

Goal Completion and State Preservation: Case Studies of GUI-Derived Agent Tools

Dekuan Li

qvnianjinri@outlook.com

Independent Researcher

Research Article

Keywords: Software agents, GUI-derived workflows, Empirical software engineering, Task completion, State preservation

Posted Date: October 22nd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11463966/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Goal Completion and State Preservation: Case Studies of GUI-Derived Agent Tools

Dekuan Li

Abstract Recorded graphical user interface (GUI) operations can be reused as parameterized tools for software agents. We evaluate their effects on complete-task coverage and application state. In one software-as-a-service (SaaS) workflow family, a frozen GUI-derived workflow attains 33/33 local native-compatible core points on three instances, versus raw code's 8/33, 8/33 and 18/33. Additional checks are 16/20 versus 0/20, 0/20 and 6/20. Two prospectively fixed variants reuse the implementation; tool agents use six and five model calls, compared with 121 and 70 for raw code. Both agents finish normally on the last instance. Residual requirements prevent whole-task acceptance, and broad collateral preservation remains unknown. An earlier Magento study finds that all eight complete records reach the requested prices, but native save routes differ in promotion-cache consistency and all introduce collateral changes. A posthoc intervention finds blocked description propagation in 4 of 12 target pairs. These case studies identify a repeated within-family completion benefit and show why workflow evaluation must also examine persistent state.

Keywords Software agents · GUI-derived workflows · Empirical software engineering · Task completion · State preservation

1 Introduction

Completing a multi-application request requires linked business operations. A code agent must connect customers, invoices, payments and event records while preserving identifiers and native accounting behavior. A recorded GUI workflow supplies examples of these operations; parameterizing its requests lets later agents reuse the business path.

Dekuan Li
Independent Researcher, Chengdu, China
E-mail: qvnianjinri@outlook.com

We ask whether this path improves execution along two dimensions: coverage of the complete request and preservation of application state. These outcomes can diverge. An invoice and payment can exist without the expected ledger balance; a price can be correct while a promotion cache is stale.

Our main comparison uses complete tasks from one SaaS-Bench family spanning Twenty CRM, BigCapital and Pretix. A development comparison is followed by two prospectively fixed parameter variants with unchanged model, budgets and workflow. Static tools reach all 33 local core points on each instance, whereas raw code reaches 8, 8 and 18. An earlier native Magento study traces side effects through product saves and a later description update.

The paper contributes two linked findings:

- A GUI-derived workflow covers more core and additional requirements than raw code on all three SaaS instances, with fewer model decisions, including a pair in which both agents finish normally.
- Native execution paths preserve different parts of application state. The Magento records connect promotion-cache consistency and description inheritance to specific save operations, showing what must be measured beyond target completion.

The raw/tool contrast evaluates the combined workflow package: recorded examples, researcher-provided parameterization and orchestration, and a restricted execution interface. It does not isolate GUI provenance from these implementation choices.

Figure 1 summarizes the interface and evaluation. We describe the GUI evidence and researcher-provided implementation separately so that the observed benefit can be tied to the workflow actually evaluated.

Research questions. RQ1 asks how a frozen workflow changes requirement coverage on complete requests within one business family. RQ2 asks which persistent-state effects remain when requested target values are achieved. We report descriptive paired outcomes and mechanisms, rather than population-level error rates or significance tests over individual checks.

2 From GUI recordings to executable workflows

2.1 Recorded evidence and workflow construction

The SaaS implementation, C006, uses 77 operations derived from sanitized HTTP exchanges recorded during real training GUI use and associated with saved GUI observations. Each operation retains its request method, route template, payload structure and response paths for extracting identifiers. New business names, amounts and dates become goal parameters; identifiers are obtained from current responses or lookups. The records supply examples of valid business calls, rather than a complete application specification.

The study implementation provides extraction and parameterization code, a goal schema, workflow ordering, value conversions, document templates,

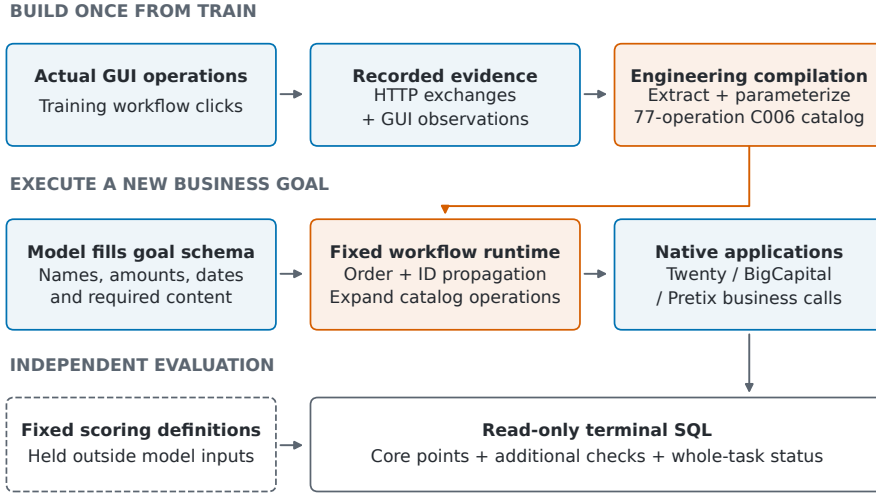


Fig. 1 The evaluated SaaS workflow. Real training GUI use supplies observed business requests and responses. Researcher-provided extraction, parameterization and orchestration turn these observations into a frozen interface; the execution model fills its goal schema. Independent terminal-state scoring uses protected evaluator definitions and answer states held outside model inputs

