VERA: VERIFICATION AND ENVIRONMENTAL RISK AWARENESS FOR FUNCTIONAL MODEL INDUCTION THROUGH OPEN-ENDED WEB EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-ended GUI exploration offers a way to acquire application knowledge before downstream tasks are specified. However, successful tool execution does not establish that a functional claim is supported, and the interactions used to test unknown functionality may involve environmental risks. We introduce VERA, a new framework for constructing functional models through evidence-based knowledge admission and contextual risk awareness. Each function is represented as a hypothesis with an observable outcome specified before execution. VERA compares subsequent interface observations against this expectation, admits supported claims, and retains unresolved or unsuccessful attempts for inspection and further exploration. Persistent hypothesis pools and context recovery sustain evidence collection, while risk annotations associate selected actions with their acquisition context. In controlled web exploration studies, evidence-based admission raises knowledge precision from 82.5% to 97.0% while retaining 32 of 33 human-supported functions. Full exploration attains greater functional coverage at additional cost, while visual context improves risk categorization on an external context challenge. These results demonstrate VERA's effectiveness in acquiring reliable functional knowledge while making the environmental risks of its acquisition explicit and traceable. The source code is available at `https://anonymous.4open.science/r/AI-Web-explorer-4516`.

1 INTRODUCTION

Graphical user interface (GUI) agents extend language model capabilities from generating responses to interacting with digital environments. By interpreting interfaces and issuing clicks, text inputs, and navigation actions, they offer a common interaction paradigm for information retrieval, document editing, online shopping, software operation, and workflows across web, desktop, and mobile applications (Deng et al., 2023; Zhou et al., 2024; Xie et al., 2024; Rawles et al., 2025). Reliable interaction involves more than locating an interface element. An agent must also understand which operations an application supports, where they are available, and what outcomes to expect. Persistent application knowledge can support navigating unfamiliar interfaces and organizing subsequent tasks, rather than requiring the agent to reconstruct the same understanding for every request.

Open-ended GUI exploration offers a practical way to acquire this knowledge before a specific user goal is given. Early systems collect application experience through interaction, demonstrations, or dynamic analysis (Zhang et al., 2025; Wen et al., 2024). Subsequent work develops function-aware exploration goals and systematic functionality coverage evaluation (Xie et al., 2025; Nica et al., 2025), while more recent systems organize acquired experience into knowledge graphs, state-machine memories, and executable routines for later decisions (Yu et al., 2026a; Zhong et al., 2026; Qin et al., 2026; Chai et al., 2026). These advances establish exploration as a source of reusable application knowledge for navigation, planning, and task execution. Reuse also extends the consequences of an exploratory mistake. A claim retained from one interaction may guide decisions in later contexts, even when the evidence that originally supported it is no longer visible to the agent.

However, coverage and subsequent task performance alone do not characterize the evidential status of the functional knowledge retained through exploration. A discovered control, a successful

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.

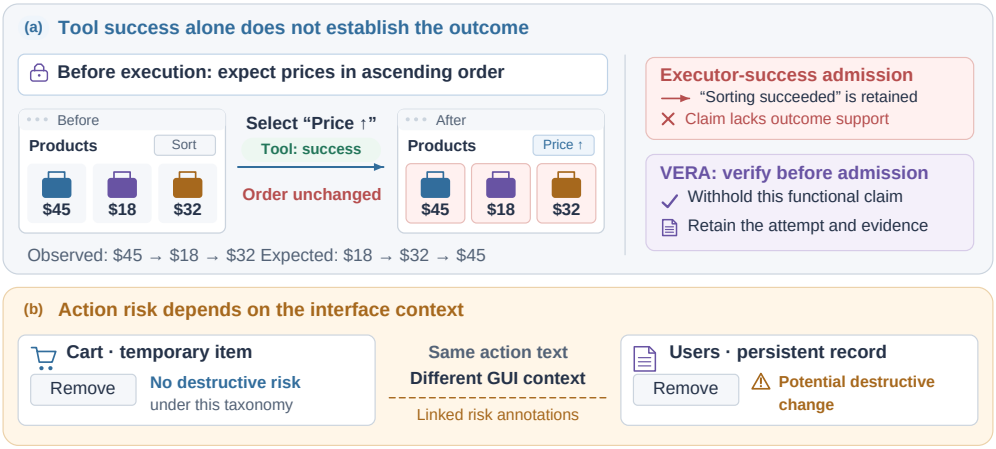


Figure 1: **From interaction signals to justified knowledge.** (a) In the schematic sorting example, the executor reports success but the observed item order does not support the saved expectation. VERA withholds the claim while retaining the attempt and its evidence. (b) The same action text can have different risk implications in different interfaces, which VERA records as contextual risk annotations.

tool invocation, and an observed functional outcome are distinct signals. Figure 1(a) illustrates how selecting a sorting option need not produce the intended reordering. Existing systems validate transitions, verify action effects, and audit learned structures (Zhong et al., 2026; Zhang et al., 2026; Chai et al., 2026), but these capabilities do not by themselves establish which criterion should admit a reusable functional claim or how much supported knowledge a stricter criterion rejects. Moreover, the interactions used to obtain evidence can enter payment workflows, prepare external communications, or modify records. Contextual risk assessment and predictive safeguards address such operations (Zheng et al., 2025; Chen et al., 2026; Nayak et al., 2026), yet identifying a risky interaction does not determine whether its functional outcome occurred. Conversely, a correctly observed outcome does not make the operation risk-free. An account of functional knowledge acquisition therefore needs to connect what is claimed, what is observed, and the context in which the evidence was obtained, while keeping evidential support distinct from operational risk.

To address these limitations, we introduce VERA, a framework for *Verification and Environmental Risk Awareness* in open-ended web exploration. VERA represents each candidate function as a testable hypothesis and saves an expected observable outcome before execution. The resulting observations are checked against this fixed expectation, rather than allowing the intended result to be redefined after the interaction. Supported claims enter an admitted-knowledge view, while unsuccessful and unresolved attempts remain in the complete record store. Persistent hypothesis pools and context recovery allow evidence collection to continue across revisited interfaces. In parallel, a risk annotator associates the current interface, selected action, and potential risk with the same hypothesis and interaction record. Pre-action risk assessment characterizes the evidence-gathering operation, post-action verification assesses outcome support, and admission determines which functional claims are retained for reuse. Risk annotations remain observational and do not alter execution or knowledge admission.

We evaluate these components through complementary studies of knowledge admission, functional discovery, and contextual risk recognition. On identical exploration traces, VERA removes six of seven unsupported admissions made by accepting executor success while retaining 32 of 33 human-supported functions. Repeated exploration studies examine how persistent recovery affects attained coverage and interaction cost, while input ablations and an external context challenge test the role of visual information in risk recognition. Together, these evaluations distinguish the reliability of retained knowledge from the extent of exploration and the potential risk of its interactions. This separation makes it possible to assess evidence verification without mistaking more activity or successful tool execution for better functional understanding.

The main contributions are as follows:

- **We propose knowledge admission as an explicit objective of open-ended exploration.** Our formulation distinguishes discovering a function from justifying its reuse, and assesses the support for a claim separately from the potential risk of the interaction that produced it. This brings knowledge reliability and acquisition context into the evaluation of functional exploration.
- **We introduce VERA to synergize functional verification and contextual risk awareness.** The framework fixes observable expectations before execution and verifies outcomes before admitting knowledge. Persistent hypotheses sustain evidence collection, while linked risk annotations make the circumstances of knowledge acquisition inspectable.
- **We demonstrate substantial gains in knowledge precision with high supported-function retention.** On identical traces, VERA reduces unsupported admissions from seven to one while retaining 32 of 33 supported functions. Repeated exploration attains higher coverage with explicitly measured recovery costs, and visual context improves acceptable risk-type accuracy by 35.4 percentage points on the external context challenge.

