.21433*, 2025. URL https://arxiv.org/abs/2508.21433v3. Version 3.

Ming Liu. More Is Not Always Better: Cross-Component Interference in LLM Agent Scaffolding. *arXiv preprint arXiv:2605.05716*, 2026. URL https://arxiv.org/abs/2605.05716v1.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Xixuan Song, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. VisualAgentBench: Towards Large Multimodal Models as Visual Foundation Agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/eea71dc576381b88f2a0ca4dedc2140d-Abstract-Conference.html.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. AgentRewardBench: Evaluating Automatic Evaluations of Web Agent Trajectories. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=fQcUZMPIvu.

Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, Lianyong Qi, and Shi Jin. ContractSkill: Repairable Contract-Based Skills for Multimodal Web Agents. *arXiv preprint arXiv:2603.20340*, 2026. URL https://arxiv.org/abs/2603.20340.

OpenAI. GPT-5.6 Sol Model. OpenAI API Documentation, n.d. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Accessed 2026-09-22.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c66e1fcc9691aae706250638f36f681b-Abstract-Conference.html.

Qwen Team. Qwen3.5-27B Model Card. Hugging Face, n.d.a. URL https://huggingface.co/Qwen/Qwen3.5-27B. Accessed 2026-09-22.

Qwen Team. Qwen3.5-4B Model Card. Hugging Face, n.d.b. URL https://huggingface.co/Qwen/Qwen3.5-4B. Accessed 2026-09-22.

Qwen Team. Qwen3.5-9B Model Card. Hugging Face, n.d.c. URL https://huggingface.co/Qwen/Qwen3.5-9B. Accessed 2026-09-22.

Sahil Sen, Akhil Kasturi, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. Is Grep All You Need? How Agent Harnesses Reshape Agentic Search. *arXiv preprint arXiv:2605.15184*, 2026. URL https://arxiv.org/abs/2605.15184v1.

ServiceNow. AgentLab. Source code repository, n.d.a. URL https://github.com/ServiceNow/AgentLab.

ServiceNow. BrowserGym. Source code repository, n.d.b. URL https://github.com/ServiceNow/BrowserGym.

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.

Segev Shlomov, Ben Wiesel, Aviad Sela, Ido Levy, Liane Galanti, and Roy Abitbol. From Grounding to Planning: Benchmarking Bottlenecks in Web Agents. *arXiv preprint arXiv:2409.01927*, 2024. URL https://arxiv.org/abs/2409.01927.

Karen Ullrich, Jingtong Su, Claudia Shi, Arjun Subramonian, Amir Bar, Ivan Evtimov, Nikolaos Tsilivis, Randall Balestriero, Julia Kempe, and Mark Ibrahim. OpenApps: Simulating Environment Variations to Measure UI-Agent Reliability. *arXiv preprint arXiv:2511.20766*, 2025. URL https://arxiv.org/abs/2511.20766.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a4b6ad6b48850c0c331d1259fc66a69c-Abstract-Conference.html.

Zhangyi Wang, Bingnan Yu, Jiexiang Xu, and Zongze Li. Action Boundary Blindness: When LLM Agents Cannot Tell Where One Action Ends and Another Begins. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36883–36899, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1711. URL https://aclanthology.org/2026.acl-long.1711/.

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

Yuyang Wanyan, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Jiabo Ye, Yutong Kou, Ming Yan, Fei Huang, Xiaoshan Yang, Weiming Dong, and Changsheng Xu. Look before you leap: A GUI-Critic-R1 model for pre-operative error diagnosis in GUI automation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=theJ4Qiz55.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training Web Agents via End-to-End Multi-Turn Reinforcement Learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7909–7928, 2025. URL https://aclanthology.org/2025.emnlp-main.401/.

Qinzhuo Wu, Pengzhi Gao, Wei Liu, and Jian Luan. BacktrackAgent: Enhancing GUI agent with error detection and backtracking mechanism. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4250–4272. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.212. URL https://aclanthology.org/2025.emnlp-main.212/.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An Illusion of Progress? Assessing the Current State of Web Agents. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.01382v4.

Jingbo Yang, Bairu Hou, Wei Wei, Shiyu Chang, and Yujia Bao. WebDART: Dynamic Decomposition and Re-planning for Complex Web Tasks. *arXiv preprint arXiv:2510.06587*, 2025a. URL https://arxiv.org/abs/2510.06587.

