Why Are GUI Agents Correct but Late? Decode on the Decision-Time Critical Path, Tested with Pre-Compiled Policy Trees

Anonymous submission

Abstract

Computer-use agents fail on transient GUI events in a specific way: the model computes the correct action and delivers it after the window has closed. We hypothesize a single cause – the expensive generative decode sits on the decision-time critical path – and test it with an instrument built to remove exactly that cause and nothing else: Adaptive Anticipatory Policy Trees (AAPT). During quiet screen phases the same frozen multimodal model compiles a bounded conditional policy tree – observable guards, pre-authorized actions, per-branch deadlines – sized to bridge the planner's own latency; at event time a low-token observer routes change-gated frames through the prepared branches, generating nothing. Under per-seed pairing, exact McNemar tests, and pre-registered endpoints, AAPT recovers a contested window (0.79 vs. 0.50, $p = 1.8 \times 10^{-3}$) with zero incorrect actions, while open-loop and predict-replan baselines that also act early – but keep decoding on the path – score zero; a preparation sweep places the advantage's onset where the sizing rule predicts. The effect is capability-gated – sub-second observer decode, schema-valid flat-tree planning, accurate branch routing – and a pre-registered oracle probe that refuted our own prediction makes the routing gate causal. A confirmation with predictions committed before any data replicates the effect on an untuned generalist of the same Qwen3.5-MoE class (126 pairs, $p = 4.9 \times 10^{-13}$). On an external benchmark the aggregate is a tie whose discordant wins dissociate perfectly: AAPT wins exactly where winning responses are pre-enumerable at compile time, reactive execution exactly where they are not.

1 Introduction

Computer-use agents overwhelmingly follow a sequential loop: capture a screenshot, invoke a multimodal language model, execute one action, observe again. The loop fails structurally when the interface does not wait: boot prompts, auto-dismissing dialogs, short-lived authentication requests, transient toasts, and reaction-time game states all present *action windows* that open and close on the environment's clock. The recurring failure is not wrong understanding but *late* understanding – the model computes the right action after the window has closed – which admits two competing explanations: the agent does not anticipate the event, or it anticipates perfectly well and is defeated by *where* its expensive computation sits. This paper is a controlled test of which cause is binding.

Our hypothesis is the second: the failure is caused by the generative decode occupying the decision-time critical path, not by a lack of anticipation or capability. (Calling a 30B-class multimodal model at frame rate is economically off the table, so the cause cannot be outrun.) The hypothesis has falsifiable consequences: (i) anticipation alone must *not* help – acting early while keeping the decode on the path should do no better than plain reaction; (ii) removing the decode should help *exactly* when preparation time exceeds the planner's own latency, with the onset at a value the theory names in advance ($L_{p95} + M$, Eq. 1); (iii) raw speed must not suffice – a model fast enough to beat the window but unable to compile or route a valid conditional policy should not benefit. Each prediction is tested below, pre-registered, and one of our own committed predictions fails in a way that sharpens the account.

The instrument that isolates the cause is *Adaptive Anticipatory Policy Trees* (AAPT): while the screen is quiet, the slow model compiles a small, bounded conditional policy – branches guarded by observable conditions, carrying pre-authorized concrete actions, expiring on explicit deadlines – sized so its wall-clock coverage bridges the planner's own latency. A lightweight observer (the *same* frozen model on a strict token budget) then watches change-gated frames and does one cheap thing: name which prepared branch the world took; the pre-authorized action fires immediately, and anything outside the tree stops execution and requests replanning (Figure 1). AAPT thus removes exactly one thing from the critical path – generation – holding the model, the information, and the task fixed, so its pattern of success and failure is evidence about the cause. The method is the manipulation, not the destination: every ingredient has precedents (Section 2), so the question is whether the causal hypothesis it embodies is true. Every comparison is paired per-seed on frozen models with exact McNemar tests, stages were pre-registered with declared primaries, and deviations are reported rather than substituted away.

Concretely, the paper answers four questions. (1) **What causes the missed deadline?** Baselines that act early but keep the decode on the path score zero where AAPT wins a contested window at a pre-registered primary (0.79 vs. 0.50, $p = 1.8 \times 10^{-3}$): removing decision-time generation, not anticipation, is the active ingredient (Sections 5.1–5.2). (2) **Does the effect obey the rule that motivates it?**

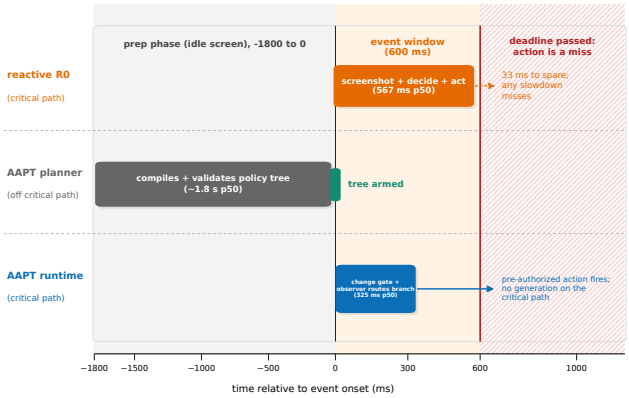

Figure 1: The critical path under a contested 600 ms window, drawn to measured latencies. The reactive loop must fit a full perceive-reason-act round trip (567 ms p50) inside the window, leaving 33 ms to spare. AAPT compiles the policy before the event and needs only a low-token routing call (∼325 ms p50) after it.