2 Related Work

2.1 Autonomous GUI Exploration

Autonomous exploration complements task-conditioned GUI control by acquiring application experience that can be reused rather than reconstructed for every task. Early systems such as AppAgent and AutoDroid collect app-specific knowledge through interaction, demonstrations, or dynamic analysis (Zhang et al., 2025; Wen et al., 2024). Another line uses exploration to obtain training data: NNetNav retrospectively labels exploratory browser trajectories, while GUI-Bee collects environment-specific experience for adapting action grounding (Murty et al., 2025; Fan et al., 2025). At the functional level, GUI-explorer generates function-aware exploration goals and extracts transition-aware knowledge, and UIExplore-Bench formalizes functionality discovery and evaluates exploration under structured and screenshot-based observations (Xie et al., 2025; Nica et al., 2025). These approaches establish exploration as a source of both reusable knowledge and learning data.

A related direction concerns how acquired structure supports subsequent decisions. Agent+P uses UI transition graphs for symbolic planning (Ma et al., 2026). GraphPilot constructs application knowledge graphs for sequence-level reasoning, while ActionEngine validates explored transitions and stores state-machine memory for programmatic execution (Yu et al., 2026a; Zhong et al., 2026). EAM combines state-aware exploration with action-group mining to construct executable memory (Qin et al., 2026); UI-KOBE studies semantic state-transition graphs as local guidance for lightweight agents, including graph refinement and re-exploration (Chai et al., 2026). These studies demonstrate how acquired representations support learning, navigation, and planning. VERA instead makes the functional claim an explicit unit of evidence assessment and knowledge admission. Its evaluation separates candidate discovery, executor status, and observed outcome support, while measuring both admission precision and retention of supported knowledge.

2.2 Knowledge Verification and Environmental Risk Awareness

Reliable reuse depends on assessing what an interaction actually establishes, not merely storing its trajectory. Research on digital-agent evaluation has developed autonomous evaluators for refinement (Pan et al., 2024) and examined the agreement of trajectory judges with expert assessments (Lù et al., 2025). At the action level, VeriGUI carries predicted effects across steps to verify execution and guide recovery (Zhang et al., 2026). At the knowledge structure level, UI-KOBE audits duplicate states and inconsistent transitions (Chai et al., 2026). Interactive Reward Agent gathers additional task-completion evidence through environment tools (Shi et al., 2026); VAGEN progressively combines trajectory evidence with active environment probing (Cui et al., 2026). VERA builds on this established role of verification, applying it to functional claims whose expected outcomes are recorded before execution. Its evaluation concerns their admission into persistent knowledge and the support retained with them, complementing action correction, graph auditing, and task completion assessment.

Environmental risk introduces a complementary concern: an interaction may accomplish its intended function yet expose data, alter resources, or initiate a sensitive workflow. ToolEmu studies conse-

quential agent failures in an emulated tool environment, and ST-WebAgentBench evaluates web-task completion under explicit safety policies (Ruan et al., 2024; Levy et al., 2026). For execution-time risk assessment, WebGuard classifies proposed web actions in their current page context, and SafePred predicts consequences to guide safety intervention (Zheng et al., 2025; Chen et al., 2026). OS-Sentinel combines formal and contextual safety judgment, while SeerGuard uses world-model predictions for pre-execution assessment (Sun et al., 2026; Yu et al., 2026b). OSGuard separately evaluates local action judgments and end-to-end task safety (Mohammadmirzaei & Flanigan, 2026). Guided exploration also studies user-sensitive states from demonstrated tasks (Nayak et al., 2026). VERA examines these risks during open-ended functional knowledge acquisition, linking each contextual assessment to the selected interaction and its supporting artifacts. The taxonomy includes entry into or preparation for sensitive workflows, and the annotations are observational. This design makes evidential support and acquisition risk separately inspectable within a common functional model.

3 VERA

VERA converts exploratory interactions into an inspectable functional model by linking each claim to a criterion fixed before execution and the observations obtained afterward. As shown in Figure 2, the explorer proposes and selects a hypothesis, executes a concrete instance, and verifies its outcome before deciding whether to admit the claim. Persistent pools retain unfinished hypotheses for later recovery. In parallel, risk annotations record the context of selected actions without controlling execution or admission.

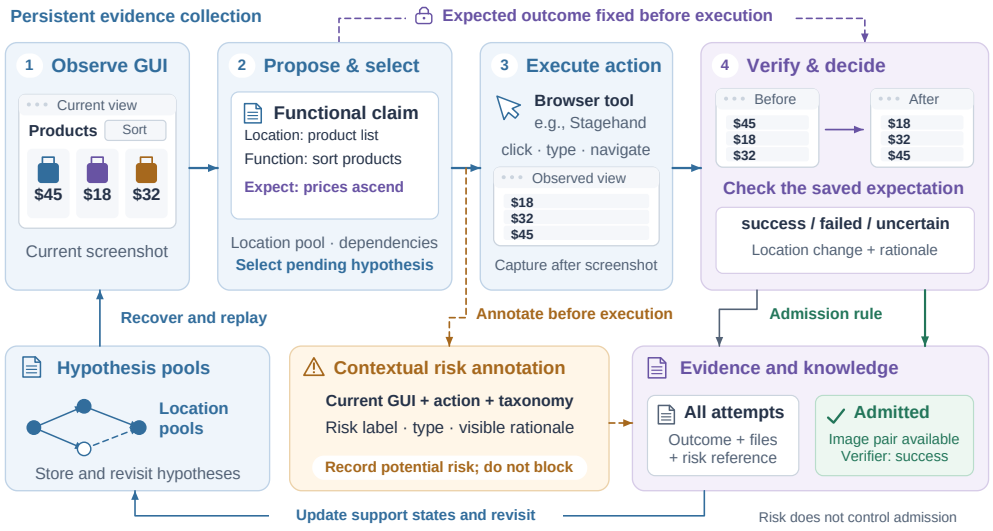


Figure 2: **The VERA framework.** Numbered stages follow the interaction sequence: observe, propose and select, execute, and verify. The expected outcome is saved before execution and passed to the verifier. All attempts remain in the record store; admission additionally requires both screenshots to be available and a successful outcome judgment. Persistent pools support recovery and continued exploration. Dashed orange links associate contextual risk annotations with the selected action and its interaction record.

3.1 Problem Formulation

We consider an application accessed through an entry URL, a browser interface, and test credentials when required. The explorer receives neither a downstream business objective nor a complete function inventory. For an ordinary exploration attempt t, let a_t denote the selected high-level action instance and o_t^- and o_t^+ the screenshots captured before and after execution, respectively. A functional hypothesis is $h = (\ell, f, e)$, where ℓ is a semantic location, f describes a function, and e specifies its expected observable outcome before execution. The concrete action instance and hy-

pothesized direct dependencies are associated metadata. A semantic location is a stable business context, such as a product list or checkout form, rather than a complete valuation of application state. Different parameters can instantiate the same function.

An attempt record τ_t links the hypothesis and action to the available observations, executor status, outcome judgment, optional risk annotation, and artifact references. The complete record store retains every attempt, whereas an admitted-knowledge view contains claims satisfying the evidence rule. SUPPORTED, UNSUPPORTED, and UNRESOLVED summarize this evidential status: a successful judgment with the required observations provides support; diagnostic observations may fail to support the expectation; missing, partial, or unjudgeable evidence remains unresolved. These summaries are distinct from runtime retry and scheduling states. An unsuccessful attempt does not establish that the application lacks the function, and human reference labels are used only for evaluation.

3.2 PERSISTENT EVIDENCE COLLECTION

VERA uses a vision language model to identify visible business operations, executable instances, expected outcomes, and direct dependencies from the current interface and exploration context. A candidate requires an observable affordance or a visible prerequisite operation. Related form fields can form one semantic operation, whereas static text and completed historical operations are not proposed as new executable functions; an empty proposal set is permitted. The expectation e is saved before execution and is not revised after observing the result. It fixes the claim being tested while allowing equivalent visible evidence to satisfy it. Hypotheses are stored in pools indexed by semantic location and selected deterministically subject to proposed dependencies and retry limits. Dependencies describe local ordering supported or suggested by observed interactions, not universally necessary business preconditions.