identifier propagation and the execution runtime. The execution agent maps a natural-language request to the supported two-milestone goal and submits its parameters through the protected interface. It can inspect the schema and available account or organizer choices, perform allowed reads, validate or preview a goal, execute the macro and inspect status. The macro expands the goal into the fixed catalog sequence. Appendix B gives the application-specific bindings.

2.2 Conditions and complete-task allocation

We evaluate three local parameter instances of SaaS-Bench’s **business_032** (UniPat-AI, 2026): DEV008, DEV009 and DEV010. Each complete request has 19 requirements connecting Twenty customer relationship management (CRM) records and follow-up tasks; BigCapital customers, invoices, payments and accounting state; and a Pretix event, ticket, quota and voucher. Each actor attempts the entire request in one episode.

DEV008 compares four conditions: A, actual pixel-based GUI interaction; B, raw code execution; C, the static GUI-derived workflow; and D, the same workflow with runtime state guards and bounded recovery. C and D share the catalog, implementation, schema, prompt and execution identity. Live dependency checks resolve selected labels before writes in both modes; D adds state guards and the recovery policy. Each condition starts from a full reset with native database-equality and identity checks. C006 is frozen before DEV008.

After DEV008, the extension fixes two new parameter variants and four raw/static slots before execution. Every allocated slot runs once using unchanged C006. These tasks test reuse within the same business family. All nine saved SaaS actor receipts, including one GUI technical repeat, request `gpt-5.6-luna` with `low` reasoning. Common scientific limits are 400 actions, 7,200 seconds, 2,000,000 input-plus-output tokens, 300 seconds per model turn and 180 seconds per command.

2.3 Use of AI tools in the research process

The research question originated with the author. OpenAI Codex assisted study design, software implementation, experiment orchestration, analysis scripts, and manuscript drafting and revision. These research-assistance activities are distinct from the participant-model executions described below. The author reviewed the manuscript and is responsible for its content; measured outcomes come from the recorded application executions and evaluator outputs.

Baseline access and information presented to the actors. The SaaS raw-code actor can execute shell commands inside each selected application container, inspect and edit application source and synthetic data, and use local services. It also has a generic authenticated HTTP helper using the configured synthetic identity; it receives no recorded GUI pathway. The protected actor instead receives the fixed workflow interface. Both retain eight recent turns and at most 12,000 characters of actor-maintained notes. Ordinary observations are limited to 16,000 characters; the protected actor’s schema response has a separate 65,536-character limit. Thus equal model and scientific budgets do not imply identical information presentation or action granularity. One workflow invocation can expand into many native operations, so model-call counts measure decisions at the supplied interface, not comparable counts of application operations.

Units and methodological material. The SaaS comparison contains three parameter instances of one business family, with one raw/static execution per instance; the additional DEV008 conditions and technical repeat do not increase that task count. The 33 core points are weights assigned to 18 checks, not independent observations. Online Resource 2 provides the three synthetic requests, their requirement-to-check mappings and weights, the original and local scoring definitions, and excerpts of the frozen actor interfaces and C006 workflow. These are study-method records exported from existing fixed sources. They are separate from the saved-result recomputation package and do not constitute a new execution or a complete runnable native environment.

2.4 Independent outcome measures

Read-only SQL against each terminal container measures 18 core checks worth 33 weighted points and 20 additional checks. Preserved backups are bound to the same immutable container IDs. We report both the original benchmark evaluator and the predeclared local native-compatible evaluator. Whole-task acceptance and collateral preservation are separate fields; a full core score is not itself complete task satisfaction.

The accounting distinction is material. The original evaluator sums invoice **BALANCE**, which retains the invoiced amount after payment in this installation. The local evaluator reads the native accounts-receivable ledger. For the DEV008 tool runs, that ledger agrees with the native customer report and adds three core points, changing 30 to 33. Raw code’s direct invoice/payment writes leave the ledger unchanged, changing its score from 11 to 8. Both definitions are retained in Appendix A. The local additional checks also replace two invoice-balance expressions with native amount-due calculations and normalize whitespace before trimming complete task bodies. Online Resource 2 retains these exact amendments alongside the original checks.

3 SaaS results: coverage and execution cost

3.1 Completion across the three instances

Figure 2 shows the paired raw/static results. Static tools attain 33/33 local core points and 16/20 additional checks on every instance. Raw code attains 8/33 and 0/20 on DEV008 and DEV009, and 18/33 and 6/20 on DEV010. Thus the local core advantage is 25, 25 and 15 points, respectively. The original evaluator gives the same ordering: 30/33 for tools versus 11/33, 11/33 and 21/33 for raw code. Its additional scores are 14/20 for tools versus 0/20, 0/20 and 6/20.

