

Software Engineering for and with GUI Agent

SHENGCHENG YU, State Key Laboratory for Novel Software Technology, Nanjing University, China and Technical University of Munich, Germany

YUCHEN LING, State Key Laboratory for Novel Software Technology, Nanjing University, China

JUNYANG XING, State Key Laboratory for Novel Software Technology, Nanjing University, China

QUAN ZHOU, State Key Laboratory for Novel Software Technology, Nanjing University, China

CHUNRONG FANG*, State Key Laboratory for Novel Software Technology, Nanjing University, China

ZHENYU CHEN, State Key Laboratory for Novel Software Technology, Nanjing University, China

Graphical user interface (GUI) agents have advanced rapidly in recent years, producing a growing body of frameworks, benchmarks, and applications. However, this growth has outpaced the maturity of the field. GUI agents remain technically brittle, incompletely engineered, and insufficiently validated for sustained real-world use. They are also evolving into closed-loop software systems. Within these systems, model reasoning is coupled with interface perception, execution feedback, recovery, and human oversight. This evolution calls for a software engineering perspective that remains largely absent from existing research. We address this gap by reviewing 336 GUI-agent papers from January 2018 to April 2026. Five research questions examine the research landscape, architectures, evaluation, software lifecycle concerns, and future opportunities. Our findings show that the field has expanded sharply since 2024, while mobile and web settings remain dominant. Architectures increasingly adopt modular perceive–reason–act loops, but recovery, human escalation, safety enforcement, and auditability remain underdeveloped. This architectural imbalance extends to evaluation. Evaluations are becoming more interactive, but they remain centered on task success and are difficult to compare across protocols. More broadly, existing studies provide limited support for testing beyond benchmarks and for maintaining agents after release. Observability, privacy engineering, and systematic human oversight are also underdeveloped. Together, these findings show that capability improvements alone cannot ensure deployment readiness. Future research should connect dependable execution with lifecycle-centered testing and reproducible evaluation. It should also integrate permission and privacy controls with cost-aware, human-centered governance. This integration is necessary to build dependable, maintainable, secure, and deployable GUI-agent systems.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; • **Computing methodologies** → **Intelligent agents**; • **Information systems** → **Language models**.

Additional Key Words and Phrases: LLM, GUI Agent

1 Introduction

Graphical user interface (GUI) agents have become a major research topic as foundation models have improved their ability to interpret instructions, perceive interfaces, and generate actions. They now operate across web navigation, mobile-device control, desktop automation, and general computer-use settings. GUI-specific model training and fine-tuning have been central to this progress. CogAgent combines GUI-oriented pre-training with multi-task fine-tuning, SeeClick strengthens visual agents through GUI-grounding pre-training, and GUICourse uses staged GUI

*Chunrong Fang is the corresponding author.

Authors' Contact Information: [Shengcheng Yu](#), State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China and Technical University of Munich, Heilbronn, Germany, shengcheng.yu@tum.de; [Yuchen Ling](#), State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, yuchenling@smail.nju.edu.cn; [Junyang Xing](#), State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, xingjunyang@smail.nju.edu.cn; [Quan Zhou](#), qzhou@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China; [Chunrong Fang](#), State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, fangchunrong@nju.edu.cn; [Zhenyu Chen](#), State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, zychen@nju.edu.cn.

data to adapt general vision-language models for interface understanding, grounding, and action generation [33, 38, 79]. At the system level, frameworks such as Agent S, UFO, and AppAgent show how foundation models can be orchestrated for multi-step interaction across operating systems and applications [2, 117, 301]. Benchmarks such as WebArena, AndroidWorld, and OSWorld then provide shared test environments for web, mobile, and desktop agents [176, 260, 331]. Together, these progress has made GUI interaction a shared challenge across several research communities.

This growth is accompanied by a change in what constitutes a GUI agent. Early work often focused on individual capabilities such as interface understanding, grounding, or action prediction. Recent systems increasingly combine these capabilities within a closed interaction loop. One line of work strengthens perception and grounding through GUI-specialized models or parsers [38, 79, 111, 122, 145, 149, 171, 255]. Another embeds those capabilities in runtimes that maintain task state, execute actions, and respond to failure [127, 198, 216, 232, 235, 252, 300]. GUI agents are therefore becoming integrated software systems. Their behavior depends on the interaction between the model, its orchestration layer, the target application, and the user. The resulting system boundary now extends well beyond the capabilities of the foundation model itself.

The maturity of the field has not kept pace with this increasing system complexity. Technical performance remains brittle under long-horizon tasks, interface changes, and unexpected environment states. Engineering support is less consistent. Recovery and monitoring are often partial, while maintenance procedures, permission controls, and human takeover policies remain uncommon. Evaluation has become more interactive, but it remains centered on task success and is often difficult to compare across protocols. Evidence of sustained value in real-world applications is also limited. An agent may complete a benchmark task while relying on excessive retries, exposing sensitive content, or failing after a minor interface update. Recent benchmarks and safety studies have begun to expose these limitations across different tasks and platforms [15, 25, 57, 96, 99, 107, 120, 195, 211, 219, 270, 272, 316, 325]. These gaps become more consequential as agents receive broader access and greater operational autonomy.

Existing surveys provide valuable maps of this fast-moving area. Some surveys organize the field around agent capabilities and benchmarks. Others focus on particular platforms, training strategies, or trustworthiness concerns [82, 110, 125, 157, 182, 201, 220, 299]. These surveys clarify what GUI agents are and how their core capabilities are developed. However, they provide limited synthesis of how GUI agents should be engineered and evaluated across the software lifecycle. This perspective is increasingly important because deployment requires more than an accurate model. The system also needs explicit interfaces and requirements, testable recovery behavior, operational logs, bounded permissions, and a policy for human handoff. A software engineering synthesis can connect these concerns and reveal where capability gains do not yet translate into dependable deployment. Without this view, recurring system risks remain separated across otherwise closely related research threads in the current literature.

This paper studies GUI agents from that software-engineering perspective. We analyze 336 papers published or posted from January 2018 to April 2026. The corpus covers research artifacts ranging from models and frameworks to benchmarks, evaluations, tools, and prior surveys. The goal is to determine how the field has evolved and whether current progress supports dependable, maintainable, secure, and deployable systems. Five research questions structure this analysis. RQ1 maps the research landscape, RQ2 examines system architectures, and RQ3 evaluates the evidence behind capability claims. RQ4 then assesses software-engineering coverage across the lifecycle, while RQ5 synthesizes the resulting gaps into a future research agenda. This progression connects descriptive evidence about the field with prescriptive guidance for engineering practice.

The main contributions of this survey are as follows:

- We map the development of GUI-agent research across publication trends, contribution types, platforms, applications, interface representations, and enabling models.
- We synthesize GUI-agent architectures as closed-loop software systems and identify weaknesses in recovery, escalation, safety enforcement, and auditability.
- We analyze how evaluation protocols support capability claims and why their results remain difficult to compare.
- We assess the coverage of software-engineering concerns across the lifecycle, including testing, maintainability, observability, security, privacy, efficiency, and human oversight.
- We derive a research agenda for connecting capability improvements with dependable execution, lifecycle management, reproducible evaluation, and risk-aware deployment.

The remainder of the paper follows this logic. Section 2 defines GUI agents, summarizes their technical evolution, and introduces the software-engineering concepts used in the survey. Section 3 describes the corpus, research questions, collection process, and coding procedure. Sections 4–8 answer the five research questions. Section 9 draws cross-cutting insights from the findings, and the final sections position this survey relative to prior reviews, discuss threats to validity, and conclude with implications for future GUI-agent research.

2 Background

This section establishes the concepts needed to analyze GUI agents. It first defines their closed-loop interaction and technical evolution, then introduces the engineering concerns created by integrating perception, reasoning, execution, and human oversight within a runtime.

2.1 GUI Agents: Definitions, Evolution, and Scope

A graphical user interface (GUI) agent is an autonomous or semi-autonomous system that completes tasks through the interface presented to human users. It interprets an instruction, observes the GUI, and executes actions such as clicking, typing, scrolling, dragging, or using system shortcuts. Tool-oriented agents usually invoke documented APIs, whereas GUI agents act through visual layouts and interaction conventions that may change without a stable interface contract. The agent must translate a semantic goal into spatially and temporally grounded operations under partial observability. GUI understanding, task planning, action grounding, and sequential decision making consequently form one interaction problem [157, 201, 220, 299].

Each action changes the interface and creates a closed feedback loop. At step t , the agent combines the instruction with the current observation, updates its task state, executes an action, and observes the transition. The loop ends when the goal is achieved, a stopping condition is reached, or control returns to the user. Perception and grounding construct actionable interface state [38, 67, 111]. Planning and memory maintain long-horizon progress [41, 56, 250, 315], while verification and recovery respond to unexpected outcomes [21, 104, 138]. AppAgent and Mobile-Agent coordinate these capabilities through screenshots, view hierarchies, and interaction histories on mobile devices [117, 216, 302]. Agent S and UFO extend the loop to desktop and multi-application workflows [2, 300, 301]. We summarize the learned and deterministic components, verification paths, recovery, and human intervention in Fig. 1.

The closed-loop abstraction predates foundation models. Earlier systems learned policies for constrained websites, grounded web-support instructions, mapped language to mobile actions, or used DOM and Android view hierarchies [116, 123, 177, 268]. They established instruction-conditioned policies, interface state representations, finite action vocabularies, and environments that returned state transitions. Their scope remained tied to specific platforms, tasks, or interface schemas and

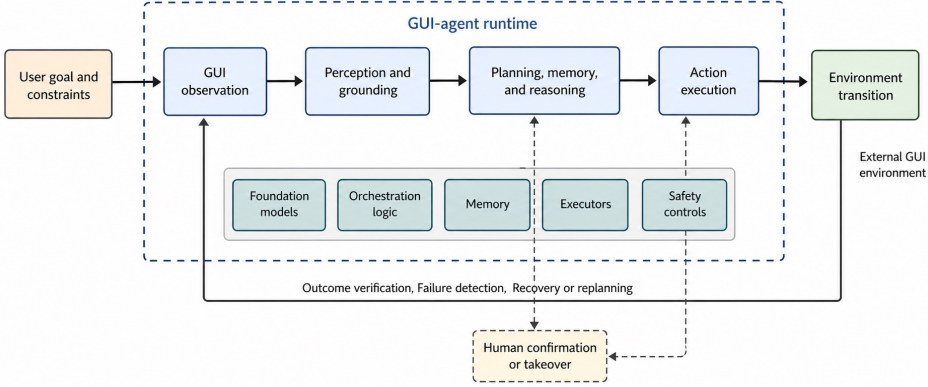


Fig. 1. GUI-agent interaction loop.

often required application-specific adaptation. Moving to a new application commonly required additional demonstrations, handcrafted selectors, or a new representation of target interface.

Large language models and multimodal foundation models expanded this paradigm. Language models improved instruction interpretation and task decomposition, while vision-language models made screenshots a practical observation channel [7, 122]. SeeClick and CogAgent connect instructions to visual elements and coordinates [38, 79], and OmniParser converts screenshots into candidate interface elements [149]. Native agents such as UI-TARS, OS-ATLAS, ShowUI, and GUI-R1 integrate perception, reasoning, and action prediction within one model [122, 145, 171, 255]. These advances support general computer use across heterogeneous applications. They also shift part of the interaction logic from platform-specific rules into learned visual and language representations.

Long-horizon benchmarks place agents in interactive websites, mobile applications, desktop software, and knowledge-work tasks [12, 54, 99, 176, 260, 331]. They expose dependencies among perception, reasoning, execution, and recovery that a grounded click cannot capture. Successful behavior in these settings requires persistent task state, appropriate termination, recovery from invalid actions, and control over irreversible effects. GUI agents must therefore be analyzed as stateful systems whose behavior depends on learned components and surrounding infrastructure.

2.2 Software Engineering of Agentic GUI Systems

Viewing GUI agents as stateful systems shifts attention to the complete runtime. Observation adapters and memory construct state, executors and environment controllers apply decisions, and safety checks, logs, and confirmation mechanisms constrain behavior. AutoDroid, AssistGUI, Agent S, OS-Copilot, and MM-Pro combine foundation models with screenshots, structured meta-data, tool wrappers, or deterministic controllers [2, 64, 235, 241, 252]. Failures can emerge from interactions among these components when interfaces move, pages load slowly, modals intercept input, or executors select the wrong window. The model may produce a valid plan while the complete system still fails because of incorrect tool execution or environment feedback.

Requirements define intended capability and acceptable behavior. Reliability and robustness concern repeated execution under interface variation, while operational quality covers latency, interaction cost, privacy, and resistance to malicious content. Robustness, safety, privacy, and human-centered studies begin to isolate these attributes [25, 107, 280, 325]. Their interactions require explicit operating boundaries. Confirmation can reduce harmful actions while increasing user effort, and context compression can lower cost while weakening state tracking. Privacy

masking may also remove information needed for task completion. Requirements must specify acceptable trade-offs among autonomy, reliability, cost, privacy, and human oversight.

Architecture assigns these responsibilities. Framework-based agents expose perception, planning, execution, memory, and reflection as inspectable modules [2, 215, 301, 323]. Native models learn tighter mappings from observations and instructions to reasoning and actions [122, 171, 255], while hybrid systems combine foundation models, specialized grounders, and deterministic controllers [38, 104, 149]. These choices shape component replacement, failure isolation, observability, platform adaptation, and testing. Modular systems expose intermediate state and policy boundaries more clearly, while native models reduce hand-built coordination and concentrate more responsibility inside learned behavior.

Testing must reflect the same boundaries. Offline datasets isolate perception, grounding, and next-action prediction [38, 111, 177], while interactive benchmarks evaluate trajectories under environment feedback [176, 260, 331]. Offline tests omit timing, recovery, and side effects, and aggregate task success can hide retries, unsafe intermediate actions, or environment dependence. Repeated runs, controlled perturbations, and stable regression oracles provide complementary evidence. Reliability testing examines variation across repeated executions, while robustness testing changes interface content, timing, or environmental conditions. Regression testing then checks whether model, prompt, wrapper, or application updates invalidate earlier behavior. GUI-Robust, WABER, and environmental-injection benchmarks evaluate anomalies, web unreliability, and active perturbations [23, 37, 97, 280].

Operation and evolution determine how long testing evidence remains valid. Applications, models, prompts, and external tools change on separate schedules, so an agent can degrade without changes to its own code [43, 141, 328]. Logs, state snapshots, trajectory replay, and failure classification distinguish model errors from environment or executor failures [21, 138, 148]. Maintainability also requires explicit contracts for observations, actions, stopping criteria, fallback rules, and permissions. Versioned interfaces and replayable traces help preserve system intent as individual components and target applications evolve.

Operation also creates a security and privacy boundary because agents may observe credentials or private documents and modify external state. Environmental injection can redirect behavior or induce privacy leakage [57, 120, 140, 316]. GUIGuard, VeriOS, and VeriSafe address visual data protection, policy compliance, human involvement, and action verification [104, 229, 253]. Safety is therefore a runtime responsibility that must be specified, tested, monitored, and audited across the lifecycle. Permission boundaries, confirmation policies, and logs connect that responsibility to concrete actions and later accountability.

3 Survey Methodology

Our methodology links the research questions to corpus construction, structured coding, and evidence synthesis. The complete corpus establishes the field-level landscape, while purpose-built analytical subsets provide the evidence required for the architectural, evaluation, and software-engineering analyses. This layered design combines broad coverage with focused coding at the level appropriate to each research question.

3.1 Research Questions

- RQ1** How has GUI agent research evolved, and what are the major trends in research contributions, platforms, application domains, and enabling models?
- RQ2** How are GUI agents architected to perceive interfaces, reason about tasks, execute actions, and recover from failures?

- RQ3** How are GUI agents evaluated, and what protocol and system factors shape the validity, reproducibility, and comparability of evaluation evidence?
- RQ4** To what extent does existing GUI agent research address software engineering concerns across the system lifecycle?
- RQ5** What open challenges and research opportunities must be addressed to engineer dependable, maintainable, secure, and deployable GUI agents?

The research questions move from descriptive mapping to engineering interpretation. RQ1 characterizes the field through publication trends, contribution types, platforms, applications, observation media, and model usage. RQ2 moves inside the resulting systems to examine how they allocate perception, state representation, planning, memory, execution, verification, recovery, and human intervention. RQ3 then assesses how benchmarks, protocols, and metrics support valid, reproducible, and comparable capability claims. RQ4 broadens the unit of analysis from individual capabilities to lifecycle concerns such as requirements, testing, maintainability, observability, security, privacy, efficiency, and oversight. Finally, RQ5 integrates the preceding results into a research agenda. This sequence connects the scale and structure of the field to the evidence and engineering practices required for dependable deployment in practice.

3.2 Literature Collection and Selection

We defined the corpus around a functional criterion. A paper was in scope when it treated a graphical interface as an observation or action environment for an autonomous or semi-autonomous agent. The scope covers web pages, mobile applications, desktop operating systems, productivity software, and cross-platform computer-use environments. Supporting work was also eligible when it directly contributed a dataset, benchmark, grounding model, UI parser, safety analysis, evaluation protocol, or engineering technique for GUI agents. Together, these criteria center the corpus on studies that combine GUI interaction with autonomous or semi-autonomous operation.

Conventional GUI testing was incorporated through an agent-centered selection boundary. That mature literature primarily studies test generation, event-sequence exploration, regression testing, crash detection, coverage, model-based testing, and record-and-replay validation. Papers from this literature entered the corpus when they explicitly studied a GUI agent or directly supported agent perception, decision making, execution, benchmark design, safety, or lifecycle engineering. This criterion concentrates the software-engineering evidence on techniques that inform GUI-agent construction and evaluation.

Corpus construction combined keyword search, backward and forward snowballing, and manual consolidation of existing paper lists. The search vocabulary covered GUI agents, computer-use agents, web agents, mobile agents, Android agents, desktop agents, UI automation, interface grounding, multimodal agents, and GUI benchmarks. We considered both peer-reviewed papers and preprints to represent rapidly emerging work alongside venue-reviewed evidence. Publication source was recorded separately so that RQ1 could distinguish corpus growth from evidence maturity. We manually consolidated duplicate records, title variants, and multiple versions of the same work. When several versions were available, we selected the most complete and current bibliographic record and harmonized its metadata with the rest of the corpus.

The resulting corpus contains 336 unique papers published or posted between 2018 and April 2026. Every retained paper contributes to at least one core category, namely framework, model, benchmark, evaluation study, assistive tool, survey, or software-engineering analysis. This rule captures the multidisciplinary character of GUI-agent research while preserving a clear boundary around systems that perceive, reason over, act on, evaluate, or engineer graphical interfaces. The corpus therefore supports both quantitative mapping and qualitative interpretation of the field.

3.3 Data Extraction, Coding, and Synthesis

The paper is the primary unit of analysis. For each paper, we first recorded its bibliographic metadata and contribution type. Technical coding covered the target platform, application domain, base model, observation medium, and agent modules. Separate fields captured the reported techniques, evaluation design, metrics, and human-in-the-loop signals. Contribution type is multi-label because one paper may introduce several artifacts, such as a framework and a benchmark. Platform and observation medium are also multi-label because a system may span several platforms or combine screenshots with HTML, XML, accessibility trees, or generated descriptions.

We developed the coding scheme iteratively from coarse contribution categories to finer engineering dimensions. The first layer distinguishes frameworks, benchmarks, models, evaluations, assistive tools, and surveys. The second layer captures functions that may be implicit in a paper. It follows the agent loop from perception and decision to execution, then records support for planning, memory, verification, reflection, and recovery. Additional fields cover operational and governance concerns such as cost, observability, maintainability, safety, privacy, and human oversight. When papers used different names for similar mechanisms, we coded the function performed in the agent loop. For example, a verifier, critic, evaluator, reward model, or post-action checker may provide verification or reflection depending on how it affects execution. This functional interpretation enables comparison across systems with different architectural vocabularies.

Each research question draws on an analytical population aligned with its evidence needs. RQ1 uses the complete 336-paper corpus. RQ2 draws module statistics from the 145 papers coded as frameworks because these papers expose explicit system structures. RQ3 uses the 252 unique papers coded as frameworks, models, or evaluation studies, where capability evidence is most directly reported. RQ4 uses the 327 records with complete software-engineering coding. Each analytical subset has a distinct denominator, which accompanies all reported counts to support transparent interpretation.

We used quantitative synthesis for fields that supported consistent aggregation. The field-level summaries describe publication patterns, research artifacts, target settings, and model use. Architecture and evaluation are aggregated separately, as are the software-engineering signals used in RQ4. Multi-label categories are reported as overlapping counts rather than mutually exclusive partitions. We then used qualitative synthesis to explain the distributions. This analysis examines how observation media shape architecture, how benchmark protocols condition performance claims, how systems evaluate recovery and verification, and how engineering concerns appear across the complete system lifecycle.

We checked related fields to strengthen internal coherence across the coding scheme. A framework record was expected to expose perception, decision, and execution responsibilities. A benchmark record required a task, environment, dataset, or evaluation-protocol contribution. Safety and privacy signals required a concrete risk, attack, defense, policy, or sensitive-data concern. For records requiring interpretation, we considered the contribution claim, method, and evaluation setup together with the terminology used by the authors. These checks make interpretive judgments explicit and tie later claims to observable properties of each paper.

The remainder of the survey follows the same analytical sequence. RQ1 establishes scale and distribution, RQ2 interprets architectural patterns, RQ3 examines evaluation evidence, and RQ4 assesses lifecycle coverage. RQ5 then integrates the results into a research agenda. Numerical summaries establish the prevalence of observed patterns, while representative papers explain their technical and engineering significance.

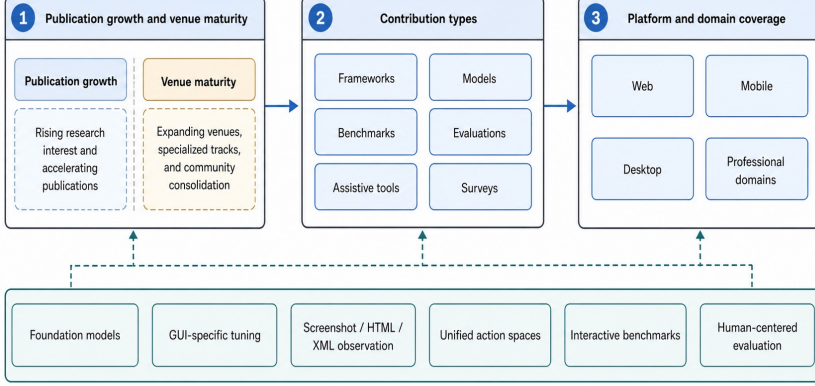


Fig. 2. GUI-agent research landscape.

4 RQ1: Research Landscape and Development Trends

We use RQ1 to establish the empirical landscape for the later analyses. The corpus contains 336 papers from January 2018 to April 2026. Only 19 appeared before 2024, while 317 were published or posted during 2024–2026. This increase coincides with stronger multimodal models, interactive benchmarks across major platforms, and a shift from isolated GUI understanding to closed-loop GUI interaction [38, 79, 176, 260, 331]. We examine this development through publication patterns, contribution types, platform and domain coverage, observation media, and model usage.

We use Fig. 2 to connect publication growth and source maturity with the technical dimensions that shape architectural and evaluation claims. The recent expansion is distributed across systems, benchmarks, evaluation studies, assistive tools, and surveys [2, 25, 226, 299, 324]. Many claims about GUI agents are produced inside tightly coupled research artifacts. A new framework may introduce a benchmark, a benchmark may embed a particular observation interface, and a model paper may also define data generation and action-space assumptions. We therefore interpret the literature as an evolving research ecosystem shaped by publication maturity, contribution type, platform boundary, task domain, observation medium, and model family.

4.1 Rapid Growth and Uneven Publication Maturity

We observe in Fig. 3 that GUI-agent research moved from scattered precursors to a high-volume research topic within a short period. Early work investigated web-interface reinforcement learning, web-support instruction grounding, and mobile GUI action prediction through narrower technical formulations [116, 123, 177, 268]. WebShop and Mind2Web then helped frame GUI interaction as grounded language-to-action decision making in realistic web tasks [48, 289]. The post-2023 expansion is substantial. We report 286 papers in 2024 and 2025 alone in the left panel of Fig. 3, and the partial 2026 count already exceeds all pre-2024 work combined. This acceleration coincides with WebArena, VisualWebArena, WebLINX, AndroidWorld, OSWorld, and later macOSWorld making interactive environments and cross-application tasks more visible as shared evaluation targets [99, 147, 176, 260, 283, 331]. In parallel, GUI-specialized multimodal agent systems such as CogAgent, SeeClick, OmniParser, OS-ATLAS, UI-TARS, and ShowUI demonstrated that visual grounding, screenshot parsing, and action prediction could be trained or adapted at a scale that earlier GUI automation work did not support [38, 79, 122, 149, 171, 255]. Contribution types are coded as multiple labels, so their counts do not sum to the corpus size.

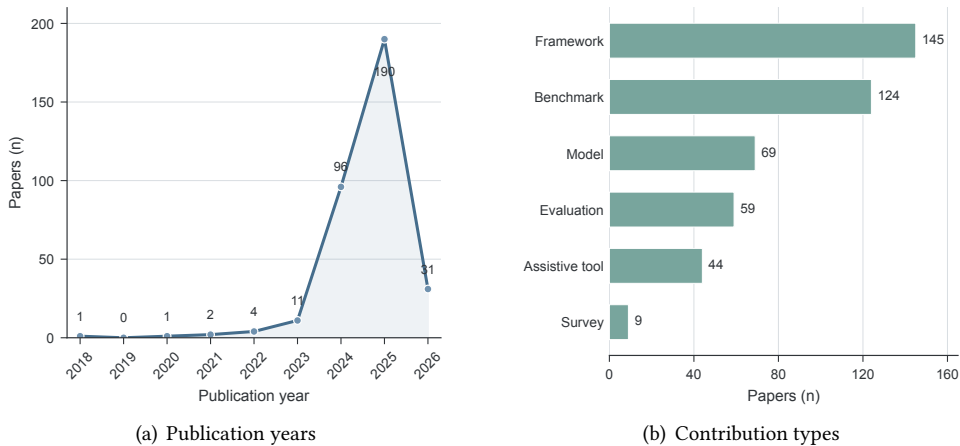


Fig. 3. Publication years and contribution types.

Table 1. Publication-source distribution.