When the current location has no selectable candidate, the explorer can return to a frontier with unfinished hypotheses by navigating to the entry point, replaying a saved semantic path, and checking the recovered location. Ordinary exploration then resumes and can generate new candidates. Returning to an entry page does not reset persistent application data, so recovery failures, repeated effects, and additional calls are recorded. Stagehand executes each selected instance through one or more browser interactions. Proposal and outcome verification use interface images, while execution can use DOM information and locators. The framework thus separates the visual evidence used to assess a claim from the browser mechanisms used to carry out its action.

3.3 FUNCTIONAL VERIFICATION AND KNOWLEDGE ADMISSION

The verifier directly receives the action description, the selected hypothesis's frozen expectation e_t, and screenshots o_t^- and o_t^+. It returns an outcome $y_t \in \{\mathsf{success}, \mathsf{failed}, \mathsf{uncertain}\}$, a semantic-location change flag, and a brief rationale grounded in visible evidence. Functional success and location change are independent: sorting can succeed without navigation, whereas navigation alone need not satisfy a functional claim. Backend state, reference labels, and the complete low-level trajectory are not verifier inputs. An attempt is admitted only when the required observations are available and the expected outcome is judged successful:

$$q_E(\tau_t) = \mathbf{1}[\mathrm{available}(\tau_t)]\,\mathbf{1}[y_t = \mathsf{success}]. \tag{1}$$

Here $\mathbf{1}[\cdot]$ is the indicator function. In the evaluated implementation, $\mathrm{available}(\tau_t)$ checks that both screenshot files exist; the verifier, not this availability check, assesses semantic support. Missing observations or unresolved outcomes do not pass the gate. Eligibility of an entire run is specified separately in the experimental protocol.

For a canonical function key k, let $\mathcal{T}(k)$ be its recorded attempts within the scope of an evaluation and let q be a binary admission rule. In the reported study, k comprises the application, semantic location, and canonical action identifier. The expected outcome remains attached to each attempt rather than becoming part of the aggregation key. The admitted function set is

$$A_q = \{k : \exists \tau \in \mathcal{T}(k),\ q(\tau) = 1\}. \tag{2}$$

Thus one supported attempt is sufficient to admit a function under q_E, but every attempt is judged against its own frozen expectation. The evaluation specifies whether attempts are pooled within or

across runs. This aggregation records observed support; it does not make all expectations equivalent or establish executability in unobserved states. Claims without an admitted attempt remain available for inspection and further evidence collection.

3.4 Contextual Environmental Risk Awareness

Before an ordinary or replay action is executed, a separate annotator receives the current screenshot, action label and description, and a versioned risk taxonomy. It returns a binary potential-risk label, one primary category when risk is identified, and a short visible-evidence rationale. The five categories are destructive changes, financial transactions, external communication, account or permission changes, and sensitive data. Their definitions cover entering or preparing sensitive workflows as well as performing the corresponding operation. For example, preparing checkout can be risk-positive without completing a payment, whereas removing a temporary cart item is not destructive by default.

A risk annotation describes an action in its current context rather than a permanent property of the abstract function. Ordinary exploration links the annotation to its hypothesis and attempt; replay links it to the corresponding recovery step. The annotator records potential risks without changing action selection, execution, verification depth, or knowledge admission. Annotation errors are logged and execution continues. Assessment is performed for selected high-level calls, not every internal browser operation. This keeps the risks of obtaining evidence inspectable alongside the support for retained claims. An auxiliary symbolic projection is described in Appendix H.

4 Experiments

4.1 Setup

Environments and data. We use SauceDemo[1] and Practice Shopping, two shopping test applications, for interactive knowledge acquisition. Runs begin at the entry point in a new browser context, which does not restore all persistent application data. The admission study uses three runs on each site, while exploration evaluates three conditions with three runs per condition and site. Risk recognition is evaluated offline on 106 actions selected during exploration and an external challenge of 100 screenshot and action cases spanning 80 websites. The external cases broaden the context-recognition evaluation to interfaces drawn from those websites.

Execution settings. Each formal run permits up to 25 ordinary high-level attempts and may stop earlier when candidates and recoverable frontiers are exhausted. Recovery calls are accounted for separately, so the comparison fixes the ordinary-attempt cap while reporting recovery cost independently. Ten-attempt exploration pilots are excluded. The interactive studies use `gpt-4o` as the configured model identifier; risk evaluation uses the same identifier at temperature 0 and records the model identifier returned by the service. Models, prompts, budgets, and available configuration details are documented in Appendix A.

Comparisons and metrics. Knowledge admission compares three rules on identical frozen proposals and traces, reporting precision, supported retention, admission yield, and F1. Exploration compares Random, Linear, and Full using the same proposer, executor, and outcome verifier; outcomes are measured by supported-function coverage and recorded calls. Risk evaluation varies action text, visual context, and taxonomy information. Binary precision, recall, and F1 are its primary metrics, with risk-type accuracy as a prespecified secondary metric. These are controlled comparisons of VERA components.

Reference labels and statistical units. Reference labels are AI-initialized and then reviewed by a human; admission review hides executor status, system outcome, and admission decisions. Review distinguishes a visible affordance from a demonstrated outcome and can accept core functional success despite unnecessarily narrow presentation details in an expectation. Admission metrics pool canonical function keys across runs, whereas exploration averages runs within each site before weighting the two sites equally. Independent coverage inventories contain 22 SauceDemo and

[1]Upstream application: `https://github.com/saucelabs/sample-app-web`.

Table 1: **Knowledge admission on identical exploration traces.** Results pool 45 canonical functions across six runs; 33 have human-supported outcomes. Unsupported counts are relative to the reference labels. Precision, retention, and F1 are percentages (%); the remaining columns are counts.

Admission policy	Admitted	Unsupported ↓	Precision ↑	Retention ↑	F1 ↑
Proposal-as-fact	45	12	73.33	100.00	84.62
Executor-success	40	7	82.50	100.00	90.41
Evidence-grounded	33	**1**	**96.97**	96.97	**96.97**

16 Practice Shopping functions and are not supplied to the explorer. Risk aggregation and paired uncertainty estimates follow their dataset-specific units, detailed in Appendix G.

4.2 Knowledge Admission

We compare **Proposal-as-fact**, admitting every proposed function; **Executor-success**, admitting a function after any executor-successful attempt; and **Evidence-grounded admission**, using Equation 1. All three operate on the same six-run corpus of 103 attempts and 45 cumulative function keys, of which 33 are human-supported. The comparison isolates admission decisions from differences in newly generated trajectories. Precision measures the supported fraction of admitted knowledge; supported retention measures the fraction of the human-supported candidate set retained. Their F1 balances reliability against rejection of useful knowledge. Supported retention is computed over the human-supported candidate set, while cumulative counts pool canonical function keys across runs. Formal metric definitions and admission yields appear in Appendix C.

Admission reliability. Table 1 shows that evidence-grounded admission raises precision from 82.50% to 96.97% relative to executor-success admission while retaining 32 of 33 supported functions. It removes six unsupported admissions and withholds one additional supported function. The accompanying F1 increase from 90.41% to 96.97% reflects higher precision with little loss of human-supported knowledge. The pattern is consistent across the two applications: evidence precision and retention are 94.12% on SauceDemo and 100% on Practice Shopping. These results evaluate the admission protocol as a whole, including the quality of the saved expectations.

Sources of disagreement. One form-completion claim was admitted without direct evidence for every required field, whereas a sorting function was withheld despite visible reordering. An overly specific cart expectation also rejected an attempt that opened checkout directly, even though human review accepted its core function. Such cases distinguish insufficient support, missed visible effects, and expectation mismatch. They also show why rejecting a claim should preserve its evidence rather than be interpreted as functional nonexistence. Appendix H provides the recorded case descriptions.

4.3 Functional Discovery and Recovery

Controlled comparison. Random samples candidates without dependency checks or recovery. Linear selects deterministically with dependency checks but no recovery. Full adds frontier recovery to Linear. A function is covered when an attempt has available observations, a human-confirmed successful outcome, and a match to the independent inventory; each inventory item counts at most once per run. Random versus Linear changes both selection and dependency handling. Full versus Linear examines the addition of recovery together with its extra cost.