A preparation sweep confirms the advantage's onset exactly where Eq. 1 predicts ($L_{p95} + M \approx 2.4$ s), starved trees fail safe, randomized delays leave the gain unchanged, and budget experiments locate the tree's efficient shape at branch count equal to the outcome-set size (Section 5.6, Appendix F[1]). (3) **When does removing the cause help, and when not?** The gain appears exactly where sub-second observer decode, schema-valid flat-tree planning, and accurate branch routing coincide; gate predictions for two unseen models were committed before their first token and both graded correct, including replication at $p = 4.9 \times 10^{-13}$ on an untuned generalist of the same Qwen3.5 mixture-of-experts (MoE) class, and a pre-registered oracle probe *refuted* our own prediction, upgrading the routing gate to causal (Sections 5.3–5.4). (4) **Where does the account stop?** The winning response must be pre-enumerable at compile time: on a third-party benchmark the aggregate is a tie whose discordant wins dissociate perfectly by mechanism (Section 5.5); and a serving compile-cache change flipped schema-following from 88% to 18% validity, so serving numerics are part of the experimental identity (Section 4).

2 Related work

AAPT sits at the intersection of four converging lines; none of the individual mechanisms is new, and the contribution defended here is their integration around *expiring action windows* (Appendix I maps the closest systems feature by feature).

GUI world models and lookahead planning. WebDreamer (Gu et al. 2024) scores imagined outcomes of candidate web actions before executing; ViMo (Luo et al. 2025) predicts future mobile GUI observations; the Computer-Using World Model (Guan et al. 2026) predicts the next desktop UI state for test-time action search; MobileDreamer (Cao et al. 2026) is the closest predictive structure – a recursive *tree-of-prediction* over candidate actions and predicted states – and Code2World (Zheng et al. 2026), WebWorld (Xiao et al. 2026), and DynaWeb (Ding et al. 2026) strengthen the substrate. In all of these the imagined future is deliberative context: the tree is never activated as a temporary executable policy, no observer routes live frames through its branches, and horizon is not set by planner latency or deadlines.

Anticipatory planning. TraceR1 (Liang et al. 2026) predicts a trajectory, executes the first action, and replans – receding-horizon control that keeps one expensive call on every step's critical path; StreamAgent (Yang et al. 2025) anticipates future evidence in streaming video but targets perception. Our P1 baseline instantiates this pattern and scores zero in the contested regime: anticipation without pre-computation lengthens the decisive path.

Speculative execution and latency hiding. Interactive Speculative Planning (Hua et al. 2024), Dynamic Speculative Planning (Guan et al. 2025), and Speculative Actions (Ye et al. 2025) overlap fast approximation with slow verification or pre-launch likely next calls, resting on speculated work being reversible; AgenticCache (Kim, Wu, and Tambe 2026) replays cached plan transitions under asynchronous validation. AAPT inverts the commitment rule: nothing is executed speculatively – several actions are *prepared* and exactly one fires when a live observation satisfies its guard before its deadline, the appropriate discipline for GUIs where plausible actions are not safely pre-launchable.

Continuous and dynamic GUI observation. The Agent-Computer Observation Interface (Li and Shi 2026) continuously samples the screen and gates changed frames – improving perception, not decision speed; DynamicGUIBench (Liu et al. 2026) and living-screen benchmarks (Yao et al. 2026) formalize continuous-time GUIs where observation bears cost. AAPT adds the conditional-execution layer above this observation layer; our transfer evaluation runs on the DynaCU-Bench tasks introduced with AOI.

Verified replay and procedural memory. PreAct (Li 2026) compiles a successful trajectory into a verify-before-replay state machine; SkillDroid (Chen et al. 2026), ActionEngine (Zhong et al. 2026), Agent Workflow Memory (Wang et al. 2024), and SKILL.nb (El Hattami, Chapados, and Pal 2026) induce reusable gated skills from *completed experience*. AAPT instead synthesizes a short-lived contingency policy for futures not yet experienced; promoting validated subtrees into PreAct-style memory is complementary future work.

Classical foundations. Conditional policies over action-observation branches are the native representation of POMDP planning – DESPOT (Ye et al. 2017), bounded-branch contingency planning (Meuleau and Smith 2003;

[1]Appendices A–M are provided as a separate technical appendix in the supplementary material.

Blumenthal and Shani 2023), macro-action planning (He, Brunskill, and Roy 2011) – and behavior trees (Iovino et al. 2020) give modular reactive policies with fallbacks. AAPT does not claim to invent conditional policy trees; it has a frozen multimodal model *synthesize and refresh* deadline-aware GUI policies of this classical shape. To our knowledge no reviewed system implements and evaluates the complete combination (Appendix I), stated as a bounded literature-review conclusion.

3 Method: Adaptive Anticipatory Policy Trees

3.1 Setting and success criterion

We consider environments with transient, sometimes irreversible decision points, where success couples correctness with timing: success = correct action $\wedge$ action before deadline. The goal is *not* to run a large multimodal model at frame rate; it is to let the slow model prepare enough conditional behavior that the system remains operational while the next slow reasoning call is unavailable. As an instrument, AAPT removes exactly one factor – generation on the decision-time critical path – holding the model, its information, and the task fixed, so the presence or absence of its advantage discriminates between the causal accounts of Section 1.

3.2 The latency-coverage rule: a falsifiable timing prediction

Let L_{p95} be the 95th-percentile end-to-end planner latency, M a safety margin (500 ms in all experiments), and T_{cover} the wall-clock interval a prepared tree remains valid for. The tree must bridge the planner's own unavailability:

$$T_{\text{cover}} \;\geq\; L_{p95} + M. \tag{1}$$

Eq. (1) is not a tuning heuristic but the hypothesis's quantitative content: it names, from independently measured quantities, the preparation budget below which the advantage cannot exist and above which it must appear – a prediction the prep-budget sweep tests directly (the measured crossover brackets $L_{p95}+M \approx 2.4$ s; Appendix F). With $\mathbb{E}[\Delta t]$ the expected transition duration, the horizon $n \approx \lceil T_{\text{cover}}/\mathbb{E}[\Delta t] \rceil$ is a *latency-coverage* quantity, not an open-ended lookahead depth; trees whose declared coverage misses the measured budget (3,500 ms here) are rejected at validation.

