Position: Context Alignment Is a First-Class Learning Problem in Agentic UI

Tassilo Klein [1] **Sebastian Wieczorek** [1]

Abstract

This position paper argues that standard context windows fail to capture the partitioned and temporally inconsistent nature of Agent–User Interaction (AG-UI). As Large Language Models evolve into agents controlling persistent interfaces, the traditional framing of context as a read-only token sequence has become a limiting abstraction. In AG-UI, context is not a static buffer but a concurrent control problem: users and agents asynchronously manipulate a shared, graph-structured state. We introduce **context alignment**, a framework where agents must synchronize their internal deliberative context with a rapidly mutating, partially observable interface. We propose a four-layer taxonomy of AG-UI context, characterize the addressability problem, and define temporal drift as the divergence between inference-time belief and execution-time state. We conclude by outlining an alignment objective that treats user interface state as a co-evolving control variable, distinct from retrieval or memory.

1. Introduction

The deployment of large language models has shifted rapidly from isolated question–answering toward persistent, agentic interaction. Modern Artificial Intelligence (AI) systems are increasingly embedded in document editors, code environments, web interfaces, dashboards, and multimodal applications, where they act continuously on shared artifacts alongside users. In these settings, agents must reason not only about tasks and tools, but about evolving interaction state, implicit user intent, and emergent partial commitments.

This evolution reflects a broader shift in interaction regimes. Early chat-based systems positioned the language model as an advisor: the user remained the sole controller of system state, and interaction unfolded through turn-based text. Tool-augmented agents extended this model by allowing the agent to execute backend actions on the user's behalf, yet control remained largely serialized and mediated.

By contrast, emerging Agent–User Interaction (AG-UI) systems place both human and agent in direct, concurrent control of shared, addressable interaction state, such as documents, code buffers, or interface components. In this regime, correctness is no longer determined solely by what the agent says or executes, but by whether its actions remain *valid at execution time* with respect to a continuously mutating interaction state.

Recent research has responded to these demands by elevating memory to a central primitive of agentic AI. A wide range of techniques, including retrieval-augmented generation (RAG) (Lewis et al., 2020), episodic memory, external scratchpads, tool-use traces, and reflection loops, have been proposed to improve long-horizon reasoning and coherence (Wei et al., 2022; Yao et al., 2022; 2023; Shinn et al., 2023; Schick et al., 2023; Madaan et al., 2023; Zhou et al., 2023; Hu et al., 2024; Lee et al., 2024; Zhang et al., 2024c; Li et al., 2025).

Modern multi-agent frameworks like AutoGen and MetaGPT incorporate structured communication protocols and role-based collaboration to coordinate specialized agents (Wu et al., 2023; Hong et al., 2024a). Increasingly, these efforts are framed through the lens of dual-process theory (Kahneman, 2011), where the base language model provides intuitive, fast (System 1) responses, while agentic architectures provide deliberate, slow (System 2) reasoning processes. In parallel, long-context models now condition on hundreds of thousands of tokens (OpenAI, 2023; Gemini Team, Google, 2024; Anthropic, 2024; Press et al., 2024). This recall capacity has enabled the emergence of generalist computer-use frameworks such as Agent S (Agashe et al., 2025), which leverage experience-augmented hierarchical planning to interact directly with OS-level interfaces.

Despite these advances, agentic systems remain brittle in interactive environments, with benchmarks repeatedly reporting failures where agents pursue stale plans or act on outdated assumptions (Kambhampati et al., 2024; Zhou et al., 2024; Liu et al., 2023b). Similarly, LongHorizonUI reports that cumulative error and contextual drift remain primary blockers for sustained reliability in long-horizon GUI tasks (Anonymous Authors, 2026b). Notably, many issues occur even when relevant information is present in context,

[1]Mantix. Correspondence to: Tassilo Klein <tk@mantix.cloud>, Sebastian Wieczorek <sw@mantix.cloud>.

Preprint. February 25, 2026.

suggesting the dominant failure mode is not forgetting, but acting on beliefs incompatible with current interaction state.

In this paper, **we argue that context alignment must be treated as a first-class learning problem, distinct from memory or retrieval.** Current approaches overwhelmingly treat context as an agent-internal object: a prompt, a memory buffer, or a retrieved set of past observations. This framing implicitly assumes that maintaining a sufficiently rich internal state is enough to ensure correct behavior.

We argue that this assumption breaks in interactive settings. Agentic systems operating under the AG-UI paradigm must instead be understood as maintaining *two coupled context processes*: **(i)** an *interaction context* that evolves rapidly in response to UI events and user dynamics, and **(ii)** a *deliberative context* that evolves more slowly through reasoning, planning, and abstraction. Failures arise when these processes fall out of synchrony.

This challenge becomes unavoidable as agents begin to generate and manipulate user interfaces directly. With the emergence of Generative User Interfaces (GenUI) (Leviathan et al., 2025), the interaction boundary becomes executable: UI elements are persistent artifacts participating in state transitions. Wang and Lu term this as *Agentic Media*, in which interfaces operate as processes that retain context and logic rather than being passive displays (Wang & Lu, 2025). Thus, generated widgets become persistent entities that *constrain future action*, requiring agents to respect their stateful semantics. In this regime, treating context as a static prompt or memory store is fundamentally insufficient. Specifically, this paper makes the following contributions:

- **A Taxonomy of AG-UI Context:** We decompose context into four layers and identify *addressability failure* and *temporal drift* as primary misalignment modes.
- **A Characterization of Alignment:** We define alignment as the minimization of drift between agent belief and execution-time state.
- **A Taxonomy of Consolidation Strategies:** We classify structural mitigation strategies and propose architectural primitives for action validity.
- **An Alignment Objective:** We outline an objective that penalizes actions unsupported by the interface at execution time.