Coverage and cost. Figure 3 shows that Full attains the highest mean coverage on both sites. Equal-site averages are 28.8% for Random, 42.6% for Linear, and 54.5% for Full. The difference between Full and Linear is 19.7 percentage points on SauceDemo and 4.2 on Practice Shopping, averaging 11.9 points. These gains accompany additional interaction: the three SauceDemo Full runs use 62 ordinary attempts and 76 recovery/reset calls, versus 34 ordinary attempts for Linear. On Practice Shopping, Full uses 68 ordinary attempts and one recovery/reset call, while Linear uses 49 ordinary attempts and no recovery calls. The result establishes greater attained coverage under the ordinary-attempt cap, with the additional recovery cost reported explicitly.

Recovery behavior. The archived analysis records two newly covered functions after recovery in Practice Shopping Run 2 and three each in SauceDemo Runs 2 and 3. This is consistent with recov-

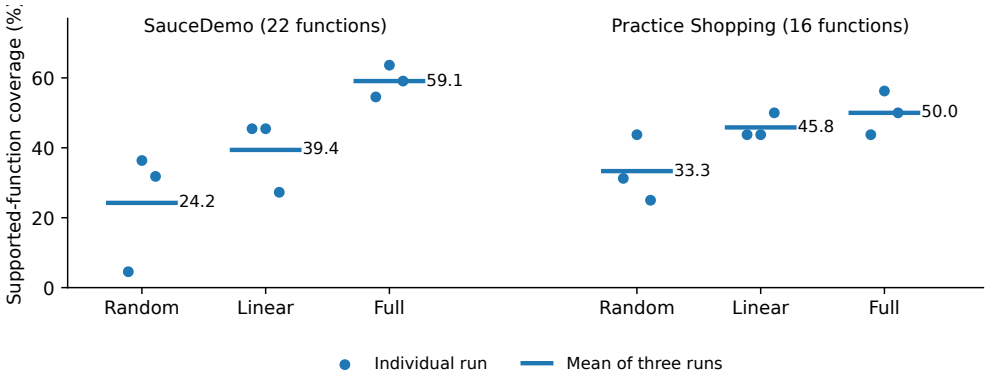


Figure 3: **Supported functional discovery.** Each point is one completed run; horizontal segments show the mean of three runs. Coverage (%) uses 22 SauceDemo and 16 Practice Shopping inventory items. All runs share a cap of 25 ordinary high-level attempts and can stop early; Full incurs additional recovery calls.

Table 2: **Risk recognition on exploration-selected actions.** Prespecified site-macro averages (%). Primary-type accuracy uses all gold-positive samples, with missed positives counted as incorrect.

Input	Precision	Recall	F1	Primary-type acc.
Action only	90.4	97.5	**93.7**	52.5
Action + taxonomy	89.0	95.0	91.9	75.8
Action + visual	**96.2**	90.0	92.4	65.0
Full	87.8	**100.0**	93.5	**85.0**

ery restoring access to unfinished hypotheses. However, two Practice Shopping Full runs invoke no replay, so not every difference between Full and Linear can be attributed to recovery alone. Exact run endpoints, the per-attempt growth curves, and the complete cost ledger are retained in Appendix D; implementation repairs and replacement runs are documented in Appendix A.

Progressive evidence acquisition. The per-attempt trajectories in Appendix D show where supported knowledge accumulates rather than only where exploration terminates. After ten ordinary attempts, the SauceDemo Full runs cover 5–8 inventory functions; their final counts reach 12–14. All three Practice Shopping Full runs cover three functions at the same checkpoint and finish with 7–9. Thus substantial evidence is collected beyond early interactions in both applications.

4.4 Contextual Risk Awareness

Exploration-linked input ablation. We retain the first occurrence of each run-local (site, location, action) key among 130 ordinary attempts from the six Full runs, yielding 106 samples. Their 32 positive labels comprise 14 financial-transaction and 18 sensitive-data cases; replay is excluded from classification accuracy. The four conditions share action text and allowed category identifiers. Taxonomy conditions additionally receive definitions and examples, and visual conditions receive the screenshot, with instructions matched to the available inputs. The four-condition evaluation produces 424 valid offline predictions. The source audit in Appendix G reports runtime risk records for all 130 ordinary attempts and source linkage for all 106 retained samples. Each sample is linked to its source interaction, pre-action screenshot, and runtime risk record. The audit therefore provides traceable source linkage for every retained sample, while the reported classification results are computed from the matched offline predictions on the frozen inputs.

As shown in Table 2, Full obtains 100% observed recall and 85.0% primary-type accuracy under site-macro averaging. Primary-type accuracy exceeds both the 75.8% obtained without visual context and the 65.0% without taxonomy definitions. Binary separation is already strong with action text alone: its F1 is 93.7%, compared with 93.5% for Full, and every paired F1 difference interval includes zero. The more consistent benefit of richer input is therefore identifying the relevant risk

category rather than improving binary detection. Per-site results, pooled counts, and paired intervals are reported in Appendix G.

External context challenge. To test supplied actions across more varied interfaces, we sample 100 cases from final-reviewed WebGuard metadata (Zheng et al., 2025): 30 candidate same-action context pairs and 40 diversity samples spanning 80 websites. Action text is reconstructed mechanically from visible target labels. Original WebGuard severity labels are used only for stratified sampling and hidden during VERA annotation and inference; they are not converted into VERA gold. Ten pilot screenshot IDs are excluded. Human-confirmed VERA labels contain 65 positives and 35 negatives. Text + taxonomy and Context-conditioned receive identical action text and taxonomy, with only the latter receiving a screenshot. All 200 predictions are valid. WebGuard metadata provides the source pool, while evaluation uses human-confirmed VERA labels under matched input conditions.

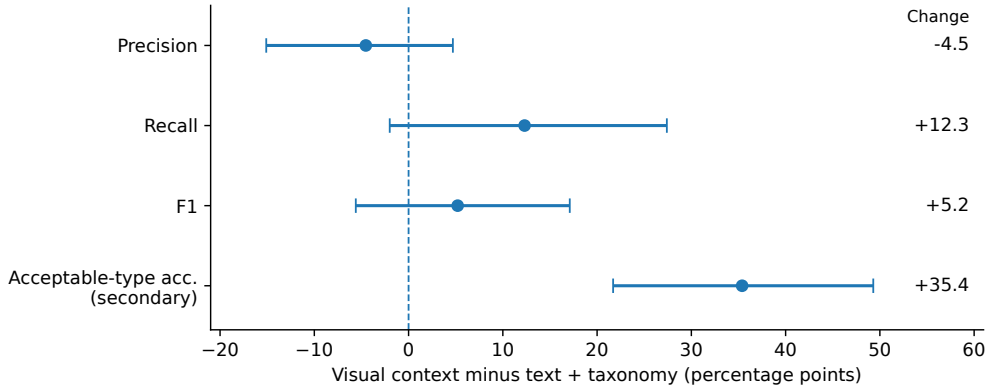


Figure 4: **Effect of visual context.** Change from text plus taxonomy to the same inputs with visual context on the external context challenge. Points are percentage-point differences from the reported counts; bars are the frozen paired 95% bootstrap intervals. Acceptable-type accuracy is a prespecified secondary endpoint. Only its difference interval excludes zero.

Figure 4 highlights a 35.4 percentage point gain in acceptable-type accuracy from visual context, with a paired 95% interval of [21.7,49.3]. Acceptable-type accuracy increases from 27.7% to 63.1%. This secondary endpoint provides the clearest evidence that the interface helps disambiguate the kind of risk associated with an action. Binary recall rises from 58.5% to 70.8%, while precision falls from 77.6% to 73.0%. F1 increases by 5.2 points, but its interval crosses zero. Thus the evidence supports improved risk categorization more clearly than an overall improvement in binary detection.

Paired error analysis. Visual context corrects 25 type decisions and introduces two new type errors among the 65 positives, compared with 16 corrections and 14 introduced errors for binary detection. These paired counts show a clearer benefit for risk categorization than for binary risk detection. Appendix G reports all metric values, intervals, and the small subset of binary-discordant context pairs.