3.3 Tree representation

A policy node is a tuple $v = (z_v, C_v, A_v, D_v, Q_v, R_v, F_v)$: a predicted state description, an observable guard, a pre-authorized primitive action or short chunk, an action deadline, a calibrated confidence, a risk class, and fallback behavior – the fallback realized not as a tree node but as the runtime abstain path (Section 3.6). The implemented tree is a *flat JSON node list* with string-id children – deliberately not a recursive structure, which weaker schema-followers reliably fail to emit (Section 5.4); the full JSON Schema and a canonical compiled example (a root `WAIT` with three guarded children, each an executable PyAutoGUI string, so the executor never synthesizes an action the planner did not authorize) are in Appendix B.

3.4 Adaptive branching under uncertainty

Each node has an individual branching factor $1 \leq K_v \leq K_{\max}$. When the current action is certain but its *outcome* is uncertain, the node allocates one branch per likely outcome, keeping the smallest set whose cumulative probability reaches a coverage target ρ:

$$\sum_{i=1}^{K_v} P(s_{v,i}) \;\geq\; \rho, \qquad K_v \leq K_{\max}. \tag{2}$$

Probability mass not represented by explicit branches is owned by the abstain path – and a compiled tree must *not* materialize that fallback as a branch carrying a default action: over-budgeted planners invent exactly such armed catch-all branches, which then execute wrong actions on ambiguous frames (Appendix E). The hard budgets ($K_{\max}$, depth $n_{\max}$, node count, coverage) are enforced numerically by the post-hoc validator; the confidence/risk thresholds and ρ are design rules realized as planner-prompt instructions (full allocation rule in Appendix J). On the three-outcome benchmark the rule is degenerate; Appendix E probes it non-degenerately with a five-outcome family, including an uneven-probability arm where covering the four most likely outcomes ($\rho = 0.95$, $K_{\max} = 4$) requires dropping exactly the rarest branch. Section 5.6 finds a sharp interior optimum: the branch budget must reach the event's outcome-set size, and depth beyond its decision structure buys nothing.

3.5 Planning as guided decoding, not fine-tuning

The planner is the frozen model itself, prompted with the task, the current screenshot, and the budget constraints, decoded under a JSON-Schema grammar (vLLM `guided_json`, backed by xgrammar (Dong et al. 2024)). No training was required, but two engineering findings – both initially misdiagnosed as capability gaps – were (Appendix J): *schema shape dominates grammar constraints* (a `oneOf` action alternation induced 88.1% tree validity; removing it fixed validity even unconstrained, guided decoding plus the simplified schema yields 99.57% pooled, and a planned LoRA fine-tune was cancelled), and *post-hoc validation plus one bounded retry closes the residual gap* (7/336 retries, 4 recovered, 0.89% final invalidity).

3.6 Runtime: observe, route, execute

The fast path never generates free-form actions. The screen is captured at 30 FPS and a pixel-difference gate suppresses static frames. On a changed frame, the low-token observer receives the frame and *only the currently eligible branch guards*, and returns – in at most 64 tokens, temperature 0 – a single JSON object {`branch_id, confidence, reason`}, with `NONE` as the abstain value. The runtime applies three suppression checks: (i) the returned id must be currently eligible; (ii) the *planner's* stored confidence for the branch must clear $\tau_m = 0.45$ – a check with a sharp boundary: when the task makes outcome probabilities explicit the planner writes those *priors* into Q_v, and a family whose largest prior is below τ_m becomes unroutable (Appendix E); (iii) the decision age must not exceed the branch deadline.

The action must also be on the pre-authorized allowlist; after firing, only the node's children become eligible; no-match, ambiguity, expiry, or verification failure falls back to replanning. A branch returned after its deadline is suppressed, never executed – which is why T2's incorrect-action rate (T2 is the protocol identifier for AAPT, Section 4.2) is zero throughout Section 5: a property of the execution design on this benchmark's error surface, not an empirical safety finding, and a budget-conditional one (Appendix E). The observer prompt, a real request/response pair, and the general match-score design the deployed single-candidate instantiation simplifies are in Appendix C.

Scope. Everything above is implemented and measured. Asynchronous tree refresh and reusable policy memory (promotion of validated subtrees into PreAct-style (Li 2026) skills) are deliberately unclaimed design sketches (Appendix J).

4 Experimental setup

4.1 The contested-window benchmark

The primary testbed is a local, instrumented timed-GUI benchmark (`key_prompt`): a server renders a web page in Firefox on a virtual display; after a trial is armed, a transient prompt appears after a fixed 750 ms delay and requests exactly one of three keys (`F12`, `Enter`, `Esc`), chosen by a hidden schedule derived deterministically from the trial seed, and disappears when the event window (250–2000 ms depending on the experiment) expires. A trial succeeds iff the agent delivers exactly the requested keypress before the server-side deadline; a different key is scored separately as an *incorrect action*. Scoring is server-side and deterministic – no judge model. The agent is told only that "a temporary prompt will request F12, Enter, or Esc"; before the event the screen is idle, so a slow planner can prepare but nothing reveals which key will be requested.

The fixed delay does not leak. No condition receives the delay value, countdown, or any pre-event cue – the delay can inform *that* an event will occur (which the task text already says), never *which* branch to take. R1, which exploits timing alone, scores zero throughout; and a paired block randomizing the delay over $[500, 1000]$ ms leaves T2's gain unchanged (discordant $15/0$, $p = 6.1 \times 10^{-5}$; Appendix F).