DEV008’s guarded workflow ties the static workflow at 33/33 and 16/20. Both finish normally after six calls. The actual GUI endpoints score 0/33 and 0/20: GUI001 ends on a controller response-decoding error, and its separate technical repeat ends on an invalid key action. Table 2 retains all five records; Appendix F distinguishes their termination causes.

3.2 What the workflow leaves incomplete

Every SaaS attempt has whole-task FAIL and collateral UNKNOWN. In the tool runs, the four unmet additional checks concern exact event cardinality and a complete task-body string. The workflow deactivates an extra ticket but retains its ticket, quota and link rows. Its document template also adds surrounding text to an actor-supplied complete body. These failures locate

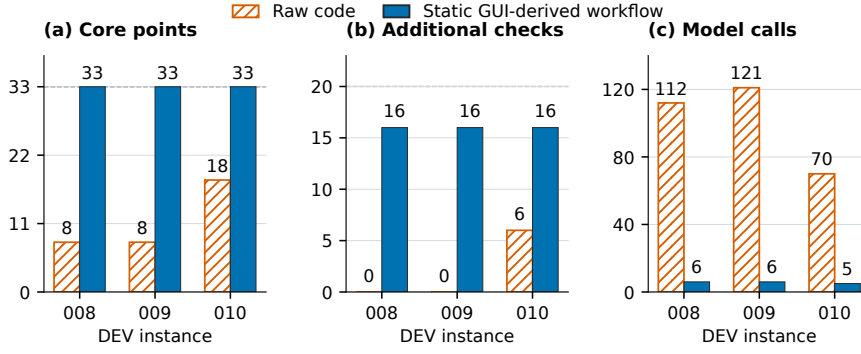


Fig. 2 Raw code and static workflow tools on three complete-task parameter instances of one SaaS business family. Core and additional scores use the predeclared local native-compatible evaluator. Each bar represents one saved execution; DEV009 and DEV010 were fixed prospectively with unchanged C006. Calls count model responses. Raw code ends on an invalid command in DEV008 and the scientific token limit in DEV009; both agents finish normally in DEV010. All six whole-task outcomes remain FAIL, and collateral preservation is UNKNOWN. Full condition and cost tables appear in Appendix A

the remaining gap in the shared workflow’s coverage. DEV008’s private exact-cardinality checks are stricter than the visible wording about the named ticket and quota, so those failures do not all represent explicit instruction violations.

3.3 Prospective reuse and model decisions

The two new tool agents finish after six and five calls, using 106,848 and 73,180 execution tokens. DEV009 raw reaches the scientific token limit after its 121st model response, with 120 actions and 2,013,080 tokens; the last proposed action is not dispatched. DEV010 raw finishes normally after 70 calls and actions, using 1,247,320 tokens. Its paired tool run therefore retains higher completion even when both actors choose normal termination.

In DEV008, static and guarded execution use 96,162 and 96,393 tokens, compared with 1,741,353 for raw code. These are measured costs to each actor’s endpoint. Appendix A reports every attempt, including the GUI interruption and repeat, with separate calls, actions, elapsed time and token accounting.

4 Magento results: target values and persistent state

4.1 Native routes and evaluation

The earlier Magento study asks whether executing a recorded business route preserves state beyond the requested price. A controller-assisted GUI acquisition uses a distinct product family, MH01, and records a native price save

and a rejected negative price. One `gpt-5.6-luna` call at `low` reasoning produces the predicate $p \geq 0$, also validated by the provided operation wrapper. Acquisition uses 7,723 input and 402 output tokens in a 17.54-second model turn.

Protocol 2.5 allocates four conditions to each of two held-out product families. Case 1 requests \$79 for the black, size-M Teton Pullover Hoodie. Case 2 requests \$69 for the black and blue size-XL Bruno Compete Hoodie variants, giving $2 \times 4 = 8$ slots. Here A is raw code, B is code with recorded GUI pathway hints, C is the static catalog, and D is the same catalog with runtime guards. These A/B labels differ from the SaaS allocation; Magento B is a code agent, not mouse interaction. All conditions request the same model and reasoning setting and use OS identity UID 10001.

A and B share source inspection, file editing and ordinary code execution inside the clone. A receives only the business goal; B also receives the GUI pathway. C and D share the catalog, schemas, prompt, identity and native write implementation. The model proposal supplies metadata, parameter/evidence bindings, a path and the nonnegative-price predicate. An engineered macro binds the observed Product-model save through ProductRepository to the shared native resource. The executor and recipe grammar are researcher-provided.

Independent evaluation checks the requested persistent prices and native quotes (G), preservation of a fixed state boundary through execution and a maintenance probe (C), and evidence completeness within that boundary (E). Safe completion is

$$S = G \wedge C \wedge E.$$

The verifier uses the fixed application source and task policy, independently of the learned predicate. Appendix C details the native audit and common budgets.