5 Conclusion

VERA synergizes functional hypotheses, execution evidence, knowledge admission, and contextual risk records within open-ended web exploration. Explicit outcome verification increases admission precision while retaining most supported functions. Persistent exploration attains higher coverage with additional interaction cost, and the external context study shows that visual information is particularly useful for risk categorization. These findings support evaluating exploration through complementary measures of knowledge reliability, discovery cost, and contextual risk. The resulting representation preserves both the basis of a functional claim and the circumstances in which its evidence was obtained. Together, these mechanisms provide a practical approach to reusable application knowledge with inspectable evidence and explicit acquisition risks, separating the success of a tool call from support for a persistent functional claim.

AI USE STATEMENT

Generative AI tools were used to assist with writing, language editing, formatting, and limited code development. All AI-assisted text and code were reviewed by the authors, and the generated code was further verified and tested. The authors take responsibility for the final manuscript.

ETHICS STATEMENT

The interactive studies use test web applications. External risk assessment uses source screenshots without executing the proposed actions. Releasing screenshots and traces requires appropriate permissions and redaction of sensitive information.

REPRODUCIBILITY STATEMENT

Sections 3–4 describe the method and evaluation. Recorded configurations, function inventories, annotation and sampling protocols, and supplementary results are provided in the appendix. Condensed prompt templates are provided in Appendix F.

REFERENCES

Yuxiang Chai, Han Xiao, Xinyu Fu, Jinpeng Chen, Rui Liu, and Hongsheng Li. Ui-kobe: Knowledge-oriented behavior exploration for lightweight graph-guided gui agents. *arXiv preprint arXiv:2605.29534*, 2026.

Yurun Chen, Zeyi Liao, Ping Yin, Taotao Xie, Keting Yin, and Shengyu Zhang. Safepred: A predictive guardrail for computer-using agents via world models. *arXiv preprint arXiv:2602.01725*, 2026.

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic reward modeling: Verifying gui agent via online proactive interaction. *arXiv preprint arXiv:2602.00575*, 2026.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023.

Yue Fan, Handong Zhao, Ruiyi Zhang, Yu Shen, Xin Eric Wang, and Gang Wu. GUI-bee: Align GUI action grounding to novel environments via autonomous exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33261–33278, 2025.

Ido Levy, Ben wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. St-webagentbench: A benchmark for evaluating safety and trustworthiness in web agents. In *International Conference on Learning Representations*, pp. 122213–122255, 2026.

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. *arXiv preprint arXiv:2504.08942*, 2025.

Shang Ma, Xusheng Xiao, and Yanfang Ye. Agent+p: Guiding ui agents via symbolic planning. *arXiv preprint arXiv:2510.06042*, 2026.

Mina Mohammadmirzaei and Jeffrey Flanigan. Osguard: A benchmark for safety in computer-use agents. *arXiv preprint arXiv:2606.15034*, 2026.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D Manning. NNetNav: Unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2025.

Aradhana Nayak, Mussadiq Nazeer, Wang Peng, and Feng Liu. GUI agent: Guided exploration of user-sensitive screens. *arXiv preprint arXiv:2606.25705*, 2026.

Andrei Cristian Nica, Akshaya Vishnu Kudlu Shanbhogue, Harshil Shah, Aleix Cambray, Tudor Berariu, Lucas Maystre, and David Barber. Toward autonomous ui exploration: The uiexplorer benchmark. *arXiv preprint arXiv:2506.17779*, 2025.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024.

Zerui Qin, Sheng Yue, Xingyuan Hua, Yongjian Fu, and Ju Ren. Executable agentic memory for gui agent. *arXiv preprint arXiv:2605.12294*, 2026.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, pp. 406–441, 2025.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, pp. 27031–27098, 2024.

Chenrui Shi, Yuwei Wu, Yang Liu, Ruining Feng, Zirui Shang, Zhi Gao, Lifeng Fan, and Che Sun. Interactive reward agent: Gui task evaluation via environment-state verification. *arXiv preprint arXiv:2607.25904*, 2026.

Qiushi Sun, Mukai Li, Zhoumianze Liu, Zhihui Xie, Fangzhi Xu, Zhangyue Yin, Kanzhi Cheng, Zehao Li, Zichen Ding, Qi Liu, Zhiyong Wu, Zhuosheng Zhang, Ben Kao, and Lingpeng Kong. OS-sentinel: Towards safety-enhanced mobile GUI agents via hybrid validation in realistic workflows. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9529–9553, 2026.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 543–557, 2024.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. GUI-explorer: Autonomous exploration and mining of transition-aware knowledge for GUI agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5650–5667, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024.

Mingxian Yu, Siqi Luo, and Xu Chen. GraphPilot: GUI task automation with one-step LLM reasoning powered by knowledge graph. *arXiv preprint arXiv:2601.17418*, 2026a.

Xue Yu, Bo Yuan, Kailin Zhao, Pengshuai Yang, Hong Hu, and Junlan Feng. SeerGuard: A safety framework for mobile GUI agents via world model prediction. *arXiv preprint arXiv:2607.15550*, 2026b.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI conference on human factors in computing systems*, pp. 1–20, 2025.

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, 2026.

Boyuan Zheng, Zeyi Liao, Scott Salisbury, Zeyuan Liu, Michael Lin, Qinyuan Zheng, Zifan Wang, Xiang Deng, Dawn Song, Huan Sun, and Yu Su. WebGuard: Building a generalizable guardrail for web agents. *arXiv preprint arXiv:2507.14293*, 2025.

Hongbin Zhong, Fazle Faisal, Luis França, Tanakorn Leesatapornwongsa, Adriana Szekeres, Kexin Rong, and Suman Nath. ActionEngine: From reactive to programmatic GUI agents via state machine memory. *arXiv preprint arXiv:2602.20502*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, pp. 15585–15606, 2024.

A Implementation and Reproducibility Details

The interactive studies use the core execution settings in Table 3. Function inventories and coverage matching refer to the tested application versions.

For reproducibility, we refer to the archived C1 experiment configurations as SauceDemo v1 and Practice Shopping v4.

Table 3: Core execution settings for the interactive studies.

Setting	C1 SauceDemo v1	C1 Practice Shopping v4	C2
Runs	3	3	3/site/condition
Stagehand / outcome alias	gpt-4o / gpt-4o	gpt-4o / gpt-4o	gpt-4o
Execution mode	observe_act	observe_act	observe_act
Ordinary attempt cap	25	25	25
Attempts/candidate	2	2	2
Replay/frontier; total	2; 4	2; 4	2; 4 (Full)
Outcome temperature	0	0	0

C1 Practice Shopping v4 and C2 allow up to 12 candidates and two visual scans per location. C1 Practice Shopping v4 uses a 1,400-token candidate/visual response cap. C2 uses a 1440×1000 viewport and no business-specific planning profile.

Run eligibility and replacements. C1 uses the three SauceDemo v1 runs and three Practice Shopping v4 runs. SauceDemo v1 requires at least 95% completeness of necessary artifacts. Practice Shopping v4 retains incomplete attempts within a valid run, but those attempts cannot pass evidence admission. Earlier Practice Shopping versions and zero-attempt diagnostic runs are excluded. In C2, pilot runs and the instrumentation-invalid v1 run are excluded; the final matrix uses v2 with reset calls included in recovery cost.

Table 4: C2 replacement provenance for the final 18-run matrix. Original failed runs are excluded from the reported matrix.

Final run	Reason for replacement	Included record
Practice Shopping Full 01	Initial external call stalled before a graph checkpoint	First replacement
Practice Shopping Full 02	Browser spawn, embedding-write, and trace-write failures in successive attempts	Third replacement
SauceDemo Random 02	Browser spawn failure before a graph checkpoint	First replacement; seed 5102 retained
SauceDemo Linear/Full 03	Login-action type incompatibility	Repaired runs; originals archived

Application and execution settings. The application entry points are `https://www.saucedemo.com/` and `https://practiceautomatedtesting.com/shopping`. SauceDemo execution includes a site adapter for test login credentials and cart-link targeting. This execution support is separate from the evaluation inventory. Random samples candidates, whereas Linear and Full use deterministic candidate selection.