2. Background and Related Perspectives

Our argument draws on insights from several research traditions that are often considered in isolation. In this section, we review these perspectives and explain why none of them, in isolation, resolves the context alignment problem.

Why Memory Is Insufficient. A growing literature frames memory as essential for agentic AI (Hu et al., 2024). Prominent architectures like ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023), and Toolformer (Schick et al., 2023) leverage these concepts, while systems like Generative Agents (Park et al., 2023) and MemGPT (Packer et al., 2023) demonstrate their utility for long-horizon coherence. These efforts are increasingly framed around the distinction between static LLM memory and interaction-driven agent memory, which must be dynamically encoded and consolidated over time (Cai et al., 2026). However, these approaches share a limiting assumption: they treat internal context (memory) as the primary locus of state, rather than modeling interaction as a co-evolving dynamical system. Similarly, while long-context models extend recall capacity (OpenAI, 2023; Gemini Team, Google, 2024), empirical studies show that attention collapse and salience dilution persist (Liu et al., 2023a; Press et al., 2024). Even perfect recall does not guarantee that an agent's beliefs remain synchronized with a rapidly evolving interaction state.

HCI and Implicit Intent. Decades of research in human–computer interaction (HCI) provide a complementary perspective. Classic work on mixed-initiative systems emphasizes that user intent is often implicit, emerging through interaction patterns rather than explicit commands (Horvitz, 1999). Empirical studies show that edits, hesitations, overrides, and undo actions carry rich semantic information about goals and preferences (Chi et al., 2001). More recently, guidelines for human–AI interaction stress the importance of maintaining predictability, respecting user control, and supporting graceful recovery from errors (Amershi et al., 2019). These principles implicitly assume that the system maintains an accurate representation of interaction state and user intent.

However, most agentic AI systems do not explicitly model these interaction dynamics. Instead, they attempt to infer intent indirectly from text or task descriptions, discarding temporal and behavioral cues that are central in HCI (Chi et al., 2001). Even sophisticated Graphical User Interface (GUI) agents (Hong et al., 2024b; Lin et al., 2024; Zhang et al., 2024b;a; Gao et al., 2025), supported by universal visual grounding models like UGround (Gou et al., 2025), primarily focus on perceiving and acting on GUI elements. Driven by these limitations, emerging research has begun to address fine-grained perception and coordinate-free grounding to improve stability (Anonymous Authors, 2026a; Wu et al., 2026). Yet, as Chen et al. observe, even with precise grounding, agents are prone to "attention drift," where background distractions or small-element mislocalization cause attention to diverge from the desired area (Chen et al., 2026). This gap motivates treating interaction context as a first-class object rather than an incidental observation stream.

Addressability $\neq$ Visual Grounding. It is essential to distinguish context alignment from visual grounding. Visual grounding solves spatial ambiguity: mapping a referent (e.g., "Submit" or "the login field") to pixel coordinates

$(x_{\min}, y_{\min}, x_{\max}, y_{\max})$. Context alignment addresses the orthogonal problem of temporal identity: ensuring the element at that location is the same semantic entity observed (e.g., 500ms) earlier. A perfect visual grounder fails if the underlying entity was swapped during inference latency. Addressability is thus a distributed systems problem, not a perception problem.

3. Empirical Signals of Context Misalignment

A central claim of this paper is that many failures observed in agentic systems stem directly from the *synchronization gap* between the agent's internal context and the evolving state of interaction, leading agents to select actions that are locally coherent but no longer valid at execution time. In this section, we synthesize empirical evidence supporting this claim. Across diverse settings, a consistent pattern emerges. Planning benchmarks show models fail when assumptions are invalidated (Valmeekam et al., 2023; Kambhampati et al., 2024). In WebArena, 34% of failures stem from DOM timing and 21% from inconsistencies (Hattami et al., 2025). OSWorld-G ($\sim$30% recognition) (Xie et al., 2025), WebCanvas (23% success) (Pan et al., 2024), and τ-bench (<25% pass^8) (Kirmani et al., 2024) reveal similar synchronization gaps. Mind2Web attributes failures to layout shifts (Deng et al., 2023), while tool-augmented agents invoke tools on outdated context (Jimenez et al., 2024).

Human–AI interaction studies further corroborate these patterns: users frequently intervene when systems fail to respect implicit intent encoded through prior interaction (Amershi et al., 2019). These results share a common structure: agents act correctly with respect to an outdated internal view of the world. Failures arise not from lacking information but because deliberative context is not reconciled with state changes during interaction. This reflects violations of a single invariant: *action validity under evolving state*. Increasing model scale, memory, or context length does not eliminate this failure mode. Instead, it often amplifies its cost by enabling agents to act more decisively on stale assumptions.

