# Agent Interaction Surfaces: Impact on Browser-Based Agent Performance

*Manju Weerasinghe[1], Priya Chawla[1], Danny Prevoznik[2]*
*[1] TollBit*
*[2] KERNEL*
*Benchmarking setup developed in collaboration with KERNEL*
May 2026

**Abstract**

This document evaluates how browser interaction surfaces affect agent performance while holding task, prompt, and model constant. Across five commerce workflows and 100 runs per site per variant (1,000 runs total), Agent Sites for eCommerce reduced mean time to completion by 24-35% and matched or exceeded default-site task completion on every site, reaching 100% completion on all five. Default sites ranged from 91% to 100% completion, with observed failures tied to the agent exhausting its step budget while navigating or recovering from UI state changes. The interaction surface is a primary driver of agent performance, independent of the model or prompt.

## 1 KEY RESULTS

- 24-35% faster time to completion across all sites tested
- Agent Site reached 100% task completion on every site; default ranged 91-100%
- Fewer interaction steps and screenshots per run (used as a proxy for token usage and cost)
- Failures on default sites frequently involve max-step loops and unstable navigation paths

Results aggregated across 100 executions per site per variant (1,000 runs total).

| Site | Time to Completion Improvement | Completion Delta | Step Reduction |
|---|---|---|---|
| Cool3C | 34.8% faster | +4-6 pts | 37.9% fewer |
| Kopari Beauty | 24.3% faster | No material change | 18.6% fewer |
| MM Lafleur | 29.9% faster | +6-10 pts | 25.4% fewer |
| Pela | 28.0% faster | +0-18 pts | 29.5% fewer |
| Quip | 26.1% faster | +0 pts | 35.2% fewer |

## 2 PROBLEM

Most websites today are built around human conversion flows. They assume a human who can visually scan a page, ignore distractions, recover from dead ends, and adapt to UI changes in real time.

When browser-based agents attempt to complete real tasks on these same sites, that assumption breaks down. Agents interact with UI patterns that are incidental for humans but costly for automation. In practice, this shows up as extra steps, higher variance across runs, and frequent failure modes that are not directly tied to the underlying task. Common issues we observe include:

- Visual clutter such as ads, modals, and promotional overlays
- Non-deterministic navigation paths and stateful UI
- Dynamically injected DOM elements and shifting page structure
- Repeated visual scanning and screenshot capture to recover context
- Interruptions from chat widgets, consent prompts, and autoplay media

The result is brittle automation, not just slower execution. Identical prompts against the same site can succeed or fail depending on timing, layout shifts, or incidental UI changes. This increases compute cost, reduces reliability, and complicates debugging.

This evaluation isolates the effect of the interaction surface on agent execution by holding task, prompt, and model constant.

## 3 WHAT WE MEAN BY AGENT SITE

Agent Sites for eCommerce are not new websites and not new content. They follow the same pattern as mobile-optimized sites: the underlying system remains the same, but the interaction surface adapts to a different mode of use.

They are an alternative interaction surface over an existing site, designed specifically for agents executing tasks through a browser. The underlying site stays the same:

- Same products and content
- Same business logic
- Same pricing and checkout flows

For the purposes of this evaluation, we hold the task and prompt constant and change only the surface the agent interacts with:

- The default, human-oriented site
- A TollBit-managed Agent Site over that same site

The goal is not to simplify the task or bypass site logic. The goal is to remove patterns that are incidental for humans but create instability, excess steps, and failure modes for agents. In practice, agent sites focus on:

- Stabilizing page structure across navigations
- Removing UI elements that introduce non-determinism for agents
- Making action paths explicit and repeatable
- Preserving full actionability (not read-only extraction)

The underlying site stays the same. The surface shifts from being optimized for human attention to agent execution.

## 4 EVALUATION PHILOSOPHY

Most discussions around browser agents change multiple variables at once - model choice, prompting strategy, tooling, retries, and task design. This makes it difficult to isolate what actually drives success or failure.

This evaluation focuses on a single question:

**Holding task, prompt, and model constant, how much does the interaction surface affect an agent's ability to complete real workflows?**

Prompts are fixed. Agent configuration is fixed. Success criteria are fixed.

The only variable that changes is the surface the agent interacts with at runtime.

## 5 EVALUATION METHODOLOGY

This evaluation isolates the impact of the interaction surface on agent performance by constraining all other variables.

### *5.0.1 Controlled Variables*

For each comparison, the following are held constant:

- Task definition
  Each workflow has a fixed task description (e.g. constrained product selection and add-to-cart).
- Agent prompt
  The exact same prompt is used for both the default site and the agent site.
- Starting point
  The agent begins from the same logical entry point for each run.
- Agent framework and model
  Comparisons are made using the same agent framework and model configuration per run. We do not

tune prompts or agent behavior between the default and Agent Site

- variants.
- Success criteria
  Task success and failure are defined identically across runs.