Source	Full name	Nature	Papers	Share
arXiv	arXiv preprint repository	Informal	204	60.7%
ACL	Annual Meeting of the Association for Computational Linguistics	Conference	20	6.0%
Findings of ACL	Findings of the Association for Computational Linguistics: ACL	Conference	16	4.8%
AAAI	AAAI Conference on Artificial Intelligence	Conference	12	3.6%
NeurIPS	Conference on Neural Information Processing Systems	Conference	12	3.6%
EMNLP	Conference on Empirical Methods in Natural Language Processing	Conference	9	2.7%
Findings of EMNLP	Findings of the Association for Computational Linguistics: EMNLP	Conference	8	2.4%
CVPR	IEEE/CVF Conference on Computer Vision and Pattern Recognition	Conference	7	2.1%
ACM MM	ACM International Conference on Multimedia	Conference	7	2.1%
LNCS	Lecture Notes in Computer Science	Conference	5	1.5%
MobiCom	ACM International Conference on Mobile Computing and Networking	Conference	4	1.2%
KDD	ACM SIGKDD Conference on Knowledge Discovery and Data Mining	Conference	4	1.2%
ICLR	International Conference on Learning Representations	Conference	3	0.9%
CHI	ACM CHI Conference on Human Factors in Computing Systems	Conference	3	0.9%
CIKM	ACM International Conference on Information and Knowledge Management	Conference	2	0.6%
UIST	ACM Symposium on User Interface Software and Technology	Conference	2	0.6%
ACL Demo	ACL System Demonstrations	Conference	2	0.6%
EMFM Workshop	International Workshop on Edge and Mobile Foundation Models	Conference	1	0.3%
WWW	The Web Conference	Conference	1	0.3%
CVPR Workshop	IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops	Conference	1	0.3%
IC-NIDC	International Conference on Network Infrastructure and Digital Content	Conference	1	0.3%
MobiSys	ACM International Conference on Mobile Systems, Applications, and Services	Conference	1	0.3%
EMNLP Demo	EMNLP System Demonstrations	Conference	1	0.3%
Interspeech	Conference of the International Speech Communication Association	Conference	1	0.3%
COLM	Conference on Language Modeling	Conference	1	0.3%
VLDB	International Conference on Very Large Data Bases	Conference	1	0.3%
WSDM	ACM International Conference on Web Search and Data Mining	Conference	1	0.3%
ISSTA	ACM SIGSOFT International Symposium on Software Testing and Analysis	Conference	1	0.3%
SAC	ACM Symposium on Applied Computing	Conference	1	0.3%
ICLR Workshop	International Conference on Learning Representations Workshop	Conference	1	0.3%
IJCAI	International Joint Conference on Artificial Intelligence	Conference	1	0.3%
IEEE IoTJ	IEEE Internet of Things Journal	Journal	1	0.3%
Pattern Recognition	Pattern Recognition	Journal	1	0.3%

We report the complete publication-source distribution of the 336-paper corpus in Table 1 to compare literature growth with formal publication cycles. Preprints account for 204 papers, or 60.7% of the corpus, so a large share of current evidence is still moving through review, revision, and venue consolidation. The formally published papers are dispersed across several research communities. NLP and vision contribute instruction following and visual grounding, while HCI

studies the user-facing interaction. Systems research supplies execution environments, and software engineering contributes methods for testing and lifecycle quality [38, 48, 235, 280, 314]. It also means that evidence maturity varies substantially. A benchmark result reported in a preprint, an ACL paper on web instruction grounding, and a CVPR paper on screenshot understanding may all influence the field, yet they often carry different assumptions about task sampling, evaluation oracles, environment stability, and reproducibility. Later sections interpret performance claims together with their evaluation settings and avoid direct score comparison when the underlying protocols differ in material ways.

Our contribution-type analysis shows a field that is still building its objects of study, as reported in the right panel of Fig. 3. Framework papers form the largest category, with 145 papers, followed by benchmarks with 124 papers. Models and evaluation papers are smaller but substantial groups. We observe an iterative loop between construction and measurement. New agents expose missing task types, benchmarks make their weaknesses visible, and changes to perception or grounding create new evaluation needs. Representative examples include framework-oriented systems for mobile and desktop automation [2, 301, 302], benchmarks for web, mobile, and desktop interaction [176, 260, 331], and GUI-specific models for grounding or action prediction [38, 79, 171]. Surveys have also appeared quickly [157, 201, 220, 299], which is typical of an area whose terminology and scope are expanding faster than its methodological conventions can stabilize across communities over time.

The contribution-type counts are multi-label, which is important for interpreting the right panel of Fig. 3. Many framework papers include benchmark construction or new evaluation protocols, while benchmark papers often encode implicit architectural assumptions through their observation format, action space, environment reset policy, and completion oracle [219, 226, 272]. This coupling accelerates research while complicating cumulative evidence. A reported improvement may come from the agent architecture or the foundation model. It may instead reflect a more favorable action abstraction, tool wrapper, or task distribution.

4.2 Mobile and Web Dominance in Current Research

We observe that mobile and web interfaces dominate the corpus, as reported in the platform panel of Fig. 4. Because platform labels are multi-label, the reported coverage overlaps and a paper can contribute to more than one platform category. Under this coding, 186 papers cover mobile environments, 158 cover web environments, and 92 include desktop settings. This pattern reflects both historical and practical factors. Mobile interfaces combine a relatively constrained action space with app-scale task diversity. Their structured view hierarchies can also complement screenshots [176, 177, 219, 304]. Web environments are attractive for a different reason. DOM structure and browser automation make public tasks easier to instrument at scale [43, 48, 147, 289, 331]. Desktop environments are harder to standardize because state is distributed across windows, files, applications, and operating-system services. That complexity also makes them essential for evaluating general computer use [2, 12, 260, 301].

Cross-platform work accounts for 73 papers, or 21.7% of the corpus, as summarized in the platform panel of Fig. 4. These papers are important because they test whether a GUI agent can rely on abstractions that survive changes in device, layout convention, interaction primitive, and interface metadata. Systems such as UI-TARS and OS-ATLAS explicitly pursue more general GUI action models across platforms [171, 255], while general-purpose computer-use frameworks and benchmarks such as Agent S, OmniACT, CRAB, and MMBench-GUI aim to coordinate perception, planning, and tool use across broader computer-use settings [2, 96, 226, 270]. Cross-platform claims carry a heavier evidentiary burden than single-platform claims. A method that works on Android by using XML hierarchies may not transfer cleanly to arbitrary desktop applications. A web agent

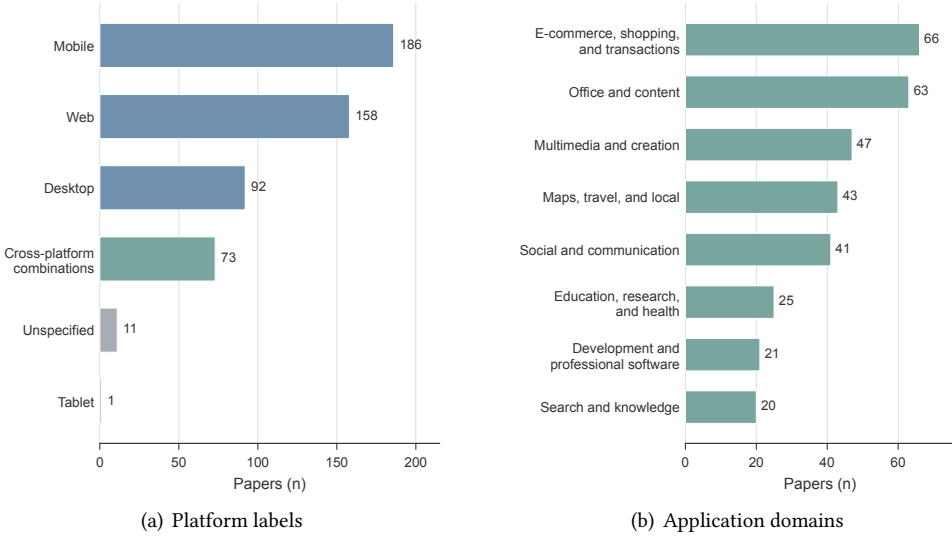


Fig. 4. Platform and application-domain distributions.

that relies on HTML or accessibility trees may face different failure modes when the interface is image-heavy, dynamically generated, or partially hidden behind authentication and personalization. Platform coverage therefore defines the assumptions under which an agent’s perception, action, and recovery mechanisms are expected to operate.

Our application-domain analysis explains why these platforms became attractive testbeds, with the distribution reported in Fig. 4. The most common domains cover everyday activities such as commerce, office work, navigation, communication, and information access. They share recognizable goals and visible intermediate states, which makes multi-step workflows easier to turn into repeatable tasks. WebShop, WebArena, VisualWebArena, AndroidWorld, WorkArena, and OfficeBench make use of this structure by translating everyday activities into repeatable tasks [54, 99, 176, 231, 289, 331]. The same panel also shows a domain spread that extends beyond consumer tasks. Scientific tools, engineering software, enterprise workflows, finance, health, and system settings point toward a harder deployment frontier. Benchmarks such as WONDERBREAD, ScienceBoard, and Spider2-V illustrate this movement toward business processes, scientific workflows, and data-science or engineering software [15, 195, 239]. In these settings, task success may depend on business semantics, personal preferences, external documents, security policies, or irreversible side effects. A click sequence can be syntactically correct while still producing a poor or unsafe real-world outcome.

This domain shift has a direct software engineering implication. Early benchmarks can often define success through page state, selected items, form completion, or textual answer matching. Real work settings require richer oracles. The evaluator may need to inspect a generated artifact, determine whether a state change is reversible, or check whether the agent respected privacy and organizational constraints. Safety, privacy, and human-centered benchmarks make this richer evaluation space explicit by measuring trustworthiness, leakage, and user-facing consequences in addition to task completion [107, 211, 314, 325]. We accordingly distinguish task completion from reliability, reproducibility, safety, privacy, and human oversight in the later analysis.

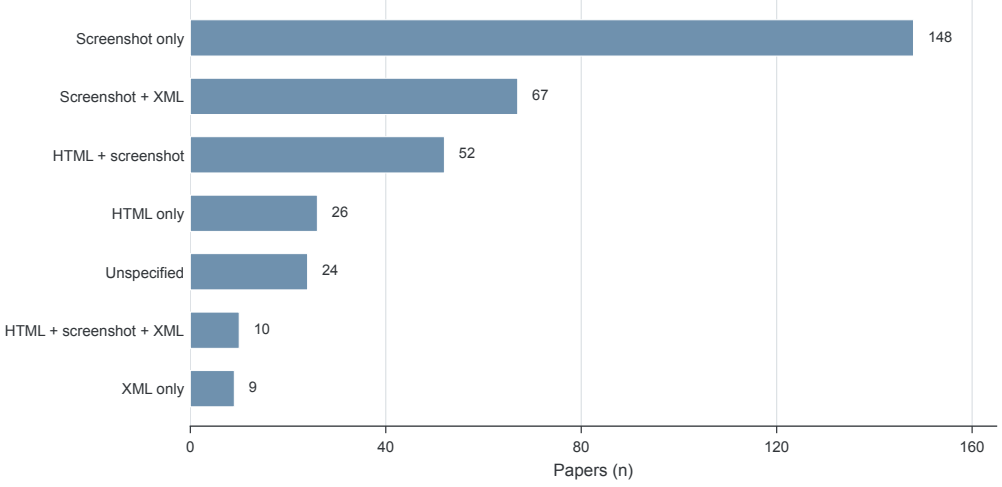


Fig. 5. Observation-medium distribution.

4.3 Screenshot-Centered Observation and Model Adaptation

The enabling technology behind recent GUI agents is also changing. We summarize the observation media used to interpret interfaces in Fig. 5 and report model-family usage and model usage patterns in Table 2 and Table 3. Screenshots appear in 277 papers, and 148 papers use screenshots as the only recorded GUI interpretation medium. This dominance reflects the appeal of vision as a platform-independent observation channel. A screenshot preserves layout, colors, icons, visual hierarchy, and spatial relations even when the underlying interface metadata is unavailable or inconsistent. Pure-vision systems and tools such as CogAgent, SeeClick, OmniParser, ScreenAgent, and You Only Look at Screens therefore occupy a central position in the field [38, 79, 149, 158, 321]. The categories in Fig. 5 are exact observation-medium combinations and therefore sum to the full 336-paper corpus used in RQ1. The two model tables also use paper-level counts from this corpus. Model-family labels are multi-label because one paper can use or compare multiple families, whereas the usage-pattern categories summarize the BASE_MODEL coding field. In the latter table, “uses” means that at least one model satisfies the condition, “only” means that all specified models satisfy it, and “combination” means that both conditions occur in the same paper.

We use the observation-medium distribution in Fig. 5 to explain why screenshots have become the default interface representation. Screenshot-only settings account for 148 papers, and another 129 papers combine screenshots with HTML, XML, or both. This pattern indicates that GUI interpretation is an alignment problem among visual appearance, structural metadata, and executable action targets. Screenshots help agents perceive what a user sees, while HTML, XML, accessibility trees, or view hierarchies can provide element text, hierarchy, clickability, and stable coordinates. Web agents often exploit DOM-level information or browser instrumentation, whereas mobile agents can use Android view hierarchies and XML states [48, 147, 235, 272]. The combined-representation categories in Fig. 5 therefore carry architectural meaning. They reveal where an agent receives extra structure, where it must infer structure from pixels, and where benchmark results may depend on platform-specific metadata that another environment cannot reliably provide.

We observe a similarly uneven model-family pattern in Table 2. GPT-4/4o and Qwen/Qwen-VL are the two most frequently mentioned families in the extracted metadata. The remaining papers form a long tail that mixes other general models with GUI-specific variants. The concentration around a few major model families reflects the practical role of strong general multimodal models

Table 2. Model-family distribution.

Model family	Papers	Share	Representative variants or coding scope
GPT-4/4o family	96	28.6%	GPT-4, GPT-4V, GPT-4o, GPT-4o-mini variants
Qwen/Qwen-VL family	91	27.1%	Qwen, Qwen-VL, Qwen2-VL, Qwen2.5-VL, Qwen3-VL variants
LLaMA/Llama family	24	7.1%	LLaMA/Llama variants excluding LLaVA-specific entries
Gemini family	21	6.2%	Gemini and Gemini Pro/Flash/Vision variants
Claude family	17	5.1%	Claude 3/3.5/3.7/4 and Sonnet/Opus variants
InternVL family	12	3.6%	InternVL and InternVL-derived variants
GPT-3.5/ChatGPT family	12	3.6%	GPT-3.5 and ChatGPT-labelled entries
UI-TARS family	11	3.3%	UI-TARS and UI-TARS-SFT/DPO/API variants
LLaVA family	10	3.0%	LLaVA and LLaVA-OneVision variants
GLM/ChatGLM family	7	2.1%	GLM, ChatGLM, and AutoWebGLM variants
OS-ATLAS family	7	2.1%	OS-ATLAS and OS-ATLAS-Pro variants
CogVLM/CogAgent family	6	1.8%	CogVLM, CogAgent, and CogAgent-Chat variants
Mistral/Codestral family	6	1.8%	Mistral and Codestral variants
PaLM/PaLI/PaliGemma family	6	1.8%	PaLM, PaLI, and PaliGemma variants
Gemma family	6	1.8%	Gemma, CodeGemma, and SLiME-Gemma variants
DeepSeek family	5	1.5%	DeepSeek-VL, DeepSeek-V3, and DeepSeek-R1-distilled variants
BLIP/InstructBLIP family	5	1.5%	BLIP-2 and InstructBLIP variants
OpenAI o-series	4	1.2%	o1/o3-labelled entries
MiniCPM family	3	0.9%	MiniCPM and MiniCPM-V variants
GPT-5 family	3	0.9%	GPT-5-labelled entries in corpus metadata
QwQ family	2	0.6%	QwQ-labelled entries
Ferret-UI family	1	0.3%	Ferret/Ferret-UI entry
ShowUI family	1	0.3%	ShowUI entry
MobileVLM family	1	0.3%	MobileVLM entry
Grok family	1	0.3%	Grok-labelled entry

Table 3. Model usage patterns.

Dimension	Usage pattern	Papers	Share	Interpretation
Availability	Uses at least one open model	163	48.5%	The paper includes one or more open-weight/open-source models
Availability	Uses only open models	95	28.3%	All specified models are coded as open
Availability	Uses at least one closed model	124	36.9%	The paper includes one or more proprietary or API-only models
Availability	Uses only closed models	56	16.7%	All specified models are coded as closed
Availability	Combines open and closed models	68	20.2%	The paper uses or compares both open and closed models
Adaptation	Uses at least one fine-tuned model	128	38.1%	The paper includes a model adapted through GUI/task-specific training or tuning
Adaptation	Uses at least one off-the-shelf model	146	43.5%	The paper includes a base model used without GUI/task-specific tuning
Adaptation	Uses only off-the-shelf models	92	27.4%	All specified models are used without GUI/task-specific tuning
Adaptation	Combines fine-tuned and off-the-shelf models	54	16.1%	The paper uses or compares both adapted and non-adapted models

as reasoning and perception engines. The long tail of GUI-specific or open model families shows a parallel effort to build models whose perception, grounding, action syntax, and reasoning traces are better aligned with interface interaction. This split is visible in framework-oriented systems that embed a foundation model inside a larger agent loop [2, 235, 301], as well as in fine-tuned or GUI-native systems such as CogAgent, SeeClick, OS-ATLAS, UI-TARS, ShowUI, and MobileGUI-RL [38, 79, 122, 171, 183, 255].

We further separate model availability from model adaptation in Table 3. Nearly half of the corpus uses at least one open model, while 36.9% uses a closed model. One fifth combines both within the same paper. Open models support inspection and deployment control. Closed models remain attractive when rapid prototyping or high-end multimodal capability is the immediate goal. The adaptation rows show a second axis of variation. Papers that use at least one fine-tuned model account for 128 cases, while 146 papers use at least one off-the-shelf model and 92 rely only on off-the-shelf models. These numbers point to two complementary paths. One orchestrates general-purpose models within an external runtime. The other adapts the model itself to GUI perception, grounding, and executable action prediction.

The adaptation path appears across mobile, web, and cross-platform research. Mobile work emphasizes visual state abstraction, action prediction, and online device control [50, 115, 154, 160, 164, 170, 179, 186, 213, 249, 310]. Web work increasingly uses curricula, large-scale trajectories, and real-world exploration [61, 76, 128, 161, 169, 206, 210, 298, 305, 306, 329]. Reinforcement-learning studies treat interaction as sequential control and use experience, evaluators, or critics to improve action selection and error detection [9, 27, 119, 134, 137, 142, 234, 258, 271, 285, 296, 332]. Cross-platform models further incorporate action histories, speech, reward learning, and generalized task execution [10, 73, 143, 228, 244, 274, 281].

Perception and assistive tools connect model adaptation to executable agent loops. Grounding methods convert screenshots into actionable regions or candidate elements [6, 32, 33, 80, 101, 151, 165, 166, 173, 202, 205, 251, 266, 287, 291, 320], while representation methods add screen schemas, region consistency, and layout-aware reading [8, 14, 55, 58, 95, 118, 180, 230, 279, 295, 308, 336]. Synthetic data and large-scale grounding resources then connect these representations to training and benchmark construction [17, 20, 59, 90, 108, 113, 114, 132, 218, 275]. Together, this work explains the heterogeneity in Table 2. The model category includes foundation models, perception modules, and policy-learning systems, all of which may occupy different positions in the same larger agent runtime used during execution.

Three technical trends follow from the combined evidence in Fig. 5, Table 2, and Table 3. First, GUI-agent research is moving toward native multimodal action models that learn from screenshots, element descriptions, action traces, and interactive feedback, reducing reliance on hand-designed prompts alone [122, 133, 171, 255]. Second, observation design is becoming a first-order architectural choice. The decision to use pixels, HTML, XML, accessibility trees, or their combination determines what the agent can perceive, how it grounds actions, what failures can be diagnosed, and how portable the system is across platforms [147, 149, 235]. Third, benchmark design and model design are becoming increasingly intertwined. A benchmark that exposes DOM nodes encourages different agent strategies from one that permits screenshots only. A benchmark with live interaction and delayed consequences places greater demands on memory and recovery than a static grounding dataset [99, 176, 260, 331].

Answer to RQ1. We find that GUI-agent research has changed from a small collection of platform-specific automation studies into a rapidly expanding, multi-community field. The concentration of papers in 2024–2026 and the prevalence of preprints, frameworks, and benchmarks indicate uneven evidence maturity and evolving methodological conventions. Mobile and web environments remain dominant, while desktop, cross-platform, and professional applications extend the field toward more consequential forms of computer use. Screenshots are the primary observation channel, often supplemented by structural metadata. Model development follows two paths. Some studies orchestrate general multimodal models within modular systems, while others adapt GUI-specific models for perception, grounding, and action. We conclude that systems, benchmarks, representations, and models are evolving together. The architectural analysis in RQ2 examines their coupling, and RQ3 assesses whether the resulting evaluations support comparable system-level claims.

5 RQ2: Architectures and Engineering Techniques

We use RQ2 to translate the field-level trends in RQ1 into system responsibilities and interfaces. Module statistics use the 145 framework papers, while broader architectural and recovery signals draw on the SE-parseable corpus where stated. Across platforms, a stable closed loop appears [2, 48, 235, 260, 301]. An agent observes the GUI, constructs an internal state, selects and executes an action, and uses the resulting state to continue, verify, recover, or stop. This loop is shared by web agents that operate through DOMs and browser automation, mobile agents that exploit

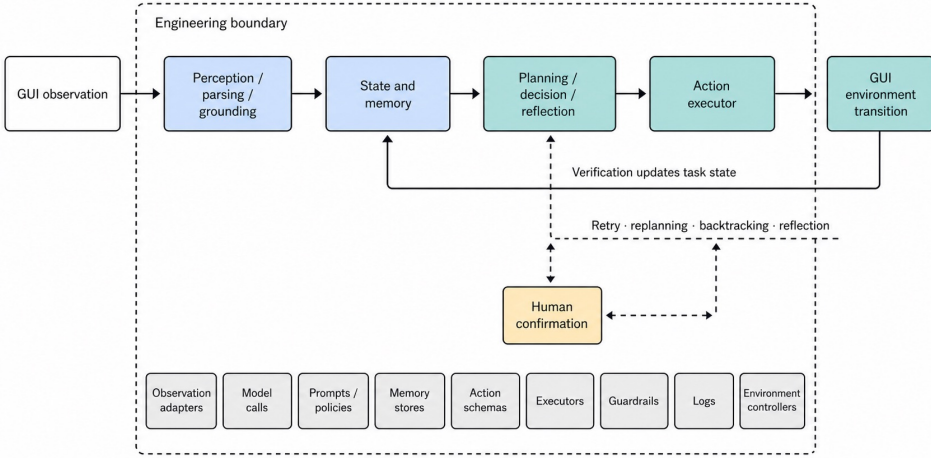


Fig. 6. Modular GUI-agent architecture.

screenshots and Android view hierarchies, and desktop agents that must coordinate across windows, applications, and files.

We use Fig. 6 to separate the universal perceive–decide–execute loop from the modules that support longer tasks and safer operation. The basic loop appears in all framework papers, while the placement of state, memory, verification, recovery, and human intervention varies. Some systems invest in visual parsing and grounding to make screenshots actionable [38, 111, 149]. Long-horizon agents instead devote more architecture to planning, memory, and reflection [2, 208, 315]. Where runtime risk is central, verifiers and action guards are joined by safety filters or human queries [45, 104, 253]. Our analysis follows the flow from representation through reasoning to execution and recovery.

Our module counts in Table 4 support interpreting GUI-agent frameworks as control systems with model components embedded in larger runtimes. Perception, decision, and execution form the universal skeleton. Perception constructs a usable state from the GUI observation. Decision selects an operation from that state and the task history, after which execution applies it to the environment. Verification, memory, planning, and reflection then indicate support for longer-horizon control. Verification checks whether an action or trajectory satisfies an expected condition, memory stores task progress and reusable interaction knowledge, planning decomposes or revises goals, and reflection critiques failures or updates strategies. We observe a maturation path from direct observation-to-action policies toward components that preserve state, reason over alternatives, check progress, and repair mistakes in long-horizon tasks. The variation in module support also complicates architectural comparisons. Two papers may both report a task-success rate, yet one may rely on a rich browser controller, structured memory, and a verifier, while another may ask a multimodal model to predict a coordinate from a screenshot. We use module placement and responsibility as core evidence for architectural comparison.

Table 4. Common architectural modules.

Module	Papers	Share of frameworks
Perception	145	100.0%
Decision	145	100.0%
Execution	145	100.0%
Verification	93	64.1%
Memory	88	60.7%
Planning	79	54.5%
Reflection	61	42.1%

5.1 Observation-Driven State Construction

The perception layer transforms raw GUI state into a representation that a planner or policy can use. Screenshots dominate the corpus and provide a portable observation across platforms. They do not directly expose which elements are clickable, editable, disabled, hidden, or semantically linked. GUI agents therefore build perception pipelines that combine visual models, OCR, UI parsers, accessibility metadata, DOM/XML structures, coordinate normalization, and element-ranking rules. The resulting representation becomes the contract between the external GUI and the agent’s internal reasoning and subsequent action selection.

Pure-vision systems treat screenshots as the primary state. CogAgent and SeeClick learn to ground instructions or elements in visual interface images, while OmniParser converts screenshots into structured visual elements that can be consumed by downstream agents [38, 79, 149]. ScreenSpot and related grounding benchmarks isolate the accuracy of matching natural-language element references to screen targets [111]. Screenshot-first agents such as ScreenAgent, You Only Look at Screens, ShowUI, OS-ATLAS, and UI-TARS further demonstrate how interface understanding can be framed as a multimodal perception-and-action problem [122, 158, 171, 255, 321]. Screenshots remain usable when DOMs, accessibility trees, or app internals are unavailable. The cost is that an agent must infer interactivity, hierarchy, and hidden state from pixels or from model priors alone during execution.

Structured-interface systems push part of this burden into platform metadata. Web agents can use HTML, DOM nodes, accessibility trees, browser APIs, and page-change feedback to identify action targets and monitor transitions [48, 147, 331]. Mobile agents can exploit Android XML, view hierarchies, and device-control APIs, as seen in AutoDroid, AndroidWorld-oriented systems, AppAgent-style mobile automation, and MobileAgentBench-style evaluation [117, 176, 219, 235]. Desktop agents such as UFO, UFO2, Agent S, and PC-Agent must handle accessibility trees, OCR, window selection, application context, and mouse/keyboard actions at the operating-system level [2, 127, 300, 301]. These systems show that structured metadata can improve grounding and action validity while introducing platform dependence. A DOM-aware method and an Android-XML-aware method may solve different engineering problems under the same label of “GUI agent”.

Mobile-control research illustrates how representation choices accumulate into system design. Earlier systems combine screenshots, XML hierarchies, task descriptions, and learned policies in different proportions [44, 193, 237, 277, 307]. Newer frameworks add exploration, page graphs, modality fusion, and adaptive control [36, 40, 52, 159, 256, 259]. Retrieval, knowledge graphs, world models, and backtracking then compensate for information absent from the current screen [70, 144, 194, 246, 269]. The common architectural point is that perception defines what later modules can store, verify, generalize, and recover.