4. Loss of Single-Serializer Paradigm

We now turn from empirical symptoms to their structural cause. The emergence of Generative User Interfaces (GenUI) introduces a qualitative shift in how agentic systems interact with users and environments. Agents increasingly synthesize persistent interface artifacts containing executable code (e.g., event handlers, sandboxed logic (Anthropic, 2024; Cloudflare, 2024)) (Leviathan et al., 2025; Cao et al., 2025; Cheng et al., 2024). Prior work in model-driven engineering has generated UI logic from high-level specifications (e.g., MOBICAT (Zafar et al., 2024)), yet these systems rely on rigid transformation rules. GenUI extends this via learning-based synthesis that generalizes beyond hand-engineered models but introduces synchronization challenges: these artifacts run asynchronously in the user's runtime, accepting input even while the agent is offline.

Modern interactive systems are commonly organized around architectural patterns such as Model–View–Controller (MVC) or Model–View–Presenter (MVP). Despite differences, these paradigms share a critical assumption: interaction state is governed by a *single-writer guarantee*. While multiple input sources may exist (e.g., keyboard shortcuts, plugins, or system events), state transitions are ultimately serialized through a shared control boundary that enforces ordering, validity, and ownership. Text-based agentic systems largely preserve this assumption by operating inside a platform-enforced serialized loop: the model interprets user instructions, executes backend actions, and renders responses through a passive textual interface. GenUI fundamentally alters this interaction pattern. Importantly, this is not merely the introduction of another writer, but the loss of the serialization boundary that previously preserved action validity.

Collaborative editors already support multiple concurrent human writers, but they do so by coupling each writer to strong consistency machinery such as operational transforms, conflict-free replicated data types, explicit locks, or merge policies. These mechanisms ensure that each writer's actions are grounded in a consistent view of shared state at the time of execution. AG-UI introduces a qualitatively different writer: an agent whose actions are computed from a *lossy, delayed, and partial observation channel* of the interaction state. As a result, the agent may emit state mutations that are well-formed with respect to its internal deliberation, yet invalid with respect to the current serialized interface state.

The novelty of AG-UI is therefore not multiplicity of writers per se, but the loss of a serialization boundary that helps enforce action validity. As control becomes shared and asynchronous between human and agent, UI state is no longer downstream of agent reasoning but a co-evolving component of the system's control loop.

This shift has two important consequences. First, interaction context acquires semantics that cannot be reconstructed from agent-internal memory or textual history alone. UI artifacts encode task phase, partial commitments, and affordances that constrain which actions are valid at any point in time. Second, the cost of misalignment increases: an invalid agent action is no longer a benign textual error, but a stateful mutation that may overwrite user work or introduce irreversible inconsistencies.

From the perspective of context modeling, GenUI renders prompt-centric notions of context insufficient. Context must instead be understood as a shared, evolving control state spanning agent deliberation, backend state, and interaction

Table 1. **Comparison of Context Paradigms.** AG-UI shifts the focus from static text processing to dynamic state synchronization.

Feature	Chat-based Agent	AG-UI Agent
Context	Linear, Append-Only Log	Graph-Structured, Partitioned
State	Implicit in Text History	Explicit, Distributed (UI, Backend, Agent)
Dynamics	Synchronous Turn-Taking	Asynchronous, Concurrent
Failure	Hallucination, Retrieval	Addressability failure, temporal drift
Objective	Next-Token Prediction	Bounded Belief-State Drift

dynamics. In this regime, the UI functions as an *actuator*: generated interface artifacts persist, write into interaction state, and impose structural constraints on action.

5. Context and Misalignment in AG-UI

Given the loss of a single-serializer paradigm, the notion of *context* itself must be reformulated. We argue that AG-UI therefore introduces a qualitatively different context regime than chat-based or tool-augmented LLM systems. In this section, we formalize the notion of context in AG-UI, propose a taxonomy of context layers, and characterize two dominant misalignment modes (*addressability failure* and *temporal drift*) that arise under asynchronous interaction.

In traditional dialogue systems, context is implicitly modeled as a linear, append-only log of text: a sequence of user messages and agent responses. The underlying assumption is that access to textual history suffices to represent system state. This assumption—that an append-only textual log is an adequate proxy for state—quietly underlies most existing agentic LLM designs and is preserved by serialized interaction loops.

AG-UI breaks this assumption at an architectural level. Interaction state is no longer linear, immutable, or textually complete, but distributed across concurrent, non-linguistic interaction processes. The interface is not a passive log but a graph-structured, persistent artifact that evolves asynchronously alongside agent deliberation and user action. Table 1 summarizes these structural differences.

5.1. A Taxonomy of AG-UI Context

To align agent reasoning with interface dynamics, "context" must be decomposed into constituent layers. In AG-UI, state is not monolithic; it is distributed across the rendered interface, the agent's internal abstractions, and the backend environment. Crucially, these layers differ in update frequency, persistence, and observability.

We propose the taxonomy in Table 2. Misalignment arises when an agent treats one layer (e.g., ephemeral interaction state) as if it were another (e.g., persistent backend state).

5.2. Partitioned Context in AG-UI

Standard LLM-based agents assume a single, globally accessible context. In AG-UI, context is *partitioned* and *partially observable*. The total system state at any moment comprises three distinct components: (1) the **ground-truth interface state** (the actual structure and content of the UI as rendered); (2) the **agent's deliberative context** (its beliefs, plans, and reasoning traces); and (3) the **backend environment state** (databases, APIs, and application logic that underlie the interface).