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.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=oWdzUpOlkX.

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

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. HarnessBench: Measuring Harness Effects across Models in Realistic Agent Workflows. *arXiv preprint arXiv:2605.27922*, 2026. URL https://arxiv.org/abs/2605.27922v1.

Qinyuan Ye, Yu Li, Yada Pruksachatkun, Jiaxin Zhang, and Chien-Sheng Wu. On the Fragility of Self-Improving Agents: Variance, Task Order, and Underspecification. *arXiv preprint arXiv:2608.18066*, 2026. URL https://arxiv.org/abs/2608.18066v2.

Alejandra Zambrano, Sara Vera Marjanovic, Imene Kerboua, Xing Han Lù, and Leila Kosseim. Does The Way You Plan Matter? An Empirical Study of Planning Representations for LLM Web Agents. *arXiv preprint arXiv:2605.29927*, 2026. URL https://arxiv.org/abs/2605.29927.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves. *arXiv preprint arXiv:2606.09498*, 2026a. URL https://arxiv.org/abs/2606.09498.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. SwarmAgentic: Towards Fully Automated Agentic System Generation via Swarm Intelligence. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1778–1818, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.93. URL https://aclanthology.org/2025.emnlp-main.93/.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. WebPilot: A Versatile and Autonomous Multi-Agent System for Web Task Execution with Strategic Exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23378–23386, 2025c. doi: 10.1609/aaai.v39i22.34505. URL https://ojs.aaai.org/index.php/AAAI/article/view/34505.

Yao Zhang, Yu Wu, Haowei Zhang, Weiguo Li, Haokun Chen, Jingpei Wu, Guohao Li, Zhen Han, and Volker Tresp. GroundedPRM: Tree-Guided and Fidelity-Aware Process Reward Modeling for Step-Level Reasoning. *arXiv preprint arXiv:2510.14942*, 2025d. URL https://arxiv.org/abs/2510.14942. Presented at the NeurIPS 2025 Workshop on Scaling Environments for Agents.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Han, and Volker Tresp. WebArbiter: A Generative Reasoning Process Reward Model for Web Agents. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=canA6Ef0RP.

Yunbei Zhang, Janet Wang, Yingqiang Ge, Weijie Xu, Jihun Hamm, and Chandan K. Reddy. Stop Comparing LLM Agents Without Disclosing the Harness. *arXiv preprint arXiv:2605.23950*, 2026c. URL https://arxiv.org/abs/2605.23950v1.

Yuzhe Zhang, Xianwei Xue, Xingyong Wu, Mengke Chen, Chen Liu, Xinran He, Run Shao, Feiran Liu, Huanmin Xu, Qiutong Pan, and Haiwei Wang. Don't Act Blindly: Robust GUI Automation via Action-Effect Verification and Self-Correction. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28924–28941, San Diego, California, United States, 2026d. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1335. URL https://aclanthology.org/2026.acl-long.1335/.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a Generalist Web Agent, if Grounded. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61349–61385, 2024a. URL https://proceedings.mlr.press/v235/zheng24e.html.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control. In *International Conference on Learning Representations*, 2024b. URL https://iclr.cc/virtual/2024/poster/18701.

Mengyu Zheng, Kai Han, Boxun Li, Haiyang Xu, Yuchuan Tian, Wei He, Hang Zhou, Jianyuan Guo, Hailin Hu, Lin Ma, Chao Xu, Guohao Dai, Lixue Xia, Yunchao Wei, Yunhe Wang, and Yu Wang. Claw-SWE-Bench: A Benchmark for Evaluating OpenClaw-style Agent Harnesses on Coding Tasks. *arXiv preprint arXiv:2606.12344*, 2026. URL https://arxiv.org/abs/2606.12344.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160, 2024a. URL https://proceedings.mlr.press/v235/zhou24r.html.