B VERA Exploration Procedure

The following summarizes the documented control flow at the selected high-level action boundary. It separates runtime execution from the evidence-based admitted view used for reporting.

Algorithm 1 Open-ended evidence production and admission

Require: Entry, browser, ordinary-attempt budget, proposal/verifier models
Initialize candidate pools, records, graph, and admitted view
while ordinary-attempt budget remains **do**
    Observe the interface and identify its semantic location
    Propose candidates when needed; save expectations and dependencies
    Select a candidate according to the experimental condition
    if no selectable candidate exists **then**
        if recovery is allowed and a frontier is recoverable **then**
            Navigate to entry, recording reset cost
            for each saved high-level action in the recovery path **do**
                Assess its risk; execute it; retain its record and call cost
            end for
            Verify the recovered location
            continue
        else
            break
        end if
    end if
    Save pre-action observation and optional risk annotation
    Execute the high-level instance; record status and available after evidence
    Verify the frozen expectation when observations are comparable
    Preserve all results; update candidate lifecycle and graph
    Update admitted view using the system evidence rule
end while
Optionally project a screened graph into a downstream representation

C KNOWLEDGE ADMISSION: DATA AND METRICS

Let C be the discovered canonical function keys, $G \subseteq C$ the human-supported reference set, and $A \subseteq C$ the admitted set. Precision P, supported retention R_s, admission yield Y, and the F1 score of P and R_s are

$$P = \frac{|A \cap G|}{|A|}, \qquad R_s = \frac{|A \cap G|}{|G|}, \qquad Y = \frac{|A|}{|C|}, \qquad F_1 = \frac{2PR_s}{P + R_s}. \tag{3}$$

Empty denominators are undefined. Retention is restricted to human-supported functions among discovered candidates; yield measures the admitted fraction of the candidate set. Neither measures whole-application coverage. Table 5 reports cumulative counts and metrics.

Table 5: C1 cumulative knowledge admission across six runs. P, R_s, Y, and F_1 are percentages (%); other entries are counts. TP/FP/FN are relative to the reference labels.

Admission policy	$\lvert A\rvert$	TP/FP/FN	P	R_s	Y	F_1
Proposal-as-fact	45	33/12/0	73.33	100.00	100.00	84.62
Executor-success	40	33/7/0	82.50	100.00	88.89	90.41
Evidence-grounded	33	32/1/1	96.97	96.97	73.33	96.97

Table 6: C1 cumulative corpus. Availability counts attempts for which both screenshot files exist.

Site	Candidates	Supported	Attempts	Image pair available
SauceDemo	26	17	48	48
Practice Shopping	19	16	55	51
Total	45	33	103	99

The reference labels distinguish a visible functional affordance from demonstrated success. For C1, the canonical key is (application, semantic location, canonical action identifier), pooling attempts

Table 7: C1 admission by application. $|A|$ and TP are counts; P, R_s, and F_1 are percentages (%).

Site	Admission policy	$\|A\|$	TP	P	R_s	F_1
SauceDemo	Proposal-as-fact	26	17	65.38	100.00	79.07
	Executor-success	21	17	80.95	100.00	89.47
	Evidence-grounded	17	16	94.12	94.12	94.12
Practice Shopping	Proposal-as-fact	19	16	84.21	100.00	91.43
	Executor-success	19	16	84.21	100.00	91.43
	Evidence-grounded	16	16	100.00	100.00	100.00

across runs within each application. Each attempt retains its own frozen expectation, which is not part of this aggregation key. The admitted sets are evaluated against human-supported core functionality. The frozen keys are evaluated without post-hoc merging. One executor-success false positive has an empty location field; key-assignment errors and insufficient observations are distinct from an absent function. Human review can accept an equivalent core outcome despite overly narrow presentation details in the frozen expectation; the false-rejection examples below illustrate this distinction. Labels were initialized by AI and then checked item by item by a single human reviewer, with executor status, system outcome, and admission decisions hidden.

D Functional Discovery: Coverage and Recovery Accounting

Table 8: C2 supported-function coverage. Run entries are covered inventory items; the denominator is 22 for SauceDemo and 16 for Practice Shopping. Mean coverage is computed within site.

Site	Condition	Run 1	Run 2	Run 3	Mean (%)
SauceDemo	Random	1/22	8/22	7/22	24.2
	Linear	10/22	10/22	6/22	39.4
	Full	12/22	14/22	13/22	59.1
Practice Shopping	Random	5/16	7/16	4/16	33.3
	Linear	7/16	7/16	8/16	45.8
	Full	7/16	9/16	8/16	50.0

Table 9: C2 execution ledger. SD: SauceDemo; PS: Practice Shopping. N: ordinary attempts; S: executor successes; I: evidence-incomplete attempts; R: recovery episodes; C_R: recovery/reset calls. S and I are not mutually exclusive. Stop codes: E, current candidates exhausted; F, no recoverable frontier; B, ordinary-attempt budget; R, total recovery limit.

Site	Condition	Run	N	S	I	R	C_R	Stop
PS	Random	1	19	7	1	0	0	E
PS	Random	2	11	9	0	0	0	E
PS	Random	3	10	9	0	0	0	E
PS	Linear	1	17	11	2	0	0	E
PS	Linear	2	16	12	1	0	0	E
PS	Linear	3	16	14	2	0	0	E
PS	Full	1	25	17	4	0	0	B
PS	Full	2	18	14	2	1	1	F
PS	Full	3	25	16	4	0	0	B
SD	Random	1	3	1	0	0	0	E
SD	Random	2	13	9	0	0	0	E
SD	Random	3	11	9	0	0	0	E
SD	Linear	1	10	10	0	0	0	E
SD	Linear	2	14	14	0	0	0	E
SD	Linear	3	10	8	0	0	0	E
SD	Full	1	20	16	0	4	28	R
SD	Full	2	21	21	0	3	20	F
SD	Full	3	21	18	0	4	28	R

The ordinary-attempt cap and recovery budget are distinct. SauceDemo Full records 62 ordinary attempts plus 76 recovery/reset calls across three runs, compared with 34 ordinary calls for Linear. On Practice Shopping, Full records 68 ordinary attempts plus one recovery/reset call, compared with 49 ordinary calls and no recovery calls for Linear. Figure 6 displays all conditions using the recorded high-level units. A recovery call counts one entry reset or one saved high-level action; failed recovery calls remain in the cost. Ordinary and recovery calls are reported separately.

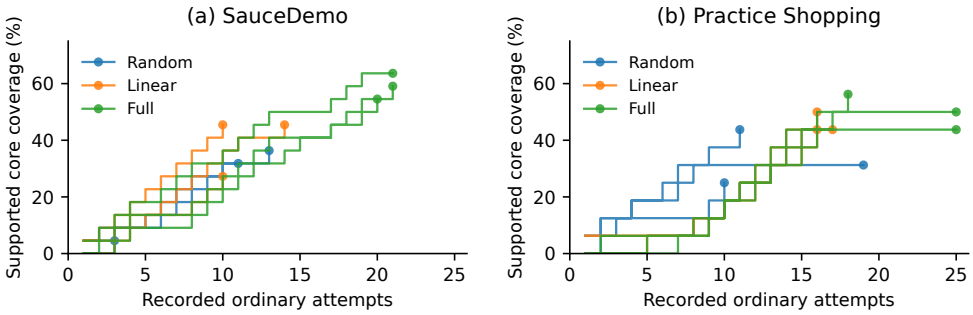


Figure 5: Recorded C2 coverage trajectories. Each line segment is one of the three runs per condition, reconstructed from the frozen per-attempt arrays; the terminal marker identifies its final ordinary attempt. Runs that stop early are not extrapolated. The horizontal axis counts ordinary high-level attempts and excludes additional recovery/reset calls.