The agent does not have direct access to the true interface state. Instead, it observes a lossy projection (typically a screen capture or partial DOM snapshot) that may omit, delay, or distort information about what currently exists on screen. Context alignment is therefore the challenge of maintaining an internal belief about the interface state that remains synchronized with the true state, despite observing it only through this imperfect channel. Alignment requires that the agent's action selection remains *supported* by the true interface state, avoiding actions that reference elements that have been deleted, moved, or modified since the last observation.

5.3. Why Linear Context Fails in AG-UI

A useful way to understand the limitations of existing context mechanisms is to ask why the linear, append-only abstraction succeeds in chat-based systems but fails in AG-UI. In dialogue settings, state evolves primarily through language. Utterances are consumed sequentially, references are linguistic, and past turns become irrelevant mainly through topical drift. Under these conditions, treating context as an ordered history is a reasonable approximation: recency correlates with relevance, and overwriting past assumptions is rare.

AG-UI breaks each of these assumptions. First, interaction state evolves through non-linguistic events such as cursor movement, selection, deletion, and reordering. Second, relevance is spatial and structural rather than temporal: an interface element created many steps ago may remain active, editable, and semantically salient. Third, users routinely invalidate prior assumptions implicitly by editing the interface rather than issuing explicit corrective instructions.

As a result, a linear context window forces the agent to conflate three distinct operations: remembering what happened, inferring what still exists, and deciding what is safe to act upon. This conflation is benign in chat, but destructive in AG-UI, where acting on stale assumptions can overwrite user work or corrupt shared state.

This clarifies why context length or retrieval depth does

Table 2. **A Taxonomy of Context in AG-UI.**

Context Layer	Definition	AG-UI Characteristic	Misalignment Failure
1. Ephemeral	Transient interaction micro-state (e.g., cursor position, hover state, pending input).	• High-frequency, event-driven. • Often omitted from agent observations.	**Race conditions**: Agent acts on outdated focus or overwrites active input.
2. UI State	Structural interface representation (e.g., DOM (Apparao et al., 1998), render tree).	• Addressable, graph-structured. • Persistent across turns.	**Stale references**: Agent targets deleted or modified UI nodes.
3. Task	Latent user intent governing the session.	• Implicit and evolving. • Expressed via edits rather than prompts.	**Goal drift**: Agent optimizes for obsolete objective after a pivot.
4. Tool/Backend	Underlying data and side-effect state.	• Schema-constrained. • Decoupled from UI rendering.	**Schema hallucination**: Affordances unsupported by backend data.

not resolve AG-UI failures. The problem is not insufficient recall, but an inappropriate representation of state: linear histories lack the primitives needed to track addressability, persistence, and concurrent modification.

5.4. The addressability problem

A defining challenge in AG-UI is *addressability*. Unlike chat, where references are linguistic ("the previous paragraph"), AG-UI actions must bind to specific, addressable interface elements: buttons, text fields, list items, or other UI components identified by unique handles.

Addressability failure. An addressability failure occurs when an agent generates an action targeting an interface element that no longer exists in the current UI state. The agent believes the element is present based on its internal model, but the element has since been deleted, renamed, or moved by the user or by prior agent actions. We can quantify this as the number of "stale references" (handles the agent believes exist but that are absent from the true interface), providing a task-agnostic signal of misalignment.

5.5. Temporal drift under asynchrony

In AG-UI, the relevant notion of correctness is not whether an action is well-formed at generation time, but whether it remains *valid at execution time* with respect to the current interaction state. We make this explicit by treating correctness as a state-dependent predicate: an action is correct only if its semantic preconditions still hold in the realized execution-time state. These preconditions include, but are not limited to, (i) **addressability** (the referenced UI handle exists and matches the intended target), (ii) **structural or schema constraints** (e.g., DOM, Abstract Syntax Tree (AST), or backend contracts), and (iii) **interaction constraints** implied by user actions such as deletions, overrides, or undo operations. Context alignment is therefore fundamentally about preserving *action validity under evolving state*, rather than maintaining internal belief consistency alone.

AG-UI violates the synchronous turn-taking assumption common in agent benchmarks. During the agent's inference delay (which may range from hundreds of milliseconds to several seconds), the user may update the interface: clicking, typing, deleting, or navigating. We call this *temporal drift*: the divergence between the true interface state at execution time and the agent's belief about what the interface looks like. Temporal drift quantifies how quickly validity conditions decay relative to interaction throughput. The faster the user acts, the more likely that an action planned under an old observation will fail when executed. An agent is *context aligned* if this drift remains bounded, that is, if the agent's beliefs about the interface stay close enough to reality that its actions remain valid at execution time. This reframes alignment as a dynamic stability guarantee for the coupled agent-interface control loop (Astrom & Murray, 2021).

Notably, these failure modes closely resemble concurrency and consistency problems studied in distributed systems and operating systems (Lamport, 1978; Mei et al., 2025), where actions computed under stale views of shared state lead to conflicts, race conditions, or rollbacks. Just as an OS must manage process isolation and shared memory synchronization to prevent corruption, AG-UI agents require explicit scheduling and context-coupling mechanisms to maintain coherence between fast interaction events and slow deliberative reasoning.

This perspective reinforces the need for structural alignment primitives rather than implicit prompt-based synchronization. These observations motivate a family of mechanisms whose purpose is not to improve recall or reasoning per se, but to preserve the *validity* of actions under asynchronous state evolution. Viewed through a systems lens, existing and proposed mitigation strategies for AG-UI inconsistency fall into four complementary classes: *pre-execution validation*, *transactional execution*, *post-execution repair*, and *representational consolidation*. These classes differ in when they

act in the agent–UI control loop and in the failure modes they mitigate, but all share a common objective: increasing the probability that an agent action remains supported by the true interaction state at execution time. Table 3 summarizes this design space.