Paired per-seed design. Every comparison is paired: per seed, each condition runs one episode with the byte-identical hidden schedule (condition order alternating), assessed with the exact two-sided McNemar test on discordant pairs. Pairing is a validity requirement: reaction latency here is nearly deterministic, so whether a 42-seed draw is contested at a fixed window is close to all-or-nothing – absolute rates do not transfer across seed draws, while within-pair comparisons remain valid. Seeds come from pre-registered disjoint ranges, key-balanced, never reused (Appendix A).

Window calibration. The contested window – where the reactive baseline is neither floored nor at ceiling – moves with serving speed and seed draw, so windows were recalibrated per stage from *reactive-only* scans before any treatment data; all deviations from pre-registered windows are reported in Section 5.

4.2 Conditions

R0 (reactive baseline): screenshot → model → one action, repeated; the critical path is a full model round trip after the event fires. **T2** (AAPT): the slow planner compiles a bounded policy tree before the event (Section 3); after it, the low-token observer routes change-gated frames and the pre-authorized action executes after the eligibility, confidence-floor, and deadline checks. **R1** (open-loop chunk): a short unconditional action sequence requested before the event, executed without observation – "just act early." **P1** (predict-and-replan): predict two steps, execute one, replan after each observation – receding-horizon anticipation in the spirit of TraceR1 (Liang et al. 2026). **O1** (oracle routing): the compiled tree routed by ground truth at zero decision latency, gated on the same change signal. If T2's advantage were merely "acting before a full round trip," R1 and P1 should share it; O1 measures the headroom lost to observer decode and routing errors.

4.3 Models

Six checkpoints are studied: Holo-3.1-35B-A3B (H Company 2026) (primary, Qwen3.5-MoE computer-use tune), Qwen3.6-35B-A3B (Qwen Team 2026) (confirmation, same class, untuned generalist), UI-Venus-1.5-30B-A3B (inclusionAI 2026), EvoCUA-32B (Meituan 2026), UI-Ins-32B (Tongyi-MiA 2025), and UI-TARS-1.5-7B (Qin et al. 2025) as pre-registered boundary models (Table 2; full inventory with measured decode speeds and serving details in Appendix J.5). All are served frozen (no fine-tuning anywhere in this paper) with vLLM (Kwon et al. 2023), pinned to exact Hugging Face revisions; decode speeds are measured per model with a token bench ($R^2 \geq 0.99$), so costs are reported in output-token equivalents that transfer across hardware. Both AAPT roles are played by the *same* frozen checkpoint in every experiment – planner at temperature 0.2 under guided JSON decoding, observer on a 64-token cap – so the comparison isolates the architecture, not a model pairing. The pixel-difference change gate is fixed across all experiments (constants in Appendix J); at 30 FPS it contributes at most one frame interval ($\leq$ 33 ms), which is why T2's reaction-time floor sits near the observer's own latency.

4.4 Serving invariants and reproducibility

A methodological incident shaped the serving protocol: changing only the vLLM launch line (a revision pin and an offline flag) changed the engine's `torch.compile` cache key, and planner tree validity collapsed from 88.1% to 18.3% with identical weights, sampler, prompts, and screenshots; relaunching with the byte-exact original command line restored the behavior. Schema-following at temperature 0.2 is, on this stack, knife-edge sensitive to kernel-level numerics. All data collection therefore verifies the compile-cache key before any trial, each model's key is frozen in provenance (Appendix D), and any key change voids collected data.

4.5 Pre-registration practice

Each stage was specified in a written plan before execution – windows, seed ranges, primary endpoints, decision rules, acceptance criteria – and deviations are reported alongside the results they affect. The strongest instance is the gate confirmation (Section 5.3): the two confirmation models, their predicted gate outcomes, and the interpretation of every possible result were committed to the repository before either model produced a single number. No model was added after results were seen; no seed re-rolls or post-hoc sample-size increases occurred anywhere.

5 Results

All results are paired per-seed comparisons with exact two-sided McNemar tests, on frozen models, with the serving invariants of Section 4.4 verified before every block; unless stated otherwise the benchmark is `key_prompt`, the model Holo-3.1-35B-A3B, and 42 pairs are run per window. Table 1 is the paper's prediction ledger: every claim below was declared before its data, including two that failed, so no subsection has to be read as post-hoc selection; the subsections work through the ledger in order, each a test of the critical-path hypothesis.

5.1 Test 1: the contested-window advantage

Below the reactive floor. At 500 ms – below R0's reaction floor – T2 succeeds 28/42 vs. R0 0/42, all 28 discordants favoring T2 ($p = 7.5 \times 10^{-9}$); at relaxed windows both reach ceiling, at 250 ms both floor. The stronger claim is that T2 beats a *functioning* reactive baseline: a four-window scan (Appendix L.3) locates the contested regime at 600–650 ms, where T2 beats a degraded-but-alive R0 by 25/0 and 19/1 discordant pairs ($p = 6.0 \times 10^{-8}$, 4.0×10^{-5}). We report an honest deviation: the pre-registered primary was 700 ms, where R0 turned out to be near ceiling – not significant ($p = 0.25$), and we do not claim it; this stage rests on the pre-registered *secondary* windows. Incorrect actions: 0% for both conditions at every window.

Declared-primary confirmation. That deviation was closed by a confirmatory re-run with 650 ms *declared as primary before data collection*, on the tuned configuration and fresh seeds. The primary passes: T2 33/42 (0.79) vs. R0 21/42 (0.50), $p = 1.8 \times 10^{-3}$, with every secondary window also significant, incorrect-action 0 everywhere, tree validity 1.0, and observer branch accuracy 0.86–0.93. This is the headline result: a pre-registered, declared-primary win over a live reactive baseline in the contested regime.