We group specialized modules beyond the basic loop in Table 5. The counts are multi-label semantic groupings of the specialized-module field across the 145 framework papers. These specialized modules serve different stages of control. Memory and retrieval preserve experience beyond the current screen. Perception enhancement turns raw observations into more actionable states, while planning and search explore how the agent should proceed from them. Reflection modules

Table 5. Specialized module themes.

Specialized theme	Papers	Share of frameworks
Memory and retrieval	51	35.2%
Perception enhancement	47	32.4%
Planning and search	44	30.3%
Reflection and self-improvement	33	22.8%
Safety, privacy, and verification	19	13.1%
Multi-agent and role decomposition	12	8.3%
Other or uncategorized modules	28	19.3%

add critics, reward signals, bootstrapping, retry logic, or post-action critique, while safety/privacy modules add filters, guards, validators, or sensitive-operation checks. Perception enhancement is the second most common theme, appearing in 47 framework papers. These modules usually sit between raw observation and decision making. Representative systems intervene at different points in this process. Less is More simplifies the GUI context before reasoning, whereas MGA combines spatial-semantic grounding with cross-step memory. AUTO-Explorer and Explorer use parsing or systematic element collection to support data acquisition, and PC-Agent combines accessibility information with OCR for active perception [19, 28, 41, 71, 127]. The architectural issue extends beyond perception accuracy. A perception module also determines the size of the action space, the cost of each step, the information exposed to the model, and the failure modes available for diagnosis. When perception produces candidate elements and confidence signals, downstream modules can verify or recover. When perception is hidden inside a model prediction, the system may achieve a compact interface while making errors harder to localize during debugging.

Specialized modules further shift perception from screen understanding toward control-ready state construction. They add reusable exploration, region structure, temporal reuse, modality fusion, and grounding confidence before an action is selected [22, 85, 184, 196, 200, 230, 264]. We interpret perception modules as state constructors. Pixel-only observation maximizes platform portability and aligns with general multimodal models. Structured metadata improves action grounding and state tracking when available, and hybrid representations are common because real interfaces rarely provide a complete view of the task state. This choice shapes every later module. A planner can reason only over the state it receives. The same boundary constrains verification and safety because neither can inspect signals omitted by the observation layer during runtime diagnosis.

5.2 Explicit State and Feedback for Long-Horizon Control

The reasoning layer determines how an agent maintains the task goal across steps. In short tasks, a model can often select the next action from the current screen and instruction. In long-horizon GUI interaction, the current screen may omit earlier choices, hidden constraints, user preferences, or pending subtasks. The agent must therefore preserve a compact task state, decompose objectives, track progress, decide when a plan is invalid, and incorporate feedback from executed actions. Our coding identifies explicit planning in 79 framework papers, memory in 88, and reflection in 61, as reported in Table 4. These modules define how agents move from single-screen grounding to multi-step control and recovery.

Planning modules appear in several forms. Dynamic planning systems revise action sequences as the GUI changes, while hierarchical planners separate high-level subgoals from low-level grounding. Dynamic Planning for GUI Automation, ScaleTrack, HiconAgent, Octo-Planner, Agent S, UFO, Agent-E, OS-Copilot, OSCAR, and PC-Agent illustrate variants of this separation [1, 2, 35, 77, 84, 225, 252, 301, 315, 332]. Search-based methods expose alternative paths more directly. Agent Alpha and Agent Q search over GUI states, with Agent Q adding MCTS-guided critique. Mirage-1 organizes multimodal skills hierarchically, while WebPilot combines strategic exploration with multi-agent execution [168, 204, 262, 317]. We find that planning becomes valuable when action consequences are uncertain, when backtracking is possible, or when a task requires information gathering before commitment.

Web agents make this design space especially visible because browser tasks combine explicit links and forms with long-horizon state changes. Existing systems explore search, long-context planning, visual context, browser control, and lightweight agent baselines [72, 75, 91, 98, 100, 102, 150, 155, 190, 199, 282, 303]. Others learn environment dynamics, workflow traces, rollback rules, or reusable skills [16, 68, 181, 233, 273, 313, 319, 326, 334]. Web-agent planning spans navigation

guidance, explicit search, learned environment models, specialized roles, and reusable skills. Each design imposes different costs for debugging and reproducibility.

Memory modules address a different bottleneck. GUI agents often operate under context-window limits, noisy histories, and repeated interface patterns. Memory designs differ in what they preserve and how they reuse it. AppAgentX and Agent S retain experience for later tasks, whereas AgentProg manages the active context through program structure. MOBA separates several forms of task and application state. R2D2 and MobileGPT instead organize experience as graphs that support navigation or reuse [2, 86, 94, 106, 208, 335]. These systems suggest that memory is both a performance mechanism and an engineering boundary. A memory store can reduce prompt length, improve reuse, and support recovery. It also raises maintenance questions about validity after app updates, stale-trajectory detection, storage of private screen content, and the influence of retrieved memories on new tasks.

Reflection modules complete the control loop by interpreting failures and updating behavior. Structured reflection for computer control, Mobile-Agent-E, AutoGLM, UItron, InfiGUIAgent, Guardian, and verifier-driven mobile agents all incorporate mechanisms for self-critique, outcome checking, curriculum improvement, or failure recovery [45, 112, 130, 133, 174, 232, 297]. Reflection is architecturally distinct from verification. Verification asks whether a condition holds. Reflection explains what went wrong and suggests the next strategy. The two are often coupled, and their distinction matters for testing. A verifier can be unit-tested against expected states or action outcomes, while a reflection module is evaluated through its effect on subsequent trajectories.

Mobile and cross-platform systems extend reflection and memory from isolated failure handling to continual adaptation. LearnAct, MobileSteward, Mobile-Agent-V, FedMobileAgent, ReachAgent, Learn-by-interact, CHOP, and PersonalAlign use demonstrations, self-evolution, decentralized user data, page reaching, interactive learning, optimized subtask planning, or long-term user records to make repeated GUI interaction more adaptive [126, 135, 146, 191, 217, 224, 248, 333]. Process-reward and inference-time guidance methods provide a complementary route by shaping the agent’s decisions during execution, as in GUI-PRA, process-reward VLM guidance, variational subgoal-conditioned RL, and language multi-agent learning with credit re-assignment [78, 83, 247, 264]. We observe adaptation at several time scales. Grounding confidence acts within a step, reflection or backtracking acts within a trajectory, memory acts across tasks, and personalized or federated data acts across users.

We report multi-label architectural patterns from 327 SE-parseable papers in Table 6 and compare them with the framework-module statistics in Table 4. The dominant architectural tendency is modular. Modular or multi-stage pipelines expose responsibilities for inspection and replacement. External execution layers move part of the system boundary into wrappers, browsers, device controllers, or operating-system tools. State-history loops add continuity across screens and across runs. Planner-grounder or planner-executor separation splits strategic task decomposition from concrete target selection and action application. End-to-end native-agent designs compress several responsibilities into trained multimodal action models, making the interface compact while reducing component-level observability. Native models remain technically central because systems such as UI-TARS, OS-ATLAS, ShowUI, and InfiGUIAgent improve the learned perception-action core [122, 133, 171, 255]. The corpus nevertheless indicates that most practical GUI-agent systems still externalize responsibilities into modules. They do so because GUI control requires translating

Table 6. Architectural patterns.

Architectural pattern	Papers	Share
Modular pipeline or multi-stage architecture	154	47.1%
External tool/browser/device/API execution layer	139	42.5%
Memory, reflection, or historical-context loop	116	35.5%
Planner-grounder or planner-executor separation	63	19.3%
End-to-end VLM or native-agent architecture	18	5.5%

model output into a safe executable command, observing effects, updating state, and handling environment-specific constraints in addition to predicting an action token.

Multi-agent decomposition is a smaller but conceptually important version of modularity. COLA, AssistEditor, WebPilot, MobileExperts, Agent S2, PC-Agent, and UFO2 distribute tasks across managers, workers, planners, editors, grounders, verifiers, or specialized decision agents [3, 63, 127, 300, 309, 317, 323]. This design can align responsibilities with different knowledge sources or action spaces. A manager can maintain global progress, a grounder can resolve coordinates, a verifier can assess completion, and a worker can execute local steps. The same design increases coordination cost and introduces failure points involving stale shared state, inconsistent subtask boundaries, conflicting judgments, and accumulated latency. We interpret multi-agent GUI frameworks as evidence of a shift from prompt engineering toward explicit software architecture.

Desktop and general-computer-use systems make this shift visible because they must coordinate multiple applications, files, windows, and specialized tools. CCAgent, AgentStore, MMAC-Copilot, Programming with Pixels, COLA, and AgentStudio distribute responsibilities across collaborative agents, agent stores, copilot modules, software-engineering tasks, Windows UI automation, or virtual-agent toolkits [4, 30, 93, 189, 322, 330]. Ponder & Press, UIPro, SpiritSight Agent, Mobile-Agent-v3, Continual GUI Agents, and OmegaUse push the same architectural pressure toward cross-platform or general-purpose execution, where a single agent loop must handle diverse layouts, tasks, and operating contexts [89, 109, 136, 227, 290, 312]. We find that modularity remains common because the surrounding software environment still requires execution adapters, policy boundaries, logs, and recovery channels even when strong native action models are available.

5.3 Gaps in Runtime Control and Action Governance

The execution layer converts an agent’s decision into an operation that changes the GUI environment. It is the point where model uncertainty becomes external state change. Across platforms, action execution ranges from simple coordinate clicks and text entry to browser commands, Android intents, accessibility-tree actions, Python scripts, API calls, long screenshots, window switching, and application-specific tools. AutoDroid and AutoDroid-V2 illustrate how mobile execution can combine Android state abstraction with generated code [235, 236]. On desktops, UFO and UFO2 divide control between host and application agents and add safeguards for sensitive actions [300, 301]. OS-Copilot, CRADLE, and OSCAR broaden the executor again by combining operating-system actions with tool generation and state-aware reasoning [198, 225, 252]. These examples make the executor a major architectural component with its own state, constraints, and failure modes.

Action schemas are one source of variation. Some systems use low-level primitives such as click, type, scroll, and wait. Others introduce higher-level actions such as opening an app, calling an API, taking a long screenshot, selecting a window, querying an element, executing code, or handing control to a user. MagicGUI includes an extended mobile action space with API calls, screenshots, takeover, waiting, and text entry [203]. PromptRPA and Prompt2Task translate user prompts into reusable mobile procedures and include assessment or intervention mechanisms [87, 88]. Enterprise-oriented workflows use demonstrate-execute-validate patterns and standard operating procedures to constrain execution [51, 238]. Higher-level actions can reduce trajectory length and improve reliability when correctly specified, but they move more semantics into the executor and require clearer preconditions, permissions, and failure handling. The runtime-control signals in Table 7 are not mutually exclusive. Its exception-handling rows use the 327 SE-parseable papers, while the explicit human-in-the-loop row uses the 145-paper framework subset.

Our analysis of Table 7 reveals a gap between autonomous execution and accountable runtime control. Invalid-action

Table 7. Runtime reliability and intervention signals.

Runtime-control signal	Papers	Share
Invalid/redundant/malformed action handling	95	29.1%
Reflection, correction, retry, or replanning	86	26.3%
Human request, confirmation, or takeover	17	5.2%
Explicit human-in-the-loop flag	3	2.1%

handling appears in 95 SE-parseable papers, covering mechanisms that reject, filter, correct, or avoid actions that cannot be executed safely or usefully. Reflection, correction, retry, or replanning appears in 86 papers, showing that many systems recognize execution failure as a normal runtime condition. The lower counts for human request, confirmation, or takeover show that escalation remains less systematically engineered, and the explicit human-in-the-loop signal appears in only a very small subset of framework papers. Verifier-driven agents such as V-Droid, VeriSafe, STEVE, and Guardian move the field toward explicit post-action checks and runtime validation [45, 104, 138, 174]. Guardrail and privacy systems such as GUIGuard and GuardAgent further show that execution should be conditioned on policy, privacy, and safety reasoning alongside task progress [229, 257].

Failure recovery has three recurring forms. Local retry or repair repeats an action, changes the target, waits for a page to load, or uses a backup grounding strategy. Plan-level revision updates a subgoal, backtracks, or searches an alternative path, as in tree-search and MCTS-inspired agents [168, 204, 317]. Knowledge update stores the failed trajectory, modifies memory, or evolves a shortcut or skill for future tasks [86, 94, 232]. Each recovery form requires different evidence. Action-level outcomes can reveal local repair, while plan revision requires trajectory comparison. Knowledge updates can be assessed only across repeated tasks or over time. A single aggregate task-success metric can hide these differences.

Human intervention is the least developed but most consequential execution mechanism. VeriOS explicitly studies proactive human-agent-GUI interaction for trustworthy OS agents, while MobileAgent and MobileGPT include human-machine interaction, SOP integration, or human repair mechanisms [51, 106, 253]. Other systems use uncertainty-aware refinement, follow-up questions for ambiguous instructions, confirmation for sensitive actions, interactive modes, or user fallback without consistently treating the human as a first-class runtime component [1, 42, 74, 300, 301]. Many GUI tasks contain decisions that are inappropriate to automate silently, including login, payment, deletion, permission granting, message sending, medical or financial operations, and enterprise workflow approval. A robust architecture should define when to ask, what evidence to present, how user feedback changes the plan, and how the intervention is logged.

Voice, RPA, and toolkit-oriented systems provide additional execution interfaces through process automation, conversational control, reusable procedures, and constrained web policies [69, 162, 185, 212, 261, 293]. Although some are not full autonomous agents, they show how users, procedures, and tools can restrict the executable action space.

Answer to RQ2. We find that GUI-agent architectures share a closed perceive–decide–execute loop whose responsibilities are increasingly distributed across modules. All 145 framework papers expose perception, decision, and execution, while verification, memory, planning, and reflection provide additional support for long-horizon control. Observation design determines the state available to the agent. Reasoning modules preserve goals and revise strategies, while executors translate uncertain model outputs into external state changes. The dominant architecture embeds foundation models within a larger runtime. Adapters and memories construct state, planners and grounders select operations, and controllers apply them under partial verification or guardrails. Recovery, human escalation, safety enforcement, and auditability remain less explicit than perception and planning. This imbalance allows task success to conceal brittle or unsafe runtime behavior and establishes the evaluation problem examined in RQ3.

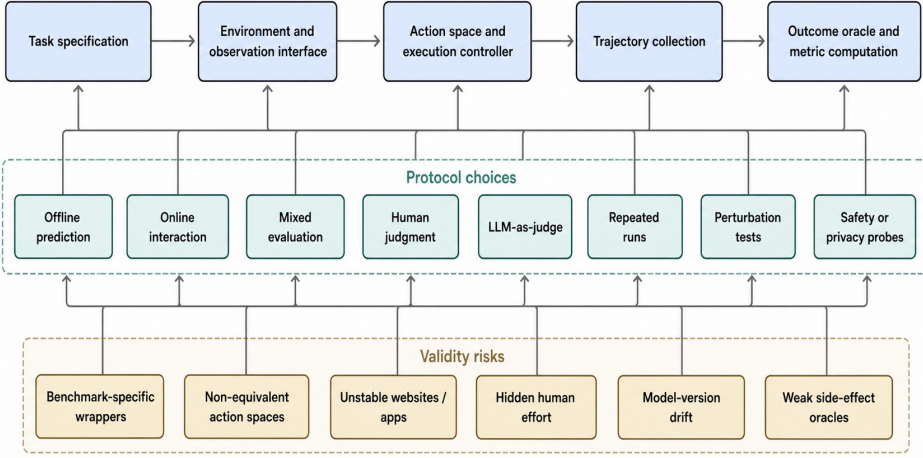


Fig. 7. GUI-agent evaluation pipeline.

6 RQ3: Evaluation and Benchmarking

We use RQ3 to examine whether current evaluations provide credible evidence for the systems characterized in RQ2. The analysis uses the 252 papers coded as frameworks, models, or evaluation studies. GUI-agent evaluation connects offline capability tests, interactive trajectory execution, and oracles that judge outcomes and behavior. The corpus contains extensive benchmark activity, while its evidence remains centered on task success. We analyze benchmarks, modalities, metrics, and validity threats as parts of one evaluation pipeline.

We use Fig. 7 to show that a GUI-agent benchmark evaluates the agent and the surrounding interaction protocol together. A reported score depends on task wording, allowed observations, action schema, environment reset policy, time or step budget, authentication state, external tools, and the oracle used to judge completion. WebArena, VisualWebArena, AndroidWorld, OSWorld, and Windows Agent Arena made this dependency visible by moving evaluation into realistic web, mobile, and desktop environments [12, 99, 176, 260, 331]. Earlier and complementary datasets such as WebShop, Mind2Web, WebLINX, OmniACT, and ScreenSpot isolate grounded language-to-action prediction, website navigation, generalist action modeling, or screen grounding [48, 96, 111, 147, 289]. The distinction matters because an offline action-prediction score and an online task-success rate answer different questions.

6.1 Protocol Heterogeneity in Interactive Benchmarks

We use the 252-paper framework, model, and evaluation subset for Table 8. Benchmark relation follows the coded paper type, while modality distinguishes offline data, live or interactive environments, and combinations of the two.

Our analysis shows that benchmark construction is a major contribution type without dominating all evaluation activity. Among the 252 system-oriented papers, 77 contribute benchmarks, datasets, environments, or protocols, while 175 mainly evaluate systems or models through existing benchmarks, self-designed tasks, or experimental environments. This balance is healthy for a young field because new benchmarks expose new capabilities and risks. It also creates a comparability challenge. When a framework introduces its own tasks, action abstractions, environment adapters, or success oracles, its reported improvement can reflect both agent design and evaluation design.

Table 8. Evaluation relations and modalities.

Dimension	Category	Papers	Share
Benchmark relation	Existing/custom evaluation	175	69.4%
Benchmark relation	Self-built/contributed benchmark	77	30.6%
Evaluation modality	Mixed online/offline	164	65.1%
Evaluation modality	Offline	52	20.6%
Evaluation modality	Online	36	14.3%

We interpret benchmark papers as technical artifacts that encode assumptions about what a GUI agent should observe, how it should act, and what counts as success.

The modality counts in Table 8 reveal a strong preference for mixed evaluation. Mixed protocols combine interactive execution with offline prediction, ablation, benchmark statistics, or trajectory analysis. Offline protocols are more controlled and typically measure perception, grounding, action prediction, classification, or static outputs. Online protocols place agents in live or simulated GUI environments and judge task completion or trajectory outcomes. This pattern is visible in modern benchmarks that combine interactive execution with offline analysis. The mixed protocols take several forms. WebShop links interaction to grounded rewards, whereas Mind2Web uses offline prediction to study cross-site generalization [48, 289]. WebLINX and conversational web-agent studies add multi-turn data that supports turn-level prediction and broader trajectory analysis [49, 147]. Interactive benchmarks then expose the agent to environment feedback. WebArena and VisualWebArena use realistic website state changes [99, 331]. AndroidWorld and related infrastructure bring the same principle to device control, while OSWorld and Windows Agent Arena extend it to desktops [12, 176, 219, 260, 272].

We use Fig. 8 to trace the expansion of benchmark scope from web navigation and mobile action prediction to general computer use, professional workflows, safety, privacy, and human-centered evaluation. Web resources such as WebShop, Mind2Web, WebArena, VisualWebArena, WebLINX, WebVLN, TurkingBench, NaviQate, VisualWebBench, WebGames, RealWebAssist, and BrowserGym emphasize shopping, navigation, browser interaction, multimodal grounding, long-horizon assistance, and trajectory analysis [31, 43, 48, 99, 129, 147, 178, 207, 267, 289, 292, 331]. Mobile resources such as AndroidWorld, AndroidLab, A3, MobileAgentBench, FedMABench, GUI-Robust, MobileSafetyBench, and See-Think-Act focus on Android task execution, procedural state, decentralized data, efficiency, robustness, and safety [18, 103, 176, 219, 222, 223, 254, 272, 280]. Desktop, game, and cross-environment benchmarks such as OSWorld, Windows Agent Arena, OmniACT, FlashAdventure, and CRAB test open-ended OS tasks, multi-application operation, desktop/web action prediction, long-horizon game interaction, and cross-environment task completion [5, 12, 96, 260, 270]. The workflow and risk-aware families extend the same logic to knowledge work, scientific workflows, safety, privacy, adversarial content, cultural awareness, and human judgment through WorkArena, WorkArena++, ScienceBoard, SafeArena, ST-WebAgentBench, AgentDAM, EIA, EVA, Computer Agent Arena, and AgentRewardBench [11, 54, 107, 120, 140, 148, 172, 195, 211, 214, 325]. These resources move evaluation beyond a single success/failure endpoint.

The density of each benchmark family reflects different sources of difficulty. Web benchmarks vary task horizon, visual demand, traversal structure, simulation control, sequential composition, and real-world assistance [62, 66, 92, 163, 187, 209, 243, 276]. Grounding resources isolate pixel-level understanding and the connection between visual evidence, user intent, and action targets [60, 284]. Together, they show that a single web score cannot distinguish semantic misunderstanding, grounding failure, lost context, or site-specific overfitting to individual sites.

Mobile benchmarks add variation in device configuration, app diversity, language, ambiguity, cross-app navigation, agent weaknesses, and reasoning-execution consistency [29, 47, 53, 65, 105,

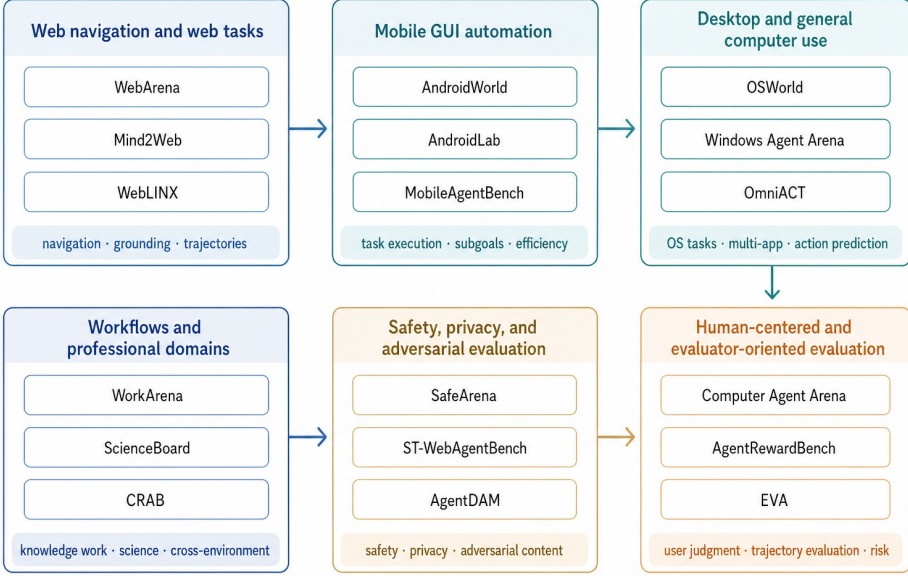


Fig. 8. GUI-agent benchmark and evaluation families.

139, 263, 311]. More diagnostic resources evaluate command feasibility, partial progress, failure categories, and evaluator reliability alongside final screen state [13, 175, 192]. Mobile evaluation covers action-target correctness, task feasibility, and execution faithfulness in the current device state.

Desktop, video, and general visual-agent benchmarks broaden the boundary to temporal procedures, multi-application work, and software-engineering-like tasks [4, 26, 121, 131, 156, 245]. We find across these families that a benchmark evaluates an agent together with a particular observation channel, action system, and environment controller.

Protocol design remains a central difficulty. Offline evaluation is repeatable and helps isolate perception, grounding, action type, click target, or action-argument correctness. It is especially useful for model development and ablation because the same examples can be reused across models. Online evaluation is closer to deployment because it includes latency, environment state, side effects, recovery, and termination. Mixed evaluation appears most often because researchers need both forms of evidence. A complete benchmark should therefore specify observation permissions, action space, reset behavior, task timeout, repetition policy, judge type, and whether external tools or human interventions are allowed. Without these details, a comparison between two agents may reflect hidden differences in scaffolding.

6.2 Dominance of Task Success in Evaluation

We use the same 252-paper subset for the paper-level, multi-label counts in Table 9. The analysis covers high-frequency metrics and less frequent categories that capture system quality. We find task success in 194 of the 252 system-oriented papers, making it the dominant metric. Task-outcome metrics ask whether the trajectory reaches the intended final state. Specialized settings may instead measure partial progress or the functional correctness of a produced artifact. This dominance is understandable because GUI agents are ultimately judged by whether they complete user tasks,

and task success can often be compared across agents with a single number. However, the table also shows why a success-rate-centered view is incomplete. A successful trajectory may still rely on excessive retries or hidden human repair. It may also expose private information, violate a safety constraint, or depend on a benchmark-specific wrapper. We separate outcome metrics from process, cost, reliability, safety, and human-centered metrics used in practice.

Grounding and action metrics remain essential because many GUI-agent failures arise before task-level judgment is possible. Grounding metrics localize failure before the final outcome. They test whether the agent selected the right target, action type, and arguments for an executable operation. These metrics are especially important for model papers and offline datasets, including ScreenSpot, SeeClick, CogAgent-style grounding, OmniACT, DreamStruct, and mobile-interface understanding tasks [38, 79, 96, 111, 167, 197]. They also help diagnose online failures. If a web task fails, the error may originate in high-level reasoning or element grounding. The action type, its text argument, or delayed page state can produce the same final outcome. Step-level metrics make these causes more visible than final task success alone.

Efficiency metrics are becoming more visible because GUI agents often operate through expensive multimodal model calls and repeated environment interactions. Compute or token cost appears in 74 papers, interaction efficiency in 58, and runtime efficiency in 44. Systems such as MobileAgentBench, WABER, Windows Agent Arena, and on-device or distributed-control work make latency, interaction length, and resource use explicit [12, 34, 97, 219, 221]. These metrics should be interpreted with platform context. A five-step desktop task and a five-step mobile task may differ in action latency, rendering delay, state recovery, and cost per observation. Efficiency also interacts with reliability. Reducing the number of steps can improve user experience, while aggressive shortcuts can increase the risk of irreversible or poorly verified actions.

Safety, privacy, and human-centered metrics form a smaller but growing layer. Safety and human-centered evaluations use a more varied set of measures. Studies report attack or safeguard success, policy compliance, privacy leakage, human agreement, usability, and user effort, but each appears far less often than task success. Their lower frequency should not be read as lower importance. These metrics become central when an agent can observe sensitive content or change external state. Submitting a form, sending a message, and altering a file carry risks that task success alone cannot represent. SafeArena, ST-WebAgentBench, AgentDAM, EIA, EVA, and pop-up attack evaluations show how adversarial or policy-oriented benchmarks can reveal harms that ordinary task-completion benchmarks overlook [107, 120, 140, 211, 318, 325]. Human-centered evaluations and Computer Agent Arena add another missing dimension by asking which agent behavior users prefer and how they judge risk, usability, and step-wise behavior [25, 214].