4.2 Correct prices, different side effects

Table 1 reports all eight complete comparison records, including a separate retry after one case-2 A infrastructure interruption. The original allocation therefore has seven evaluable first attempts and one interrupted slot. All eight complete records pass canonical price, actor-end quote and post-reindex goals, and each has complete bounded evidence and a confirmed collateral error.

Figure 3 illustrates two state mechanisms. In case 2, the additional raw-code attempt uses `saveAttribute(price)`. Two native row updates change prices from 63 to 69 without a source edit, but the existing 20% promotion retains a cached price of 50.40 rather than $69 \times 0.8 = 55.20$. The stale values cover six dated business keys, three dates for each of two products, and persist after the original price/stock/search reindex. Repository-based execution in B, C and D regenerates all six values to 55.20.

The full-object saves also create non-price EAV rows. Case-1 A adds eight store-1 attribute rows copied from the defaults. B adds twelve, including four

Table 1 All complete records, including the separate case-2 A retry. Entity-attribute-value (EAV) counts distinct non-price row additions, not duplicate phase findings. Rule counts target entities with inconsistent CatalogRule derived prices. The two units are not added together. Case-1 C retains its original runner-error terminal despite an evaluable target state. Cached input is already included in tokens

Case	Condition	Goal	EAV	Rule	Safe	Calls	Tokens	Seconds
1	A: Code	yes	8	0	no	9	100,671	207.86
1	B: GUI hints	yes	12	0	no	9	100,595	196.94
1	C: Static tools	yes	4	0	no	3	22,698	65.11
1	D: Guarded tools	yes	4	0	no	3	22,675	64.89
2	A: Code	yes	0	2	no	7	81,755	158.99
2	B: GUI hints	yes	24	0	no	7	76,150	166.96
2	C: Static tools	yes	8	0	no	5	39,243	106.02
2	D: Guarded tools	yes	8	0	no	4	36,488	85.87

image attributes set to `no_selection`; C and D each add four image defaults at store 0. In case 2, B adds 24 rows across two targets, while C and D each add eight. These changes persist through maintenance. The row counts represent distinct additions; actor-end and post-reindex observations are not separate writes. No model execution in this allocation modifies application source.

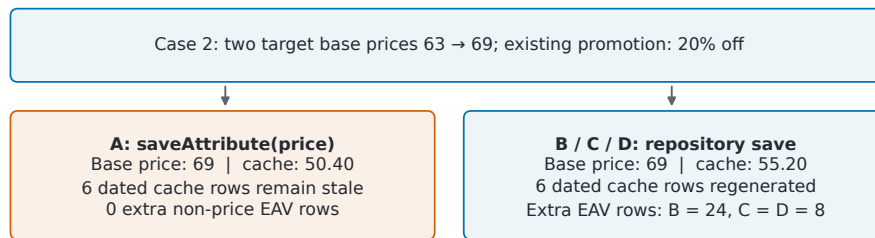
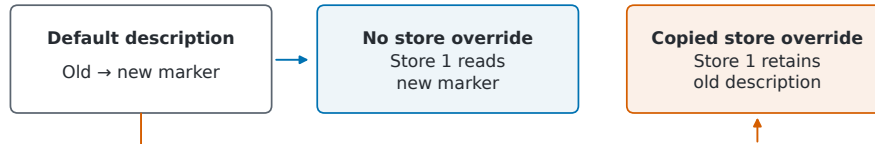
4.3 Posthoc description propagation

A stored override can initially display the same value as its default yet change later behavior. After inspecting early main-study records, we committed a separate exploratory protocol and applied one fixed intervention to all eight complete units and twelve episode-target pairs. Each unit compares its sealed pre-agent baseline with its preserved post-reindex state. We save a new default-scope description through the original repository and read the default and store-1 values with a fresh native getter.

The assay observes four blocked pairs and eight normally propagating pairs, with zero other mechanisms and zero unknown pairs. Blocking occurs in case-1 A, case-1 B and both case-2 B targets: a model-added store-1 override retains its old description after the default update. C and D propagate the description normally. Table 6 and Appendix D retain the complete denominator and operational criterion. This posthoc intervention measures a consequence of the saved state; the main experiment’s original scores and model usage remain unchanged.

5 Discussion and limitations

The SaaS results support reuse of a recorded business path as a practical execution interface. The shared workflow resolves identifiers, sequences dependent operations and applies application-specific representations while the agent

(a) One price goal, different persistent state**(b) Copied store values can block later inheritance**

Posthoc assay: 4/12 blocked pairs (C1 A/B; C2 B) • 8/12 propagate

Fig. 3 Two observed Magento state mechanisms. The case-2 price-only path changes the requested prices but leaves promotion-derived state stale; repository saves refresh that state and also create non-price EAV rows. The separate posthoc description assay tests a later default-scope update: a model-added store override can retain the old value and block propagation. Description blocking occurs in case-1 A/B and case-2 B, whereas C/D propagate normally

supplies goal parameters. Its remaining errors also follow from that shared implementation: retained event rows and extra body text recur across instances. Improving coverage therefore requires validating the generated business state against the whole request, beyond whether each recorded call executes.