Costs and misses. The win is not free: T2 generates $3.2\times$ R0's completion tokens per trial, almost all of it tree compilation (283.5 tokens, p50 1.81 s, off the critical path); the runtime observer is *cheaper* per call than R0's decisive calls (59.3 vs. 105.8 tokens; full ledger in Appendix L.4). Of T2's nine misses on this block, five are correct routes converted too late, three wrong-branch routes whose presses also landed post-deadline, one a correctly suppressed late decision: late conversion, not tree quality, dominates – consistent with the O1 headroom analysis below.

5.2 Test 2: pre-computation is the mechanism

All five conditions were compared at the contested windows (Appendix L.2 shows the full comparison figure). T2 dominates both anticipatory baselines: at 650 ms, T2 reaches 0.69 while R1 and P1 score *zero* (both $p = 3.7 \times 10^{-9}$; T2>R0 $p = 3.8 \times 10^{-6}$), mirrored at 600 ms. Incorrect actions: 0 in every condition. Why do R1 and P1 fail? Every reactive-family condition needs two model calls (`WAIT`, then the decisive action), and anticipatory prompting *lengthens* outputs: measured per-decision critical paths are 273 ms/30 tokens for the T2 observer vs. 411 ms/48 (R0), 459 ms/60 (R1), 484 ms/66 (P1) – their decisive call lands *later* than plain R0's. Acting early does not help by itself; only moving the reasoning off the critical path turns anticipation into a latency win.

The oracle O1 (0.86 at 650 ms, 0.79 at 600 ms) sits above T2 at both windows, but neither per-window test survives Holm–Bonferroni, so we report the O1−T2 gap as an *estimated headroom*: 0.167 (95% CI $[0.04, 0.30]$) and 0.143 ($[0.04, 0.25]$). The direction is consistent (discordants 8/1, 6/0), and all eight 650 ms oracle-only pairs are `routed_late` misses for T2 – never a wrong branch: the binding constraint of the architecture is the observer, not the planner.

5.3 Test 3: replication within the declared class

Fresh-seed repeats (same model). Two further 42-pair blocks on fresh seeds at a 550 ms sub-cliff window replicate the direction: pooled 84 pairs, T2 0.619 vs. R0 0.143, 40 T2-only pairs, *zero* R0-only ($p = 1.8 \times 10^{-12}$). These runs also exposed the knife-edge property motivating the paired design (Section 4.1): per-seed-set difficulty variance, not time drift (Appendix K.3).

Pre-registered confirmation on an untuned generalist. The boundary map (Section 5.4) yields a falsifiable hypothesis: the gain appears iff a model has (a) sub-second observer decode, (b) schema-valid flat-tree planning under guided decoding, and (c) branch-routing accuracy $\geq$0.85. We tested this *confirmatorily*: two models were selected by predicted gate outcome, with predictions and their interpretation committed before any data. Qwen3.6-35B-A3B – an *untuned generalist* of Holo's architecture class – passed all three gates (decode 180–201 tok/s; validity 30/30; branch accuracy 0.969), and the pre-declared consequence followed: T2>R0 replicates in all three 42-pair blocks at the declared 600 ms primary, pooling to 126 pairs at 0.778 vs. 0.341, discordant 60/5, $p = 4.9 \times 10^{-13}$, every block individually significant (Appendix L.5). Two conclusions fixed in advance are now licensed: the gain is not a Holo idiosyncrasy, and it does not require an agentic fine-tune. The pre-fixed claim ceiling still applies: this upgrades the scope to *the Qwen3.5-MoE class, including an untuned generalist* – the confirmation model shares Holo's architecture family, so it is not evidence of architecture-agnostic transfer.

The second pre-registered model, UI-TARS-1.5-7B, was predicted at risk on gate (b) and fails it persistently: root-only `WAIT` trees with outcome coverage 0/30 before and

Table 1: The prediction ledger: what was declared before each block's data, and what happened. Failed predictions are retained at full weight, and every deviation is reported in the listed section.

Stage	Declared before data (primary endpoint)	Outcome
Contested scan	windows, 700 ms primary (T2>R0)	n.s. (R0 near ceiling); secondaries carry it (§5.1)
Confirmation	650 ms declared primary (T2>R0)	**0.79 vs. 0.50,** $p = 1.8 \times 10^{-3}$ (§5.1)
Mechanism	conditions, windows (T2 vs. R1/P1/O1)	R1=P1=0; O1 headroom estimate (§5.2)
$k \times n$ ablation	grid, 600 ms primary (frontier shape)	knee at $k=$ outcome-set size, $n=1$ (§5.6)
Fresh-seed repeats	550 ms, seeds (direction repeats)	pooled $p = 1.8 \times 10^{-12}$, 40/0 discordant (§5.3)
Boundary map	gate predictions (per-model gates)	each model fails exactly one gate (§5.4)
Gate confirmation	predictions committed pre-token (2 unseen models)	both graded correct; $p = 4.9 \times 10^{-13}$ (§5.3)
Oracle probe (R1 rev.)	committed: oracle does *not* beat R0 on UI-Venus	**prediction failed**: 22/42 vs. 12/42, $p = 0.021$ – routing gate causal (App. G)
DynaCU power run (R1 rev.)	3 episodes, majority-of-3; 80% concentration rule	confirmed: 100%/100% concentration; aggregate stays a tie (§5.5)
Five further Round-1 revision entries (five-outcome family, budget cells, uneven arm, prep sweep, delay jitter) continue in Appendix L.1.		

after the one permitted adaptation round leave gate (c) unreachable, so no paired block was run; its reactive arm also floors at 0/30 – a contract-expression boundary, not a claim about the model's GUI skill.