Risk-aware evaluation is now large enough to be treated as its own benchmark family. Risk-aware benchmarks manipulate the GUI in several ways. Some inject fine print or hidden instructions, while others introduce backdoors, unprivileged interference, or broader environmental distraction [24, 39, 124, 152, 153, 242, 265, 286, 288]. We use these studies to assess whether an agent follows malicious cues or leaks information under manipulation. Other evaluation papers probe whether API-based web agents, grounded GPT-4V web agents, and preliminary Claude computer-use agents

Table 9. Metric families.

Family	Metric	Papers
Task outcome	Task success rate	194
Task outcome	Task completion/progress	44
Task outcome	Exact match	28
Grounding/action	Benchmark accuracy	73
Grounding/action	Action prediction correctness	51
Grounding/action	UI grounding accuracy	48
Grounding/action	Action type accuracy	44
Efficiency/cost	Compute/token cost	74
Efficiency/cost	Interaction efficiency	58
Efficiency/cost	Runtime efficiency	44
Reliability	Error/failure rate	22
Safety/privacy	Attack success rate	17
Safety/privacy	Constraint/policy compliance	14
Human/judge	Human agreement	14
Human/judge	LLM-as-judge score	14
Human/judge	Usability score	11

Table 10. Evaluation validity and reproducibility concerns.

Concern	Primary affected protocol element
Observation mismatch	Screenshots, DOM/XML, accessibility trees, OCR, tool descriptions
Action-space mismatch	Coordinates, element IDs, browser actions, device APIs, code/tool execution
Oracle ambiguity	Exact match, state match, reward, LLM-as-judge, human labels
Environment instability	Websites, apps, authentication, pop-ups, localization, model APIs
Cost and hidden assistance	Retries, prompt length, tool calls, human intervention, cached knowledge
Safety and privacy coverage	Malicious content, sensitive data, irreversible actions, policy constraints

actually satisfy their claimed interaction capabilities under realistic protocols [81, 188, 327]. Together with safety and privacy metrics in Table 9, these studies shift evaluation from capability ranking toward behavioral auditing of complete agent trajectories.

The metric distribution also exposes a measurement gap around recovery and long-term dependability. Error/failure rate is present in 22 papers, termination correctness in 4, and recovery success in only 2 according to the coded long-tail metrics. This is small relative to the architectural emphasis on verification, reflection, memory, and replanning observed in RQ2. We find that many systems implement recovery mechanisms while evaluating them indirectly through aggregate success. A stronger evaluation design would measure how often agents enter loops, how quickly they detect failed actions, whether retry policies are bounded, whether recovery changes the failure mode, and whether human escalation occurs at appropriate times. AgentRewardBench provides an important step by evaluating automatic assessments of web-agent trajectories, including side effects and repetition cycles [148].

6.3 System-Boundary Constraints on Cross-Benchmark Comparability

Evaluation validity is the main pressure point for cumulative progress. GUI-agent scores depend on the agent, the model version, the prompt, the observation adapter, the action wrapper, and the environment state. A benchmark that exposes HTML or accessibility nodes may favor agents designed around structured metadata. A benchmark that permits screenshots only may favor visually trained models. A benchmark with deterministic resets and static tasks may understate failures that appear on live websites, dynamic mobile apps, or desktop workflows. These dependencies define the conditions under which each score should be interpreted in later comparisons.

We summarize the recurring threats to GUI-agent evaluation in Table 10. Observation mismatch and action-space mismatch are especially important because they connect directly to RQ2. Some agents receive screenshots only, while others receive DOM, XML, accessibility trees, OCR, element lists, or tool-generated descriptions. Benchmarks also differ in whether they permit coordinate clicks, element IDs, browser actions, device APIs, code execution, high-level tools, or takeover options. Two agents may both operate on the same task description while receiving different state representations and executing through different abstractions. In such cases, the benchmark measures a system stack, not an isolated model. This issue appears in comparisons between web agents using DOM-level information, mobile agents using XML hierarchies, and screenshot-only agents using visual grounding [48, 149, 235, 321]. We identify observation channels, action schemas, tool permissions, and environment instrumentation as first-order protocol details for reproducible comparison.

Oracle ambiguity is another source of weak comparability. Exact match works for tasks with a short textual answer or stable state label. Functional correctness is more appropriate for generated files, data-analysis workflows, or scientific tasks. Human judgment may be necessary for usability, risk, preference, and ambiguous outcomes. LLM-as-judge can scale trajectory evaluation, but its agreement with human labels and sensitivity to prompt design must be established [148, 214]. WorkArena and ScienceBoard illustrate why richer oracles are needed for knowledge work and

scientific workflows [54, 195]. A benchmark that judges only final page state may miss side effects, low-quality intermediate decisions, or dangerous actions that were reversed before completion.

Reproducibility is difficult because GUI environments are live software systems. Websites update layouts, mobile apps change UI flows, operating-system dialogs appear unpredictably, and third-party services alter authentication or rate limits. BrowserGym and WABER are useful because they focus attention on ecosystem support, reliability, and efficiency when evaluating web agents with existing benchmarks [43, 97]. AndroidLab and Windows Agent Arena also reflect a push toward systematic benchmarking infrastructure for mobile and OS agents [12, 272]. Still, reproducible comparison requires more than publishing tasks. It requires environment versioning, seed control, model-version reporting, action logs, trajectory replays, judge prompts, timeout rules, and failure categories for later diagnosis.

We treat cross-benchmark comparability cautiously. A web-navigation score, an Android task-success score, a desktop OSWorld score, and an offline grounding score each measure different bundles of capability and infrastructure. Leaderboards are useful for local progress within a benchmark family, while broader claims require triangulation across benchmarks with different observation channels, task horizons, domains, and risk profiles. For example, strong results on WebArena and VisualWebArena support claims about web interaction under those protocols. They do not establish mobile, desktop, cross-environment, or scientific-workflow generality without evidence from the corresponding benchmark families [99, 176, 195, 260, 270, 331].

Answer to RQ3. We find that GUI-agent evaluation is increasingly interactive and benchmark-rich while remaining dominated by task success. Among 252 system-oriented papers, 77 contribute evaluation resources, 164 combine online and offline evidence, and 194 report task success. The field has moved beyond static prediction without achieving uniform comparability. Scores remain conditioned by what the agent can observe and do. Wrappers, environment state, judge design, cost accounting, and hidden assistance further shape the result. We interpret scores as evidence about a complete system and protocol. Credible evaluation must connect outcomes to the process that produced them. Benchmarks should disclose what the agent can observe and do, how the environment is controlled, and how outcomes are judged. Reports should combine task success with the process, cost, risk, human burden, and reproducibility evidence required by the deployment claim. RQ4 examines whether these qualities are engineered across the lifecycle and sustained beyond an isolated benchmark run.

7 RQ4: GUI Agents through a Software Engineering Lens

We use RQ4 to examine whether the capabilities, architectures, and evaluations identified in RQ1–RQ3 are supported across the software lifecycle. We analyze how requirements are specified, architectures accommodate change, quality is tested, runtime behavior is observed, and deployment risks are governed. This perspective is necessary because GUI actions can change accounts, files, communications, permissions, or external services. Engineering concerns that appear secondary in offline evaluation become central in real workflows.

We organize the analysis from requirements and architecture to testing, operation, governance, and evolution in Fig. 9. Each stage reframes a capability question as a system question. Perception and grounding become requirements on observation quality and action safety. Planning becomes an architectural and verification concern. Benchmarks become one part of a broader testing strategy. Trajectories become potential runtime evidence for debugging, auditing, and regression analysis. Human intervention becomes an engineering mechanism for boundary-setting and accountability. We find that GUI-agent research has accumulated many relevant components, while several engineering practices remain underdeveloped as explicit system mechanisms across the lifecycle.

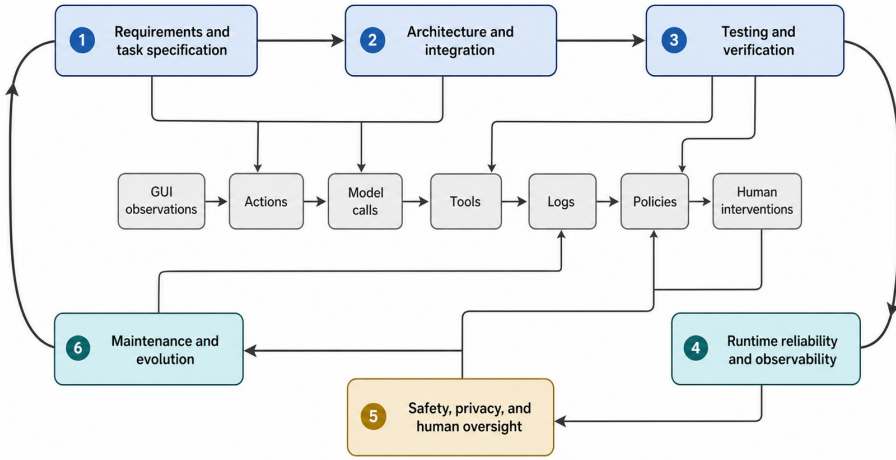


Fig. 9. Software-engineering lifecycle of GUI agents.

Table 11. Requirement signals.

Requirement signal	Papers	Share
GUI perception, localization, and action execution	276	84.4%
Cross-platform generalization and adaptation	134	41.0%
Planning, memory, and long-horizon task handling	118	36.1%
Low human intervention and autonomy	24	7.3%

7.1 Limited Testability of Capability Requirements

GUI-agent requirements span both task capability and operating quality. In the 327-paper SE-parseable subset, most papers emphasize perception, localization, and action execution. A smaller group addresses long-horizon control, memory, cross-platform generalization, and adaptation. These requirements motivate modular frameworks across mobile, web, and desktop environments [2, 117, 127, 235, 252, 301], as well as interface-grounding models such as SeeClick, ShowUI, and UI-TARS [38, 122, 171].

Our coding shows that GUI-agent requirements are dominated by the ability to observe and manipulate interfaces, followed by generalization and long-horizon task execution, as reported in Table 11. This distribution explains the architectural patterns identified in RQ2. Modular pipelines, planner-executor separation, memory loops, tool layers, and reflection mechanisms are engineering responses to requirements that cannot be satisfied by a single prediction step. The requirement profile changes with the platform. Mind2Web and WebArena emphasize task decomposition and navigation on the web [48, 331]. Mobile benchmarks add device state and application diversity, while desktop environments introduce files, windows, and operating-system control [12, 176, 219, 260, 272].

From a software-engineering perspective, however, requirement specification remains less mature than capability description. Many papers state that an agent should be autonomous or robust, while fewer define the contract that would make either claim testable. Autonomy may refer to reduced human correction or to longer unassisted tasks, which are not equivalent. Robustness may concern ordinary interface variation, hostile content, or transfer to a new domain. Efficiency is similarly ambiguous unless the paper distinguishes machine cost from interaction time and human effort.

Table 12. Quality-assurance signals.

Quality signal	Papers	Share
Benchmark or ablation testing	160	48.9%
User study, human labels, or real-environment testing	51	15.6%
Traditional testing gap explicitly noted	33	10.1%
Invalid, redundant, or malformed action handling	95	29.1%
Reflection, retry, correction, or replanning	86	26.3%
Human help, confirmation, or takeover for failures	17	5.2%

Without clearer requirement dimensions, systems with different assumptions can be compared under the same headline claim. We identify explicit task boundaries, allowed observation channels, action authority, risk level, intervention policy, and environment assumptions as necessary elements of a testable requirement specification.

7.2 Benchmark-Centered Testing and Underspecified Runtime Failures

Current GUI-agent quality assurance is heavily benchmark-centered. This is understandable because benchmarks provide a shared measurement substrate for a rapidly moving field, and RQ3 shows that interactive benchmarks have become one of the main drivers of progress. Yet the SE statistics reveal a mismatch between evaluation practice and software testing practice. Benchmark and ablation experiments are the main testing form in 160 papers, while only a small fraction explicitly discusses traditional testing gaps. Exception handling is also limited. Invalid-action handling and retry or replanning mechanisms appear in fewer than one third of the SE-parseable papers.

Our analysis of Table 12 indicates that the field has invested far more in capability benchmarks than in systematic verification. The difference matters because GUI-agent failures are often stateful. A wrong click may change the screen and invalidate the next observation. Repeated retries can create duplicate submissions, while an unsafe recovery step may hide side effects behind a correct final state. Recent work on GUI testing, web-agent evaluation, and reward-based trajectory assessment begins to expose these problems by analyzing failure modes, action validity, and process-level behavior [37, 97, 148, 280]. STEVE, AgentRewardBench, VeriSafe, and related work further show that verification and recovery must reason over trajectories, intermediate states, and external effects, with final success serving as only one signal [21, 104, 138].

Robustness research has also expanded toward adversarial and environmental stressors. Studies of malicious interfaces, GUI attacks, environmental injection, and web-agent security show that GUI agents can be misled through screen content, UI state, prompts embedded in pages, tool outputs, or environment manipulation [57, 120, 316, 318, 325]. These works are important because they move robustness beyond ordinary distribution shift. A software system that controls GUIs must handle benign variation and hostile manipulation within the same execution loop. This calls for testing at several levels. Component checks can isolate perception and grounding, while integration tests exercise planner-executor coordination. Saved trajectories support regression, and dynamic or adversarial environments provide system-level stress tests. Existing benchmarks provide valuable starting points. We find that deployed GUI agents need a broader testing pyramid that connects component checks, integration tests, trajectory regression, and system-level stress tests.

7.3 Gaps in Maintainability and Observability

Maintainability and observability are the clearest gaps when GUI agents are viewed as long-lived software systems. Many architectures are modular at the research-prototype level, and RQ2 identified frequent use of separate perception, planning, grounding, memory, and execution modules. However, maintainability is rarely articulated as an engineering goal. The SE extraction shows that

Table 13. Operational engineering signals.

Operational signal	Papers	Share
Modular or pluggable maintainability signal	25	7.6%
Missing maintenance process or API evolution	279	85.3%
Cross-platform or cross-task extension	101	30.9%
Data, training, or environment scale extension	96	29.4%
Module, tool, or model replacement extension	100	30.6%
Trajectory, history, log, or replay signal	54	16.5%
Missing monitoring, dashboard, or audit mechanism	251	76.8%

only 25 papers contain a modular or pluggable signal that can be interpreted as maintainability-related, while 279 papers lack discussion of maintenance processes, API evolution, or prompt maintenance. This gap is consequential because several interfaces can change independently. The model API and prompt may evolve without the target application, while permissions and browser or device wrappers follow their own versions. Benchmark environments introduce another source of drift during long-term operation.

We separate three related operational issues in Table 13. First, extensibility is discussed more often than maintainability. Cross-platform, data-scale, and module-replacement signals appear in roughly one third of the SE-parseable subset, often in systems that aim to generalize across apps, operating systems, or model backbones [127, 255, 300, 301]. Second, maintainability remains implicit. A modular diagram alone leaves open the need for versioned interfaces, compatibility tests, prompt migration rules, and a process for adapting to UI updates. Third, observability is underdeveloped as a runtime capability. Trajectories are often stored to train models or score benchmarks. Far fewer systems turn the same evidence into operational logs, replay tools, alerts, or audit trails.

Efficiency has a stronger presence, although it is usually measured as a performance attribute more than as an operational budget. Efficiency appears in several forms in the SE extraction. Runtime or interaction efficiency is discussed in 249 papers, while 192 address token, compute, or training cost. Another 98 discuss efficiency in parameters, data, or inference. MobileAgentBench, Octopus, DistRL, and EcoAgent illustrate different efficiency directions, including mobile execution cost, on-device models, distributed reinforcement learning, and economical agent operation [34, 219, 221, 294]. In deployment, these costs interact with reliability and oversight. A system that retries aggressively may improve success rate while increasing latency, API cost, and duplicated side effects. A system that asks for frequent human confirmation may reduce risk while increasing user burden. A system that compresses observations may reduce cost while removing evidence needed for debugging or safety checks. We frame efficiency as a lifecycle constraint that should be reported together with retries, action counts, model calls, context length, human-intervention frequency, and failure-recovery cost.

7.4 Governance Gaps between Risk Recognition and Control

Safety, security, privacy, and human oversight form the governance layer of GUI-agent engineering. The corpus recognizes these risks more often than it implements controls for them. Safety or security risks appear in 164 papers, while policy, permission, refusal, or defense mechanisms appear in 48. Privacy shows the same asymmetry, with 212 risk signals but only 33 mitigation signals. Human oversight is also limited. Runtime confirmation, help, takeover, or correction appears in 47 papers, and only three framework papers are explicitly marked as human-in-the-loop in the framework subset used for RQ2.

Our analysis of Table 14 shows that GUI-agent governance is widely recognized and remains thinly institutionalized. This is a pressing issue because GUI agents combine broad observation

with direct actuation. The risk depends on both platform and task. A web page can embed adversarial instructions, while a mobile app can expose private messages or credentials. Desktop workflows often span files and communications and may reach terminals or payment dialogs. GUI-specific safety work such as GUIGuard, SafeArena, VeriSafe, VeriOS, and EIA begins to address this problem by studying risk detection, unsafe actions, policy compliance, proactive verification, and environment-level attacks [104, 120, 211, 229, 253]. Environmental injection, prompt injection through web pages, and visual adversarial manipulation further show that the GUI surface is both the agent’s input channel and an attack channel [57, 107, 316].

Privacy requires an equally concrete treatment. Screenshots avoid exposing raw interface meta-data, but they can still contain credentials, private communications, or proprietary content. Conversely, DOM, XML, and accessibility representations may reveal hidden metadata or structured identifiers. The choice of observation medium therefore changes the privacy boundary without eliminating privacy risk. Systems such as VeriOS and human-centered studies of computer-use agents point toward finer-grained interaction policies, confirmation points, and user-aware control [214, 253]. MobileAgent and MobileGPT also illustrate how human-machine interaction, SOP integration, and memory-like mechanisms can support more controllable mobile agents [51, 106]. Still, the low HITL count suggests that human oversight is rarely designed as a first-class lifecycle mechanism in deployed systems.

Defense and alignment papers show what a more concrete governance layer could contain. LaSM studies pop-up attack defense through layer-wise scaling, while user-aligned web navigation frames task execution around ethical and personalized constraints [46, 278]. These works complement the risk-oriented benchmarks by showing that mitigation can be introduced inside the perception, decision, or policy layer. We argue that a GUI agent should expose where risk is detected, where policy is applied, and how a user or developer can inspect the resulting decision during later audits.

An SE-oriented governance model should begin with action authority and sensitive-data handling. Policy enforcement and user confirmation constrain that authority, while audit logs preserve accountability. These mechanisms should be tied to task risk. Low-risk information lookup may tolerate higher autonomy. External communication and irreversible changes require stricter gating, especially when credentials or payments are involved. We treat governance as a lifecycle mechanism connected to requirements, architecture, evaluation, logging, and maintenance. Otherwise, the system may appear safe in benchmark tasks while lacking the operational controls needed for deployment in real settings.

Answer to RQ4. We find that software-engineering concerns are present in GUI-agent research and unevenly covered across the lifecycle. Capability requirements and modular architectures are common. Testing beyond benchmarks and explicit verification are less developed, and the gap widens further for maintenance, observability, privacy controls, and systematic human oversight. Among 327 SE-parseable papers, benchmark or ablation testing appears in 160, invalid-action handling in 95, retry or replanning in 86, trajectory or replay signals in 54, and runtime human oversight in 47. By contrast, 279 lack a maintenance-process signal and 251 lack a monitoring or audit signal. The field is moving from model capability toward system construction without yet establishing lifecycle-managed deployment in practice. Closing this gap requires explicit requirements and testable execution contracts. Observable behavior and maintainable interfaces

Table 14. Governance signals.

Governance signal	Papers	Share
Safety or security risk	164	50.2%
Policy, permission, refusal, or defense	48	14.7%
High-risk action gating	45	13.8%
Sensitive-data privacy risk	212	64.8%
Privacy mitigation	33	10.1%
Runtime human oversight	47	14.4%
Explicit HITL in framework subset	3	2.1%

Table 15. Priority research gaps and opportunities.

Gap area	Evidence signal	Research opportunity
Dynamic dependability	95 invalid-action handling, 86 retry/replanning, 17 human recovery	Recovery contracts, stop rules, rollback, and state-aware execution
Lifecycle quality	160 benchmark/ablation tests, 33 testing-gap notes, 279 missing maintenance signals	GUI-agent test pyramid, trajectory regression, interface-version management
Evaluation comparability	Task success dominates, with protocols varying by observation, action space, oracle, and environment	Protocol disclosure, multi-objective leaderboards, reproducible benchmark infrastructure
Governance and deployment	164 safety risks, 48 defenses, 212 privacy risks, 33 privacy mitigations, 3 HITL frameworks	Permission systems, privacy controls, audit trails, risk-adaptive human oversight
Efficiency and sustainability	249 runtime-efficiency signals, 192 token/compute/training-cost signals	Cost budgets spanning model calls, actions, retries, latency, and supervision

must be supported by operational budgets and risk-aware governance throughout deployment and maintenance across the system lifecycle.

8 RQ5: Open Challenges and Research Opportunities

We use RQ5 to convert the preceding evidence into a research roadmap. The field combines rapid and unevenly mature growth with weak runtime-control boundaries, success-centered evaluation, and limited lifecycle support. These gaps require a connected system agenda spanning capability, evaluation, and deployment.

We organize the agenda into capability, quality, evaluation, and deployment layers in Fig. 10. The first concerns dependability in dynamic, heterogeneous, long-horizon environments. The second concerns lifecycle-centered testing, debugging, and maintenance. The third concerns continuous and risk-aware evaluation. The fourth concerns safe, efficient, and human-centered deployment. These opportunities are connected and involve trade-offs because GUI agents are closed-loop software systems. A stronger planner may increase task success while making failures harder to diagnose. Richer observations can improve grounding but expand the privacy boundary, and safer confirmation policies can increase user burden. We identify a need for multi-objective engineering methods that assess capability improvements within their operational context.

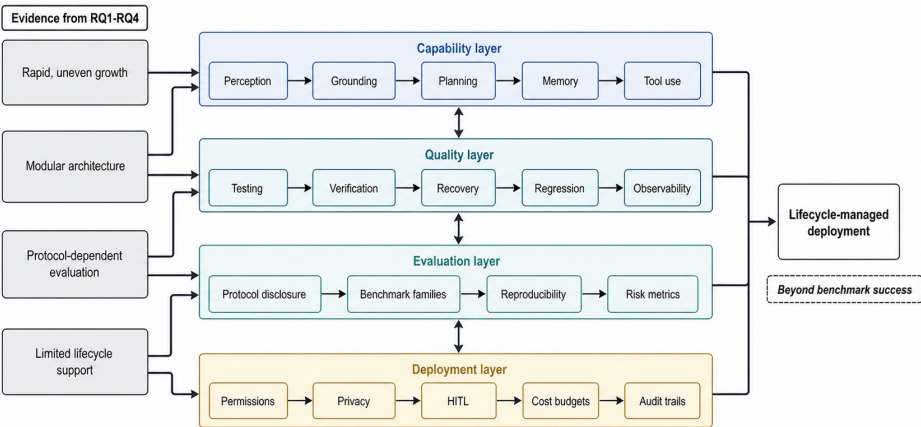


Fig. 10. Research roadmap for dependable GUI agents.

We link each proposed direction to evidence reported earlier in the survey in Table 15. The dependability row follows from the limited recovery and human-escalation signals in RQ2 and RQ4. The lifecycle and comparability rows respond to the testing, maintenance, and protocol gaps in RQ3 and RQ4. The governance row reflects the large difference between risk recognition and implemented controls, while the efficiency row treats cost as a cross-cutting deployment constraint.

8.1 Execution Contracts for Dependable Interaction

The first challenge is dependable execution under changing interfaces and uncertain feedback. Visual observations improve portability, while DOM, XML, and accessibility structures improve precision but introduce platform and privacy assumptions [38, 48, 149, 171, 235]. Neither representation prevents errors from accumulating across long tasks, hidden side effects, dialogs, loading states, and cross-application transitions. Interactive benchmarks and modular frameworks expose these pressures across web, mobile, and desktop environments [2, 176, 260, 301, 331]. The open problem is to make the full control loop dependable across perception, planning, execution, and recovery.

A promising direction is to define explicit execution contracts for GUI agents. Such contracts should define valid actions and acceptable state changes. They should also govern the transition from retry to stopping, confirmation, or recovery. Current reflection, retry, and replanning mechanisms remain unevenly represented across the literature. STEVE, VeriSafe, and related verification-oriented studies indicate that trajectory-level reasoning can help diagnose and repair execution failures [21, 104, 138]. We see an opportunity for transaction-aware GUI execution in which high-risk operations are checked before and after execution, reversible steps are preferred when possible, and irreversible operations are gated by risk-sensitive policies.

Another opportunity is hierarchical state abstraction. Many agents keep raw screenshots or action histories. Fewer maintain an explicit belief about task progress and the options available after failure. A dependable GUI agent should track both the visible interface state and the task-level commitment state. After filling a form, for example, the system should distinguish local edits from external submission and service confirmation. This distinction can prevent duplicate actions and make failures easier to explain. Research on memory-augmented, reflective, and planner-executor systems already points in this direction [41, 56, 250, 315]. We identify testable and auditable state abstraction as a necessary runtime mechanism.

8.2 Trajectory-Based Lifecycle Testing

The second challenge is to move GUI-agent quality assurance beyond benchmark execution. Benchmarks are necessary because they supply tasks, environments, and shared scores. They are insufficient as the sole quality mechanism because deployed agents must survive continuous change. Models and prompts evolve alongside wrappers, target interfaces, infrastructure, and user policy. Benchmark or ablation testing appears in 160 SE-parseable papers, while maintenance-process signals are missing from 279 papers. We use this gap to motivate a lifecycle quality stack that connects benchmark tasks to software testing, debugging, observability, and maintenance.

A GUI-agent test pyramid would include several layers. At the bottom, component tests should isolate perception and element localization from action serialization and permission checks. In the middle, integration tests should validate planner-grounder-executor coordination under controlled UI states. At the top, end-to-end tests should run realistic tasks in versioned environments. Existing resources already cover parts of this stack. ScreenSpot supports perception and localization checks, while Mind2Web provides evidence about action prediction and task decomposition [48, 111]. At the system level, BrowserGym, WABER, AndroidLab, and Windows Agent Arena contribute infrastructure for online evaluation [12, 43, 97, 272]. We see an opportunity to connect these resources into a coherent lifecycle process for deployed agents.

Debugging also needs better abstractions. A failed trajectory can mix model error with tool misuse or delayed environment feedback. Policy violations and oracle ambiguity may further obscure the cause. If the system stores only final success and screenshots, debugging becomes manual and fragile. Structured traces can turn those failures into reusable regression cases. The trace must preserve the environment and model configuration together with policy decisions and recovery attempts. AgentRewardBench and GUI testing work illustrate the value of process-level analysis and reward signals over trajectories [148, 280]. We propose treating every failed trajectory as a potential test artifact. Its metadata should support replay and minimization before the failure enters regression tracking.