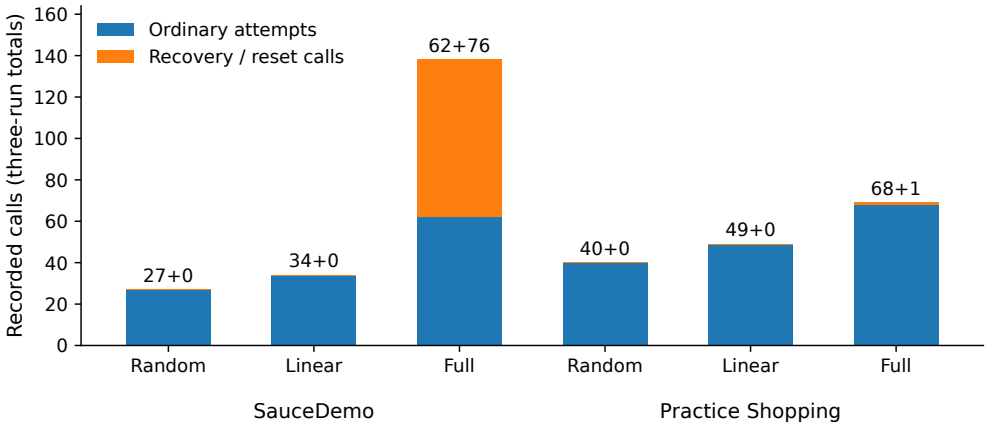


Figure 6: Recorded C2 interaction cost across three runs per condition. Ordinary candidate attempts and additional entry/reset or replay calls are shown separately. Bars use high-level calls, not atomic browser actions; all conditions retain their original early stopping behavior.

Coverage is computed per run against the inventories below. Common global controls count once; object or parameter variants do not automatically become separate inventory items. An attempt covers an inventory item only when before/after observations are available, human review confirms a successful outcome, and both its semantic location and functional effect match the item. Each attempt matches at most one item, and each item counts once per run. Proposal-only, wrong-location, unmatched, and unresolved cases do not increase coverage. Parameter values within a filtering dimension remain instances; the C2 inventory pools filtering dimensions into one filtering item. AI-initialized mappings were reviewed and frozen before final scoring. The inventories are evaluation-side resources and are not supplied to the explorer.

Table 10: SauceDemo core-function inventory (22 items) from the study record.

ID	Function and context
SD01	Fill credentials on the login page
SD02	Submit login
SD03	Sort the product list
SD04	Open product details from the list
SD05	Add an item from the list
SD06	Remove an item from the list
SD07	Use the global cart entry
SD08	Add an item from product details
SD09	Remove an item from product details
SD10	Return from details to the product list
SD11	Open product details from the cart
SD12	Remove an item from the cart
SD13	Continue shopping from the cart
SD14	Start checkout from the cart
SD15	Fill checkout information
SD16	Submit checkout information
SD17	Cancel on the checkout-information page
SD18	Open product details from checkout overview
SD19	Complete an order from checkout overview
SD20	Cancel checkout overview
SD21	Return to products from completion
SD22	Generate an order PDF from completion

Table 11: Practice Shopping core-function inventory (16 items) from the study record.

ID	Function and context
PS01	Search products
PS02	Filter products
PS03	Clear filters
PS04	Sort products
PS05	Navigate product pagination
PS06	Open product details
PS07	Close the product-detail dialog
PS08	Add an item from the product page
PS09	Add an item from the detail dialog
PS10	Use the global cart/checkout entry
PS11	Close checkout and continue shopping
PS12	Fill billing information
PS13	Select a payment method
PS14	Fill payment information
PS15	Submit an order
PS16	Download an invoice from the success dialog

Table 12: Risk labels cover preparation as well as execution of sensitive workflows. They are not severity levels or observed losses.

Category	Meaning and relevant boundary
Destructive change	Preparing or performing hard-to-recover deletion, overwrite, or reset; temporary cart removal is not positive by default.
Financial transaction	Entering or preparing payment/paid-obligation workflows, including checkout and payment details; viewing prices is not positive by default.
External communication	Preparing, sending, publishing, or exposing content to external audiences; reading public content alone is not positive.
Account and permission	Preparing or changing credentials, identity, security, membership, or access controls.
Sensitive data	Preparing to enter, upload, reveal, or transmit identity, financial, or authentication information; ordinary search terms are not positive.

E Risk Taxonomy and Evaluation Boundaries

Risk gold uses pre-execution inputs, not after screenshots or outcome labels. The external set contains sampled source screenshots, not synthetically generated interfaces. Its pairs share normalized action targets across different screenshots. Pair IDs define resampling clusters. On the exploration-linked set, primary-type accuracy compares the predicted primary category to the gold primary category over all positives. On the external set, acceptable-type accuracy is

$$\frac{1}{|G^+|} \sum_{i \in G^+} \mathbf{1}[\hat{p}_i = 1 \ \wedge \ \hat{c}_i \in C_i^{\text{gold}}],$$

Here G^+ is the set of gold-positive cases, $\hat{p}_i$ is the binary risk prediction, $\hat{c}_i$ is the predicted category, and C_i^{gold} is the human-approved set of acceptable categories for case i. Missing risk predictions and missed positives count as incorrect.

F Prompt Documentation

The following condensed templates specify the inputs, instructions, and response fields; repeated wording and examples are omitted for brevity.

Candidate proposal (C1/C2). Inputs: current screenshot, candidate limit, and available exploration context such as the semantic location and existing or completed action identifiers.

> Inspect the frontmost active interaction surface. Assign a stable semantic location and propose distinct business operations supported by visible controls. Exclude hidden or speculative operations, static content, and already completed operations. Related required fields may form one composite action. Select a representative visible target for each operation and distinguish the function from its concrete parameters. Include only indispensable direct dependencies supported by the visible surface. For each candidate, specify a concise GUI-observable outcome before execution; do not assert hidden backend state. Return at most the candidate limit, using fewer or no candidates when appropriate.

Response summary: location identifier and candidate records containing action identifier, description, target, expected outcome, execution policy, and dependencies. Later source versions additionally request an explicit `execution_instance`. The available C1 v1 source snapshot groups filtering broadly, whereas the C1 v4 snapshot and C2 source distinguish business filter dimensions while treating values within a dimension as parameters. For targeted scans, the builder requests newly enabled, changed, or disabled actions justified by the supplied business-fact delta and avoids repeating completed actions.

Outcome verification (C1/C2). Inputs: action description, the expectation fixed before execution, and before/after screenshots.

> Compare the screenshots for this executed action. Decide whether visible evidence supports the fixed expected outcome; do not substitute a different success criterion. Prefer a definite success or failed judgment when the evidence permits it; use uncertain for missing, incomparable, or conflicting images. Judge semantic location change separately: altered values, counts, filtering, or sorting on the same active surface do not imply a new location, whereas a newly active stable page, dialog, or workflow surface may. Return the outcome, location-change flag, and concise visible evidence.

Response fields: `outcome` (success, failed, or uncertain), `location_change` (boolean), and `evidence` (list of short observations).

Executor-goal template (C1/C2). Inputs: the selected business action, visible page evidence, and available site or exploration context.

> Execute the selected business action only. Stop after its first visible completion or clear failure and report visible evidence. Use exploration memory as context for that action, rather than choosing a different business goal from it.

Contextual risk assessment (C3). Inputs: selected action text, with screenshot and taxonomy definitions/examples included according to the input condition.

> Assess whether the proposed action may perform a risk-sensitive operation or enter or prepare a workflow in which it may occur. Ordinary browsing, reading, searching, filtering, and display operations are not risky by themselves. When the supplied evidence reasonably indicates potential risk, prefer marking the action as risky. Select the single category that best explains the primary risk. Explain the relevant consequence or workflow transition in one concise sentence rather than repeating the category name. Return only potential_risk, risk_type, and evidence; use a null type for a non-risk prediction.

Response fields: `potential_risk` (boolean), `risk_type` (category or null), and `evidence` (string). All four exploration-linked conditions receive action text and the allowed category keys. Action-only omits screenshots and category definitions/examples; action+taxonomy adds definitions/examples; action+visual adds the screenshot; Full supplies both. The external study compares text+taxonomy with context-conditioned assessment.