5.4 Test 4: three capability gates; routing is causal

Table 2 summarizes all six models. T2>R0 appears on exactly the two models that pass all three gates, and each failing model localizes a different gate. EvoCUA-32B: near-perfect validity but ∼25 tok/s decode, so routing lands past every deadline (T2 0/43 even at 4–10 s windows, vs. R0 22/43). UI-Venus-1.5-30B-A3B: fast and schema-valid, but routes wrongly (branch accuracy 0.39) and builds low-coverage trees; its paired block is a tie ($p = 0.625$). A pre-registered oracle probe makes the routing gate *causal*: ground-truth routing over the same frozen planner flips the tie to a win (22/42 vs. 12/42, $p = 0.021$) – against our own committed prediction – with success exactly equal to $\{\text{schema-valid tree} \wedge \text{true outcome covered}\}$. The probe session's trees had higher coverage than the original block's (recall 0.59 vs. 0.22), so the licensed statement is conditional: when tree coverage clears the reactive baseline, routing quality alone separates tie from win, and gate (c) and planner outcome-recall are jointly, not singly, binding (Appendix G). UI-Ins-32B fails both axes at once (T2 0/50 while R0 works at ≈0.5). UI-TARS fails gate (b) as above.

5.5 Test 5: transfer is conditional on pre-enumerability

To test transfer beyond the local benchmark, we paired a reactive agent and an AAPT agent (both frozen Holo) on DynaCU-Bench, a third-party dynamic-browser benchmark (Li and Shi 2026), using its 39 deterministic DOM-scored tasks in the deadline-focused categories. A pre-registered three-episode power run – majority-of-3, with the concentration decision rule committed before any episode was scored – confirms both parts of the single-episode finding. The aggregate remains a tie: reactive 7/39 vs. AAPT 6/39, discordant 5/4, $p = 1.0$. The dissociation, however, is confirmed (Appendix L.6): *every* AAPT-only win (4/4) is a dashboard deadline task whose winning response is pre-enumerable at compile time – the observer dispatched pre-armed responses 140–215 ms after gated frames while the reactive agent's ∼437 ms round trip missed the windows – and *every* reactive-only win (5/5) is a carousel or toast/form task needing multi-step sequencing or values revealed only after compile time. One single-episode pattern did not survive the majority rule: the reaction games floor at 0 for both arms, so the AAPT side of the split rests entirely on dashboards. The supported external claim is deliberately narrow: a pre-computed bounded policy tree transfers its deadline advantage when outcomes and actions are pre-enumerable; it does not replace reactive sequencing, late value reading, or positional grounding.

5.6 Test 6: the sizing rules hold quantitatively

These experiments test the quantitative predictions of Sections 3.2 and 3.4, each with an outcome that could have falsified the corresponding rule; full grids, the frontier figure, and the token sweep are in Appendix K. The $k \times n$ sweep (42 pairs per cell, 600 ms recalibrated primary) confirms the knee at $(k{=}3, n{=}1)$ – branch budget exactly the event's outcome-set size at minimal depth, 42/42 and strictly dominating R0 (19/0 discordant, $p = 3.8 \times 10^{-6}$, window-robust) – while budgets below the outcome-set size fail by planner *validity*, not just coverage ($k{=}2$ is the "uncanny valley," 0.405), and both $k < 3$ cells score *below* the reactive baseline: an underbudgeted tree is worse than no tree. The five-outcome replication (Appendix E) confirms both findings track the outcome-set size and adds that mis-set budgets do not just miss, they *execute wrong actions*. Extra depth is Pareto-dominated: the planner builds the same four-node depth-1 tree even when 13 nodes are allowed. We record $(3, 1)$ as the recommended operating point; the confirmatory experiments deliberately kept the pre-registered $(3, 2)$ configuration so replication is not confounded with tuning. The observer token sweep selects a 64-token cap by the pre-registered rule (32 truncates the observer's JSON, 128 adds latency and drifts branch accuracy below the gate), lifting the contested-regime T2 ceiling from ∼0.70 to 0.79/0.88 at 600/650 ms; and the window-sensitivity analysis (Appendix K.3) confirms the contested window is a property of the model-serving-seed triple – shifts across sessions are seed-draw difficulty variance, not server drift – kept honest by the recalibration protocol of Section 4.1.

Table 2: Cross-model gate matrix: the gain appears exactly where all three gates (column heads) pass; "n/r" = not reached. [1] After one documented prompt-adaptation round. [2] A ground-truth-routed probe (O1) flips this to 22/42 vs. 12/42, $p = 0.021$ (Appendix G). [3] Stopped by protocol: no branch set to route (§5.3).

Model	(a) decode < 1 s	(b) flat-tree validity	(c) branch acc. ≥ 0.85	Outcome (T2 vs. R0)
Holo-3.1-35B-A3B (MoE)	✓ $\sim$0.2 s	✓ $\sim$1.0	✓ 0.86–0.93	T2$\gg$R0 ($\times$4 blocks)
Qwen3.6-35B-A3B (MoE)	✓ 0.15–0.45 s	✓ 1.00	✓ 0.969	T2$\gg$R0 ($\times$3 blocks)
UI-Venus-1.5-30B-A3B (MoE)	✓ $\sim$0.15 s	✓ 0.95[1]	$\times$ 0.39	n.s. (11/42 vs. 13/42)[2]
EvoCUA-32B (dense)	$\times$ $\sim$2.5 s	✓ 0.98	n/r	T2 0/43 vs. R0 22/43
UI-Ins-32B (dense)	$\times$ $\sim$1.5 s	$\times$ 0.10[1]	n/r	T2 0/50 (R0 ≈ 0.5)
UI-TARS-1.5-7B (dense)	✓ 0.5–1.1 s	$\times$ 0.50[1], cover 0/30	n/r	paired block not run[3]

6 Discussion

The capability-gate model is the organizing result. The six-model study produced a *predictive* model of where the gain lives: three capabilities must hold simultaneously – (a) sub-second observer decode, (b) schema-valid flat-tree planning, (c) accurate branch routing – each demonstrated by a model that fails exactly it, then tested confirmatorily with gate predictions committed before either unseen model's first token, both graded correct. This turns a fragile single-checkpoint observation into a twice-confirmed hypothesis with a mapped boundary – and a gate battery practitioners can measure in minutes.