Maintenance is the long-term form of the same problem. GUI agents depend on external software that changes outside the agent developer’s control. A website redesign, a mobile OS update, an API deprecation, or a model-provider change can invalidate assumptions without changing the agent code. Research opportunities include interface-version monitoring, prompt migration tests, model-version differential testing, wrapper compatibility suites, and deprecation-aware tool schemas. Modular architectures help because they expose replaceable components, but modularity becomes maintainability only when interfaces are versioned and tested. This is especially important for systems that advertise cross-platform generality, such as OS-Atlas, UFO, OS-Copilot, and PC Agent [127, 252, 255, 301].

8.3 Protocol, Risk, and Drift in Continuous Evaluation

The third challenge is to keep evaluation meaningful as models, tasks, and environments change. Interactive benchmarks now cover web, mobile, desktop, cross-environment, professional, and scientific workflows [54, 176, 195, 260, 270, 331]. We identify comparability, reproducibility, and risk coverage across benchmark families as the remaining evaluation problem.

Reproducibility requires more than releasing tasks. Interactive GUI environments contain websites, apps, operating systems, model APIs, authentication states, localization settings, external services, and time-sensitive content. A reported score can depend on hidden assistance, cached knowledge, action limits, retries, wrappers, browser settings, judge prompts, and environment versions. Future benchmarks should publish execution protocols as first-class artifacts. The protocol must define observation and action access together with tool permissions and interaction limits. Reproduction also requires model and environment versions, judge configuration, failure categories, and intervention rules. BrowserGym and WABER are useful examples because they treat benchmark infrastructure and evaluation reliability as part of the research object [43, 97]. The same infrastructure requirements apply to mobile, desktop, and cross-environment benchmarks.

Risk-aware evaluation should also become standard. A GUI agent that reaches a correct final state after exposing private data, clicking an unsafe confirmation, or repeatedly submitting a form should be judged differently from a safe trajectory. Safety and privacy benchmarks such as SafeArena, GUIGuard, and VeriSafe begin to supply such evidence by examining unsafe actions, risk detection, and verification mechanisms [104, 211, 229]. Environmental injection, adversarial web content, and malicious GUI manipulation further show that safety evaluation must be integrated with ordinary task evaluation as part of the same protocol [57, 120, 316].

Continuous evaluation is the practical extension of reproducibility. A benchmark result from a fixed date may become stale when the underlying website changes, the base model is updated, or the agent’s prompt is modified. Future leaderboards should therefore distinguish static benchmark snapshots from continuously monitored environments. They should also report uncertainty, drift, and environmental failures. In such a setting, a leaderboard would measure task success and performance stability under controlled updates.

8.4 Bounded Authority and Human Oversight in Deployment

The final challenge is deployment under real user, organizational, and regulatory constraints. GUI agents observe user interfaces and perform actions through them, which gives them unusual power compared with agents that only produce text. They can read private content and access accounts. Their actions may also alter files, contact other people, or confirm payments and permissions. Privacy and security are widely recognized, while defense, mitigation, and human oversight mechanisms appear far less often than risk discussion. We identify a need for deployment architectures that combine capability with least privilege, privacy protection, auditability, and user control.

Permission design is a central problem. Current agents often operate with broad screen access and broad action authority. Future systems should expose finer-grained authority. Read-only observation and reversible editing should be separated from external communication, credential use, and irreversible state changes. Each capability should have a policy and logging requirement. Work on GUI safety, proactive verification, and policy compliance provides early building blocks for this direction [104, 211, 229, 253]. We propose making permissions part of the agent architecture and evaluation protocol so that reported performance is interpreted together with action authority.

Privacy engineering is equally important. Screenshot-based agents, DOM-based agents, and accessibility-based agents expose different kinds of sensitive information. A deployment-ready system should minimize what is captured, mask or redact sensitive regions when possible, keep local processing local when appropriate, and record data lifecycles for logs and trajectories. Research on attacks and environmental manipulation shows that privacy and security interact because malicious UI content can influence the agent while it observes private user state [107, 318, 325]. We propose evaluating privacy leakage, prompt injection, and action risk together.

Human-centered deployment requires a richer account of human participation. We find sparse HITL evidence in the corpus. Only three framework papers are explicitly marked as human-in-the-loop, although many real scenarios require confirmation, correction, takeover, or preference judgment. Existing work on computer-use agents, VeriOS, MobileAgent, and MobileGPT suggests that users can participate through confirmation, SOP integration, repair, and oversight [51, 106, 214, 253]. We propose interaction policies for task setup, mid-execution confirmation, failure recovery, and post-hoc audit. The key question is where human judgment adds safety, accountability, or preference alignment without making the system unusable in routine practice.

Efficiency should be evaluated under the same deployment lens. GUI-agent cost has both computational and operational components. Model calls and latency capture only part of it. Retries, environment setup, human supervision, logging, and evaluation maintenance can dominate in long-running deployments. MobileAgentBench, Octopus, DistRL, and EcoAgent point to several efficiency paths, including benchmarked mobile cost, compact/on-device models, reinforcement learning efficiency, and economical agent operation [34, 219, 221, 294]. We identify cost budgets that include machine and human costs as a research priority. A safe and useful agent should report what task completion required. At minimum, this includes model calls, actions, retries, elapsed time, and human confirmation, with sensitive observations reported when relevant.

Answer to RQ5. We identify the transformation of benchmark-performing GUI agents into dependable, maintainable, secure, and deployable systems as the central challenge. Our synthesis yields four priorities. Future work should establish dependable execution under dynamic state, lifecycle-centered testing and maintenance, continuous and risk-aware evaluation, and governed human-centered deployment with explicit cost budgets. These priorities are interdependent. Execution contracts shape recovery and evaluation, observability supports maintenance and auditing, and permission or privacy controls affect both capability and user burden. We conclude that GUI-agent research should treat capability, quality, evaluation, and governance as one lifecycle problem.

9 Discussion

Across the five research questions, we find that the relevant unit of analysis is the complete interactive software system. Model behavior is mediated by observation and action interfaces and further shaped by memory, runtime checks, environment controllers, evaluation protocols, and user-facing policies. An improvement to visual grounding can raise task success while leaving recovery or action authority unchanged. A stronger planner can improve long-horizon control while increasing latency and making failures harder to localize. GUI-agent progress should consequently be interpreted through the architecture and operating conditions that produce observed behavior.

Observation design is the first system boundary that cuts across model development, architecture, evaluation, and deployment. Screenshots provide a common interface across web, mobile, and desktop settings, yet they omit stable element identity, hidden state, and explicit interaction semantics. DOM, XML, and accessibility representations expose more structure while introducing platform dependence and access to potentially sensitive metadata. OmniParser, SeeClick, ShowUI, and UI-TARS illustrate screenshot-centered perception and action, whereas Mind2Web and AutoDroid show how structured web or mobile state can support grounding and execution [38, 48, 122, 149, 171, 235]. We find no single representation that dominates across all operating conditions. Papers should state the observation channels available to the agent and explain how those channels affect portability, privacy, and diagnostic access.

Agent construction and benchmark construction also remain closely coupled. WebArena, VisualWebArena, AndroidWorld, OSWorld, WorkArena, ScienceBoard, and CRAB have expanded evaluation from static prediction to interactive tasks across major platforms and professional workflows [54, 99, 176, 195, 260, 270, 331]. These environments expose capability gaps that offline datasets cannot capture. They also determine observation privileges, action abstractions, interaction budgets, reset policies, and completion oracles. We therefore interpret benchmark scores as evidence about an agent stack under a specific protocol. Cross-benchmark claims require explicit reporting of wrappers, retries, tool access, environment versions, judge settings, and human assistance. BrowserGym, WABER, AndroidLab, and Windows Agent Arena provide useful foundations by treating evaluation infrastructure and reliability as part of the research contribution [12, 43, 97, 272].

The transition from action prediction to closed-loop control changes the engineering problem. GUI agents modify external state, and some changes persist after a task fails. Memory, verification, reflection, and replanning help manage this uncertainty, although their presence does not establish dependable recovery. STEVE, VeriSafe, AgentRewardBench, and GUI testing studies begin to make trajectories and intermediate outcomes explicit objects of analysis [104, 138, 148, 280]. We find that deployed systems need execution contracts defining valid actions, expected state transitions, retry limits, rollback conditions, and escalation rules. Structured trajectories should record the model and environment configuration together with actions, policy decisions, failures, and recovery attempts. The same evidence can support debugging, regression testing, continuous evaluation, and audit.

Model and framework design create a related architectural choice. GUI-native systems such as UI-TARS, CogAgent, ShowUI, and OS-ATLAS demonstrate the value of task-specific data and unified perception-action learning [79, 122, 171, 255]. Framework-oriented systems such as AppAgent, UFO, OS-Copilot, CRADLE, and Agent S expose state construction, planning, execution, and verification as separate responsibilities [2, 117, 198, 252, 301]. Native models can reduce coordination overhead, while modular systems provide clearer locations for testing, replacement, policy enforcement, and failure diagnosis. The corpus points toward hybrid systems in which learned models provide perception, reasoning, and action priors and explicit runtime modules retain authority over verification, permissions, logging, and recovery. This design becomes maintainable only when module interfaces, prompts, model versions, and environment adapters are versioned and tested.

Deployment adds coupled constraints involving safety, privacy, usability, and efficiency. Stronger confirmation policies can reduce harmful actions while increasing user effort. Privacy filters can limit exposure while removing information needed for grounding or recovery. Detailed logs improve auditability while creating sensitive data that must be governed. GUIGuard, SafeArena, VeriSafe, and environmental-injection studies demonstrate that risk can originate in model behavior, interface content, or excessive action authority [57, 104, 120, 211, 229, 316]. VeriOS, MobileAgent, MobileGPT, and human-centered studies further indicate that confirmation, takeover, and preference judgment should be designed as runtime mechanisms [51, 106, 214, 253]. We interpret autonomy as bounded delegation with explicit observation access, action authority, and accountability. Operational cost should include model calls, actions, retries, latency, environment recovery, logging, and human supervision.

These findings have direct implications for research and practice. Researchers should align each claim with the evidence needed to support it. Grounding models may emphasize target and action correctness, whereas enterprise or scientific agents require additional evidence on permissions, recovery, privacy, oracle validity, and reproducibility. System builders should define testable interfaces for observation, state construction, planning, execution, verification, and intervention. Benchmark designers should version tasks and environments and publish protocols that cover observation access, action permissions, resets, timeouts, authentication, judging, and intervention. Practitioners should separate read-only and reversible work from external communication, credential use, and irreversible state changes. We identify system boundaries, lifecycle evidence, action authority, recovery contracts, and human oversight as the shared vocabulary needed to compare GUI-agent research across vision-language modeling, NLP, HCI, mobile computing, security, and software engineering.

10 Related Surveys and Positioning

Prior surveys have organized the rapidly expanding terminology around GUI agents, multimodal mobile agents, and computer-use systems. Broad reviews use capabilities such as perception, instruction understanding, planning, grounding, action, and benchmark construction as their main analytical dimensions [157, 201, 220, 299]. They establish the pipeline shared by many agents and explain how GUI interaction relates to multimodal foundation models and interactive evaluation.

Specialized surveys narrow this view by platform or technical concern. Mobile-agent reviews emphasize phone interaction, Android representations, and application automation [125, 240], while OS-agent reviews focus on desktop control and general computer use [82]. Trustworthiness- and reinforcement-learning-oriented surveys examine safety, robustness, alignment, and sequential decision-making strategies [110, 182]. Together, these studies demonstrate the field's breadth across NLP, computer vision, HCI, mobile computing, security, reinforcement learning, and systems.

These perspectives are complementary, but their primary unit of analysis is usually a capability, model, platform, benchmark, or risk category. They provide less synthesis of how these elements interact across requirements, architecture, testing, operation, maintenance, and governance. The present survey addresses this gap by treating the GUI agent and its surrounding runtime as the unit of engineering analysis.

The distinctive contribution of this survey is its lifecycle-centered software-engineering perspective on GUI agents. GUI agents are analyzed as systems whose requirements and interfaces connect many components and dependencies. Execution contracts, tests, and logs determine how those systems are assured, while permissions and maintenance shape their operation over time. This perspective changes the interpretation of familiar research artifacts. A benchmark defines an execution protocol, an action space defines an interface contract, a trajectory can serve as

debugging and audit evidence, and a model update creates a maintenance event. The literature contains these connections, but rarely synthesizes them as one lifecycle problem.

The five research questions operationalize this positioning. They move from field mapping and architectural responsibility to evaluation evidence and lifecycle coverage, then integrate the resulting gaps into a deployment-oriented research agenda. This sequence distinguishes the survey from reviews that stop at capability taxonomies or benchmark comparison alone in prior work.

The survey also differs from a conventional GUI testing review. GUI testing has a mature literature on test generation, event exploration, regression, crash detection, and record-and-replay validation. We include this work only when it directly contributes to agent perception, decision making, execution, evaluation, safety, or lifecycle engineering. This boundary preserves GUI agents as the object of study while allowing established testing concepts to inform their engineering analysis.

11 Threats to Validity

The first threat concerns corpus coverage and evidence maturity. GUI-agent research evolves rapidly and uses inconsistent terminology across several communities. We combined keyword search, snowballing, deduplication, and manual screening to reduce omission risk, although the resulting corpus cannot constitute an exhaustive census. The high proportion of preprints also means that methods and claims may change after review. Because the 2026 coverage ends in April, we interpret the 336-paper corpus as a structured snapshot of the field at that time.

The second threat concerns coding and interpretation. The structured labels simplify papers that often combine several contributions, and lifecycle concerns such as maintainability or human oversight may remain implicit in system descriptions. We use multi-label and functional coding to reduce this loss and treat absent signals as a lack of explicit evidence under the coding scheme. The software-engineering synthesis also reflects analytical judgment and may give less attention to fine-grained model or data optimization. Internal-coherence checks improve consistency, although they do not provide independently measured inter-rater agreement from multiple coders.

The third threat concerns comparability across analytical subsets and evaluation protocols. RQ1 uses all 336 papers, RQ2 uses 145 framework papers for module statistics, RQ3 uses 252 system-oriented papers, and RQ4 uses 327 SE-parseable records. Benchmark results also depend on changing models, observation access, action spaces, wrappers, environments, and oracles. We report the relevant denominators and protocol assumptions with each analysis and avoid direct comparisons across incompatible subsets or benchmark settings.

12 Conclusion

GUI agents have evolved from isolated interface-understanding and action-prediction models into closed-loop systems that operate across web, mobile, desktop, and cross-platform environments. This survey analyzed that evolution from a software-engineering perspective using 336 papers published or posted between 2018 and April 2026. The five research questions connected the field's growth to its architecture and evaluation evidence. They then examined lifecycle coverage and future opportunities. The results show rapid expansion alongside incomplete maturity. Frameworks and benchmarks dominate the recent literature, screenshots provide a common observation channel, and modular perceive-decide-execute loops have become a recurring architecture. Evaluation is also more interactive and realistic than earlier static prediction tasks. However, its evidence remains centered on task success and conditioned by the surrounding protocol. Across the lifecycle, testing beyond benchmarks and explicit recovery contracts remain limited. Maintainability, observability, privacy controls, auditability, and systematic human oversight are also underdeveloped. These findings establish the central conclusion of the survey. GUI-agent capability and software quality cannot be developed independently. Dependable progress begins with explicit execution contracts

and lifecycle-centered testing. It also requires reproducible, risk-aware evaluation and observable runtime behavior. Permission, privacy, operational cost, and human governance must be designed as part of the same system. Future research should evaluate GUI agents as complete system stacks and report the boundaries under which their capabilities hold. As GUIs remain an important interface to consequential digital work, building agents that operate through them dependably, securely, and accountably will remain a shared agenda for AI and software-engineering research communities.

References

- [1] Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. 2024. Agent-E: From Autonomous Web Navigation to Foundational Design Principles in Agentic Systems. *arXiv:2407.13032* [cs.AI] [doi:10.48550/arXiv.2407.13032](https://doi.org/10.48550/arXiv.2407.13032)
- [2] Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. 2024. Agent S: An Open Agentic Framework that Uses Computers Like a Human. *arXiv:2410.08164* [cs.AI] [doi:10.48550/arXiv.2410.08164](https://doi.org/10.48550/arXiv.2410.08164)
- [3] Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. Agent S2: A Compositional Generalist-Specialist Framework for Computer Use Agents. *arXiv:2504.00906* [cs.AI] [doi:10.48550/arXiv.2504.00906](https://doi.org/10.48550/arXiv.2504.00906)
- [4] Pranjal Aggarwal and Sean Welleck. 2025. Programming with Pixels: Computer-Use Meets Software Engineering. *arXiv:2502.18525* [cs.SE] [doi:10.48550/arXiv.2502.18525](https://doi.org/10.48550/arXiv.2502.18525)
- [5] Jaewoo Ahn, Junseo Kim, Heeseung Yun, Jaehyeon Son, Dongmin Park, Jaewoong Cho, and Gunhee Kim. 2025. FlashAdventure: A Benchmark for GUI Agents Solving Full Story Arcs in Diverse Adventure Games. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 23365–23395. [doi:10.18653/v1/2025.emnlp-main.1192](https://doi.org/10.18653/v1/2025.emnlp-main.1192)
- [6] Neeraj Anand, Rishabh Jain, Sohan Patnaik, Balaji Krishnamurthy, and Mausoom Sarkar. 2025. AFRAgent : An Adaptive Feature Renormalization Based High Resolution Aware GUI agent. *arXiv:2512.00846* [cs.CV] [doi:10.48550/arXiv.2512.00846](https://doi.org/10.48550/arXiv.2512.00846)
- [7] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. ScreenAI: A Vision-Language Model for UI and Infographics Understanding. *arXiv:2402.04615* [cs.CV] [doi:10.48550/arXiv.2402.04615](https://doi.org/10.48550/arXiv.2402.04615)
- [8] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. 2021. UIBert: Learning Generic Multimodal Representations for UI Understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. 1705–1712. [doi:10.24963/ijcai.2021/235](https://doi.org/10.24963/ijcai.2021/235)
- [9] Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. In *Advances in Neural Information Processing Systems 37*. 12461–12495. [doi:10.52202/079017-0397](https://doi.org/10.52202/079017-0397)
- [10] Hao Bai, Yifei Zhou, Li Li, Sergey Levine, and Aviral Kumar. 2025. Digi-Q: Learning VLM Q-Value Functions for Training Device-Control Agents. In *International Conference on Learning Representations*. https://proceedings.iclr.cc/paper_files/paper/2025/hash/519abe71ee55aac4fe821bbd731c6645-Abstract-Conference.html
- [11] Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. 2024. WorkArena++: Towards Compositional Planning and Reasoning-based Common Knowledge Work Tasks. In *Advances in Neural Information Processing Systems 37*. 5996–6051. [doi:10.52202/079017-0195](https://doi.org/10.52202/079017-0195)
- [12] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Buckner, Lawrence Jang, and Zack Hui. 2024. Windows Agent Arena: Evaluating Multimodal OS Agents at Scale. *arXiv:2409.08264* [cs.AI] [doi:10.48550/arXiv.2409.08264](https://doi.org/10.48550/arXiv.2409.08264)
- [13] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. 2022. A Dataset for Interactive Vision-Language Navigation with Unknown Command Feasibility. *Lecture Notes in Computer Science* (2022), 312–328. [doi:10.1007/978-3-031-20074-8_18](https://doi.org/10.1007/978-3-031-20074-8_18)
- [14] Andrea Burns, Kate Saenko, and Bryan Plummer. 2024. Tell Me What’s Next: Textual Foresight for Generic UI Representations. In *Findings of the Association for Computational Linguistics ACL 2024*. 4590–4611. [doi:10.18653/v1/2024.findings-acl.273](https://doi.org/10.18653/v1/2024.findings-acl.273)
- [15] Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao, Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. 2024. Spider2-V: How Far Are Multimodal Agents From Automating Data Science and Engineering Workflows?. In *Advances in Neural Information Processing Systems 37*. 107703–107744. [doi:10.52202/079017-3421](https://doi.org/10.52202/079017-3421)
- [16] Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2024. Web Agents with World Models: Learning and Leveraging Environment

- Dynamics in Web Navigation. arXiv:2410.13232 [cs.CL] doi:10.48550/arXiv.2410.13232
- [17] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2025. AMEX: Android Multi-annotation Expo Dataset for Mobile GUI Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*. 2138–2156. doi:10.18653/v1/2025.findings-acl.110
 - [18] Yuxiang Chai, Shunye Tang, Han Xiao, Weifeng Lin, Hanhao Li, Jiayu Zhang, Liang Liu, Pengxiang Zhao, Guangyi Liu, Guozhi Wang, Shuai Ren, Rongduo Han, Haining Zhang, Siyuan Huang, and Hongsheng Li. 2025. A3: Android Agent Arena for Mobile GUI Agents with Essential-State Procedural Evaluation. arXiv:2501.01149 [cs.AI] doi:10.48550/arXiv.2501.01149
 - [19] Iason Chaimalas, Arnas Vyšniauskas, and Gabriel Brostow. 2025. Explorer: Robust Collection of Interactable GUI Elements. arXiv:2504.09352 [cs.HC] doi:10.48550/arXiv.2504.09352
 - [20] Rajat Chawla, Adarsh Jha, Muskaan Kumar, Mukunda NS, and Ishaan Bhola. 2024. GUIDE: Graphical User Interface Data for Execution. arXiv:2404.16048 [cs.HC] doi:10.48550/arXiv.2404.16048
 - [21] Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. 2025. GUI-Shepherd: Reliable Process Reward and Verification for Long-Sequence GUI Tasks. arXiv:2509.23738 [cs.AI] doi:10.48550/arXiv.2509.23738
 - [22] Chen Chen, Jiawei Shao, Dakuan Lu, Haoyi Hu, Xiangcheng Liu, Hantao Yao, and Wu Liu. 2026. GUI-Eyes: Tool-Augmented Perception for Visual Grounding in GUI Agents. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 35 (2026), 29350–29358. doi:10.1609/aaai.v40i35.40175
 - [23] Chiyu Chen, Xinhao Song, Yunkai Chai, Yang Yao, Haodong Zhao, Lijun Li, Jie Li, Yan Teng, Gongshen Liu, and Yingchun Wang. 2025. GhostEI-Bench: Do Mobile Agents Resilience to Environmental Injection in Dynamic On-Device Environments? arXiv:2510.20333 [cs.CR] doi:10.48550/arXiv.2510.20333
 - [24] Chaoran Chen, Zhiping Zhang, Bingcan Guo, Shang Ma, Ibrahim Khalilov, Simret A Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. 2025. The Obvious Invisible Threat: LLM-Powered GUI Agents' Vulnerability to Fine-Print Injections. arXiv:2504.11281 [cs.HC] doi:10.48550/arXiv.2504.11281
 - [25] Chaoran Chen, Zhiping Zhang, Ibrahim Khalilov, Bingcan Guo, Simret A Gebreegziabher, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. 2025. Toward a Human-Centered Evaluation Framework for Trustworthy LLM-Powered GUI Agents. arXiv:2504.17934 [cs.HC] doi:10.48550/arXiv.2504.17934
 - [26] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2024. GUI-World: A Video Benchmark and Dataset for Multimodal GUI-oriented Understanding. arXiv:2406.10819 [cs.CV] doi:10.48550/arXiv.2406.10819
 - [27] Gongwei Chen, Lirong Jie, Lexiao Zou, Weili Guan, Miao Zhang, and Liqiang Nie. 2025. Enhancing GUI Agent with Uncertainty-Aware Self-Trained Evaluator. In *Advances in Neural Information Processing Systems* 38. https://proceedings.neurips.cc/paper_files/paper/2025/hash/d067d16e3e5fe8fa8a3e62909907659a-Abstract-Conference.html
 - [28] Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. 2025. Less is More: Empowering GUI Agent with Context-Aware Simplification. (2025), 5901–5911. doi:10.1109/iccv51701.2025.00558
 - [29] Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye Hao, Jun Wang, and Kun Shao. 2024. SPA-Bench: A Comprehensive Benchmark for SmartPhone Agent Evaluation. arXiv:2410.15164 [cs.AI] doi:10.48550/arXiv.2410.15164
 - [30] Liang Chen, Haozhe Zhao, Yinzhen Huang, Yang Luo, Tsekai Lin, Weichu Xie, Ruoyu Wu, Peiyi Wang, Runxin Xu, Ming Wu, and Baobao Chang. 2025. CCAgent: Coordinating Collaborative Data Scaling for Operating System Agents via Web3. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 280–290. doi:10.1145/3746252.3761392
 - [31] Qi Chen, Dilepa Pitawela, Chongyang Zhao, Gengze Zhou, Hsiang-Ting Chen, and Qi Wu. 2024. WebVLN: Vision-and-Language Navigation on Websites. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 2 (2024), 1165–1173. doi:10.1609/aaai.v38i2.27878
 - [32] Ruihan Chen, Qiming Li, Xiaocheng Feng, Weihong Zhong, Xiaoliang Yang, Yuxuan Gu, Zekun Zhou, Yunfei Lu, Haoyu Ren, Kun Chen, Dandan Tu, and Bing Qin. 2025. MPR-GUI: Benchmarking and Enhancing Multilingual Perception and Reasoning in GUI Agents. arXiv:2512.00756 [cs.AI] doi:10.48550/arXiv.2512.00756
 - [33] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025. GUICourse: From General Vision Language Model to Versatile GUI Agent. (2025), 21936–21959. doi:10.18653/v1/2025.acl-long.1065
 - [34] Wei Chen and Zhiyuan Li. 2024. Octopus v2: On-device language model for super agent. arXiv:2404.01744 [cs.CL] doi:10.48550/arXiv.2404.01744