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*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Hanhao Li, Yi Yao, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Li Xiaowan, Yuhui Liu, Xiangru Tang, Jian Yang, Ge Zhang, Xitong Gao, Yuchen Eleanor Jiang, Changwang Zhang, Jun Wang, Jiaheng Liu, and Wangchunshu Zhou. OAgents: An empirical study of building effective agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13354–13369. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-emnlp.720. URL https://aclanthology.org/2025.findings-emnlp.720/.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Antony Kellermann, Jasjeet Sekhon, Jacob Steinhardt, Sarah Schwettmann, Arvind Narayanan, Matei A. Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing Best Practices in Building Rigorous Agentic Benchmarks. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f316275b44ee2de533102913828a8107-Abstract-Datasets_and_Benchmarks_Track.html.

A Catalog and Measurement Details

This appendix gives the configuration catalog used in § 2 and the execution and measurement details referenced in § 3.

A.1 Configuration Catalog

§ 2 and § 4.1 use the 30 configurations in Table A.1 and Figure A.1. FMT identifies the baseline configuration defined in § 2.1. Entries identify the design basis, framework feature, or content control used by each configuration. CTR identifies content controls, DIF identifies separate early-stopping rules, and PRD identifies the joint state-and-verification configuration.

A.2 Execution and Measurement Details

Task scope and baseline. The OpenApps evaluation uses composite tasks under the default condition, excluding the simple-task and appearance-perturbation settings of the original study. FMT uses the GenericAgent message template with the response-format contract from WebAgent-R1 (Wei et al., 2025) that requests textual reasoning and one action. It does not include screenshots, HTML, past error logs, or past reasoning.

Input and output budgets. Baseline input-fitting targets are 40,000 tokens on WebArena and 24,000 on OpenApps. On WebArena, both history configurations bypass baseline fitting. Full history estimates the entire prompt against a 50,000-token budget, approximating tokens from character counts, and restores old page bodies from newest to oldest while space remains. It does not truncate the base prompt if it already exceeds the estimated-token budget; masking leaves old page placeholders in place. These policies do not guarantee a final request bound. GPT-5.6 Sol is configured with temperature 1.0 and an Azure completion-token cap of 4096, which includes reasoning tokens.

Workflow-library construction. The WebArena library contains 25 workflows distilled by GPT-4o from 69 successful trajectories.

WebArena Postcheck. Stored-log inspection can expose evaluation errors that aggregate scores conceal (Kapoor et al., 2026). Our post-execution review examines task-success labels. A script joins the reference evaluation specification, evaluator output, and final answer recovered from saved steps to prepare review sheets for language-judged tasks and designated exact-match cases. The sheets include both rewarded and unrewarded answers. Mechanical flags identify cases for semantic adjudication; they do not determine correctness.

Adjudication checks task requirements, reference answers, and available execution evidence. The criteria accept semantically equivalent answers, check the requested entities and quantities, and require every listed element of a `must_include` reference. When the reference specifies infeasibility, the answer must convey that condition; a fabricated substantive answer is not credited. Missing answer text is checked against the recorded terminal action so that an explicit infeasibility report is not mistaken for silence.

The retained records include model-assisted semantic adjudication. A designated subset received a second review by GPT-5.6 following an initial Claude-assisted review. The second reviewer received task intent, reference criteria, evaluator output, and final answer, with configuration identity and the initial verdict withheld. Disagreement-resolution records link the adopted decisions to retained task evidence. The second review is limited to this subset.

Task-linked adjudications confirm success labels or correct false positives and false negatives. The adopted decisions are linked to supporting task evidence and incorporated into the reported success counts.

Table A.1. The 30 configurations by functional category, with the operation each performs in our framework and its design basis. The catalog does not imply that the design space is complete or optimal, that its components are independent, or that they are equivalent across environments. The catalog makes the implemented changes and their design bases explicit for comparison.