G Contextual Risk: Sampling, Uncertainty, and Errors

Table 13: External C3 results (%). Acceptable-type accuracy counts a prediction as correct when the predicted risk category matches any human-approved category on all 65 gold-positive cases. The difference is in percentage points, calculated before rounding; intervals use paired context clusters.

Metric	Text + taxonomy	Context-conditioned	Difference [95% CI]
Precision	77.6	73.0	-4.5 $[-15.1,\ 4.7]$
Recall	58.5	70.8	12.3 $[-2.0,\ 27.4]$
F1	66.7	71.9	5.2 $[-5.6,\ 17.1]$
Acceptable-type acc.	27.7	63.1	35.4 $[21.7,\ 49.3]$

Source linkage and runtime records. The archived source audit, conducted before the formal inference matrix, reports that all 130 ordinary attempts from the six Full runs have unique attempt identifiers, decodable pre-action screenshots, and runtime risk records. Each of the 106 deduplicated samples links to its source application, run, attempt, action, semantic location, screenshot, and runtime risk record. The classification tables evaluate offline predictions on these frozen pre-action inputs; runtime judgments serve as linked interaction records.

Inference configuration. Both formal tracks request `gpt-4o` at temperature 0 with prompt version `c3-ablation-prompt-v1`, a 300-token response cap, and a structured JSON response. Visual conditions request high-detail screenshots. Full uses the runtime risk system prompt; reduced-input conditions use fixed corresponding instructions. Category identifiers are available in every condition; full taxonomy definitions and examples are included only in taxonomy conditions. The ablation therefore varies input availability with instructions matched to each condition. Formal results exclude pilot predictions; the external sampling additionally excludes the ten pilot screenshot IDs.

For exploration-linked inference, 424/424 outputs are valid: 398 responses return `gpt-4o-2024-11-20` and 26 return `gpt-4o`. For external inference, 182/200 calls return that dated identifier and 18 return `gpt-4o`, evenly distributed across conditions.

External sampling and labels. The deterministic seed is 20260920. Sampling starts from 6,491 WebGuard metadata rows and retains 3,281 eligible rows after final-review, pilot, screenshot-ID, and action-text filtering. The final set contains 100 unique screenshot IDs with no duplicate image hashes: 60 samples in 30 same-action pairs and 40 diversity samples. The latter use source strata SAFE 14, LOW 13, and HIGH 13 for sampling only. Ten pilot screenshots are excluded. Failed downloads were retried for the same screenshot IDs, preserving the sampled set, and all retained screenshots passed validation. The 100 cases span 80 websites and 66 normalized action targets. The selected action is reconstructed as "Click the target labeled 'target text'." Source labels, reviewer reasons, and page URLs are excluded from the blinded inputs.

The external AI-initial labels were all confirmed by human review without correction. The resulting reference is human-reviewed but remains susceptible to anchoring on the AI initialization. There are 65 positives and 35 negatives. Primary explanatory categories are sensitive data (24), account/permission (21), external communication (18), financial transaction (1), and destructive change (1); acceptable gold sets may contain more than one category. The last two categories each contain one example and are reported as composition information rather than reliable per-category estimates.

Context-pair metrics. For each sample i, let $p_i \in \{0, 1\}$ denote its reference risk label and $\hat{p}_i \in \{0, 1\}$ its predicted risk label. Let D contain the eight same-action pairs whose two reference labels differ. Pair joint accuracy is

$$J = \frac{1}{|D|} \sum_{(i,j) \in D} \mathbf{1}[\hat{p}_i = p_i \ \wedge \ \hat{p}_j = p_j].$$

The correct risk-flip rate additionally requires $\hat{p}_i \neq \hat{p}_j$. Since $p_i \neq p_j$ for every pair in D, the two rates coincide on this subset. The other 22 same-action pairs remain in sample-level metrics and in the paired-cluster resampling.

Uncertainty. Both analyses use 10,000 bootstrap resamples with seed 20260920. Exploration-linked comparisons resample runs within sites in a paired manner. External comparisons resample the 30 candidate pairs as clusters and 40 diversity samples as singleton clusters; the same sampled clusters are used for both conditions. This resampling preserves within-pair dependence, with uncertainty defined at the pair and diversity-sample cluster level. Table 14 reports paired F1 differences for the exploration-linked ablations, and Table 15 records the external uncertainty estimates. Intervals reflect the corresponding resampling units and within-dataset variation.

Table 14: Exploration-linked site-macro F1 differences and paired 95% bootstrap intervals, in percentage points.

Comparison	Difference	95% interval
Full − Action + taxonomy	1.6	$[-3.8, 7.1]$
Full − Action + visual	1.0	$[-2.4, 6.2]$
Full − Action only	-0.2	$[-3.1, 3.5]$

Full site-macro F1 has interval [89.9%,96.5%], precision [81.7%,93.5%], and observed recall remains 100% in each run-level resample. With no positive misses in the recorded runs, the reported recall interval reflects resampling uncertainty over the observed positive cases. Full site precision/recall/F1/primary-type accuracy are 92.3/100.0/96.0/75.0% on Practice Shopping and 83.3/100.0/90.9/95.0% on SauceDemo. Full pooled primary-type accuracy is 87.5%, distinct from its 85.0% site-macro value.

Table 15: External C3 cluster-bootstrap intervals. Per-condition intervals are percentages; difference intervals are percentage points. Pair metrics use the eight binary-discordant pairs.

Metric	Text + taxonomy	Context-conditioned	Difference
Precision	[65.9,88.9]	[61.0,84.4]	[−15.1,4.7]
Recall	[43.9,72.6]	[58.7,82.4]	[−2.0,27.4]
F1	[54.5,76.7]	[61.8,80.6]	[−5.6,17.1]
Acceptable-type acc.	[15.9,40.6]	[50.0,75.5]	[21.7,49.3]
Pair joint / flip	[0.0,0.0]	[0.0,60.0]	[0.0,60.0]

Confusion counts and paired errors. Text + taxonomy has TP=38, FP=11, FN=27, TN=24; Context-conditioned has TP=46, FP=17, FN=19, TN=18. Acceptable-type counts are 18/65 and 41/65. Context preserves 48 correct binary decisions, corrects 16, introduces 14 errors, and leaves 22 wrong. On type correctness among positives, it preserves 16 correct cases, corrects 25, introduces two errors, and leaves 22 wrong. Pair joint accuracy and correct risk-flip rate both change from 0/8 to 2/8. The wide pair interval reflects the small realized denominator.

H Admission Errors and Symbolic Compatibility

H.1 Admission failure cases

The SauceDemo key (checkout information form, `complete_information_fields`) was admitted although the archived review did not directly support completion of all required fields. Conversely, (product catalog, `sort_products`) was withheld despite human-confirmed visual reordering in at least one attempt. In Practice Shopping, an expectation that was too specific about the cart interface rejected an attempt that opened checkout directly. Other attempts allowed the cumulative function to be admitted, while an individual run graph could still lose a useful transition. These cases distinguish insufficient evidence, missed effects, and expectation mismatch. They also illustrate why cumulative admission and per-run planning availability are different outcomes.

H.2 Projection interface

A minimal symbolic projection maps locations to predicates, selected functional relations to actions, and observed successful outcomes to result facts. It applies graph-level filtering over successful edges, semantic observations, locations, roles, and dependency information, whereas Equation 1 governs attempt-level admission from observed screenshot evidence.

Risk annotations are retained as metadata in the current projection. Because observed ordering and omitted business conditions may vary across execution contexts, we distinguish syntactic compatibility, symbolic reachability, evidence consistency, and real execution success. The projection serves as a downstream consumer of the learned functional knowledge.

H.3 Observed compatibility

All three Practice Shopping C1 runs export a symbolic model. Two produce plans toward `place_order_succeeded`, whereas one reaches only `view_cart_succeeded` because a needed transition was withheld. These exports demonstrate interface compatibility and symbolic reachability. No configured safety rule matched the projected actions. The exported plans therefore demonstrate symbolic interface compatibility between the learned functional model and the downstream planning pipeline.