The Magento study adds a complementary requirement. Native routes update different dependencies: a narrow price write leaves derived promotion state stale, while full-object saves can create persistent scope overrides. The description intervention makes one delayed consequence observable. Workflow interfaces should be evaluated against both their intended outputs and the state changes of the business operations they reuse.

The comparisons cover two Magento base tasks and three SaaS parameter instances within one business family, using one requested executor model and reasoning setting; the provider's backend weight snapshot is unknown. DEV008 follows development, whereas DEV009 and DEV010 are prospectively fixed and executed once per arm. Workflow knowledge, operation implementation and execution restrictions change together between raw code and tools, so the comparison identifies their combined effect. Static and guarded modes tie where both are evaluated. Magento's learned predicate overlaps shared validation, and the original repository's negative-price response was not independently established. These records do not isolate a gain from learned guards.

SaaS collateral preservation remains unmeasured. Magento’s conclusions concern its declared state boundary and native probes; image rendering, live customer-group-1 transactions and arbitrary concurrency lie outside them. The posthoc description assay measures a conditional mechanism rather than a prospective autonomous-error rate. The actual GUI attempts have distinct interrupted and invalid-action endpoints, limiting comparisons with normally completed execution.

Execution accounting includes failed attempts and reports Magento acquisition separately. Setup, engineering and manuscript work are excluded. DEV008’s tool runs used internal VM storage, whereas raw and GUI ran after migration to external storage, so elapsed times are not controlled speed comparisons or time-to-success estimates.

6 Related work

The broader evaluation problem is the test oracle problem: deciding whether observed behavior is correct remains separate from generating or executing a test (Barr et al., 2015). Our target, collateral and evidence measures make that distinction concrete for stateful business operations. Runeson and Höst (Runeson and Höst, 2009) emphasize explicit cases, units of analysis and validity boundaries in software-engineering case-study reporting. Here the comparisons use local benchmark fixtures and saved-state interventions; they are exploratory task comparisons, rather than industrial field observations.

WebArena provides reproducible web environments and tasks that assess functional completion (Zhou et al., 2023). WebArena-Verified distributes a curated dataset and deterministic evaluators using responses and network traces (ServiceNow, 2025). We reuse a pinned Magento environment and adapt the native price-change structure of its task 779. Our additional state-preservation boundary is part of this study, not an official benchmark score. OSWorld evaluates open-ended tasks in real web and desktop applications with execution-based checks (Xie et al., 2024).

Unbrowse learns callable first-party routes from browsing traffic and supports cached direct API execution (Tham et al., 2026). VIGIL enforces behavioral specifications over tool-event traces, including temporal dependencies, argument relationships and value flow (Li et al., 2026a). WeaveBench studies long-horizon hybrid-interface tasks with trajectory-aware assessment of shortcut behavior (Li et al., 2026b). These address route acquisition, runtime policy enforcement and execution-aware evaluation, respectively. Our case studies compare the completion coverage of GUI-derived workflows and inspect persistent state changes caused by their underlying business operations.

7 Conclusion

A frozen GUI-derived workflow achieves higher core and additional completion than raw code on three complete SaaS task instances, with fewer model

decisions. The two new variants reproduce that benefit within the same business family. Magento’s route-dependent promotion state and later description blocking show why practical workflow evaluation must examine persistent state alongside task coverage. The resulting design lesson is to reuse observed business paths while checking the full requested outcome and the state those paths leave behind.

Statements and Declarations

Funding. The author personally funded this research. No external funding or donated computing resources were received.

Competing interests. The author declares no financial or non-financial competing interests relevant to this work. The study was conducted independently; no company information was used.

Author contributions. Dekuan Li conceived the research question, directed the study, reviewed the experimental evidence, and revised and approved the manuscript. AI assistance to design, implementation, orchestration, analysis and writing is described in the Methods section. Dekuan Li is the sole author and takes responsibility for the work.

Ethics approval and consent. The reported experiments involved software agents and synthetic test fixtures, with no recruited human participants or identifiable participant data. Ethics approval, consent to participate and consent for publication are not applicable.

Data, materials and code availability. Online Resource 1 provides saved numerical inputs, analysis scripts, expected outputs and figure-generation code. It supports offline reproduction of the reported calculations. Online Resource 2 provides the disclosed SaaS task, evaluator and frozen-workflow method materials, with provenance and any redactions described in its inventory. To exclude credentials and private operational information from this release, native databases, authentication material and private raw execution logs are not distributed or offered as part of these supplements. The supplements do not claim to recreate the original live environments or rerun the participant models.

Online Resource 1. Numerical reproduction package for the Magento main study, its posthoc inheritance assay, SaaS DEV008 and DEV009/010, and all three figures. Filename: `ESM_1.zip`.

Online Resource 2. SaaS methods evidence and provenance for task prompts, actor capabilities, evaluator definitions and the frozen workflow. Filename: `ESM_2.zip`.