Differences in outcomes are not due to changes in prompting, model choice, or task design.

#### *5.0.2 Independent Variable*

The only variable that changes between paired runs is the site surface the agent interacts with:

- The default, human-oriented website
- A TollBit-managed Agent Site over the same underlying site

The underlying content, products, pricing, and business logic remain unchanged.

#### *5.0.3 Metrics Collected*

We evaluate agent performance using metrics that map directly to reliability, execution cost, and stability:

- Task success rate
  Whether the agent completes the workflow as defined.
- Time to completion
  Total elapsed time to reach task completion or failure.
- Interaction steps
  Number of discrete actions taken by the agent.
- Visual observations (e.g. screenshots captured during execution)
  Used as a proxy for vision-based token usage and agent effort.
- Failure modes
  Captured as a primary technical reason code (e.g., bot_detection, element_not_found, timeout) plus a qualitative surface bucket when observable (e.g., popup/overlay, consent prompt, navigation loop, checkout gating).
- Variance across runs
  Distribution of outcomes across repeated executions of the same task.

#### *5.0.4 Repeated Runs*

Each task is executed multiple times per surface to account for agent non-determinism. Results are aggregated across repeated runs. The goal is not to produce a single "best case" outcome, but to observe how surface design affects consistency and failure rates under repeated execution.

## 6 SCOPE OF EVALUATION

This evaluation focuses on structured commerce workflows where agents are required to complete end-to-end tasks.

Representative workflows include:

- Navigating to a product or listing
- Applying filters or constraints
- Selecting an item
- Adding the item to cart

These workflows were chosen because task completion depends directly on navigation stability, actionability, and consistency across runs.

While agents interact with many other types of sites (e.g. forums, content platforms, authenticated flows), those are not included in this evaluation.

## 7 EXPERIMENTAL SETUP

This section describes the agent, environment, and run configuration used in the evaluation. The benchmarking setup was developed jointly with KERNEL.

- Agent framework(s) used
  - Browser runtime: KERNEL cloud browsers, launched in stealth mode with session replay recording enabled. Each run received a fresh, isolated browser session.
  - Navigation layer: Playwright `page.goto` was issued inside the KERNEL session with a 10-second navigation timeout and up to two navigation attempts before the run was marked as a navigation failure.
  - Agent loop: Anthropic Computer Use sampling loop. The agent received screenshots and a fixed set of computer-control tools (click, scroll, type, key, screenshot) and was required to declare its own outcome via a structured `report_task_result` tool call so that success and failure were both explicit rather than inferred.
  - System prompt: A single fixed system prompt was used across every run and each variant for all sites. It instructs the agent to prefer keyboard-based scrolling, to take a verification screenshot after each action, and to call `report_task_result` exactly once at the end of the task with a success flag and, on failure, a structured failure mode and surface bucket.
- Model
  - Model: Claude Sonnet 4.6
  - Tool version: Anthropic computer-use tool surface `computer_20251124`
  - Reasoning: Adaptive extended-thinking enabled.
  - Output budget: 16,384 tokens per assistant turn.
  - The same model and configuration were used for both the default and Agent Site variants of every site.
- Run configuration (retries, parallelism, staggering)
  - Sample size: 100 invocations per (site x variant), across 5 sites and 2 variants, for 1,000 total runs.
  - Step cap: Each run was capped at 40 agent steps. Runs that reached the cap without producing a successful `report_task_result` were classified as `max_steps_reached`.
  - Invocation timeout: A 15-minute hard timeout was enforced at the runner level. Runs that exceeded this were classified as timeouts.
  - Replay capture: Every run produced a video replay for post-hoc inspection of failure modes.
  - Variant routing: The default variant navigated directly to the canonical site URL. The Agent Site variant navigated to the same underlying site through a TollBit-managed Agent Site. The underlying products, pricing, and business logic were identical between the two variants.
- Workflows and Prompts
  - Pela Case - "Add an iPhone 17 pro silly goose case with magsafe to cart."
  - MM LaFleur - "Find a blue dress with pockets in a size small, and add it to cart."
  - Kopari Beauty - "Search for 2 different creams that total less than $100. Add both to the cart."
  - Quip - "Search for a single Rev Oscillating Toothbrush in the Black Night colour to add to cart."
  - Cool3C - "Find the Kokomo vacuum cleaner product and add it to the cart."
- Environment notes
  - All runs were unauthenticated and single-session; no persistent cookies, accounts, or personalization carried between runs.
  - Runs executed during a single April 2026 evaluation window so that site state was as consistent as possible across the dataset.
  - The benchmark runner persists every step (action type, duration, screenshot flag, error flag, timestamp) for every run so that aggregate metrics in this report are computed from the recorded run data.