6. Maintaining Context Alignment in AG-UI

Having characterized the sources of context misalignment in AG-UI, we now turn to the question of how it can be mitigated in practice.

6.1. Alignment as a Learning Objective

If context alignment is a learning problem, what is the objective? Standard objectives (e.g., maximum likelihood or standard RL) implicitly assume the conditioning context remains valid at execution time. In AG-UI, this assumption is violated: user actions asynchronously mutate the interface, invalidating the agent's training distribution. Maximizing the likelihood of an action given a history is insufficient if that history is a stale representation of the current interface. To address this, we propose that alignment requires a different objective: one that explicitly penalizes actions whose validity decays under asynchronous state mutation, akin to optimistic concurrency control in database systems (Kung & Robinson, 1981).

From an ML perspective, this connects to real-time decision-making under computation-induced delay, where the world can change while an action is being computed (Anokhin et al., 2025). Let a_t be the action at time t, s_t the state at inference, and $s_{t+\Delta}$ at execution. We propose: $\mathcal{L}(\theta) = \mathcal{L}_{\text{task}}(a_t, s_t) + \lambda \mathbb{E}_{s_{t+\Delta} \sim p(\cdot|s_t)}[1 - V(a_t, s_{t+\Delta})]$, where $\mathcal{L}_{\text{task}}$ is the standard objective, $V(a, s) \in \{0, 1\}$ indicates validity, and $p(\cdot|s_t)$ is the distribution over future states under user/environment dynamics. When $V{=}1$, penalty is zero; when $V{=}0$ (element deleted/modified), penalty is maximal. This specifies what alignment should achieve: actions robust to state mutation during Δ. Instantiation requires estimating user behavior or architectural safeguards. Current next-token objectives do not encode this validity constraint.

Optimization Challenges. Since $V(a, s_{t+\Delta})$ depends on unknown user dynamics, this frames context alignment as an RL challenge: maximizing validity reward over non-stationary trajectories.

6.2. Motivating Example: Micro-Drift in Collaborative Code Editing

To ground these ideas, consider a scenario in a collaborative code editor (e.g., VS Code). An agent observes a function `processData` at lines 10–15 and begins generating a docstring (inference takes ∼2.5s). Meanwhile, the user renames the function to `processDataAsync` and moves it. When the agent finishes and attempts to write to the original location, it fails: the target no longer exists. This is an **addressability failure**: the action targets a stale handle. From the user's perspective, the agent hallucinated. To prevent this, the agent must either (a) implement a *locking* mechanism that prevents concurrent edits or (b) re-verify the code structure immediately before execution, ensuring that the action is grounded in the current state rather than a memory of the past.

Table 3. **Consolidation Strategies for AG-UI.**

Class	Definition	Mitigation Target
Pre-execution validation	Verify intended action validity against current state before execution.	Addressability failures; stale assumptions.
Transactional execution	Apply mutations atomically, committing only if preconditions hold.	Partial application; race conditions; inconsistent states.
Post -execution repair	Detect and recover from inconsistencies after execution failure.	Irreversible corruption; cascading failures.
Structural consolidation	Represent state with stable identities and dependency-aware invalidation rules.	Zombie references; stale beliefs from linear histories.

6.3. Design Principles for Aligned AG-UI

The design principles below operationalize the consolidation strategy classes (Table 3) within the AG-UI setting, moving from abstract mechanisms to concrete architectural constraints. Current methods for iterative UI refinement demonstrate that agents can improve generation over multiple turns. However, existing approaches often lack mechanisms to maintain synchrony between agent state and runtime execution. We argue that reliable refinement requires deeper architectural coupling:

a) Explicit context handshakes (Strategy: Pre-execution validation). Agents should emit a "handshake" hash (e.g., locality-sensitive hash) encoding the target's semantic state—a *learned* fingerprint invariant to benign noise (layout shifts, ad loads) but sensitive to semantic changes (button deletion, field rename). This enables *optimistic concurrency control* (Kung & Robinson, 1981): the agent learns to predict drift probability $P(\text{drift})$ and decides whether to proceed optimistically or invoke costly locking. The middleware performs semantic verification before commit; mismatches trigger re-grounding rather than full restarts.

b) State-conditioned rollback (Strategy: Transactional

execution). Actions must be reversible. Given inevitable misalignment, environments must support snapshots and rollbacks, exposing the "undo" stack as a primitive. These principles require a *context coupling interface (CCI)*: a synchronization layer mediating handshakes between deliberative context and the interaction manifold. The CCI grounds slow, deliberative reasoning of the agent (System 2) in fast-evolving UI state (System 1), enabling the agent to verify assumptions before execution (Figure 1). Notably, benchmarks show that even top models achieve only $\sim$50% success on one-shot widget generation, yet improve to 74% with a single repair step (Anonymous, 2026), empirically validating the need for such explicit feedback loops.

c) Optimistic resolution (Strategy: Post-execution repair). Agents should learn recovery policies that repair misalignment without restarting. Since locking the UI is impractical, agents must employ optimistic concurrency control: upon drift detection (e.g., fingerprint mismatch), a specialized repair policy should map the original intent to the new state (e.g., searching for a renamed function by semantic similarity) rather than discarding all reasoning.

d) Graph-aware state tracking (Strategy: Structural consolidation). Context should be modeled as a dependency graph, not a linear buffer. Unlike unstructured text, a graph-aware context explicitly models ownership: if a parent node is deleted, all latent beliefs about its children are invalidated. This is analogous to garbage collection, preventing "zombie node" references where agents target deleted elements persisting in history.