References

- Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S (2015) The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41(5):507–525.
<https://doi.org/10.1109/TSE.2014.2372785>.
- Ying Li et al. VIGIL: Runtime Enforcement of Behavioral Specifications in AI Agent Skills. *arXiv:2606.26524v1*, 2026.
<https://arxiv.org/abs/2606.26524>.
- Wanli Li et al. WeaveBench: A Long-Horizon, Real-World Benchmark for Computer-Use Agents with Hybrid Interfaces. *arXiv:2606.09426v3*, 2026.
<https://arxiv.org/abs/2606.09426>.
- Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14:131–164. <https://doi.org/10.1007/s10664-008-9102-8>.
- ServiceNow. WebArena-Verified. Public benchmark repository, 2025. Pinned commit 6473f72db5dcefc97b5725b59e734504edc28a21.
<https://github.com/ServiceNow/webarena-verified>.
- Lewis Tham et al. Internal APIs Are All You Need: Shadow APIs, Shared Discovery, and the Case Against Browser-First Agent Architectures. *arXiv:2604.00694v1*, 2026. <https://arxiv.org/abs/2604.00694>.
- UniPat-AI. SaaS-Bench. Public benchmark repository, 2026. Pinned tree aaa042140b585f80ad531bc8c9af296581426873.
<https://github.com/UniPat-AI/SaaS-Bench/tree/aaa042140b585f80ad531bc8c9af296581426873>.
- Tianbao Xie et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv:2404.07972*, 2024. <https://arxiv.org/abs/2404.07972>.
- Shuyan Zhou et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv:2307.13854*, 2023.
<https://arxiv.org/abs/2307.13854>.

A Complete SaaS scores and execution accounting

All five DEV008 attempts and all four preallocated extension attempts remain in Tables 2–5. The two scoring definitions are reported side by side. A call is a model response, including one proposing a rejected action; actions follow the accepted-decision counter and may include `finish` in the static arm.

DEV008 totals are 266 calls, 264 actions and 2,711,894 tokens: 2,644,922 input plus 66,972 output. Its 381,440 cached input tokens are a subset of input. DEV009/010 total 202 calls, 201 actions and 3,440,428 tokens: 3,349,747 input plus 90,681 output, including 716,800 cached input tokens. Cached tokens are not added twice. The five DEV008 attempts concern one complete development task; its technical repeat is not an independent task sample.

Table 2 DEV008: one complete three-application task, four conditions and one GUI-only technical repeat. O/L gives the original and predeclared local native-compatible evaluators. Core scores are weighted points out of 33; additional scores count checks out of 20. GUI001 is the collected endpoint of a controller interruption, not a normal actor finish. The repeat is a separate attempt on the same task

Condition / attempt	Core (O/L)	Additional (O/L)	Whole	Collateral
D: guarded tools	30/33 / 33/33	14/20 / 16/20	FAIL	UNKNOWN
C: static tools	30/33 / 33/33	14/20 / 16/20	FAIL	UNKNOWN
B: raw code	11/33 / 8/33	0/20 / 0/20	FAIL	UNKNOWN
A: GUI001	0/33 / 0/33	0/20 / 0/20	FAIL	UNKNOWN
A: GUI002 (repeat)	0/33 / 0/33	0/20 / 0/20	FAIL	UNKNOWN

Table 3 Saved execution costs and stopping causes for DEV008. Tokens are input plus output, including all original failed attempts; cached input is already included. Seconds are actor duration. D/C ran before the VM disk migration and raw/GUI after it, limiting cross-condition timing comparisons

Condition / attempt	Ending	Calls / actions	Tokens	Seconds
D: guarded tools	Normal finish	6 / 6	96,393	63.84
C: static tools	Normal finish	6 / 6	96,162	65.22
B: raw code	Invalid command	112 / 111	1,741,353	1480.11
A: GUI001	Controller interruption	127 / 127	696,937	1909.57
A: GUI002 (repeat)	Invalid key	15 / 14	81,049	194.72
All five attempts		266 / 264	2,711,894	—

Table 4 Two new complete-task parameter variants of the same SaaS business-workflow family, each evaluated with raw code and static workflow tools. O/L denotes original and local native-compatible scoring. Core scores are weighted points out of 33; additional scores count checks out of 20. All four terminal attempts are retained

Task	Condition	Core (O/L)	Additional (O/L)	Whole	Collateral
DEV009	B: raw code	11/33 / 8/33	0/20 / 0/20	FAIL	UNKNOWN
DEV009	C: static tools	30/33 / 33/33	14/20 / 16/20	FAIL	UNKNOWN
DEV010	B: raw code	21/33 / 18/33	6/20 / 6/20	FAIL	UNKNOWN
DEV010	C: static tools	30/33 / 33/33	14/20 / 16/20	FAIL	UNKNOWN

Table 5 Saved actor termination and execution cost for the two new fixed tasks. Input-plus-output tokens include all failed attempts; cached input is a subset of input and is not added again. These counts are separate from the five DEV008 attempts

Task / arm	Stop	Calls	Actions	Tokens	Seconds
DEV009 / B	Scientific limit	121	120	2,013,080	1384.08
DEV009 / C	Normal finish	6	6	106,848	100.18
DEV010 / B	Normal finish	70	70	1,247,320	1039.28
DEV010 / C	Normal finish	5	5	73,180	92.10
All four attempts		202	201	3,440,428	—

B SaaS request bindings and execution interface

C006 fixes each application’s request method and route template, the GraphQL document or JSON/form field structure, observed value types and array shapes, and paths for extracting returned identifiers. Twenty creation responses supply identifiers for subsequent field updates. BigCapital invoice creation retains the observed single-line request structure. Pretix routes accept current organizer and event slugs; form submissions obtain fresh controls and authentication fields. The capture does not establish a unique click attribution for every request.