## 8 RESULTS & OBSERVATIONS

### *8.1 Overview*

Each site was evaluated across 100 independent executions per variant (default vs Agent Site), for 1,000 total runs, using identical task definitions, prompts, and agent configuration.

Results are reported as distributions across runs rather than single aggregates to capture variability in agent behavior.

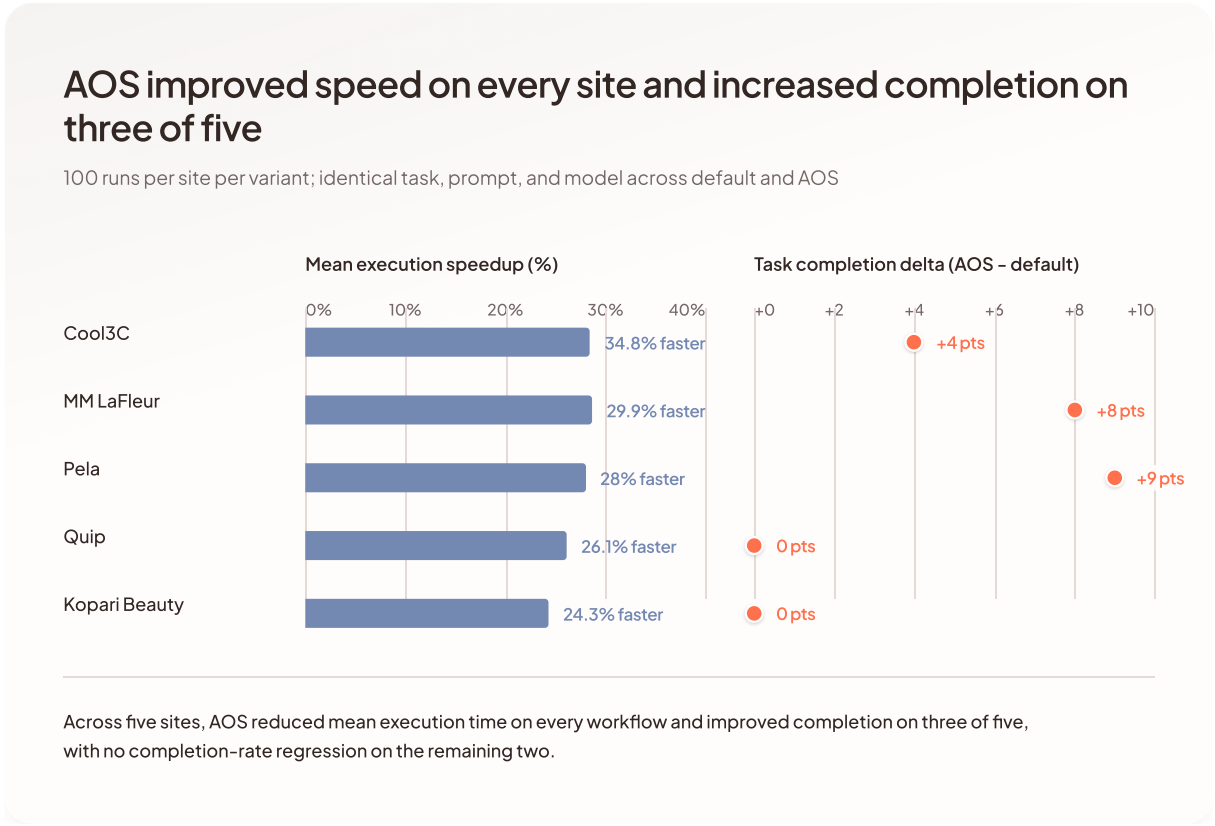


Figure 1

### *8.2 Time to Completion*

Agent Site reduces mean time to completion across all sites, with observed improvements ranging from 24.3% to 35.1%.