7. Evaluating Context Alignment under Concurrency

The preceding sections argue that context alignment in AG-UI is fundamentally about preserving *action validity under asynchronous state evolution*. Despite this, most existing benchmarks implicitly assume serialized interaction: state changes occur *between* agent actions, not during agent deliberation. As a result, they cannot directly probe the failure modes introduced by concurrent user interaction.

SYNC: A Concurrent Interaction Evaluation Regime. To address this gap, we propose **SYNC** (Synchronization under Concurrency), an evaluation regime in which user actions are injected into the environment *while the agent is computing its next action*. Rather than assuming turn-based interaction, SYNC models true concurrency: the user may edit a document, delete an interface element, or modify backend state during the latency window of the agent's inference pass. An agent succeeds under SYNC only if it detects stale preconditions, aborts invalid actions, and re-plans based on the execution-time state. Unlike static benchmarks, SYNC exposes the temporal misalignment that defines AG-UI. Agents optimized for sequential environments would be expected to exhibit systematic degradation under concurrency, including precondition violations, action conflicts, and reduced task completion due to undetected drift. By construction, SYNC separates failures of reasoning or perception from failures of synchronization, clarifying what context alignment methods must ultimately achieve.

8. Alternative Views

We now address several credible alternative positions that arise naturally from existing perspectives in machine learning and agentic systems.

Is context alignment reducible to robustness or distribution shift? One might argue that the failures described in this paper are instances of robustness failures under non-stationarity. However, robustness typically concerns preserving performance under perturbations of input distributions, whereas context alignment concerns the validity of actions under evolving interaction state. In interactive settings, the environment changes *because of* agent and user actions, and the relevant notion of correctness is conditional on implicit constraints and commitments that emerge during interaction. These constraints are not perturbations but part of the task semantics itself. Robustness alone does not capture this structural coupling.

Is context alignment simply belief-state modeling under another name? The proposed framing is closely related to belief-state models such as Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1998), and this connection is intentional. However, classical belief-state formulations assume a fixed latent world state and a stationary transition model. In contrast, interactive agentic systems involve latent variables (such as user intent, task phase, and implicit commitments) that are shaped by interaction dynamics and may change as a result of agent actions. Moreover, the observation space in these systems includes semantically rich interaction events (e.g., edits, overrides, undo operations) that are not naturally modeled as noisy sensory inputs. Context alignment extends belief-state modeling by explicitly accounting for the asymmetric update rates and semantic mismatch between interaction context and deliberative abstraction.

Can end-to-end RL learn alignment implicitly? While theoretically possible, relying on end-to-end learning obscures the locus of alignment and complicates generalization (Liu et al., 2023b; Zhou et al., 2024). Explicit modeling provides structured inductive bias, enabling agents to handle surface-form variations with shared interaction semantics. Alignment is thus a specification of *what* must be learned, not an alternative to learning.

Is this primarily an HCI problem? Although motivated by

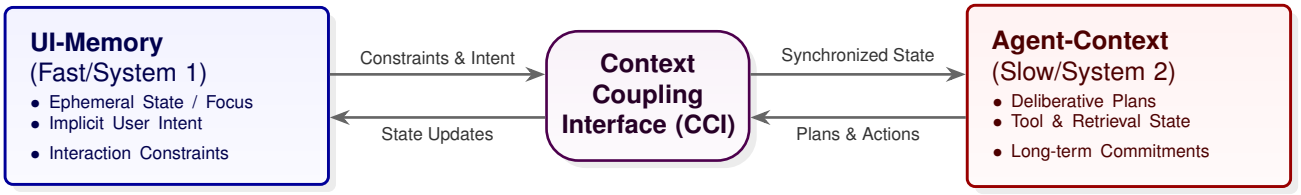

Figure 1. **Co-evolving context.** Two coupled processes: **UI-Memory** (Fast/System 1) and **Agent-Context** (Slow/System 2). System 1/2 denotes speed mismatch, not cognitive modes. The CCI mediates propagation of intent, constraints, and actions.

HCI, alignment is a fundamental *learning* problem concerning state updates, partial observability, and non-stationary dynamics. Metrics like action validity and rollback consistency define learning objectives independent of interface design. Alignment sits at the intersection of control, learning, and interaction.

Can better prompting or tool design solve this? One might argue that instructing agents to "always verify state before acting" suffices without architectural changes. However, this approach requires the agent to know *what* to verify and *when*, which itself requires modeling the non-stationary dynamics of interaction. Empirically, prompt-based verification degrades under increased interaction throughput: as users act faster, the frequency of required re-verification exceeds reasonable prompting overhead. Moreover, identifying which handles to verify requires precisely the structural state tracking we argue must be learned. Prompting can mitigate symptoms but does not address the underlying representation gap.