The researcher-provided implementation supplies the goal schema, deterministic ordering, conversions, templates, identifier propagation and execution/recovery runtime. The model fills fields for CRM records, invoice dates and amounts, event settings and follow-up instructions. It cannot create new endpoint implementations through the catalog interface. C and D share catalog, schemas, identity and prompts; D alone adds runtime state guards and bounded recovery. No additional GUI or guarded slot is allocated in the extension, and the later optional body-composition repair is not deployed in its runs.

C Magento allocation, audit and execution accounting

The two cases are intentionally selected adaptations following earlier exploration, not random population samples. Original application mechanisms remain, and common initialization and audit changes are recorded in immutable provenance. Each attempt has limits of 64 decisions, 2,400 seconds, 512,000 total model tokens, 240 seconds per model turn and 60 seconds per actor command. New clones pass source, identity, state, isolation, history-retention, ordinary-route, positive-control, harmful-control, learning-evidence and freeze checks before formal model calls. These engineering controls are outside the model-error counts. The TRAIN learner receives acquisition evidence rather than formal goals, answer states or oracle rules, and the native read helper is protected against replacement.

The audit retains 370 original keyed MySQL tables, one keyless table, application source outside the declared exclusions, statement history and existing-index search histories. A controller-run original price/stock/search reindex precedes target and adjacent quote checks. Native dependencies include source-bound target price indexes and CatalogRule regeneration. Unsupported index-lifecycle observations stay unknown. The verifier retains a confirmed collateral error even when the target is incomplete.

Quote probes use customer group 0. CatalogRule checks also inspect derived state for group 1, without a live group-1 checkout. Additional image rows are protected-state changes, not measured rendering failures. Four phase findings for the case-2 promotion discrepancy refer to the same two entities observed twice. Store-specific copies count as structural changes even when their initial bytes equal the default. The full-object paths are product resource save for case-1 A and product repository save for B/C/D. Magento C/D use an engineered native macro grounded in the GUI save path, rather than verbatim request replay or a model-generated executor.

The nine actual attempts consume 487,454 known input-plus-output tokens in 48 model calls. Acquisition adds 8,125 tokens in one call. The eight complete records have an agent-duration subtotal of 1052.64 seconds; the total including the interruption is unknown. Cached input is included once. These execution counts exclude setup, controller audits, engineering and manuscript preparation, and are not subscription billing or financial break-even estimates.

D Posthoc inheritance assay: criterion and complete results

For each complete unit, the sealed pre-agent state supplies the baseline and the preserved post-reindex state supplies the treatment. The intervention saves a new default-scope description through the original product repository. Independent fresh-process getters read

default scope and store 1 before and after each save and must agree with native EAV rows. The two-target task preserves target order and reads both targets after each step. Complete initial-state recovery, native/source/history evidence and authenticated temporary-object cleanup are required; incomplete conditions remain unknown.

Blocking requires a baseline with no store-1 override that propagates the new marker, a treatment with a model-added override, a successful default save, retention of the override's old value, and a fresh store-1 getter returning that old value. Other mechanisms, normal propagation and unknowns are retained. The completed assay has eight complete units and zero unknown units; Table 6 records all twelve target pairs. It measures the native getter rather than front-end rendering.

Table 6 Posthoc native description assay. All eight units and twelve episode–target pairs are retained. “Blocked” concerns this native business getter only; no front-end rendering or general safety guarantee is measured. B is code execution with GUI hints; C and D use the same fixed-recipe macro

Case	Arm	Targets	Blocked	Propagated	Other	Unknown
C1	A	1	1	0	0	0
C1	B	1	1	0	0	0
C1	C	1	0	1	0	0
C1	D	1	0	1	0	0
C2	A	2	0	2	0	0
C2	B	2	2	0	0	0
C2	C	2	0	2	0	0
C2	D	2	0	2	0	0

E Evidence and reproduction status

The Magento analysis is generated from the fixed eight-slot allocation and the independently preserved interruption ledger. It keeps the original reports, changed-instrument reanalyses, model requests and responses, expanded native calls, before/after observations and audit commitments. Every number in Table 1 is transcribed by the accompanying generator from an authenticated saved analysis.

The final analysis SHA-256 is:

59719ac8d2dd3e7525bfab26f14bec83bcf790bb9e686055aa6fa95cf312f78c.