- [35] Wei Chen, Zhiyuan Li, Zhen Guo, and Yikang Shen. 2026. Octo-Planner: On-Device Language Model for Planner-Action Agents. *Lecture Notes in Computer Science* (2026), 141–156. doi:10.1007/978-3-032-18011-7_9
- [36] Weizhi Chen, Ziwei Wang, Leyang Yang, Sheng Zhou, Xiaoxuan Tang, Jiajun Bu, Yong Li, and Wei Jiang. 2025. PG-Agent: An Agent Powered by Page Graph. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 6878–6887. doi:10.1145/3746027.3755189
- [37] Yurun Chen, Xueyu Hu, Keting Yin, Juncheng Li, and Shengyu Zhang. 2025. Evaluating the Robustness of Multimodal Agents Against Active Environmental Injection Attacks. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 11648–11656. doi:10.1145/3746027.3755646
- [38] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. (2024), 9313–9332. doi:10.18653/v1/2024.acl-long.505
- [39] Pengzhou Cheng, Haowen Hu, Zheng Wu, Zongru Wu, Tianjie Ju, Daizong Ding, Zhuosheng Zhang, and Gongshen Liu. 2025. Hidden Ghost Hand: Unveiling Backdoor Vulnerabilities in MLLM-Powered Mobile GUI Agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. 7781–7805. doi:10.18653/v1/2025.findings-emnlp.411
- [40] Pengzhou Cheng, Zheng Wu, Zongru Wu, Tianjie Ju, Aston Zhang, Zhuosheng Zhang, and Gongshen Liu. 2025. OS-Kairos: Adaptive Interaction for MLLM-Powered GUI Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*. 6701–6725. doi:10.18653/v1/2025.findings-acl.348
- [41] Weihua Cheng, Junming Liu, Yifei Sun, Botian Shi, Yirong Chen, and Ding Wang. 2025. MGA: Memory-Driven GUI Agent for Observation-Centric Interaction. arXiv:2510.24168 [cs.AI] doi:10.48550/arXiv.2510.24168
- [42] Ziming Cheng, Zhiyuan Huang, Juntong Pan, Zhaohui Hou, and Mingjie Zhan. 2025. Navi-plus: Managing Ambiguous GUI Navigation Tasks with Follow-up Questions. arXiv:2503.24180 [cs.CV] doi:10.48550/arXiv.2503.24180
- [43] Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omid Shayegan, Lawrence Keunho Jang, Xing Han Lü, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2024. The BrowserGym Ecosystem for Web Agent Research. arXiv:2412.05467 [cs.LG] doi:10.48550/arXiv.2412.05467
- [44] Filippos Christianos, Georgios Papoudakis, Thomas Coste, Jianye Hao, Jun Wang, and Kun Shao. 2024. Lightweight Neural App Control. arXiv:2410.17883 [cs.AI] doi:10.48550/arXiv.2410.17883
- [45] Gaole Dai, Shiqi Jiang, Ting Cao, Yuanchun Li, Yuqing Yang, Rui Tan, Mo Li, and Lili Qiu. 2025. Advancing Mobile GUI Agents: A Verifier-Driven Approach to Practical Deployment. arXiv:2503.15937 [cs.AI] doi:10.48550/arXiv.2503.15937
- [46] Preetam Prabhu Srikanth Dammu. 2025. Towards Ethical and Personalized Web Navigation Agents: A Framework for User-Aligned Task Execution. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*. 1074–1076. doi:10.1145/3701551.3707420
- [47] Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Liujianfeng Liujianfeng, Ang Li, Jian Luan, Bin Wang, Rui Yan, and Shuo Shang. 2024. Mobile-Bench: An Evaluation Benchmark for LLM-based Mobile Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8813–8831. doi:10.18653/v1/2024.acl-long.478
- [48] Xiang Deng, Yu Gu, Boyuan Zhang, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems* 36. 28091–28114. doi:10.52202/075280-1220
- [49] Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024. On the Multi-turn Instruction Following for Conversational Web Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8795–8812. doi:10.18653/v1/2024.acl-long.477
- [50] Lei Ding, Jeshwanth Bheemanpally, and Yi Zhang. 2024. Enhancing Mobile "How-to" Queries with Automated Search Results Verification and Reranking. arXiv:2404.08860 [cs.IR] doi:10.48550/arXiv.2404.08860
- [51] Tinghe Ding. 2024. MobileAgent: enhancing mobile control via human-machine interaction and SOP integration. arXiv:2401.04124 [cs.HC] doi:10.48550/arXiv.2401.04124
- [52] Jinhan Dong, Lei Jin, Zhihong Zhang, Wei Tang, Runqing Zhang, Liqiang Xu, and Junliang Xing. 2025. MT-Agent: Constructing a GUI Agent via Modality Enhancement and Text-Guided Fusion. *IEEE Internet of Things Journal* (2025), 1. doi:10.1109/jiot.2025.3600573
- [53] Lingzhong Dong, Ziqi Zhou, Shuaibo Yang, Haiyue Sheng, Pengzhou Cheng, Zongru Wu, Zheng Wu, Gongshen Liu, and Zhuosheng Zhang. 2025. Say One Thing, Do Another? Diagnosing Reasoning-Execution Gaps in VLM-Powered Mobile-Use Agents. arXiv:2510.02204 [cs.CL] doi:10.48550/arXiv.2510.02204
- [54] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? arXiv:2403.07718 [cs.LG] doi:10.48550/arXiv.2403.07718
- [55] Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. 2026. Test-Time Reinforcement Learning for GUI Grounding via Region Consistency. *Proceedings of the AAAI Conference*

- on *Artificial Intelligence* 40, 36 (2026), 30593–30601. doi:10.1609/aaai.v40i36.40314
- [56] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks. arXiv:2503.09572 [cs.CL] doi:10.48550/arXiv.2503.09572
 - [57] Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. 2025. WASP: Benchmarking Web Agent Security Against Prompt Injection Attacks. arXiv:2504.18575 [cs.CR] doi:10.48550/arXiv.2504.18575
 - [58] Yue Fan, Lei Ding, Ching-Chen Kuo, Shan Jiang, Yang Zhao, Xinze Guan, Jie Yang, Yi Zhang, and Xin Eric Wang. 2024. Read Anywhere Pointed: Layout-aware GUI Screen Reading with Tree-of-Lens Grounding. arXiv:2406.19263 [cs.CL] doi:10.48550/arXiv.2406.19263
 - [59] Yue Fan, Handong Zhao, Ruiyi Zhang, Yu Shen, Xin Eric Wang, and Gang Wu. 2025. 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*. 33249–33266. doi:10.18653/v1/2025.emnlp-main.1688
 - [60] Moghis Fereidouni, Adib Mosharrof, and A. B. Siddique. 2024. Grounded Language Agent for Product Search via Intelligent Web Interactions. arXiv:2404.10887 [cs.CL] doi:10.48550/arXiv.2404.10887
 - [61] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2023. Multimodal Web Navigation with Instruction-Finetuned Foundation Models. arXiv:2305.11854 [cs.LG] doi:10.48550/arXiv.2305.11854
 - [62] Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. 2023. Exposing Limitations of Language Model Agents in Sequential-Task Compositions on the Web. arXiv:2311.18751 [cs.LG] doi:10.48550/arXiv.2311.18751
 - [63] Difei Gao, Siyuan Hu, Zechen Bai, Qinghong Lin, and Mike Zheng Shou. 2024. AssistEditor: Multi-Agent Collaboration for GUI Workflow Automation in Video Creation. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 11255–11257. doi:10.1145/3664647.3684998
 - [64] Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Donzong Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, Hengxu Wang, Luwei Zhou, and Mike Zheng Shou. 2024. AssistGUI: Task-Oriented PC Graphical User Interface Automation. (2024), 13289–13298. doi:10.1109/cvpr52733.2024.01262
 - [65] Longxi Gao, Li Zhang, Shihe Wang, Pengzhi Gao, Wei Liu, Jian Luan, Shangguang Wang, Yuanchun Li, and Mengwei Xu. 2024. MobileViews: A Million-scale and Diverse Mobile GUI Dataset. arXiv:2409.14337 [cs.HC] doi:10.48550/arXiv.2409.14337
 - [66] Divyansh Garg, Shaun VanWeelden, Diego Caples, Andis Draguns, Nikil Ravi, Pranav Putta, Naman Garg, Tomas Abraham, Michael Lara, Federico Lopez, James Liu, Atharva Gundawar, Prannay Hebbar, Youngchul Joo, Jindong Gu, Charles London, Christian Schroeder de Witt, and Sumeet Motwani. 2025. REAL: Benchmarking Autonomous Agents on Deterministic Simulations of Real Websites. arXiv:2504.11543 [cs.AI] doi:10.48550/arXiv.2504.11543
 - [67] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. arXiv:2410.05243 [cs.AI] doi:10.48550/arXiv.2410.05243
 - [68] 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. 2024. Is Your LLM Secretly a World Model of the Internet? Model-Based Planning for Web Agents. arXiv:2411.06559 [cs.AI] doi:10.48550/arXiv.2411.06559
 - [69] Yanchu Guan, Dong Wang, Zhixuan Chu, Shiyu Wang, Feiyue Ni, Ruihua Song, and Chenyi Zhuang. 2024. Intelligent Agents with LLM-based Process Automation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5018–5027. doi:10.1145/3637528.3671646
 - [70] Ziyi Guan, Jason Chun Lok Li, Zhijian Hou, Pingping Zhang, Donglai Xu, Yuzhi Zhao, Mengyang Wu, Jinpeng Chen, Thanh-Toan Nguyen, Pengfei Xian, Wenao Ma, Shengchao Qin, Graziano Chesi, and Ngai Wong. 2025. KG-RAG: Enhancing GUI Agent Decision-Making via Knowledge Graph-Driven Retrieval-Augmented Generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 5396–5405. doi:10.18653/v1/2025.emnlp-main.274
 - [71] Xiangwu Guo, Difei Gao, and Mike Zheng Shou. 2025. AUTO-Explorer: Automated Data Collection for GUI Agent. arXiv:2511.06417 [cs.AI] doi:10.48550/arXiv.2511.06417
 - [72] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis. arXiv:2307.12856 [cs.LG] doi:10.48550/arXiv.2307.12856
 - [73] Wenkang Han, Zhixiong Zeng, Jing Huang, Shu Jiang, Liming Zheng, Longrong Yang, Haibo Qiu, Chang Yao, Jingyuan Chen, and Lin Ma. 2025. UITron-Speech: Towards Automated GUI Agents Based on Speech Instructions. arXiv:2506.11127 [cs.CL] doi:10.48550/arXiv.2506.11127
 - [74] Chao Hao, Shuai Wang, and Kaiwen Zhou. 2025. Uncertainty-Aware GUI Agent: Adaptive Perception through Component Recommendation and Human-in-the-Loop Refinement. arXiv:2508.04025 [cs.AI] doi:10.48550/arXiv.2508.04025

04025

- [75] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. 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)*. 6864–6890. doi:10.18653/v1/2024.acl-long.371
- [76] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. 2025. OpenWebVoyager: Building Multimodal Web Agents via Iterative Real-World Exploration, Feedback and Optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 27545–27564. doi:10.18653/v1/2025.acl-long.1336
- [77] Yanheng He, Jiahe Jin, Shijie Xia, Jiadi Su, Runze Fan, Haoyang Zou, Xiangkun Hu, and Pengfei Liu. 2024. PC Agent: While You Sleep, AI Works – A Cognitive Journey into Digital World. arXiv:2412.17589 [cs.AI] doi:10.48550/arXiv.2412.17589
- [78] Zhitao He, Zijun Liu, Peng Li, Yi R. Fung, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2025. Advancing Language Multi-Agent Learning with Credit Re-Assignment for Interactive Environment Generalization. arXiv:2502.14496 [cs.CL] doi:10.48550/arXiv.2502.14496
- [79] Wenyi Hong, Weihao Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2024. CogAgent: A Visual Language Model for GUI Agents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14281–14290. doi:10.1109/cvpr52733.2024.01354
- [80] Jakub Hoscilowicz, Bartosz Maj, Bartosz Kozakiewicz, Oleksii Tymoshchuk, and Artur Janicki. 2024. ClickAgent: Enhancing UI Location Capabilities of Autonomous Agents. arXiv:2410.11872 [cs.HC] doi:10.48550/arXiv.2410.11872
- [81] Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. 2024. The Dawn of GUI Agent: A Preliminary Case Study with Claude 3.5 Computer Use. arXiv:2411.10323 [cs.AI] doi:10.48550/arXiv.2411.10323
- [82] Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shawn Wang, Xinchun Xu, Shuofei Qiao, Kun Kuang, Tiejong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. 2024. OS Agents: A Survey on MLLM-Based Agents for General Computing Devices Use. (2024). doi:10.20944/preprints202412.2294.v1
- [83] Zhiyuan Hu, Shiyun Xiong, Yifan Zhang, See-Kiong Ng, Anh Tuan Luu, Bo An, Shuicheng Yan, and Bryan Hooi. 2025. Guiding VLM Agents with Process Rewards at Inference Time for GUI Navigation. arXiv:2504.16073 [cs.CL] doi:10.48550/arXiv.2504.16073
- [84] Jing Huang, Zhixiong Zeng, Wenkang Han, Yufeng Zhong, Liming Zheng, Shuai Fu, Jingyuan Chen, and Lin Ma. 2025. ScaleTrack: Scaling and back-tracking Automated GUI Agents. arXiv:2505.00416 [cs.AI] doi:10.48550/arXiv.2505.00416
- [85] Kung-Hsiang Huang, Haoyi Qiu, Yutong Dai, Caiming Xiong, and Chien-Sheng Wu. 2025. GUI-KV: Efficient GUI Agents via KV Cache with Spatio-Temporal Awareness. arXiv:2510.00536 [cs.CL] doi:10.48550/arXiv.2510.00536
- [86] Tenghao Huang, Kinjal Basu, Ibrahim Abdelaziz, Pavan Kapanipathi, Jonathan May, and Muhao Chen. 2025. R2D2: Remembering, Replaying and Dynamic Decision Making with a Reflective Agentic Memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 30318–30330. doi:10.18653/v1/2025.acl-long.1464
- [87] Tian Huang, Chun Yu, Weinan Shi, Zijian Peng, David Yang, Weiqi Sun, and Yuanchun Shi. 2024. PromptRPA: Generating Robotic Process Automation on Smartphones from Textual Prompts. arXiv:2404.02475 [cs.HC] doi:10.48550/arXiv.2404.02475
- [88] Tian Huang, Chun Yu, Weinan Shi, Zijian Peng, David Yang, Weiqi Sun, and Yuanchun Shi. 2025. Prompt2Task: Automating UI Tasks on Smartphones from Textual Prompts. *ACM Transactions on Computer-Human Interaction* 32, 3 (2025), 1–45. doi:10.1145/3716132
- [89] Zhiyuan Huang, Ziming Cheng, Junting Pan, Zhaohui Hou, and Mingjie Zhan. 2025. SpiritSight Agent: Advanced GUI Agent with One Look. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 29490–29500. doi:10.1109/cvpr52734.2025.02746
- [90] Zheng Hui, Yinzheng Li, Dan zhao, Tianyi Chen, Colby Banbury, and Kazuhito Koishida. 2025. WinClick: GUI Grounding with Multimodal Large Language Models. arXiv:2503.04730 [cs.CL] doi:10.48550/arXiv.2503.04730
- [91] Iat Long Iong, Xiao Liu, Yuxuan Chen, Hanyu Lai, Shuntian Yao, Pengbo Shen, Hao Yu, Yuxiao Dong, and Jie Tang. 2024. OpenWebAgent: An Open Toolkit to Enable Web Agents on Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. 72–81. doi:10.18653/v1/2024.acl-demos.8
- [92] Lawrence Jang, Yinzheng Li, Dan Zhao, Charles Ding, Justin Lin, Paul Pu Liang, Rogerio Bonatti, and Kazuhito Koishida. 2024. VideoWebArena: Evaluating Long Context Multimodal Agents with Video Understanding Web Tasks. arXiv:2410.19100 [cs.CV] doi:10.48550/arXiv.2410.19100

- [93] Chengyou Jia, Minnan Luo, Zhuohang Dang, Qiushi Sun, Fangzhi Xu, Junlin Hu, Tianbao Xie, and Zhiyong Wu. 2025. AgentStore: Scalable Integration of Heterogeneous Agents As Specialized Generalist Computer Assistant. In *Findings of the Association for Computational Linguistics: ACL 2025*. 8908–8934. doi:10.18653/v1/2025.findings-acl.466
- [94] Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. 2025. AppAgentX: Evolving GUI Agents as Proficient Smartphone Users. arXiv:2503.02268 [cs.AI] doi:10.48550/arXiv.2503.02268
- [95] Yiqiao Jin, Stefano Petrangeli, Yu Shen, and Gang Wu. 2025. <scp>ScreenLLM:</scp> Stateful Screen Schema for Efficient Action Understanding and Prediction. In *Companion Proceedings of the ACM on Web Conference 2025*. 2008–2013. doi:10.1145/3701716.3718379
- [96] Raghav Kapoor, Yash Parag Butala, Melissa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2024. OmniACT: A Dataset and Benchmark for Enabling Multimodal Generalist Autonomous Agents for Desktop and Web. *Lecture Notes in Computer Science* (2024), 161–178. doi:10.1007/978-3-031-73113-6_10
- [97] Su Kara, Fazle Faisal, and Suman Nath. 2025. WABER: Evaluating Reliability and Efficiency of Web Agents with Existing Benchmarks. In *ICLR Workshop on Foundation Models in the Wild*. <https://www.microsoft.com/en-us/research/publication/waber-evaluating-reliability-and-efficiency-of-web-agents-with-existing-benchmarks/>
- [98] Jihyung Kil, Chan Hee Song, Boyuan Zheng, Xiang Deng, Yu Su, and Wei-Lun Chao. 2024. Dual-View Visual Contextualization for Web Navigation. arXiv:2402.04476 [cs.CV] doi:10.48550/arXiv.2402.04476
- [99] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. 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)*. 881–905. doi:10.18653/v1/2024.acl-long.50
- [100] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree Search for Language Model Agents. arXiv:2407.01476 [cs.AI] doi:10.48550/arXiv.2407.01476
- [101] SeokJoo Kwak, Jihoon Kim, Boyoun Kim, Jung Jae Yoon, Wooseok Jang, Jeonghoon Hong, Jaeho Yang, and Yeong-Dae Kwon. 2025. MEGA-GUI: Multi-stage Enhanced Grounding Agents for GUI Elements. arXiv:2511.13087 [cs.AI] doi:10.48550/arXiv.2511.13087
- [102] Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. AutoWebGLM: A Large Language Model-based Web Navigating Agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5295–5306. doi:10.1145/3637528.3671620
- [103] Juyong Lee, Dongyoon Hahm, June Suk Choi, W. Bradley Knox, and Kimin Lee. 2026. MobileSafetyBench: Evaluating Safety of Autonomous Agents in Mobile Device Control. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 44 (2026), 37565–37573. doi:10.1609/aaai.v40i44.41090
- [104] Jungjae Lee, Dongjae Lee, Chihun Choi, Youngmin Im, Jaeyoung Wi, Kihong Heo, Sangeun Oh, Sunjae Lee, and Insik Shin. 2025. VeriSafe Agent: Safeguarding Mobile GUI Agent via Logic-based Action Verification. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*. 817–831. doi:10.1145/3680207.3765248
- [105] Juyong Lee, Taywon Min, Minyong An, Dongyoon Hahm, Haeone Lee, Changyeon Kim, and Kimin Lee. 2024. Benchmarking Mobile Device Control Agents across Diverse Configurations. arXiv:2404.16660 [cs.HC] doi:10.48550/arXiv.2404.16660
- [106] Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steve Ko, Sangeun Oh, and Insik Shin. 2024. MobileGPT: Augmenting LLM with Human-like App Memory for Mobile Task Automation. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 1119–1133. doi:10.1145/3636534.3690682
- [107] Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2024. ST-WebAgentBench: A Benchmark for Evaluating Safety and Trustworthiness in Web Agents. arXiv:2410.06703 [cs.AI] doi:10.48550/arXiv.2410.06703
- [108] Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Li Qing, and Zhaoxiang Zhang. 2025. AutoGUI: Scaling GUI Grounding with Automatic Functionality Annotations from LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10323–10358. doi:10.18653/v1/2025.acl-long.510
- [109] Hongxin Li, Jingran Su, Jingfan Chen, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025. UIPro: Unleashing Superior Interaction Capability For GUI Agents. arXiv:2509.17328 [cs.CV] doi:10.48550/arXiv.2509.17328
- [110] Jiahao Li and Kaer Huang. 2025. A Survey on GUI Agents with Foundation Models Enhanced by Reinforcement Learning. arXiv:2504.20464 [cs.AI] doi:10.48550/arXiv.2504.20464
- [111] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025. ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 8778–8786. doi:10.1145/3746027.3755688
- [112] Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. 2023. A Zero-Shot Language Agent for Computer Control with Structured Reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 11261–11274. doi:10.18653/v1/2023.findings-emnlp.753

- [113] Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the Effects of Data Scale on UI Control Agents. In *Advances in Neural Information Processing Systems 37*. 92130–92154. doi:10.52202/079017-2925
- [114] Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the Effects of Data Scale on UI Control Agents. In *Advances in Neural Information Processing Systems 37*. 92130–92154. doi:10.52202/079017-2925
- [115] Wei Li, Fu-Lin Hsu, William Bishop, Folawiyo Campbell-Ajala, Max Lin, and Oriana Riva. 2024. UINav: A Practical Approach to Train On-Device Automation Agents. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*. 36–51. doi:10.18653/v1/2024.naacl-industry.4
- [116] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping Natural Language Instructions to Mobile UI Action Sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8198–8210. doi:10.18653/v1/2020.acl-main.729
- [117] Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. AppAgent v2: Advanced Agent for Flexible Mobile Interactions. arXiv:2408.11824 [cs.HC] doi:10.48550/arXiv.2408.11824
- [118] Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujuan Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-UI 2: Mastering Universal User Interface Understanding Across Platforms. arXiv:2410.18967 [cs.CV] doi:10.48550/arXiv.2410.18967
- [119] Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, Hui Li, and Rongrong Ji. 2025. UI-AGILE: Advancing GUI Agents with Effective Reinforcement Learning and Precise Inference-Time Grounding. arXiv:2507.22025 [cs.AI] doi:10.48550/arXiv.2507.22025
- [120] Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. 2024. EIA: Environmental Injection Attack on Generalist Web Agents for Privacy Leakage. arXiv:2409.11295 [cs.CR] doi:10.48550/arXiv.2409.11295
- [121] Kevin Lin, Linjie Li, Difei Gao, Qinchen Wu, Mingyi Yan, Zhengyuan Yang, Lijuan Wang, and Mike Shou. 2024. VideoGUI: A Benchmark for GUI Automation from Instructional Videos. In *Advances in Neural Information Processing Systems 37*. 69329–69360. doi:10.52202/079017-2214
- [122] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025. ShowUI: One Vision-Language-Action Model for GUI Visual Agent. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 19498–19508. doi:10.1109/cvpr52734.2025.01816
- [123] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration. arXiv:1802.08802 [cs.AI] doi:10.48550/arXiv.1802.08802
- [124] Guohong Liu, Jialei Ye, Jiacheng Liu, Yuanchun Li, Wei Liu, Pengzhi Gao, Jian Luan, and Yunxin Liu. 2025. Hijacking JARVIS: Benchmarking Mobile GUI Agents against Unprivileged Third Parties. In *Proceedings of the 2nd International Workshop on Edge and Mobile Foundation Models*. 12–18. doi:10.1145/3737902.3768354
- [125] Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, Xiaoyu Liang, WenHao Wang, Tianze Wu, Zhengxi Lu, Siheng Chen, LiLinghao, Hao Wang, Guanqing Xiong, Yong Liu, and Hongsheng Li. 2025. LLM-Powered GUI Agents in Phone Automation: Surveying Progress and Prospects. arXiv:2504.19838 [cs.HC] doi:10.48550/arXiv.2504.19838
- [126] Guangyi Liu, Pengxiang Zhao, Liang Liu, Zhiming Chen, Yuxiang Chai, Shuai Ren, Hao Wang, Shibo He, and Wenchao Meng. 2025. LearnAct: Few-Shot Mobile GUI Agent with a Unified Demonstration Benchmark. arXiv:2504.13805 [cs.HC] doi:10.48550/arXiv.2504.13805
- [127] Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfeng Yuan, Changsheng Xu, Weiming Hu, and Fei Huang. 2025. PC-Agent: A Hierarchical Multi-Agent Collaboration Framework for Complex Task Automation on PC. arXiv:2502.14282 [cs.CV] doi:10.48550/arXiv.2502.14282
- [128] Jiarun Liu, Jia Hao, Chunhong Zhang, and Zheng Hu. 2025. WEPO: Web Element Preference Optimization for LLM-based Web Navigation. *Proceedings of the AAAI Conference on Artificial Intelligence* 39, 25 (2025), 26614–26622. doi:10.1609/aaai.v39i25.34863
- [129] Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024. VisualWebBench: How Far Have Multimodal LLMs Evolved in Web Page Understanding and Grounding? arXiv:2404.05955 [cs.CL] doi:10.48550/arXiv.2404.05955
- [130] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Jat Long Iong, Jiada Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. 2024. AutoGLM: Autonomous Foundation Agents for GUIs. arXiv:2411.00820 [cs.HC] doi:10.48550/arXiv.2411.00820

- [131] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiada 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. 2024. VisualAgentBench: Towards Large Multimodal Models as Visual Foundation Agents. *arXiv:2408.06327* [cs.AI] [doi:10.48550/arXiv.2408.06327](#)
- [132] Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. 2025. UI-E2I-Synth: Advancing GUI Grounding with Large-Scale Instruction Synthesis. In *Findings of the Association for Computational Linguistics: ACL 2025*. 15668–15684. [doi:10.18653/v1/2025.findings-acl.809](#)
- [133] Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchun Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. 2026. InfiGUIAgent: A Multimodal Generalist GUI Agent with Native Reasoning and Reflection. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1035–1051. [doi:10.18653/v1/2026.eacl-long.45](#)
- [134] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025. InfiGUI-R1: Advancing Multimodal GUI Agents from Reactive Actors to Deliberative Reasoners. *arXiv:2504.14239* [cs.AI] [doi:10.48550/arXiv.2504.14239](#)
- [135] Yuxuan Liu, Hongda Sun, Wei Liu, Jian Luan, Bo Du, and Rui Yan. 2025. MobileSteward: Integrating Multiple App-Oriented Agents with Self-Evolution to Automate Cross-App Instructions. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*. 883–893. [doi:10.1145/3690624.3709171](#)
- [136] Ziwei Liu, Borui Kang, Hangjie Yuan, Zixiang Zhao, Wei Li, Yifan Zhu, and Tao Feng. 2026. Continual GUI Agents. *arXiv:2601.20732* [cs.LG] [doi:10.48550/arXiv.2601.20732](#)
- [137] Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. 2025. ARPO: End-to-End Policy Optimization for GUI Agents with Experience Replay. *arXiv:2505.16282* [cs.CV] [doi:10.48550/arXiv.2505.16282](#)
- [138] Fanbin Lu, Zhisheng Zhong, Ziqin Wei, Shu Liu, Chi-Wing Fu, and Jiaya Jia. 2025. STEVE: A Step Verification Pipeline for Computer-use Agent Training. *arXiv:2503.12532* [cs.CV] [doi:10.48550/arXiv.2503.12532](#)
- [139] Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2024. GUIOdyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices. *arXiv:2406.08451* [cs.CV] [doi:10.48550/arXiv.2406.08451](#)
- [140] Yijie Lu, Tianjie Ju, Manman Zhao, Xinbei Ma, Yuan Guo, and Zhuosheng Zhang. 2025. EVA: Red-Teaming GUI Agents via Evolving Indirect Prompt Injection. *arXiv:2505.14289* [cs.AI] [doi:10.48550/arXiv.2505.14289](#)
- [141] Yuheng Lu, Qian Yu, Hongru Wang, Zeming Liu, Wei Su, Yanping Liu, Yuhang Guo, Maocheng Liang, Yunhong Wang, and Haifeng Wang. 2025. TransBench: Breaking Barriers for Transferable Graphical User Interface Agents in Dynamic Digital Environments. In *Findings of the Association for Computational Linguistics: ACL 2025*. 12464–12478. [doi:10.18653/v1/2025.findings-acl.645](#)
- [142] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, Guanqing Xiong, and Hongsheng Li. 2026. UI-R1: Enhancing Efficient Action Prediction of GUI Agents by Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 21 (2026), 17608–17616. [doi:10.1609/aaai.v40i21.38816](#)
- [143] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, Guanqing Xiong, and Hongsheng Li. 2026. UI-R1: Enhancing Efficient Action Prediction of GUI Agents by Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 21 (2026), 17608–17616. [doi:10.1609/aaai.v40i21.38816](#)
- [144] Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. 2025. ViMo: A Generative Visual GUI World Model for App Agents. *arXiv:2504.13936* [cs.HC] [doi:10.48550/arXiv.2504.13936](#)
- [145] Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. 2025. GUI-R1 : A Generalist R1-Style Vision-Language Action Model For GUI Agents. *arXiv:2504.10458* [cs.CV] [doi:10.48550/arXiv.2504.10458](#)
- [146] Yibo Lyu, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. 2026. PersonalAlign: Hierarchical Implicit Intent Alignment for Personalized GUI Agent with Long-Term User-Centric Records. *arXiv:2601.09636* [cs.AI] [doi:10.48550/arXiv.2601.09636](#)
- [147] Xing Han Lü, Zdeněk Kasner, and Siva Reddy. 2024. WebLINX: Real-World Website Navigation with Multi-Turn Dialogue. *arXiv:2402.05930* [cs.CL] [doi:10.48550/arXiv.2402.05930](#)
- [148] Xing Han Lü, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. 2025. AgentRewardBench: Evaluating Automatic Evaluations of Web Agent Trajectories. *arXiv:2504.08942* [cs.LG] [doi:10.48550/arXiv.2504.08942](#)
- [149] Yadong Lü, Jianwei Yang, Yelong Shen, and Ahmed Hassan Awadallah. 2024. OmniParser for Pure Vision Based GUI Agent. *arXiv:2408.00203* [cs.CV] [doi:10.48550/arXiv.2408.00203](#)