Can we simply reduce agent latency? A related objection is that as model inference becomes faster, drift approaches zero, rendering alignment unnecessary. While tempting, this argument fails on two counts. First, network latency and rendering overhead impose a physical floor on inference delay ($\geq$100ms), long enough for a user to click, type, or change focus (Firoiu et al., 2018). Second, thinking time (System 2 reasoning) is a feature, not a bug; we *want* agents to deliberate for seconds or minutes on complex tasks. Reliability cannot depend on racing the user; it must rely on architectural guarantees that allow slow reasoning to safely act on fast-moving state.

8.1. Limitations and Scope

This position paper focuses on *interactive, user-facing agentic systems* where humans and agents share concurrent control of addressable interface state. We do not propose a single algorithmic solution; our contribution is *problem formulation*: treating alignment between deliberative context and interaction state as an explicit learning objective, distinct from memory or retrieval. While GenUI makes these failures especially salient, the core challenge—preserving action validity under asynchronous state mutation—also arises in simpler interfaces (e.g., shared terminals or device controllers). Future work should test whether the proposed design principles yield measurable gains.

9. Call to Action

- **ML Researchers: Representation Learning.** Learning how interface state evolves requires compact embeddings for interaction micro-states (e.g., hesitation) that predict future user behavior, internalizing a physics of interaction to bound belief uncertainty.
- **HCI Researchers: Latent Intent Inference.** Mechanisms are needed to disentangle explicit instructions from implicit intent, suppressing actions inconsistent with predicted user trajectories.
- **Systems Developers: Adaptive Synchronization.** Agents must learn policies balancing validity against latency, defaulting to pessimistic locking in high-contention interfaces and optimistic execution otherwise.
- **Safety Communities: Structural Verification.** Interface structure constraints provide deterministic safety guarantees. Verifying action support against DOM structure prior to execution shields against unsafe extrapolations.
- **Cognitive Architectures & Benchmarking.** Shifting to "context-as-process" requires primitives like interruptible reasoning and state-conditioned rollback. Benchmarks should measure drift and intervention rates using environments exposing ground-truth for counterfactual analysis.
- **Foundation Model Pre-training.** Pre-training on *interaction traces* (state-action-outcome trajectories) enables models to internalize physics of UI mutation as a prior.

10. Conclusion

We have argued that failures in agentic AI often stem from *context alignment* errors (addressability failures and temporal drift) rather than insufficient reasoning. By treating the user interface as a co-evolving control variable and distinguishing interaction context from deliberative context, we can move beyond the "context-as-buffer" paradigm. Addressing this alignment gap is foundational; future architectures must treat the UI as a concurrent process, ensuring that long-horizon reasoning remains synchronized with the fast-moving reality of user interaction.

References

Agashe, S., Jimenez, C. E., Fan, B., Yu, H., Yao, S., and Yang, D. Agent s: An open agentic framework that uses computers like a human. In *International Conference on Learning Representations (ICLR)*, 2025. Introduces Agent-Computer Interface (ACI) and hierarchical planning for GUI agents.

Amershi, S., Weld, D. S., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S. T., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., and Horvitz, E. Guidelines for human–ai interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2019.

Anokhin, I., Rishav, R., Riemer, M., Chung, S., Rish, I., and Kahou, S. E. Handling delay in real-time reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=YOc5t8PHf2.

Anonymous. Widgeteval: Benchmarking foundation models on dynamic widget generation for apps. Submitted to ICLR 2026, 2026. URL https://openreview.net/forum?id=4Ah9yNUAJL. OpenReview Submission 9892.

Anonymous Authors. Gui-arp: Enhancing grounding with adaptive region perception for gui agents. ICLR 2026 Submission, 2026a. Dynamic region perception for fine-grained grounding.

Anonymous Authors. Longhorizonui: A unified framework for robust long-horizon task automation of gui agent. ICLR 2026 Submission, 2026b. Targets cumulative error and contextual drift in MLLM-based agents.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Anthropic Technical Report*, 2024.

Anthropic. Model context protocol: Standardizing assistant connectivity. https://www.anthropic.com/news/model-context-protocol, 2024. Accessed: 2025-05-15.

Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Le Hors, A., Nicol, G., Robie, J., Sutor, R., Wilson, C., and Wood, L. Document object model (dom) level 1 specification. W3c recommendation, W3C, 1998. Version 1.0.

Astrom, K. J. and Murray, R. M. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

Cai, Z., Hua, W., Li, K., Ma, Y., Nie, E., Schuetze, H., Stanczak, K., and Taylor, M. E. Memagents: Memory for llm-based agentic systems. ICLR 2026 Workshop, 2026. Distinguishes static LLM memory from dynamic, interaction-driven agent memory.

Cao, Y., Jiang, P., and Xia, H. Generative and malleable user interfaces with generative and evolving task-driven data model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*, 2025. URL https://arxiv.org/abs/2503.04084.

Chen, J., Chen, L., Wang, D., Su, Q., Chu, Z., Gan, L., Zhuang, C., and Gu, J. V2p: Visual attention calibration for gui grounding via background suppression and center peaking. ICLR 2026 Submission, 2026. Discusses attention drift in GUI grounding and mislocalization of small elements.

Cheng, R., Barik, T., Leung, A., and Nichols, J. Biscuit: Scaffolding LLM-generated code with ephemeral UIs in notebooks. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*, 2024. doi: 10.1145/3654777.3676375. Discusses generative UI scaffolds involving executable code.

Chi, E. H., Pirolli, P., Chen, K., and Pitkow, J. Modeling user intent in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2001.