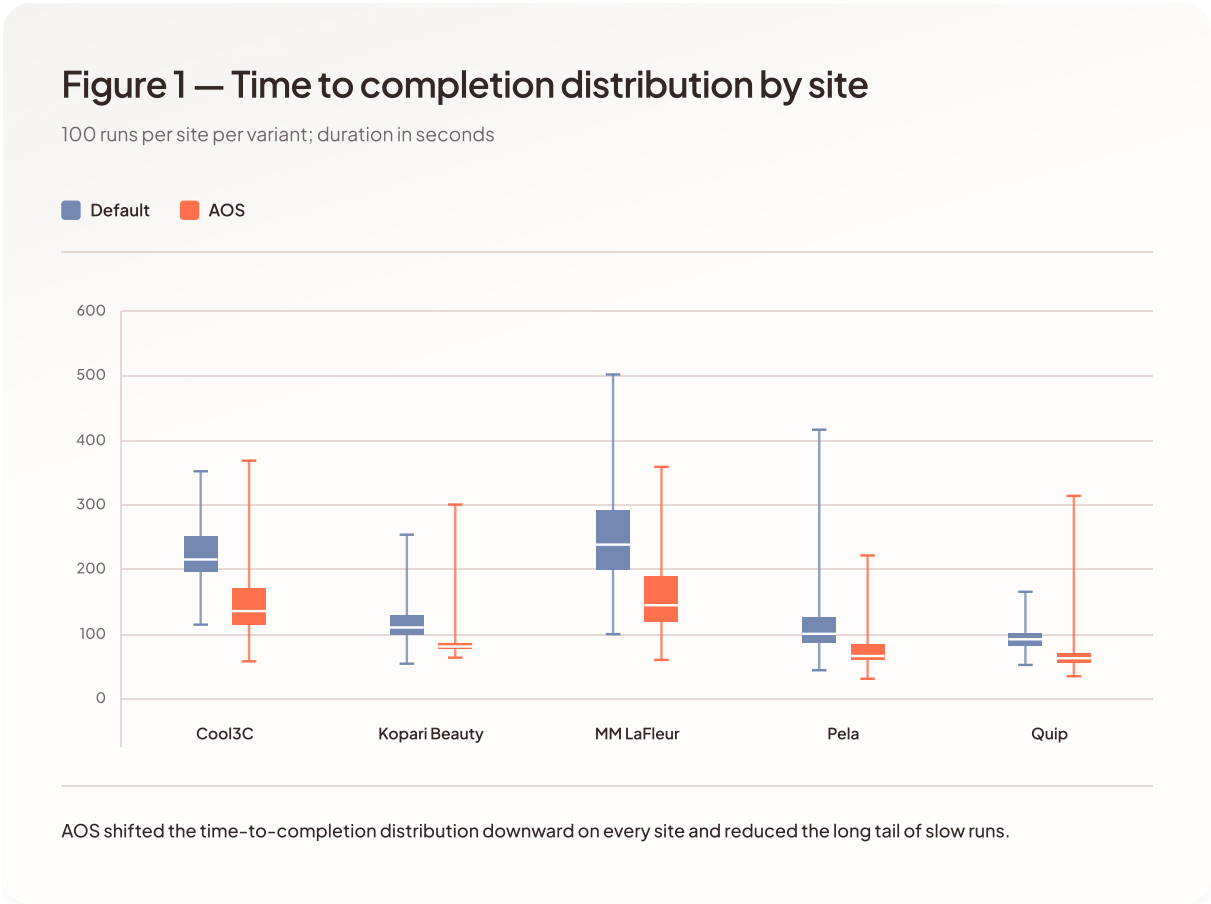


Figure 2

Across runs, default sites exhibit wider spread in time to completion, including long-tail executions. Agent Site reduces long-tail execution cases.

### *8.3 Task Completion*

Agent Site reached 100% task completion on all five sites. Default sites failed to reach full completion on three of five workflows.

Agent Site increases completion rates on MM LaFleur and Cool3C, and maintains comparable completion on Kopari, Pela, and Quip.