The DEV008 tables and accompanying CSV/JSON are generated separately by `generated/saas-dev008-generate.py`. It pins the saved final comparison and progress record, verifies all five original actor receipts and their common budgets, and checks the totals without executing an evaluator, model or application. Its comparison snapshot SHA-256 is:

add1fe0bc047e2d8a1e1084ff525c9e99df6c648d147be43114c735fa5b4b86b. The original four rows and GUI technical repeat remain separately identifiable. Local native-compatible scores are not substituted for the original evaluator scores.

The four extension rows are generated by `generated/saas-extension-generate.py`. It verifies the final summary against the original actor, terminal and independent collection records, confirms common budgets and recomputes aggregate usage before writing the CSV/JSON and Tables 4–5. It does not rerun scoring. The final extension snapshot SHA-256 is:

053b7433953f4b30c85d654f74e5cfe1307cd3a899ca6c64ecc2dde88e7ccd84.

Online Resource 1 contains the previously prepared numerical reproduction attachment. That attachment contains four research components (the Magento main study, its posthoc

inheritance assay, the five DEV008 attempts, and the four DEV009/010 attempts) plus the three manuscript figures and their offline generator. It includes saved numerical inputs, expected outputs and file manifests. The unchanged inner 82-file attachment has SHA-256: `d86e8901cf829900957cf49bc218b2da98d99888ae2bd8d28f791e8501aa5d25`.

The research components retain their original scoring definitions, failures and technical repeats. Their numerical inputs and scripts are preserved byte for byte. Fresh extraction and offline recomputation reproduce the saved results and all three new figures; the figure component contains only the saved inputs required for plotting. These materials reproduce calculations from saved results. Native applications, private execution logs, account files, native databases and historical draft manuscripts are not included.

F Execution interruptions and technical repeats

F.1 Magento records and inheritance assay

The first case-2 A attempt made one real model call before a host filesystem ENOSPC interruption. Its 7,179 known tokens and missing original terminal remain in the first-attempt table. After a verified storage repair, one additional attempt used the unchanged task, model, method and budgets. It is a supplemental retry, not a replacement for the original failure or a new base task. The interrupted attempt’s total agent duration is unknown; late audit closure does not supply it.

Two case-1 capture reports originally had incomplete assessments because of verbose-binlog parsing defects. Their original reports are retained alongside separately committed raw-only reanalyses. The original case-1 C terminal is still `runner_error`: the host parsed an empty arguments string on a schema-valid finish response. A later bounded terminal-parser repair does not relabel that original terminal or rerun the participant. No method or oracle policy was tuned using these formal failures.

Two interrupted assay controller attempts are retained: a successful Docker commit followed by failure to parse its deprecation-prefixed image identity, and a host-capacity stop before a later commit. The first completed baseline, the next two completed pairs and another completed baseline are authenticated and reused through explicit versioned checkpoint references. The completed conditions are not re-executed or relabeled; the repaired continuation performs only the remaining eleven native conditions. The complete assay comprises sixteen logical conditions and uses zero new model calls. Its native getters do not measure image display, customer-group-1 transactions, arbitrary concurrency or observed front-end behavior.

F.2 SaaS DEV008 endpoints

Raw code ends on a command containing a literal NUL character, rejected before its 112th tool dispatch. Original GUI001 ends when the controller cannot decode the response file for call 127, after the last legitimate click has executed. A publication race is the supported diagnosis, although the fault-time bytes were not retained. A narrowly versioned atomic-publication repair passes six response-publication host checks, six request-publication host checks and seven actual transport checks before one GUI-only technical repeat from a fresh initial state. The original GUI record is retained.

GUI002 then ends on its 15th model response: the actor requests the key `Enter`, while the explicit allowed-key list contains `Return`. Validation refuses the action before GUI dispatch, leaving 15 calls and 14 actions. This invalid-action stop differs from GUI001’s controller interruption; neither is a normal actor finish or budget exhaustion. The collected zero-score endpoints stay separate.

G Later development audit, separate from the experimental allocation

The 8 October 2026 audit examines four preserved decision chains, not additional participant episodes or a replacement result set. A prospective batch-policy amendment separates scientific noncompletion from infrastructure interruption; ordinary complete measured business failures did not all stop the old runner. This changes continuation policy, not the task goal, scientific caps or historical success scores.

An initial-state comparison was amended to recognize a finite, source-authenticated search-service lifecycle retry. Original raw search-state inequality remains reported; it is not converted into complete raw equality. In another interruption, an exporter had intentionally streamed bytes into content-addressed storage, while a consumer incorrectly required an ordinary stdout file. The original stop record and partial successful business operations remain separate from any later continuation. A zero-model infrastructure repair is not an additional successful agent experiment.

The reviewed earlier mouse-GUI attempt retains a confirmed non-target price error, incomplete evidence and unknown later functional checks. Its original terminal and budget-version records remain separate. These examples support reporting development limitations; they do not justify extrapolating a universal designer-model bias or imputing unmeasured GUI success.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [ESM1.zip](#)
- [ESM2.zip](#)