- [150] Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. LASER: LLM Agent with State-Space Exploration for Web Navigation. arXiv:2309.08172 [cs.CL] doi:10.48550/arXiv.2309.08172
- [151] Longhui Ma, Di Zhao, Siwei Wang, Zhao Lv, and Miao Wang. 2026. Beyond element-level understanding: Explicit relational understanding for GUI agents. *Pattern Recognition* 176 (2026), 113262. doi:10.1016/j.patcog.2026.113262
- [152] Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2025. Caution for the Environment: Multimodal LLM Agents are Susceptible to Environmental Distractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 22324–22339. doi:10.18653/v1/2025.acl-long.1087
- [153] Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2025. Caution for the Environment: Multimodal LLM Agents are Susceptible to Environmental Distractions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 22324–22339. doi:10.18653/v1/2025.acl-long.1087
- [154] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. CoCo-Agent: A Comprehensive Cognitive MLLM Agent for Smartphone GUI Automation. In *Findings of the Association for Computational Linguistics ACL 2024*. 9097–9110. doi:10.18653/v1/2024.findings-acl.539
- [155] Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. 2024. NNetNav: Unsupervised Learning of Browser Agents Through Environment Interaction in the Wild. arXiv:2410.02907 [cs.CL] doi:10.48550/arXiv.2410.02907
- [156] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. 2025. UI-Vision: A Desktop-centric GUI Benchmark for Visual Perception and Interaction. arXiv:2503.15661 [cs.CV] doi:10.48550/arXiv.2503.15661
- [157] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. 2025. GUI Agents: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2025*. 22522–22538. doi:10.18653/v1/2025.findings-acl.1158
- [158] Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. ScreenAgent: A Vision Language Model-driven Computer Control Agent. arXiv:2402.07945 [cs.HC] doi:10.48550/arXiv.2402.07945
- [159] Songqin Nong, Xiaoxuan Tang, Jingxuan Xu, Sheng Zhou, Jianfeng Chen, Tao Jiang, and Wenhao Xu. 2025. CRAFT-GUI: Curriculum-Reinforced Agent For GUI Tasks. arXiv:2508.11360 [cs.AI] doi:10.48550/arXiv.2508.11360
- [160] Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. MobileFlow: A Multimodal LLM For Mobile GUI Agent. arXiv:2407.04346 [cs.CV] doi:10.48550/arXiv.2407.04346
- [161] Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan Awadallah. 2025. Explorer: Scaling Exploration-driven Web Trajectory Synthesis for Multimodal Web Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*. 6300–6323. doi:10.18653/v1/2025.findings-acl.326
- [162] Lihang Pan, Bowen Wang, Chun Yu, Yuxuan Chen, Xiangyu Zhang, and Yuanchun Shi. 2023. AutoTask: Executing Arbitrary Voice Commands by Exploring and Learning from Mobile GUI. arXiv:2312.16062 [cs.HC] doi:10.48550/arXiv.2312.16062
- [163] Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. WebCanvas: Benchmarking Web Agents in Online Environments. arXiv:2406.12373 [cs.CL] doi:10.48550/arXiv.2406.12373
- [164] Georgios Papoudakis, Thomas Coste, Zhihao Wu, Jianye Hao, Jun Wang, and Kun Shao. 2025. AppVLM: A Lightweight Vision Language Model for Online App Control. arXiv:2502.06395 [cs.AI] doi:10.48550/arXiv.2502.06395
- [165] Joonhyung Park, Peng Tang, Sagnik Das, Srikanth Appalaraju, Kunwar Yashraj Singh, R. Manmatha, and Shabnam Ghadar. 2025. R-VLM: Region-Aware Vision Language Model for Precise GUI Grounding. In *Findings of the Association for Computational Linguistics: ACL 2025*. 9669–9685. doi:10.18653/v1/2025.findings-acl.501
- [166] Pawel Pawlowski, Krystian Zawistowski, Wojciech Lapacz, Adam Wiacek, Marcin Skorupa, Sebastien Postansque, and Jakub Hoscilowicz. 2025. TinyClick: Single-Turn Agent for Empowering GUI Automation. In *Interspeech 2025*. 3035–3039. doi:10.21437/interspeech.2025-176
- [167] Yi-Hao Peng, Faria Huq, Yue Jiang, Jason Wu, Xin Yue Li, Jeffrey P. Bigham, and Amy Pavel. 2024. DreamStruct: Understanding Slides and User Interfaces via Synthetic Data Generation. *Lecture Notes in Computer Science* (2024), 466–485. doi:10.1007/978-3-031-72691-0_26
- [168] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents. arXiv:2408.07199 [cs.AI] doi:10.48550/arXiv.2408.07199

- [169] Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2024. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. arXiv:2411.02337 [cs.CL] doi:10.48550/arXiv.2411.02337
- [170] Yijun Qian, Yujie Lu, Alexander Hauptmann, and Oriana Riva. 2024. Visual Grounding for User Interfaces. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*. 97–107. doi:10.18653/v1/2024.naacl-industry.9
- [171] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjuan Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. arXiv:2501.12326 [cs.AI] doi:10.48550/arXiv.2501.12326
- [172] Haoyi Qiu, Alexander Fabbri, Divyansh Agarwal, Kung-Hsiang Huang, Sarah Tan, Nanyun Peng, and Chien-Sheng Wu. 2025. Evaluating Cultural and Social Awareness of LLM Web Agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*. 3978–4005. doi:10.18653/v1/2025.findings-naacl.222
- [173] Abdur Rahman, Rajat Chawla, Muskaan Kumar, Arkajit Datta, Adarsh Jha, Mukunda NS, and Ishaan Bhola. 2024. V-Zen: Efficient GUI Understanding and Precise Grounding With A Novel Multimodal LLM. arXiv:2405.15341 [cs.AI] doi:10.48550/arXiv.2405.15341
- [174] Dezhi Ran, Hao Wang, Zihe Song, Mengzhou Wu, Yuan Cao, Ying Zhang, Wei Yang, and Tao Xie. 2024. Guardian: A Runtime Framework for LLM-Based UI Exploration. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 958–970. doi:10.1145/3650212.3680334
- [175] Dezhi Ran, Mengzhou Wu, Hao Yu, Yuetong Li, Jun Ren, Yuan Cao, Xia Zeng, Haochuan Lu, Zexin Xu, Mengqian Xu, Ting Su, Liangchao Yao, Ting Xiong, Wei Yang, Yuetang Deng, Assaf Marron, David Harel, and Tao Xie. 2025. Beyond Pass or Fail: Multi-Dimensional Benchmarking of Foundation Models for Goal-based Mobile UI Navigation. arXiv:2501.02863 [cs.SE] doi:10.48550/arXiv.2501.02863
- [176] Christopher Rawles, Sarah Clinckemaeille, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyi Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2024. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. arXiv:2405.14573 [cs.AI] doi:10.48550/arXiv.2405.14573
- [177] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the Wild: A Large-Scale Dataset for Android Device Control. arXiv:2307.10088 [cs.LG] doi:10.48550/arXiv.2307.10088
- [178] Mobina Shahbandeh, Parsa Alian, Noor Nashid, and Ali Mesbah. 2024. NaviQAt: Functionality-Guided Web Application Navigation. arXiv:2409.10741 [cs.SE] doi:10.48550/arXiv.2409.10741
- [179] Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. 2023. From Pixels to UI Actions: Learning to Follow Instructions via Graphical User Interfaces. In *Advances in Neural Information Processing Systems* 36. 34354–34370. doi:10.52202/075280-1490
- [180] Huawei Shen, Chang Liu, Gengluo Li, Xinlong Wang, Yu Zhou, Can Ma, and Xiangyang Ji. 2024. Falcon-UI: Understanding GUI Before Following User Instructions. arXiv:2412.09362 [cs.CL] doi:10.48550/arXiv.2412.09362
- [181] Junhong Shen, Atishay Jain, Zedian Xiao, Ishan Amlekar, Mouad Hadji, Aaron Podolny, and Ameet Talwalkar. 2024. ScribeAgent: Towards Specialized Web Agents Using Production-Scale Workflow Data. arXiv:2411.15004 [cs.CL] doi:10.48550/arXiv.2411.15004
- [182] Yucheng Shi, Wenhao Yu, Jingyuan Huang, Wenlin Yao, Wenhui Chen, and Ninghao Liu. 2025. Towards Trustworthy GUI Agents: A Survey. arXiv:2503.23434 [cs.LG] doi:10.48550/arXiv.2503.23434
- [183] Yucheng Shi, Wenhao Yu, Zaitang Li, Yong-Lin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. 2025. MobileGUI-RL: Advancing Mobile GUI Agent through Reinforcement Learning in Online Environment. arXiv:2507.05720 [cs.LG] doi:10.48550/arXiv.2507.05720
- [184] Kunal Singh, Shreyas Singh, and Mukund Khanna. 2025. Trishul: Towards Region Identification and Screen Hierarchy Understanding for Large VLM Based GUI Agents. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 170–179. doi:10.1109/cvprw67362.2025.00022
- [185] Paloma Sodhi, S. R. K. Branavan, Yoav Artzi, and Ryan McDonald. 2023. SteP: Stacked LLM Policies for Web Actions. arXiv:2310.03720 [cs.LG] doi:10.48550/arXiv.2310.03720
- [186] Yunpeng Song, Yiheng Bian, Yongtao Tang, Guiyu Ma, and Zhongmin Cai. 2024. VisionTasker: Mobile Task Automation Using Vision Based UI Understanding and LLM Task Planning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–17. doi:10.1145/3654777.3676386
- [187] Yixiao Song, Katherine Thai, Chau Minh Pham, Yapei Chang, Mazin Nada, and Mohit Iyyer. 2025. BEARCUBS: A benchmark for computer-using web agents. arXiv:2503.07919 [cs.AI] doi:10.48550/arXiv.2503.07919
- [188] Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. 2024. Beyond Browsing: API-Based Web Agents. arXiv:2410.16464 [cs.CL] doi:10.48550/arXiv.2410.16464

- [189] Zirui Song, Yaohang Li, Meng Fang, Zhenhao Chen, Zecheng Shi, Yuan Huang, and Ling Chen. 2024. MMAC-Copilot: Multi-modal Agent Collaboration Operating System Copilot. arXiv:2404.18074 [cs.AI] doi:10.48550/arXiv.2404.18074
- [190] Trisanth Srinivasan and Santosh Patapati. 2025. WebNav: An Intelligent Agent for Voice-Controlled Web Navigation. arXiv:2503.13843 [cs.AI] doi:10.48550/arXiv.2503.13843
- [191] Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arik. 2025. Learn-by-interact: A Data-Centric Framework for Self-Adaptive Agents in Realistic Environments. arXiv:2501.10893 [cs.LG] doi:10.48550/arXiv.2501.10893
- [192] Jiahui Sun, Zhichao Hua, and Yubin Xia. 2025. AutoEval: A Practical Framework for Autonomous Evaluation of Mobile Agents. arXiv:2503.02403 [cs.AI] doi:10.48550/arXiv.2503.02403
- [193] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. META-GUI: Towards Multi-modal Conversational Agents on Mobile GUI. (2022), 6699–6712. doi:10.18653/v1/2022.emnlp-main.449
- [194] Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. 2024. OS-Genesis: Automating GUI Agent Trajectory Construction via Reverse Task Synthesis. arXiv:2412.19723 [cs.AI] doi:10.48550/arXiv.2412.19723
- [195] Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, Lingpeng Kong, and Zhiyong Wu. 2025. ScienceBoard: Evaluating Multimodal Autonomous Agents in Realistic Scientific Workflows. arXiv:2505.19897 [cs.AI] doi:10.48550/arXiv.2505.19897
- [196] Yuchen Sun, Shanhuai Zhao, Tao Yu, Hao Wen, Samith Va, Mengwei Xu, Yuanchun Li, and Chongyang Zhang. 2025. GUI-Xplore: Empowering Generalizable GUI Agents with One Exploration. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 19477–19486. doi:10.1109/cvpr52734.2025.01814
- [197] Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong, Chen, Abhanshu Sharma, and James Stout. 2022. Towards Better Semantic Understanding of Mobile Interfaces. arXiv:2210.02663 [cs.HC] doi:10.48550/arXiv.2210.02663
- [198] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. 2024. Cradle: Empowering Foundation Agents Towards General Computer Control. arXiv:2403.03186 [cs.AI] doi:10.48550/arXiv.2403.03186
- [199] Brian Tang and Kang G. Shin. 2024. Steward: Natural Language Web Automation. arXiv:2409.15441 [cs.AI] doi:10.48550/arXiv.2409.15441
- [200] Fei Tang, Yongliang Shen, Hang Zhang, Siqi Chen, Guiyang Hou, Wenqi Zhang, Wenqiao Zhang, Kaitao Song, Weiming Lu, and Yueting Zhuang. 2025. Think Twice, Click Once: Enhancing GUI Grounding via Fast and Slow Systems. arXiv:2503.06470 [cs.AI] doi:10.48550/arXiv.2503.06470
- [201] Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. A Survey on (M)LLM-Based GUI Agents. arXiv:2504.13865 [cs.HC] doi:10.48550/arXiv.2504.13865
- [202] Jiaqi Tang, Yu Xia, Yi-Feng Wu, Yuwei Hu, Yuhui Chen, Qing-Guo Chen, Xiaogang Xu, Xiangyu Wu, Hao Lu, Yanqing Ma, Shiyin Lu, and Qifeng Chen. 2025. LPO: Towards Accurate GUI Agent Interaction via Location Preference Optimization. arXiv:2506.09373 [cs.LG] doi:10.48550/arXiv.2506.09373
- [203] Liujian Tang, Shaokang Dong, Yijia Huang, Minqi Xiang, Hongtao Ruan, Bin Wang, Shuo Li, Zhiheng Xi, Zhihui Cao, Hailiang Pang, Heng Kong, He Yang, Mingxu Chai, Zhilin Gao, Xingyu Liu, Yingnan Fu, Jiaming Liu, Xuanjing Huang, Yu-Gang Jiang, Tao Gui, Qi Zhang, Kang Wang, Yunke Zhang, and Yuran Wang. 2025. MagicGUI: A Foundational Mobile GUI Agent with Scalable Data Pipeline and Reinforcement Fine-tuning. arXiv:2508.03700 [cs.HC] doi:10.48550/arXiv.2508.03700
- [204] Sizhe Tang, Rongqian Chen, and Tian Lan. 2026. Agent Alpha: Tree Search Unifying Generation, Exploration and Evaluation for Computer-Use Agents. arXiv:2602.02995 [cs.AI] doi:10.48550/arXiv.2602.02995
- [205] Xingjian Tao, Yiwei Wang, Yujun Cai, Zhicheng Yang, and Jing Tang. 2025. Understanding GUI Agent Localization Biases through Logit Sharpness. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. 23361–23374. doi:10.18653/v1/2025.findings-emnlp.1268
- [206] Lucas-Andrei Thil, Mirela Popa, and Gerasimos Spanakis. 2024. Navigating WebAI: Training Agents to Complete Web Tasks with Large Language Models and Reinforcement Learning. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 866–874. doi:10.1145/3605098.3635903
- [207] George Thomas, Alex J. Chan, Jikun Kang, Wenqi Wu, Filippos Christianos, Fraser Greenlee, Andy Toulis, and Marvin Purtorab. 2025. WebGames: Challenging General-Purpose Web-Browsing AI Agents. arXiv:2502.18356 [cs.LG] doi:10.48550/arXiv.2502.18356

- [208] Shizuo Tian, Hao Wen, Yuxuan Chen, Jiacheng Liu, Shanhui Zhao, Guohong Liu, Ju Ren, Yunxin Liu, and Yuanchun Li. 2025. AgentProg: Empowering Long-Horizon GUI Agents with Program-Guided Context Management. arXiv:2512.10371 [cs.AI] doi:10.48550/arXiv.2512.10371
- [209] Shulin Tian, Ziniu Zhang, Liangyu Chen, and Ziwei Liu. 2025. MMInA: Benchmarking Multihop Multimodal Internet Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*. 13682–13697. doi:10.18653/v1/2025.findings-acl.703
- [210] Brandon Trabucco, Gunnar Sigurdsson, Robinson Piramuthu, and Ruslan Salakhutdinov. 2025. InSTA: Towards Internet-Scale Training For Agents. arXiv:2502.06776 [cs.LG] doi:10.48550/arXiv.2502.06776
- [211] Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. 2025. SafeArena: Evaluating the Safety of Autonomous Web Agents. arXiv:2503.04957 [cs.LG] doi:10.48550/arXiv.2503.04957
- [212] Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. 2024. GPTVoiceTasker: LLM-Powered Virtual Assistant for Smartphone. arXiv:2401.14268 [cs.HC] doi:10.48550/arXiv.2401.14268
- [213] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction with Mobile UI using Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17. doi:10.1145/3544548.3580895
- [214] Bowen Wang, Xinyuan Wang, Jiaqi Deng, Tianbao Xie, Ryan Li, Yanzhe Zhang, Junli Wang, Dunjie Lu, Zicheng Gong, Gavin Li, Toh Jing Hua, Wei-Lin Chiang, Ion Stoica, Diyi Yang, Yu Su, Yi Zhang, Zhiguo Wang, Victor Zhong, and Tao Yu. 2026. Computer Agent Arena: Toward Human-Centric Evaluation and Analysis of Computer-Use Agents. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=3x4SDbXbgl>
- [215] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-Agent-v2: Mobile Device Operation Assistant with Effective Navigation via Multi-Agent Collaboration. arXiv:2406.01014 [cs.CL] doi:10.48550/arXiv.2406.01014
- [216] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-Agent: Autonomous Multi-Modal Mobile Device Agent with Visual Perception. arXiv:2401.16158 [cs.CL] doi:10.48550/arXiv.2401.16158
- [217] Junyang Wang, Haiyang Xu, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Jitao Sang. 2025. Mobile-Agent-V: Learning Mobile Device Operation Through Video-Guided Multi-Agent Collaboration. arXiv:2502.17110 [cs.CL] doi:10.48550/arXiv.2502.17110
- [218] Ke Wang, Tianyu Xia, Zhangxuan Gu, Yi Zhao, Shuheng Shen, Changhua Meng, Weiqiang Wang, and Ke Xu. 2024. E-ANT: A Large-Scale Dataset for Efficient Automatic GUI Navigation. arXiv:2406.14250 [cs.CV] doi:10.48550/arXiv.2406.14250
- [219] Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. 2024. MobileAgentBench: An Efficient and User-Friendly Benchmark for Mobile LLM Agents. arXiv:2406.08184 [cs.AI] doi:10.48550/arXiv.2406.08184
- [220] Shuai Wang, Weiwen Liu, J. C. Chen, Yuqi Zhou, Weinan Gan, X. Zeng, Yuhao Che, Shicheng Yu, Xinlong Hao, Shao Kun, Bin Wang, Chuhan Wu, Y.S. Wang, Ruiming Tang, and Jianye Hao. 2024. GUI Agents with Foundation Models: A Comprehensive Survey. arXiv:2411.04890 [cs.AI] doi:10.48550/arXiv.2411.04890
- [221] Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. 2024. DistRL: An Asynchronous Distributed Reinforcement Learning Framework for On-Device Control Agents. arXiv:2410.14803 [cs.LG] doi:10.48550/arXiv.2410.14803
- [222] Wenhao Wang, Zijie Yu, Rui Ye, Jianqing Zhang, Siheng Chen, and Yanfeng Wang. 2025. FedMABench: Benchmarking Mobile Agents on Decentralized Heterogeneous User Data. arXiv:2503.05143 [cs.AI] doi:10.48550/arXiv.2503.05143
- [223] Wenhao Wang, Zijie Yu, Rui Ye, Jianqing Zhang, Guangyi Liu, Liang Liu, Siheng Chen, and Yanfeng Wang. 2025. FedMABench: Benchmarking Mobile GUI Agents on Decentralized Heterogeneous User Data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 26398–26419. doi:10.18653/v1/2025.emnlp-main.1341
- [224] Wenhao Wang, Mengying Yuan, Zijie Yu, Guangyi Liu, Rui Ye, Tian Jin, Siheng Chen, and Yanfeng Wang. 2025. MobileA3gent: Training Mobile GUI Agents Using Decentralized Self-Sourced Data from Diverse Users. In *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+NLP)*. 79–112. doi:10.18653/v1/2025.hcinlp-1.8
- [225] Xiaoqiang Wang and Bang Liu. 2024. OSCAR: Operating System Control via State-Aware Reasoning and Re-Planning. arXiv:2410.18963 [cs.AI] doi:10.48550/arXiv.2410.18963
- [226] Xuehui Wang, Zhenyu Wu, Jingjing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yuan Huang, Yiqian Liu, Xiao Zhang, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhao Wang. 2025. MMBench-GUI: Hierarchical Multi-Platform Evaluation Framework for GUI Agents.