Cloudflare. Ai agents: Code mode. https://blog.cloudflare.com/code-mode/, 2024. Accessed: 2025-05-15.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, 2023.

Firoiu, V., Ju, T., and Tenenbaum, J. At human speed: Deep reinforcement learning with action delay. *arXiv preprint arXiv:1810.07286*, 2018.

Gao, Z., Zhang, B., Li, P., Ma, X., Fan, Y., Yuan, T., Wu, Y., Jia, Y., Zhu, S.-C., and Li, Q. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage. In *International Conference on Learning Representations (ICLR)*, 2025. Spotlight.

Gemini Team, Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Gou, B., Kai, W., Dong, Y., Wang, H., Liu, Y., Li, Y., and Tang, G. Uground: Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations (ICLR)*, 2025. Oral; addresses universal visual grounding for GUI element localization.

Hattami, A. E., Thakkar, M., Chapados, N., and Pal, C. Webarena verified: Reliable evaluation for web agents. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=94tlGxmqkN.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024a.

Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., and Tang, J. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*, 2024b.

Horvitz, E. Principles of mixed-initiative user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1999.

Hu, Y., Liu, S., Yue, Y., Zhang, G., Liu, B., Zhu, F., Lin, J., Guo, H., Dou, S., Xi, Z., Chen, J., Wang, X., Cao, R., Zhang, L., Li, W., Wang, Z., Wang, P., and Zhang, Q. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2024.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. *Planning and Acting in Partially Observable Stochastic Domains*. MIT Press, 1998.

Kahneman, D. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L., and Murthy, A. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Kirmani, S. Y., Rajeswaran, A., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Kung, H.-T. and Robinson, J. T. On optimistic methods for concurrency control. *ACM Transactions on Database Systems (TODS)*, 6(2):213–226, 1981.

Lamport, L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7): 558–565, 1978.

Lee, K.-H., Chen, X., Furuta, H., Canny, J. F., and Fischer, I. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning (ICML)*, 2024.

Leviathan, Y., Valevski, D., Kalman, M., Lumen, D., Segalis, E., Molad, E., Pasternak, S., Natchu, V., Nygaard, V., Venkatachary, S., Manyika, J., and Matias, Y. Generative ui: Llms are effective ui generators. Technical report, Google Research, November 2025. URL https://generativeui.github.io/.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Li, Z.-Z., Zhang, D., Zhang, M.-L., Zhang, J., Liu, Z., Yao, Y., Xu, H., Zheng, J., Wang, P.-J., Chen, X., Zhang, Y., Yin, F., Dong, J., Li, Z., Bi, B.-L., Mei, L.-R., Fang, J., Liang, X., Guo, Z., Song, L., and Liu, C.-L. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

Lin, K. Q., Li, L., Gao, D., Yang, Z., Wu, S., Bai, Z., Lei, W., Wang, L., and Shou, M. Z. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., and Tang, J. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023b.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Chen, P., Lu, S., Honovich, O., and Singh, A. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Mei, K., Zhu, X., Xu, W., Jin, M., Hua, W., Li, Z., Xu, S., Ye, R., Ge, Y., and Zhang, Y. AIOS: LLM agent operating system. In *Conference on Language Modeling (COLM)*, 2025.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Pan, Y., Kong, D., Zhou, S., Cui, C., Leng, Y., Jiang, B., Liu, H., Shang, Y., Zhou, S., Wu, T., and Wu, Z. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Press, O., Smith, N. A., and Lewis, M. Do long-context language models really understand long contexts? In *International Conference on Machine Learning*, 2024.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., and Cancedda, N. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Shinn, N., Labash, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning. *arXiv preprint arXiv:2206.10498*, 2023.

Wang, Y. and Lu, Y. Agentic media: Reimagining the future of communication. Technical Report MSR-TR-2025-62, Microsoft, October 2025. URL https://www.microsoft.com/en-us/research/publication/agentic-media/.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Jiang, L., Zhang, X., Liu, J., Awadallah, A., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Wu, Q., Cheng, K., Yang, R., et al. Gui-actor: Coordinate-free visual grounding for gui agents. ICLR 2026 Submission, 2026. Attention-based coordinate-free grounding head.

Xie, T., Deng, J., Li, X., Yang, J., Wu, H., Chen, J., Hu, W., Wang, X., Xu, Y., Wang, Z., Xu, Y., Wang, J., Sahoo, D., Yu, T., and Xiong, C. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025. NeurIPS 2025 Datasets and Benchmarks Track (Spotlight).

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *Advances in Neural Information Processing Systems*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Zafar, H., Ur Rehman Khan, S., Mashkoor, A., and Nisa, H. U. Mobicat: a model-driven engineering approach for automatic gui code generation for android applications. *Frontiers in Computer Science*, 6, 2024. doi: 10.3389/fcomp.2024.1397805.

Zhang, C., He, S., Qian, J., Li, B., Li, L., Qin, S., Kang, Y., Ma, M., Liu, G., Lin, Q., Rajmohan, S., Zhang, D., and Zhang, Q. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024a.

Zhang, C., Yang, Z., Liu, J., Han, Y., Chen, X., Huang, Z., Fu, B., and Yu, G. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2024b.

Zhang, Y., Sun, R., Chen, Y., Pfister, T., Zhang, R., and Arik, S. Ö. Chain of agents: Large language models collaborating on long-context tasks. *arXiv preprint arXiv:2406.02818*, 2024c. NeurIPS 2024.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.