What buys the gain is moving decode off the critical path. It is tempting to read the headline as arithmetically inevitable – a 567 ms round trip cannot fit a 600 ms window – but the arithmetic makes two further predictions the data refute: the early-acting baselines score zero (their decisive calls are *longer* than R0's), and UI-Venus passes the speed gate yet ties, until the oracle probe shows its deficit was routing. The round-trip numbers say a contested window exists; which computation must move, how much preparation that requires (Eq. 1), and which capabilities must coexist are the paper's findings, each declared before its data.

Economics: the observer premise is an architecture question. The A3B mixture-of-experts models decode at 170–201 tok/s, putting a 30-token routing call at 150–450 ms; the dense 32B models put the same call at 1.2–2.5 s – past a 600 ms window before reasoning quality enters. Costs are in output-token equivalents, so any deployment can convert its measured tok/s into a feasible-window bound; a survey of documented response windows (Appendix H) bounds where this matters – real sub-second windows exist, and windows beyond $\sim$10 s need no anticipation.

When not to use AAPT. A precompiled tree needs its winning response to be *expressible at compile time*: it loses to plain reactive execution on multi-step sequencing, late-revealed values, and late-revealed coordinates (Section 5.5). In the DynaCU tie, AAPT's and reactive's wins are disjoint task populations – arguing for a hybrid controller rather than either architecture alone.

Observer headroom is the next lever. The estimated oracle gap ($\mathrm{O1} - \mathrm{T2} \approx 0.14$–$0.17$; Section 5.2) says the compiled trees are better than the live system routing them; a small distilled guard matcher, or OCR/template guards bypassing the model on easy branches, could close part of the gap without touching the planner.

Reproducibility of schema-following. The compile-cache incident (Section 4.4) has a general moral: near-threshold structured-output behavior can be flipped by kernel-level numerics most provenance practices do not record; such studies should pin and report the compiled-kernel identity of their serving stack.

Limitations. Every carried caveat from the experimental record is stated in full in Appendix M; we summarize them here. *Scope*: the positive result is bounded twice – it replicates within the Qwen3.5-MoE class and is capability-gated everywhere else; three of five non-Holo models show no gain at all. *Breadth*: the main evidence chain runs on one scenario family plus one external benchmark, designed to isolate the mechanism. *Knife-edge windows*: absolute rates are not comparable across seed draws; only paired per-seed comparisons carry evidential weight. *Deviations*: two pre-registered primary windows missed their regime (both reported, one closed by a fresh declared-primary run); a validity sub-target was near-missed (96.69% vs. 97%). *Late binding*: pre-compiled trees cannot bind parameters revealed at fire time (the `click_target` boundary). *Transfer*: the DynaCU dissociation is a boundary claim over 4 vs. 5 discordant tasks, with all AAPT-side wins in one category. *Thin coverage*: UI-Venus rests on two blocks with between-session tree-coverage instability; UI-TARS's reactive 0/30 is a harness-portability datum. *Headroom*: the oracle gap means reported T2 numbers understate what the trees contain. *Unvalidated components*: asynchronous refresh and policy memory are unvalidated sketches; no fine-tuning was performed anywhere.

7 Conclusion

Under contested GUI deadlines, the bottleneck of computer-use agents is not what the model knows but when it decodes. AAPT beats a live reactive baseline at its pre-registered primary window; the advantage is caused by pre-computation, not early acting; it replicates across fresh seeds and Qwen3.5-MoE checkpoints, including an untuned generalist; and it is capability-gated, each gate demonstrated by a model that fails it – nothing architecture-agnostic, and external transfer conditional on pre-enumerable responses.

References

Blumenthal, O.; and Shani, G. 2023. Rollout Heuristics for Online Stochastic Contingent Planning. In *Proceedings of AREA 2023 (EPTCS 391)*, 89–101.

Cao, Y.; Zhong, Y.; Zeng, Z.; Zheng, L.; Huang, J.; Qiu, H.; Shi, P.; Mao, W.; and Wan, G. 2026. MobileDreamer: Generative Sketch World Model for GUI Agent. arXiv:2601.04035.

Chen, Q.; Bellucci, A.; Sun, Z.; and Jacucci, G. 2026. SkillDroid: Compile Once, Reuse Forever. arXiv:2604.14872.

Ding, H.; Liu, P.; Wang, J.; Ji, Z.; Cao, M.; Zhang, R.; Ai, L.; Yang, E.; Shi, T.; and Yu, L. 2026. DynaWeb: Model-Based Reinforcement Learning of Web Agents. arXiv:2601.22149.

Dong, Y.; Ruan, C. F.; Cai, Y.; Lai, R.; Xu, Z.; Zhao, Y.; and Chen, T. 2024. XGrammar: Flexible and Efficient Structured Generation Engine for Large Language Models. arXiv:2411.15100.

El Hattami, A.; Chapados, N.; and Pal, C. 2026. SKILL.nb: Selective Formalization and Gated Execution for Durable Agent Workflows. arXiv:2606.08049.

Gu, Y.; Zheng, B.; Gou, B.; Zhang, K.; Chang, C.; Srivastava, S.; Xie, Y.; Qi, P.; Sun, H.; and Su, Y. 2024. Is Your LLM Secretly a World Model of the Internet? Model-Based Planning for Web Agents. arXiv:2411.06559.