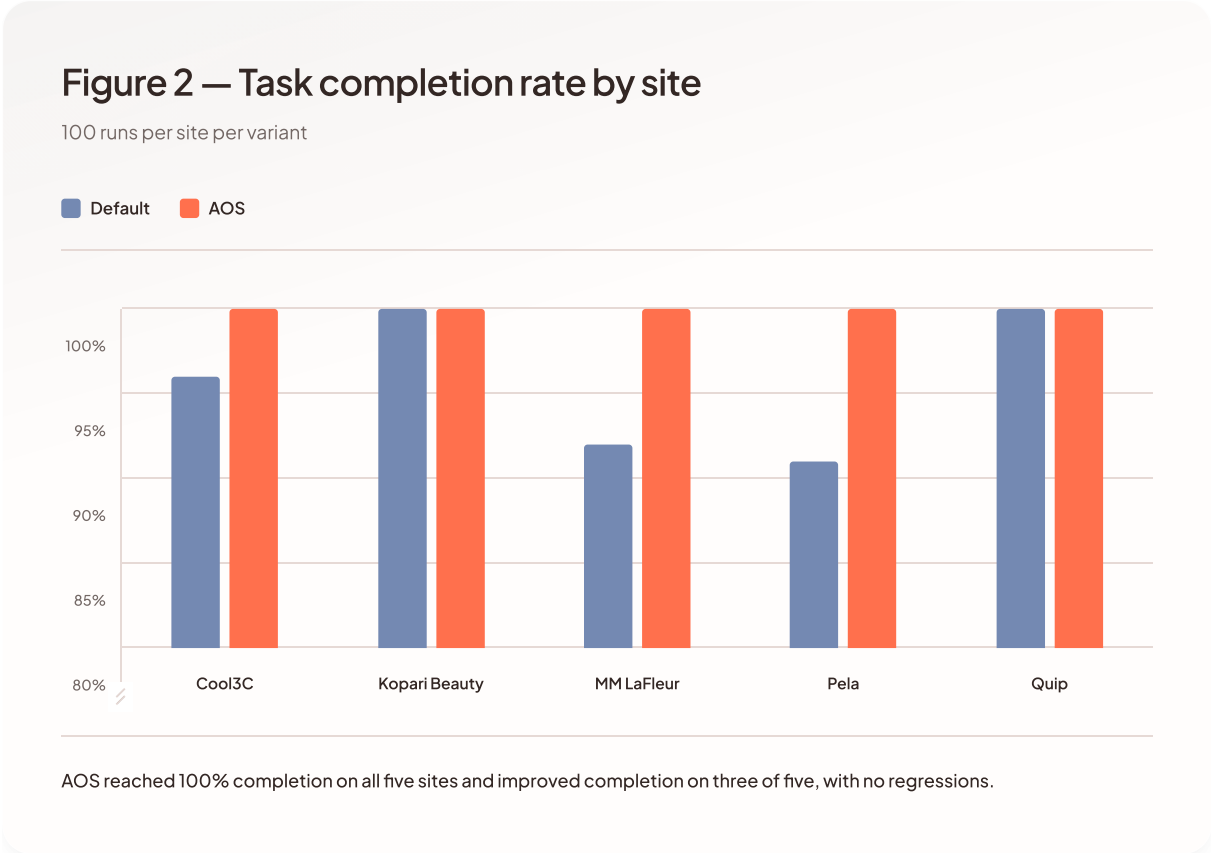


Figure 3

Failures on the default variant occurred intermittently across runs for identical tasks, indicating instability in execution rather than task complexity.

### *8.4 Interaction Steps*

Step count differs between variants.

Default runs include longer execution paths, with repeated navigation and re-evaluation of page state. Agent Site reduced the total steps required to complete each workflow, which in turn reduced the number of screenshots captured per run (each step produces one screenshot under the agent's verification protocol). Step and screenshot counts serve as proxies for token usage and vision cost.