Identifier and name	Operation	Design basis
OBS: Observation representation		
OBS1 extra element properties	Add visibility/clickability tags and missing focus markers to AXTree.	AgentLab (ServiceNow, n.d.a) and BrowserGym (Le Sellier De Chezelles et al., 2025); framework adaptation.
OBS2 viewport filtering	Filter visible elements; truncate fields; process attributes, redundant text.	WebAgent-R1 (Wei et al., 2025) preprocessing, viewport option from its evaluation entry; adaptation.
OBS3 rule-based tree pruning	Prune the accessibility tree by rules.	AgentOccam (Yang et al., 2025b) `prune_tree`; adaptation.
OBS4 default AXTree text	Return host-preprocessed, flattened AXTree text.	BrowserGym (Le Sellier De Chezelles et al., 2025) default, also in the AWM (Wang et al., 2025b) port; framework path.
OBS5 LLM line retrieval	A model selects ranges of numbered tree lines.	FocusAgent (Kerboua et al., 2025) retriever via AgentLab; adaptation.
CTX: Context management		
CTX1 full observation history	Retain past page bodies and model responses in a multi-turn history.	WebAgent-R1 (Wei et al., 2025) template/parser; variant.
CTX2 no interaction history	Disable host and action history; keep past error logs and past reasoning disabled.	AgentLab (ServiceNow, n.d.a) switches; framework configuration.
CTX3 recent-observation append	Append up to three previous flattened observation texts.	Recent-observation window design (Lindenbauer et al., 2025).
CTX4 observation masking	Replace every past page body with a placeholder; keep responses and current observation.	WebAgent-R1 (Wei et al., 2025); adaptation; cf. masking (Lindenbauer et al., 2025).
CTX5 abstract examples and hints	Host variant with abstract examples and hints.	AgentLab (ServiceNow, n.d.a) examples/hints; framework configuration.
ST: State and memory		
ST1 structured task state	Deterministically log goal, actions, URL, repeats; show recent actions.	Avenir-Web (Li et al., 2026) task checklist; source-inspired.
ST2 in-task notes	Extract, summarize, and lexically select notes.	In-task memory module.
ST3 offline workflow memory	Inject website-routed workflows from a library fixed before evaluation.	AWM (Wang et al., 2025b) offline setting; the AWM WebArena runs use online memory; source-inspired.
CTR1 irrelevant notes [control]	Inject task-irrelevant notes into memory.	Memory content control with unrelated text in the note field.
CTL: Planning and control		
CTL1 subgoal planning	Initial subgoals; URL changes advance a pointer.	WebDART (Yang et al., 2025a); source-inspired.
CTL2 static planning	Inject one fixed plan from the initial observation.	PlanAhead (Zambrano et al., 2026) static planner; adaptation.
CTL3 model-based planning	LLM calls propose actions, imagine outcomes, score, and select.	WebDreamer (Gu et al., 2024) model-based planning; adaptation.
CTR2 irrelevant plan [control]	Inject a task-irrelevant plan into planning.	Planning content control with an unrelated plan.
ACT: Action interface		
ACT1 parse-error retry	Query the model again after a parse failure.	AgentLab (ServiceNow, n.d.a) retry; framework configuration.
ACT2 deterministic action-format repair	Validate restricted formats; try deterministic escaping and wrapper removal.	Motivated by AgentOccam (Yang et al., 2025b), SWE-agent (Yang et al., 2024) interfaces.

Table A.1, continued

Identifier and name	Operation	Design basis
VER: Verification and recovery		
VER1 execution-error feedback	Inject `last_action_error` into the feedback position.	BrowserGym (Le Sellier De Chezelles et al., 2025) errors, code (ServiceNow, n.d.b); framework signal.
VER2 early-stopping rules	Handle repeated actions, parsing-failure sequences, and post-answer fallback.	WebArena (Zhou et al., 2024b) rules, WebAgent-R1 code; adaptation.
VER3 no-change retry	Resend the last action when URL and tree summary are unchanged.	Retry mechanism.
VER4 pre-execution critic	The action model judges goal, URL, truncated tree, and proposal; bounded regeneration.	GUI-Critic-R1 (Wanyan et al., 2025) critic; BacktrackAgent (Wu et al., 2025) retry; source-inspired.
VER5 URL-conditioned self-feedback	Feedback from the previous action, response, and current URL.	BacktrackAgent (Wu et al., 2025); source-inspired.
VER6 action-effect verification	A model states expected effects; compare visible changes.	VeriGUI (Zhang et al., 2026d); adaptation.
DIF1 repeated-action early stop [split of VER2]	Enable repeated-action early stop; disable parse-error early stop.	WebArena (Zhou et al., 2024b) rules, WebAgent-R1 code; VER2 split.
DIF2 parse-error early stop [split of VER2]	Enable parse-error early stop; disable repeated-action early stop.	WebArena (Zhou et al., 2024b) rules, WebAgent-R1 code; VER2 split.
CTR3 fixed irrelevant feedback [control]	Inject fixed uninformative feedback with action history.	Feedback content control with fixed uninformative text.
PRD1 postcondition check with repair prompts [joint ST + VER]	Check URL or page-text end conditions; prompt repair if the unmet count stalls.	ContractSkill (Lu et al., 2026) conditions and repair; source-inspired.