- arXiv:2507.19478 [cs.CV] doi:10.48550/arXiv.2507.19478
- [227] Yiqin Wang, Haoji Zhang, Jingqi Tian, and Yansong Tang. 2025. Ponder & Press: Advancing Visual GUI Agent towards General Computer Control. In *Findings of the Association for Computational Linguistics: ACL 2025*. 1461–1473. doi:10.18653/v1/2025.findings-acl.76
 - [228] Yuanlei Wang, Liuzhou Zhang, Haohao Luo, and Ying Shen. 2025. INREACT: An Inspire-Then-Reinforce Training Framework For Multimodal GUI Agent. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. 9148–9160. doi:10.18653/v1/2025.findings-emnlp.486
 - [229] Yanxi Wang, Zhiling Zhang, Wenbo Zhou, Weiming Zhang, Jie Zhang, Qiannan Zhu, Yu Shi, Shuxin Zheng, and Jiyan He. 2026. GUIGuard: Toward a General Framework for Privacy-Preserving GUI Agents. arXiv:2601.18842 [cs.CR] doi:10.48550/arXiv.2601.18842
 - [230] Ziwei Wang, Weizhi Chen, Leyang Yang, Sheng Zhou, Shengchu Zhao, Hanbei Zhan, Jiongchao Jin, Liangcheng Li, Zirui Shao, and Jiajun Bu. 2025. MP-GUI: Modality Perception with MLLMs for GUI Understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 29711–29721. doi:10.1109/cvpr52734.2025.02766
 - [231] Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024. OfficeBench: Benchmarking Language Agents across Multiple Applications for Office Automation. arXiv:2407.19056 [cs.CL] doi:10.48550/arXiv.2407.19056
 - [232] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. arXiv:2501.11733 [cs.CL] doi:10.48550/arXiv.2501.11733
 - [233] Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025. Inducing Programmatic Skills for Agentic Tasks. arXiv:2504.06821 [cs.CL] doi:10.48550/arXiv.2504.06821
 - [234] 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. 2025. Look Before You Leap: A GUI-Critic-R1 Model for Pre-Operative Error Diagnosis in GUI Automation. arXiv:2506.04614 [cs.AI] doi:10.48550/arXiv.2506.04614
 - [235] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. AutoDroid: LLM-powered Task Automation in Android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 543–557. doi:10.1145/3636534.3649379
 - [236] Hao Wen, Shizuo Tian, Borislav Pavlov, Wenjie Du, Yixuan Li, Ge Chang, Shanhui Zhao, Jiacheng Liu, Yunxin Liu, Ya-Qin Zhang, and Yuanchun Li. 2025. AutoDroid-V2: Boosting SLM-based GUI Agents via Code Generation. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*. 223–235. doi:10.1145/3711875.3729134
 - [237] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2023. DroidBot-GPT: GPT-powered UI Automation for Android. arXiv:2304.07061 [cs.SE] doi:10.48550/arXiv.2304.07061
 - [238] Michael Wornow, Avanika Narayan, Krista Opsahl-Ong, Quinn McIntyre, Nigam Shah, and Christopher Ré. 2024. Automating the Enterprise with Foundation Models. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2805–2812. doi:10.14778/3681954.3681964
 - [239] Michael Wornow, Avanika Narayan, Ben Viggiano, Ishan Khare, Tathagat Verma, Tibor Thompson, Miguel Hernandez, Sudharsan Sundar, Chloe Trujillo, Krrish Chawla, Rongfei Lu, Justin Shen, Divya Nagaraj, Joshua Martinez, Vardhan Agrawal, Althea Hudson, Nigam Shah, and Christopher Ré. 2024. WONDERBREAD: A Benchmark for Evaluating Multimodal Foundation Models on Business Process Management Tasks. In *Advances in Neural Information Processing Systems* 37. 115963–116021. doi:10.52202/079017-3682
 - [240] Biao Wu, Yanda Li, Zhiwei Zhang, Yunchao Wei, Meng Fang, and Ling Chen. 2024. Foundations and Recent Trends in Multimodal Mobile Agents: A Survey. arXiv:2411.02006 [cs.AI] doi:10.48550/arXiv.2411.02006
 - [241] Benlong Wu, Yuang Qi, Xiuwei Shang, Weiming Zhang, Nenghai Yu, and Kejiang Chen. 2025. MMPro: A Decoupled Perception-Thinking-Execution Framework for Secure GUI Agent. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 4679–4687. doi:10.1145/3746027.3755553
 - [242] Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024. WIPI: A New Web Threat for LLM-Driven Web Agents. arXiv:2402.16965 [cs.CR] doi:10.48550/arXiv.2402.16965
 - [243] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. WebWalker: Benchmarking LLMs in Web Traversal. arXiv:2501.07572 [cs.CL] doi:10.48550/arXiv.2501.07572
 - [244] Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. 2025. GUI-Actor: Coordinate-Free Visual Grounding for GUI Agents. arXiv:2506.03143 [cs.CL] doi:10.48550/arXiv.2506.03143
 - [245] Qinchun Wu, Difei Gao, Kevin Qinghong Lin, Zhuoyu Wu, Xiangwu Guo, Peiran Li, Weichen Zhang, Hengxu Wang, and Mike Zheng Shou. 2024. GUI Action Narrator: Where and When Did That Action Take Place? arXiv:2406.13719 [cs.CV] doi:10.48550/arXiv.2406.13719

- [246] Qinzhuo Wu, Pengzhi Gao, Wei Liu, and Jian Luan. 2025. BacktrackAgent: Enhancing GUI Agent with Error Detection and Backtracking Mechanism. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 4250–4272. doi:10.18653/v1/2025.emnlp-main.212
- [247] Qingyuan Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. 2025. Advancing Autonomous VLM Agents via Variational Subgoal-Conditioned Reinforcement Learning. arXiv:2502.07949 [cs.LG] doi:10.48550/arXiv.2502.07949
- [248] Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. 2025. ReachAgent: Enhancing Mobile Agent via Page Reaching and Operation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 4760–4775. doi:10.18653/v1/2025.naacl-long.244
- [249] Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Liujian Liujianfeng, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024. MobileVLM: A Vision-Language Model for Better Intra- and Inter-UI Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 10231–10251. doi:10.18653/v1/2024.findings-emnlp.599
- [250] Wenyi Wu, Kun Zhou, Ruoxin Yuan, Vivian Yu, Stephen Wang, Zhiting Hu, and Biwei Huang. 2025. Auto-scaling Continuous Memory for GUI Agent. arXiv:2510.09038 [cs.AI] doi:10.48550/arXiv.2510.09038
- [251] Zongru Wu, Pengzhou Cheng, Zheng Wu, Tianjie Ju, Zhuosheng Zhang, and Gongshen Liu. 2025. Smoothing Grounding and Reasoning for MLLM-Powered GUI Agents with Query-Oriented Pivot Tasks. arXiv:2503.00401 [cs.CL] doi:10.48550/arXiv.2503.00401
- [252] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. OS-Copilot: Towards Generalist Computer Agents with Self-Improvement. arXiv:2402.07456 [cs.AI] doi:10.48550/arXiv.2402.07456
- [253] Zheng Wu, Heyuan Huang, Xingyu Lou, Xiangmou Qu, Pengzhou Cheng, Zongru Wu, Weiwen Liu, Weinan Zhang, Jun Wang, Zhaoxiang Wang, and Zhuosheng Zhang. 2025. VeriOS: Query-Driven Proactive Human-Agent-GUI Interaction for Trustworthy OS Agents. arXiv:2509.07553 [cs.CL] doi:10.48550/arXiv.2509.07553
- [254] Zongru Wu, Rui Mao, Zhiyuan Tian, Pengzhou Cheng, Tianjie Ju, Zheng Wu, Lingzhong Dong, Haiyue Sheng, Zhuosheng Zhang, and Gongshen Liu. 2025. See, Think, Act: Teaching Multimodal Agents to Effectively Interact with GUI by Identifying Toggles. arXiv:2509.13615 [cs.AI] doi:10.48550/arXiv.2509.13615
- [255] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. OS-ATLAS: A Foundation Action Model for Generalist GUI Agents. arXiv:2410.23218 [cs.CL] doi:10.48550/arXiv.2410.23218
- [256] Zhiqiang Xia, Xinyuan Zhang, Yang Li, Yuchen Liu, Runyu Shi, and Jiaming Xu. 2025. G-TADS: GUI Task-Ability Decoupling Strategy for High-Adaptability Multimodal Intelligent Agents. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. doi:10.1109/icme59968.2025.11209200
- [257] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2025. GuardAgent: Safeguard LLM Agents via Knowledge-Enabled Reasoning. In *International Conference on Machine Learning*. <https://openreview.net/forum?id=2nBcjCZrrP>
- [258] Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. 2025. UI-Genie: A Self-Improving Approach for Iteratively Boosting MLLM-based Mobile GUI Agents. arXiv:2505.21496 [cs.CL] doi:10.48550/arXiv.2505.21496
- [259] Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. 2025. GUI-explorer: Autonomous Exploration and Mining of Transition-aware Knowledge for GUI Agent. arXiv:2505.16827 [cs.AI] doi:10.48550/arXiv.2505.16827
- [260] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems* 37. 52040–52094. doi:10.52202/079017-1650
- [261] Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2023. OpenAgents: An Open Platform for Language Agents in the Wild. arXiv:2310.10634 [cs.CL] doi:10.48550/arXiv.2310.10634
- [262] Yuquan Xie, Zaijing Li, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Dongmei Jiang, and Liqiang Nie. 2025. Mirage-1: Augmenting and Updating GUI Agent with Hierarchical Multimodal Skills. arXiv:2506.10387 [cs.AI] doi:10.48550/arXiv.2506.10387
- [263] Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. 2024. Understanding the Weakness of Large Language Model Agents within a Complex Android Environment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6061–6072. doi:10.1145/3637528.3671650
- [264] Tao Xiong, Xavier Hu, Yurun Chen, Yuhang Liu, Changqiao Wu, Pengzhi Gao, Wei Liu, Jian Luan, and Shengyu Zhang. 2025. GUI-PRA: Process Reward Agent for GUI Tasks. arXiv:2509.23263 [cs.AI] doi:10.48550/arXiv.2509.23263

- [265] Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. 2024. AdvWeb: Controllable Black-box Attacks on VLM-powered Web Agents. arXiv:2410.17401 [cs.CR] doi:10.48550/arXiv.2410.17401
- [266] Hai-Ming Xu, Qi Chen, Lei Wang, and Lingqiao Liu. 2025. Attention-Driven GUI Grounding: Leveraging Pretrained Multimodal Large Language Models Without Fine-Tuning. *Proceedings of the AAAI Conference on Artificial Intelligence* 39, 8 (2025), 8851–8859. doi:10.1609/aaai.v39i8.32957
- [267] Kevin Xu, Yeganeh Kordi, Tanay Nayak, Adi Asija, Yizhong Wang, Kate Sanders, Adam Byerly, Jingyu Zhang, Benjamin Van Durme, and Daniel Khashabi. 2025. TurkingBench: A Challenge Benchmark for Web Agents. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 3694–3710. doi:10.18653/v1/2025.naacl-long.188
- [268] Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica S Lam. 2021. Grounding Open-Domain Instructions to Automate Web Support Tasks. arXiv:2103.16057 [cs.CL] doi:10.48550/arXiv.2103.16057
- [269] Ran Xu, Kaixin Ma, Wenhao Yu, Hongming Zhang, Joyce C. Ho, Carl Yang, and Dong Yu. 2025. Retrieval-augmented GUI Agents with Generative Guidelines. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 17877–17886. doi:10.18653/v1/2025.emnlp-main.902
- [270] Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, Anjie Yang, Zhaoxuan Jin, Jianbo Deng, Philip Torr, Bernard Ghanem, and Guohao Li. 2025. CRAB: Cross-environment Agent Benchmark for Multimodal Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*. 21607–21647. doi:10.18653/v1/2025.findings-acl.1113
- [271] Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. 2025. MobileRL: Online Agentic Reinforcement Learning for Mobile GUI Agents. arXiv:2509.18119 [cs.LG] doi:10.48550/arXiv.2509.18119
- [272] Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2025. AndroidLab: Training and Systematic Benchmarking of Android Autonomous Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2144–2166. doi:10.18653/v1/2025.acl-long.107
- [273] Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2024. AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials. arXiv:2412.09605 [cs.CL] doi:10.48550/arXiv.2412.09605
- [274] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguis: Unified Pure Vision Agents for Autonomous GUI Interaction. arXiv:2412.04454 [cs.CL] doi:10.48550/arXiv.2412.04454
- [275] Yibin Xu, Liang Yang, Hao Chen, Hua Wang, Zhi Chen, and Yaohua Tang. 2025. DeskVision: Large Scale Desktop Region Captioning for Advanced GUI Agents. arXiv:2503.11170 [cs.CL] doi:10.48550/arXiv.2503.11170
- [276] Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An Illusion of Progress? Assessing the Current State of Web Agents. arXiv:2504.01382 [cs.AI] doi:10.48550/arXiv.2504.01382
- [277] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. 2023. GPT-4V in Wonderland: Large Multimodal Models for Zero-Shot Smartphone GUI Navigation. arXiv:2311.07562 [cs.CV] doi:10.48550/arXiv.2311.07562
- [278] Zihe Yan, Jiaping Gui, Zhuosheng Zhang, and Gongshen Liu. 2025. LaSM: Layer-wise Scaling Mechanism for Defending Pop-up Attack on GUI Agents. arXiv:2507.10610 [cs.CR] doi:10.48550/arXiv.2507.10610
- [279] Jiaxi Yang and Haowen Hou. 2025. RWKV-UI: UI Understanding with Enhanced Perception and Reasoning. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. doi:10.1109/icme59968.2025.11210007
- [280] Jingqi Yang, Zeng Song, Jiawei Chen, Mingli Song, Zhou Sheng, linjun sun, Xiaogang Ouyang, Chun Chen, and Can Wang. 2025. GUI-Robust: A Comprehensive Dataset for Testing GUI Agent Robustness in Real-World Anomalies. arXiv:2506.14477 [cs.AI] doi:10.48550/arXiv.2506.14477
- [281] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, and Jianfeng Gao. 2025. Magma: A Foundation Model for Multimodal AI Agents. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14203–14214. doi:10.1109/cvpr52734.2025.01325
- [282] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. arXiv:2410.13825 [cs.AI] doi:10.48550/arXiv.2410.13825
- [283] Pei Yang, Hai Ci, and Mike Zheng Shou. 2025. macOSWorld: A Multilingual Interactive Benchmark for GUI Agents. arXiv:2506.04135 [cs.AI] doi:10.48550/arXiv.2506.04135
- [284] Qi Yang, Weichen Bi, Haiyang Shen, Yaoqi Guo, and Yun Ma. 2025. PixelWeb: The First Web GUI Dataset with Pixel-Wise Labels. arXiv:2504.16419 [cs.CV] doi:10.48550/arXiv.2504.16419

- [285] Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin Peng, Huan Zhang, Jianfeng Gao, and Tong Zhang. 2026. GUI-Libra: Training Native GUI Agents to Reason and Act with Action-aware Supervision and Partially Verifiable RL. *arXiv:2602.22190* [cs.LG] [doi:10.48550/arXiv.2602.22190](https://doi.org/10.48550/arXiv.2602.22190)
- [286] Xiao Yang, Jiawei Chen, Jun Luo, Zhengwei Fang, Yinpeng Dong, Hang Su, and Jun Zhu. 2025. MLA-Trust: Benchmarking Trustworthiness of Multimodal LLM Agents in GUI Environments. *arXiv:2506.01616* [cs.AI] [doi:10.48550/arXiv.2506.01616](https://doi.org/10.48550/arXiv.2506.01616)
- [287] Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2025. Aria-UI: Visual Grounding for GUI Instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*. 22418–22433. [doi:10.18653/v1/2025.findings-acl.1152](https://doi.org/10.18653/v1/2025.findings-acl.1152)
- [288] Yulong Yang, Xinshan Yang, Shuaidong Li, Chenhao Lin, Zhengyu Zhao, Chao Shen, and Tianwei Zhang. 2024. Systematic Categorization, Construction and Evaluation of New Attacks against Multi-modal Mobile GUI Agents. *arXiv:2407.09295* [cs.CR] [doi:10.48550/arXiv.2407.09295](https://doi.org/10.48550/arXiv.2407.09295)
- [289] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems 35*. 20744–20757. [doi:10.52202/068431-1508](https://doi.org/10.52202/068431-1508)
- [290] Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. 2025. Mobile-Agent-v3: Fundamental Agents for GUI Automation. *arXiv:2508.15144* [cs.AI] [doi:10.48550/arXiv.2508.15144](https://doi.org/10.48550/arXiv.2508.15144)
- [291] Jian Ye, Xin Zhao, Xuanchen Feng, Xiang Gao, Xin Zhang, Lanting Li, and Wentao Hong. 2025. GG-R1: A Fine-Grained RFT Approach for GUI Grounding. In *2025 9th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*. 16–20. [doi:10.1109/ic-nidc67200.2025.11390507](https://doi.org/10.1109/ic-nidc67200.2025.11390507)
- [292] Suyu Ye, Haojun Shi, Darren Shih, Hyokun Yun, Tanya G. Roosta, and Tianmin Shu. 2026. RealWebAssist: A Benchmark for Long-Horizon Web Assistance with Real-World Users. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 40 (2026), 34441–34449. [doi:10.1609/aaai.v40i40.40742](https://doi.org/10.1609/aaai.v40i40.40742)
- [293] Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2023. ProAgent: From Robotic Process Automation to Agentic Process Automation. *arXiv:2311.10751* [cs.RO] [doi:10.48550/arXiv.2311.10751](https://doi.org/10.48550/arXiv.2311.10751)
- [294] Biao Yi, Xueyu Hu, Yurun Chen, Shengyu Zhang, Hongxia Yang, and Fan Wu. 2026. EcoAgent: An Efficient Device-Cloud Collaborative Multi-Agent Framework for Mobile Automation. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 35 (2026), 29838–29846. [doi:10.1609/aaai.v40i35.40230](https://doi.org/10.1609/aaai.v40i35.40230)
- [295] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. *Lecture Notes in Computer Science* (2024), 240–255. [doi:10.1007/978-3-031-73039-9_14](https://doi.org/10.1007/978-3-031-73039-9_14)
- [296] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and Bo Li. 2025. Enhancing Visual Grounding for GUI Agents via Self-Evolutionary Reinforcement Learning. *arXiv:2505.12370* [cs.AI] [doi:10.48550/arXiv.2505.12370](https://doi.org/10.48550/arXiv.2505.12370)
- [297] Zhixiong Zeng, Jing Huang, Liming Zheng, Wenkang Han, Yufeng Zhong, Lei Chen, Longrong Yang, Yingjie Chu, Yuzhi He, and Lin Ma. 2025. Ultron: Foundational GUI Agent with Advanced Perception and Planning. *arXiv:2508.21767* [cs.CV] [doi:10.48550/arXiv.2508.21767](https://doi.org/10.48550/arXiv.2508.21767)
- [298] Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. 2026. TongUI: Internet-Scale Trajectories from Multimodal Web Tutorials for Generalized GUI Agents. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 15 (2026), 12367–12375. [doi:10.1609/aaai.v40i15.38229](https://doi.org/10.1609/aaai.v40i15.38229)
- [299] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liquan Li, Si Qin, Kang Yu, Minghua Ma, G. M. Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Large Language Model-Brained GUI Agents: A Survey. *arXiv:2411.18279* [cs.AI] [doi:10.48550/arXiv.2411.18279](https://doi.org/10.48550/arXiv.2411.18279)
- [300] Chaoyun Zhang, He Huang, Chiming Ni, Jian Mu, Si Qin, Shilin He, Lu Wang, Fangkai Yang, Pu Zhao, Chao Du, Liquan Li, Yu Kang, Zhao Jiang, Suzhen Zheng, Rujia Wang, Jiaxu Qian, Minghua Ma, Jian-Guang Lou, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2025. UFO2: The Desktop AgentOS. *arXiv:2504.14603* [cs.AI] [doi:10.48550/arXiv.2504.14603](https://doi.org/10.48550/arXiv.2504.14603)
- [301] Chaoyun Zhang, Liquan Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. UFO: A UI-Focused Agent for Windows OS Interaction. *arXiv:2402.07939* [cs.HC] [doi:10.48550/arXiv.2402.07939](https://doi.org/10.48550/arXiv.2402.07939)
- [302] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. AppAgent: Multimodal Agents as Smartphone Users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–20. [doi:10.1145/3706598.3713600](https://doi.org/10.1145/3706598.3713600)
- [303] Danqing Zhang, Balaji Rama, Jingyi Ni, Shiyang He, Fu Zhao, Kunyu Chen, Arnold Chen, and Junyu Cao. 2025. LiteWebAgent: The Open-Source Suite for VLM-Based Web-Agent Applications. *arXiv:2503.02950* [cs.AI] [doi:10.48550/arXiv.2503.02950](https://doi.org/10.48550/arXiv.2503.02950)

- 48550/arXiv.2503.02950
- [304] Danyang Zhang, Zhennan Shen, Rui Xie, Situo Zhang, Tianbao Xie, Zihan Zhao, Siyuan Chen, Lu Chen, Hongshen Xu, Ruisheng Cao, and Kai Yu. 2023. Mobile-Env: Building Qualified Evaluation Benchmarks for LLM-GUI Interaction. arXiv:2305.08144 [cs.AI] doi:10.48550/arXiv.2305.08144
 - [305] Junlei Zhang, Zichen Ding, Chang Ma, Zijie Chen, Qiushi Sun, Zhenzhong Lan, and Junxian He. 2025. Breaking the Data Barrier – Building GUI Agents Through Task Generalization. arXiv:2504.10127 [cs.AI] doi:10.48550/arXiv.2504.10127
 - [306] Junlei Zhang, Zichen Ding, Chang Ma, Zijie Chen, Qiushi Sun, Zhenzhong Lan, and Junxian He. 2025. Breaking the Data Barrier – Building GUI Agents Through Task Generalization. arXiv:2504.10127 [cs.AI] doi:10.48550/arXiv.2504.10127
 - [307] Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the Zoo: Chain-of-Action-Thought for GUI Agents. (2024), 12016–12031. doi:10.18653/v1/2024.findings-emnlp.702
 - [308] Jiwen Zhang, Ya-Qi Yu, Minghui Liao, WenTao Li, Jihao Wu, and Zhongyu Wei. 2025. UI-Hawk: Unleashing the Screen Stream Understanding for Mobile GUI Agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 18228–18247. doi:10.18653/v1/2025.emnlp-main.920
 - [309] Jiayi Zhang, Chuang Zhao, Yihan Zhao, Zhaoyang Yu, Ming He, and Jianping Fan. 2024. MobileExperts: A Dynamic Tool-Enabled Agent Team in Mobile Devices. arXiv:2407.03913 [cs.AI] doi:10.48550/arXiv.2407.03913
 - [310] Li Zhang, Longxi Gao, and Mengwei Xu. 2025. Does Chain-of-Thought Reasoning Help Mobile GUI Agent? An Empirical Study. arXiv:2503.16788 [cs.AI] doi:10.48550/arXiv.2503.16788
 - [311] Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024. LlamaTouch: A Faithful and Scalable Testbed for Mobile UI Task Automation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–13. doi:10.1145/3654777.3676382
 - [312] Le Zhang, Yixiong Xiao, Xinjiang Lu, Jingjia Cao, Yusai Zhao, Jingbo Zhou, Lang An, Zikan Feng, Wanxiang Sha, Yu Shi, Congxi Xiao, Jian Xiong, Yankai Zhang, Hua Wu, and Haifeng Wang. 2026. OmegaUse: Building a General-Purpose GUI Agent for Autonomous Task Execution. arXiv:2601.20380 [cs.AI] doi:10.48550/arXiv.2601.20380
 - [313] Ruichen Zhang, Mufan Qiu, Zhen Tan, Mohan Zhang, Vincent Lu, Jie Peng, Kaidi Xu, Leandro Z. Agudelo, Peter Qian, and Tianlong Chen. 2025. Symbiotic Cooperation for Web Agents: Harnessing Complementary Strengths of Large and Small LLMs. arXiv:2502.07942 [cs.MA] doi:10.48550/arXiv.2502.07942
 - [314] Shuning Zhang, Jingruo Chen, Zhiqi Gao, Jiajing Gao, Xin Yi, and Hewu Li. 2025. Characterizing Unintended Consequences in Human-GUI Agent Collaboration for Web Browsing. arXiv:2505.09875 [cs.HC] doi:10.48550/arXiv.2505.09875
 - [315] Shaoqing Zhang, Zhuosheng Zhang, Kehai Chen, Xinbei Ma, Muyun Yang, Tiejun Zhao, and Min Zhang. 2024. Dynamic Planning for LLM-based Graphical User Interface Automation. (2024), 1304–1320. doi:10.18653/v1/2024.findings-emnlp.70
 - [316] Yitong Zhang, Xing-Bin Li, Liyi Cai, and Jia Li. 2025. Environmental Injection Attacks against GUI Agents in Realistic Dynamic Environments. arXiv:2509.11250 [cs.CR] doi:10.48550/arXiv.2509.11250
 - [317] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2025. WebPilot: A Versatile and Autonomous Multi-Agent System for Web Task Execution with Strategic Exploration. *Proceedings of the AAAI Conference on Artificial Intelligence* 39, 22 (2025), 23378–23386. doi:10.1609/aaai.v39i22.34505
 - [318] Yanzhe Zhang, Tao Yu, and Diyi Yang. 2025. Attacking Vision-Language Computer Agents via Pop-ups. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8387–8401. doi:10.18653/v1/2025.acl-long.411
 - [319] Zhisong Zhang, Tianqing Fang, Kaixin Ma, Wenhao Yu, Hongming Zhang, Haitao Mi, and Dong Yu. 2025. WebRollback: Enhancing Web Agents with Explicit Rollback Mechanisms. arXiv:2504.11788 [cs.CL] doi:10.48550/arXiv.2504.11788
 - [320] Zhizheng Zhang, Wenxuan Xie, Xiaoyi Zhang, and Yan Lu. 2023. Reinforced UI Instruction Grounding: Towards a Generic UI Task Automation API. arXiv:2310.04716 [cs.CV] doi:10.48550/arXiv.2310.04716
 - [321] Zhuosheng Zhang and Aston Zhang. 2024. You Only Look at Screens: Multimodal Chain-of-Action Agents. In *Findings of the Association for Computational Linguistics ACL 2024*. 3132–3149. doi:10.18653/v1/2024.findings-acl.186
 - [322] Di Zhao, Longhui Ma, Siwei Wang, Miao Wang, and Zhao Lv. 2025. COLA: A Scalable Multi-Agent Framework For Windows UI Task Automation. arXiv:2503.09263 [cs.MA] doi:10.48550/arXiv.2503.09263
 - [323] Di Zhao, Longhui Ma, Siwei Wang, Miao Wang, and Zhao Lv. 2025. COLA: Collaborative Multi-Agent Framework with Dynamic Task Scheduling for GUI Automation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 4570–4593. doi:10.18653/v1/2025.emnlp-main.227
 - [324] Yuyang Zhao, Wentao Shi, Fuli Feng, and Xiangnan He. 2025. AppAgent-Pro: A Proactive GUI Agent System for Multidomain Information Integration and User Assistance. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 6767–6771. doi:10.1145/3746252.3761473

- [325] Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. 2025. AgentDAM: Privacy Leakage Evaluation for Autonomous Web Agents. *arXiv:2503.09780* [cs.AI] [doi:10.48550/arXiv.2503.09780](#)
- [326] Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025. SkillWeaver: Web Agents can Self-Improve by Discovering and Honing Skills. *arXiv:2504.07079* [cs.AI] [doi:10.48550/arXiv.2504.07079](#)
- [327] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a Generalist Web Agent, if Grounded. *arXiv:2401.01614* [cs.IR] [doi:10.48550/arXiv.2401.01614](#)
- [328] Boyuan Zheng, Boyu Gou, Scott Salisbury, Zheng Du, Huan Sun, and Yu Su. 2024. WebOlympus: An Open Platform for Web Agents on Live Websites. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 187–197. [doi:10.18653/v1/2024.emnlp-demo.20](#)
- [329] Jiani Zheng, Lu Wang, Fangkai Yang, Chaoyun Zhang, Lingrui Mei, Wenjie Yin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2025. VEM: Environment-Free Exploration for Training GUI Agent with Value Environment Model. *arXiv:2502.18906* [cs.LG] [doi:10.48550/arXiv.2502.18906](#)
- [330] Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. 2024. AgentStudio: A Toolkit for Building General Virtual Agents. *arXiv:2403.17918* [cs.AI] [doi:10.48550/arXiv.2403.17918](#)
- [331] 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. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv:2307.13854* [cs.AI] [doi:10.48550/arXiv.2307.13854](#)
- [332] Xurui Zhou, Gongwei Chen, Yuquan Xie, Zaijing Li, Kaiwen Zhou, Shuai Wang, Shuo Yang, Zhuotao Tian, and Rui Shao. 2025. HiconAgent: History Context-aware Policy Optimization for GUI Agents. *arXiv:2512.01763* [cs.CV] [doi:10.48550/arXiv.2512.01763](#)
- [333] Yuqi Zhou, Shuai Wang, Sunhao Dai, Qinglin Jia, Zhaocheng Du, Zhenhua Dong, and Jun Xu. 2025. CHOP: Mobile Operating Assistant with Constrained High-frequency Optimized Subtask Planning. *arXiv:2503.03743* [cs.AI] [doi:10.48550/arXiv.2503.03743](#)
- [334] Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine, and Erran Li. 2024. Proposer-Agent-Evaluator(PAE): Autonomous Skill Discovery For Foundation Model Internet Agents. *arXiv:2412.13194* [cs.LG] [doi:10.48550/arXiv.2412.13194](#)
- [335] Zichen Zhu, Hao Tang, Yansi Li, Dingye Liu, Hongshen Xu, Kunyao Lan, Danyang Zhang, Yixuan Jiang, Hao Zhou, Chenrun Wang, Situo Zhang, Liangtai Sun, Yixiao Wang, Yuheng Sun, Lu Chen, and Kai Yu. 2025. MobA: Multifaceted Memory-Enhanced Adaptive Planning for Efficient Mobile Task Automation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*. 535–549. [doi:10.18653/v1/2025.naacl-demo.43](#)
- [336] Meng Ziyang, Yu Dai, Zezheng Gong, Shaoxiong Guo, Minglong Tang, and Tongquan Wei. 2024. VGA: Vision GUI Assistant - Minimizing Hallucinations through Image-Centric Fine-Tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 1261–1279. [doi:10.18653/v1/2024.findings-emnlp.68](#)