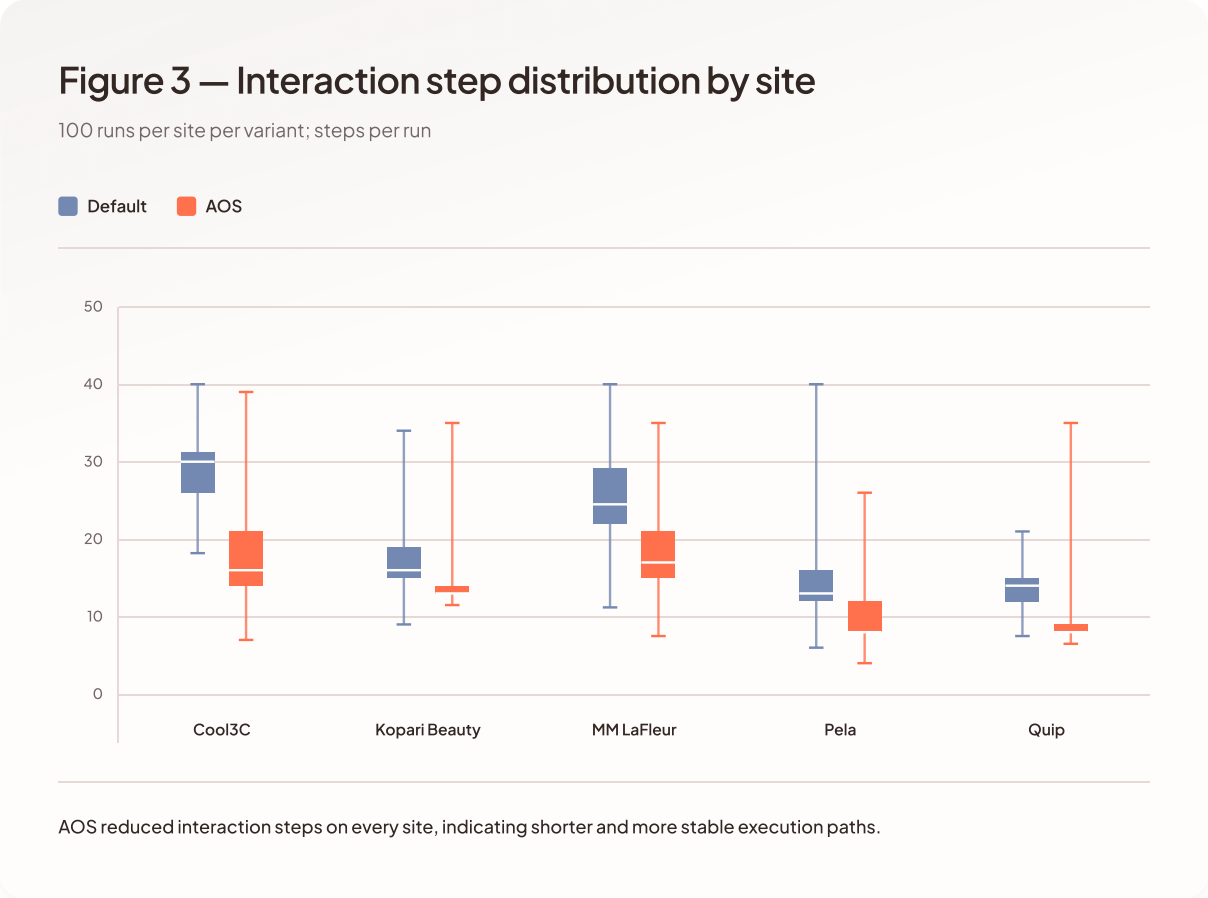


Figure 4

### *8.5 Variability*

Run-to-run variability is observed across all sites.

Default sites show:

- variation in time to completion
- variation in step count
- inconsistent completion outcomes on three of five sites

Agent Site shows more consistent execution paths across runs, with zero failed runs across the full 500-run Agent Site dataset.