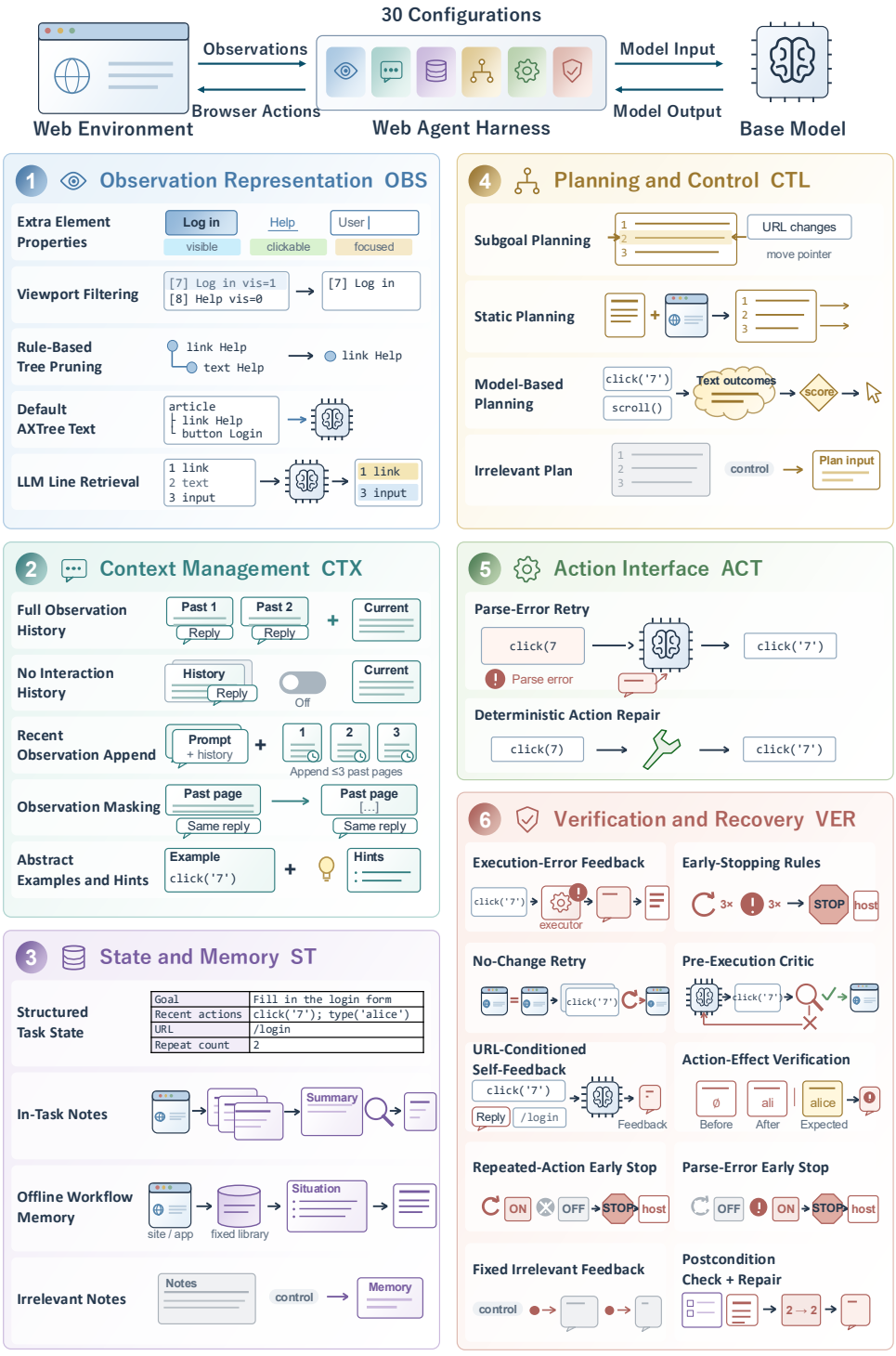


Figure A.1. All 30 configurations, grouped into the six functional categories of Figure 1; Table A.1 gives their operations and design sources.