Guan, Y.; Hua, W.; Lan, Q.; Fei, S.; Ding, D.; Acharya, D.; Wang, C.; and Wang, W. Y. 2025. Dynamic Speculative Agent Planning. arXiv:2509.01920.

Guan, Y.; Yu, R.; Zhang, J.; Wang, L.; Li, L.; Qiao, B.; Qin, S.; Huang, H.; Yang, F.; Zhao, P.; Wutschitz, L.; Kessler, S.; Inan, H. A.; Sim, R.; Rajmohan, S.; Lin, Q.; and Zhang, D. 2026. Computer-Using World Model. arXiv:2602.17365.

H Company. 2026. Holo-3.1-35B-A3B model card. Hugging Face.

He, R.; Brunskill, E.; and Roy, N. 2011. Efficient Planning under Uncertainty with Macro-actions. *Journal of Artificial Intelligence Research*, 40: 523–570.

Hua, W.; Wan, M.; Vadrevu, S.; Nadel, R.; Zhang, Y.; and Wang, C. 2024. Interactive Speculative Planning: Enhance Agent Efficiency through Co-design of System and User Interface. arXiv:2410.00079.

inclusionAI. 2026. UI-Venus-1.5-30B-A3B model card. Hugging Face.

Iovino, M.; Scukins, E.; Styrud, J.; Ögren, P.; and Smith, C. 2020. A Survey of Behavior Trees in Robotics and AI. arXiv:2005.05842.

Kim, H.; Wu, Y.; and Tambe, T. 2026. AgenticCache: Cache-Driven Asynchronous Planning for Embodied AI Agents. In *Proceedings of the 9th Conference on Machine Learning and Systems*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles*.

Li, B. 2026. PreAct: Computer-Using Agents that Get Faster on Repeated Tasks. arXiv:2606.17929.

Li, B.; and Shi, N. 2026. Agent-Computer Observation Interfaces Enable Dynamic Computer Use. arXiv:2606.29472.

Liang, Y.; Zhou, S.; Gu, Y.; Tan, H.; Wu, G.; Dernoncourt, F.; Kil, J.; Rossi, R. A.; and Zhang, R. 2026. Anticipatory Planning for Multimodal AI Agents. arXiv:2603.16777.

Liu, E.; Pan, L.; Gao, Z.; Yang, Y.; Shi, C.; Liu, Y.; Wu, J.; and Li, Q. 2026. Benchmarking and Improving GUI Agents in High-Dynamic Environments. arXiv:2604.25380.

Luo, D.; Tang, B.; Li, K.; Papoudakis, G.; Song, J.; Gong, S.; Hao, J.; Wang, J.; and Shao, K. 2025. ViMo: A Generative Visual GUI World Model for App Agents. arXiv:2504.13936.

Meituan. 2026. EvoCUA-32B model card. Hugging Face.

Meuleau, N.; and Smith, D. 2003. Optimal Limited Contingency Planning. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*.

Qin, Y.; Ye, Y.; Fang, J.; Wang, H.; Liang, S.; Tian, S.; Zhang, J.; Li, J.; Li, Y.; Huang, S.; Zhong, W.; Li, K.; Yang, J.; Miao, Y.; Lin, W.; Liu, L.; Jiang, X.; Ma, Q.; Li, J.; Xiao, X.; Cai, K.; Li, C.; Zheng, Y.; Jin, C.; Li, C.; Zhou, X.; Wang, M.; Chen, H.; Li, Z.; Yang, H.; Liu, H.; Lin, F.; Peng, T.; Liu, X.; and Shi, G. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. arXiv:2501.12326.

Qwen Team. 2026. Qwen3.6-35B-A3B model card. Hugging Face.

Tongyi-MiA. 2025. UI-Ins-32B model card. Hugging Face.

Wang, Z. Z.; Mao, J.; Fried, D.; and Neubig, G. 2024. Agent Workflow Memory. arXiv:2409.07429.

Xiao, Z.; Tu, J.; Zou, C.; Zuo, Y.; Li, Z.; Wang, P.; Yu, B.; Huang, F.; Lin, J.; and Liu, Z. 2026. WebWorld: A Large-Scale World Model for Web Agent Training. arXiv:2602.14721.

Yang, H.; Tang, F.; Zhao, L.; Zhuang, X.; Lu, Y.; An, X.; Hu, M.; Zhang, X.; Swikir, A.; He, J.; Ge, Z.; Khan, M. H.; and Razzak, I. 2025. StreamAgent: Towards Anticipatory Agents for Streaming Video Understanding. arXiv:2508.01875.

Yao, J.; Huang, H.; Wu, D.; Chen, W.; Ai, H.; Wen, H.; Liu, Z.; and Guo, Y. 2026. Benchmarking Living-Screen-Native GUI Agents on Short-Video Platforms. arXiv:2606.04701.

Ye, N.; Ahuja, A.; Liargkovas, G.; Lu, Y.; Kaffes, K.; and Peng, T. 2025. Speculative Actions: A Lossless Framework for Faster Agentic Systems. arXiv:2510.04371.

Ye, N.; Somani, A.; Hsu, D.; and Lee, W. S. 2017. DESPOT: Online POMDP Planning with Regularization. *Journal of Artificial Intelligence Research*, 58: 231–266.

Zheng, Y.; Zhong, L.; Wang, Y.; Dai, R.; Liu, K.; Chu, X.; Lv, L.; Torr, P.; and Lin, K. Q. 2026. Code2World: A GUI World Model via Renderable Code Generation. arXiv:2602.09856.

Zhong, H.; Faisal, F.; França, L.; Leesatapornwongsa, T.; Szekeres, A.; Rong, K.; and Nath, S. 2026. ActionEngine: From Reactive to Programmatic GUI Agents via State Machine Memory. arXiv:2602.20502.