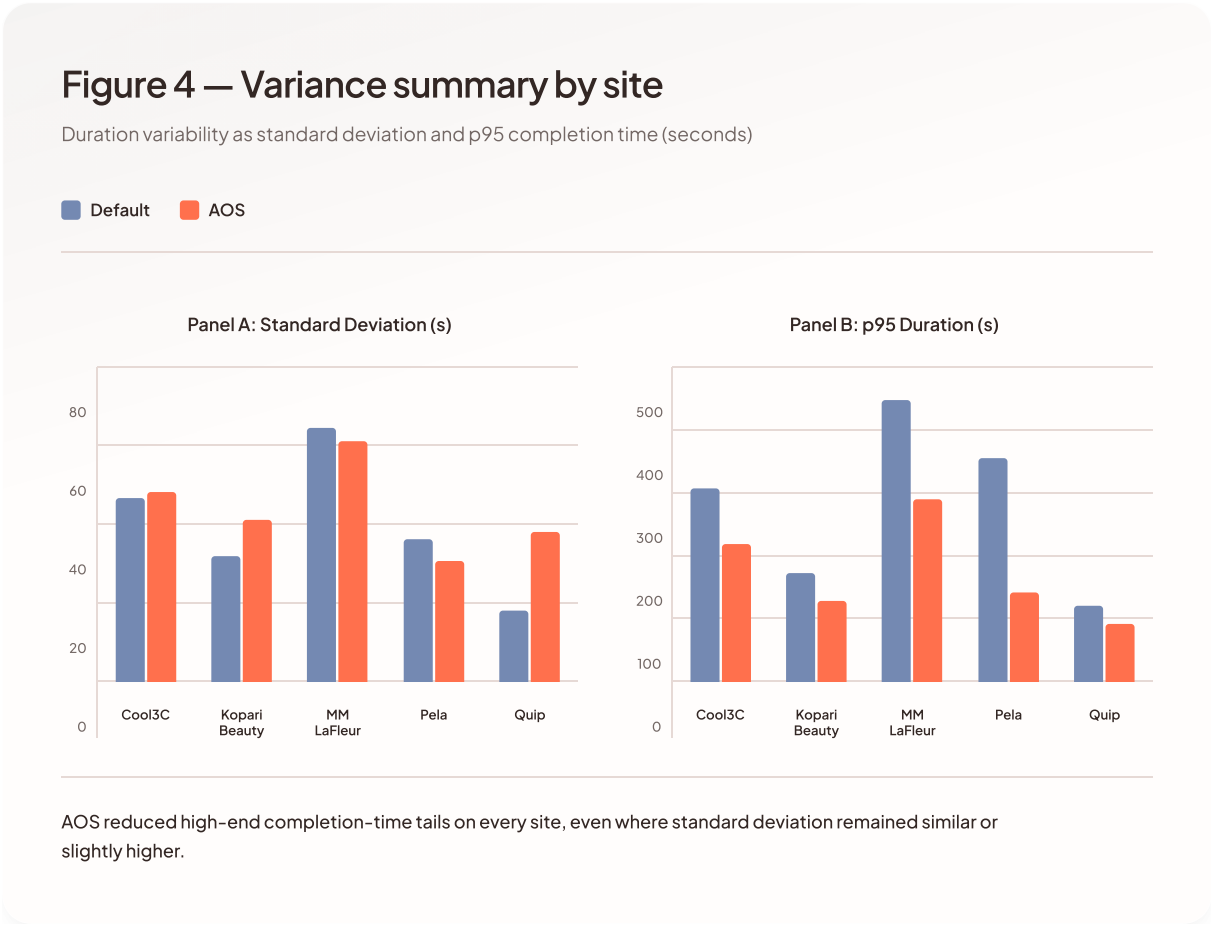


Figure 5

### *8.6 Failure Modes*

In this 1,000-run dataset, every recorded terminal failure (21 / 500 default runs, 0 / 500 Agent Site runs) ended as `max_steps_reached`, meaning the agent exhausted its 40-step budget without producing a successful report. Per-site terminal failure counts on the default variant were: Cool3C 4, MM LaFleur 8, Pela 9, Kopari 0, and Quip 0.

`max_steps_reached` is not a root-cause label. We therefore reviewed the replay for each of the 21 default-variant failures and assigned a dominant qualitative cause. That replay review suggests three recurring patterns:

- Bot detection / challenge flow: 12/21 failures, including both explicit challenge flows and silent same-page request blocking that never surfaced to the agent as a distinct error state.
- Agent decision loop: 8/21 failures, including bad-URL detours, backing out of the correct product, popup distraction, and repeated scrolling or hesitation near the correct add-to-cart action.
- Page-load / stuck-page: 1/21 failures, where the page entered a prolonged loading state and never recovered within the step budget.

By site, the replay-based dominant-cause breakdown was: Cool3C 4 decision-loop failures; MM LaFleur 4 bot-detection failures and 4 decision-loop failures; Pela 8 bot-detection failures and 1 page-load failure. These failures occurred intermittently across otherwise identical runs.

On Agent Site, navigation-related failures were reduced. In the final dataset, no Agent Site runs terminated in failure.

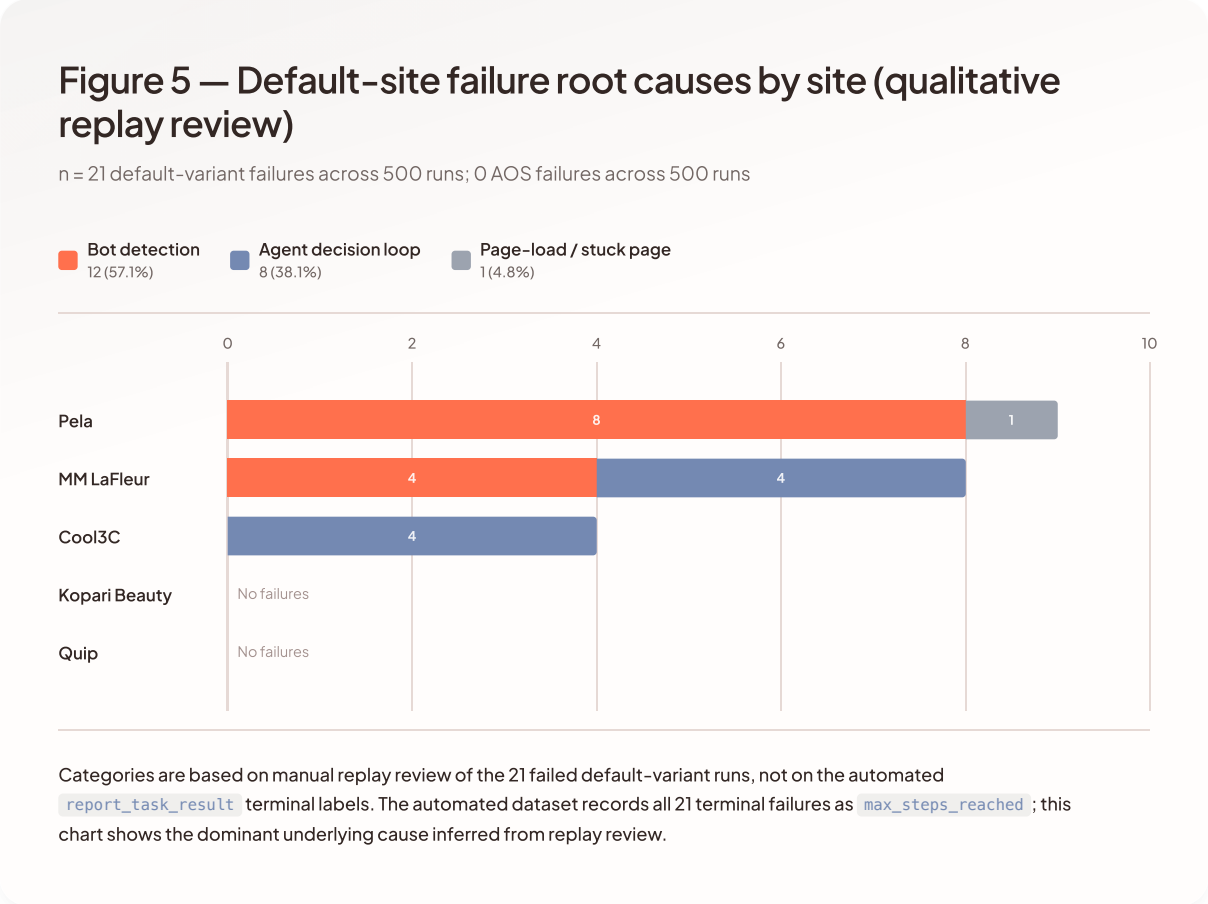


Figure 6

## 9 LIMITATIONS

This evaluation is based on controlled browser-based runs and reflects observed behavior under the specific setup used in this benchmark.

**Bot detection and environment effects**

Replay review shows that bot detection remained a meaningful source of failure on some default sites, particularly when add-to-cart or filtering actions were blocked without a clear error signal.

In these cases, failures were triggered by repeated API activity tied to normal site behavior rather than explicit agent errors. For example, some sites persist cart state by issuing multiple background requests on each page load, even when the cart has not changed. Similarly, product listing and filtering flows can generate additional fetch requests as results are updated.

These request patterns trigger bot detection during rapid, step-by-step navigation, causing actions to fail silently without a distinct challenge or error response. In these scenarios, the agent continues retrying within the same page context until it exhausts its step budget.

Notably, these patterns can also surface under fast human navigation, indicating that the behavior is tied to site-level request patterns rather than agent-specific actions.

**Run instability on default sites**

Default site failures were not deterministic. Identical tasks and prompts produced both successful and failed runs.

Across the 500 default-variant runs:

- 21 runs (4.2%) failed to complete the task within the 40-step cap
- 100% of those runs ended with the terminal label `max_steps_reached`, meaning the agent exhausted its 40-step budget without producing a successful report
- Per-site failure counts: Cool3C 4, MM LaFleur 8, Pela 9, Kopari 0, Quip 0

- Replay review assigned dominant underlying causes of 12 bot-detection failures, 8 agent decision-loop failures, and 1 page-load / stuck-page failure
- The automated label set recorded 0 explicit `bot_detection` terminal failures, but the replay review shows that this is an undercount caused by silent same-page blocking rather than a true zero

This variability is tied to execution-path differences rather than task definition.

**Failure classification granularity**

Failure modes are derived from the agent's structured `report_task_result` output and from runner-side classification of timeouts and infrastructure errors. These labels represent final recorded states, not definitive root-cause diagnoses.

Replay review of failed runs highlights a key limitation of automated labeling: some blocking behavior does not surface as an explicit error state and instead appears as repeated failed actions within the same page. In these cases, runs terminate as `max_steps_reached` rather than being attributed to a specific failure type.

To address this, we supplemented automated labels with manual replay review of all 21 failed default-variant runs and report those replay-based dominant-cause counts in the Results section. These should be interpreted as an additional layer of analysis rather than a replacement for the automated labels.

**Prompt and workflow sensitivity**
Prompts were specified tightly enough to make success criteria unambiguous. The prompts shown in Experimental Setup are the post-clarification versions and were held constant across both variants once finalized. Examples include:

- MM LaFleur - color, pocket presence, and size were all specified to disambiguate which dress to add
- Quip - exact product line ("Rev Oscillating Toothbrush") and color ("Black Night") were specified to avoid agents converging on different SKUs
- Kopari - quantity ("2 different creams") and price ceiling ("less than $100") were specified so success criteria were unambiguous
- Cool3C - exact product name ("Kokomo vacuum cleaner") was specified given the marketplace-style listing density

Prompts were held constant within each comparison after these adjustments, but small changes in task definition can affect execution outcomes.

**Limited site set**
The evaluation includes 5 commerce sites with different UI patterns and structures:

- Pela – direct-to-consumer storefront with product variants and cart flows
- MM LaFleur – structured catalog with filtering, sizing constraints, and multi-step selection
- Kopari Beauty – search-driven product discovery with cart aggregation
- Quip – product + variant selection with optional subscription paths
- Cool3C – marketplace-style interface with non-English UI and less conventional navigation patterns

This evaluation is limited to unauthenticated, single-session commerce workflows that stop at add-to-cart. Results may differ for authenticated, personalized, or multi-session experiences.

**Metric scope**
The evaluation measures:

- time to completion
- task success
- interaction steps

Direct measurement of token usage and compute cost is not included.

## 10 DISCUSSION

These results suggest that interaction surface design is a first-order factor in browser-based agent performance.

Improvements are not limited to speed, but extend to consistency and reduction in failure modes.

This has implications for how websites expose functionality to automated systems, particularly as agents move from retrieval to execution.