BEYOND ORACLE COMMUNICATION: BENCHMARKING INTERACTIVE INTENT ALIGNMENT UNDER MISCOMMUNICATION AND EVOLVING USER INTENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LLM agents increasingly tackle complex tasks through interactive, long-horizon exchanges with users, while existing benchmarks generally assume that users always accurately and sufficiently communicate a fixed intent. However, this *oracle communication assumption* rarely holds in practice: users may miscommunicate, change their goals, and run out of patience. We define this task setting as **interactive intent alignment**, where agents must recover and continuously track the user's current intent despite imperfect communication and evolving goals. To study this setting, we introduce **Drift-Bench++**, a principled benchmark construction pipeline for verified executable tasks with controlled misalignment and intent shifts, along with an interaction protocol featuring finite patience, diverse simulated users, and silent interaction-conditioned shifts. We further develop **GRIP**, a comprehensive evaluation protocol covering task grounding, user realism, inquiry effectiveness, and adaptation to evolving intent. Across diverse environments, models, and interaction conditions, stronger interaction consistently helps but remains far from oracle performance; Validation on deployed PRODAGENT sessions further shows that the modeled failures are prevalent and consequential in deployment. By providing a unified, executable benchmark for interactive intent alignment, Drift-Bench++ offers a foundation for evaluating and advancing agents under realistic communication and evolving intent.

Website

1 INTRODUCTION

Modern LLM agents have made rapid progress on increasingly complex tasks and are increasingly evaluated in interactive, long-horizon settings (Barres et al., 2025; Qian et al., 2025c; Kong et al., 2026; Yu et al., 2026). Yet most current benchmarks implicitly adopt an *Oracle Communication Assumption* (Bao et al., 2026): throughout the interaction, user messages are assumed to always accurately and sufficiently convey the user's underlying intent. In practice, however, this assumption rarely holds: users may omit details, rely on incorrect assumptions, or communicate ambiguously (Ji et al., 2026; Chen et al., 2026b); moreover, their intent may evolve without being fully communicated (El Alaoui et al., 2025; Mitra et al., 2026; Ghose et al., 2026). As a result, benchmark success can obscure whether an agent can detect misalignment, seek clarification, and remain synchronized with the user. These capabilities directly shape real-world reliability and user experience.

Recent work has begun relaxing the *Oracle Communication Assumption* through studies of *intent misalignment* and *intent shifting* (Pu et al., 2026; Zou et al., 2026). **However, three gaps remain:** (1) The two challenges are largely studied in isolation, despite a partially communicated intent shift immediately recreating misalignment. (2) Real users do not have unlimited patience, yet clarification is rarely constrained by a finite interaction budget, leaving the trade-off between information gathering and timely action underexplored. (3) User intent may evolve during the conversation itself, including when an agent's question prompts the user to reconsider or recognize a new preference, as illustrated in Figure 1(a); yet existing shifts are still largely predefined by schedules or task outcomes and explicitly announced, effectively reinstating oracle communication after each change. Together, these limitations expose a broader gap in current evaluation: existing benchmarks lack a unified setting for studying how agents remain aligned with users throughout realistic interaction. **We define this setting as Interactive Intent Alignment**: Given a latent user intent that may be miscommunicated and may evolve without being fully communicated, an agent must keep its actions aligned with the user's current intent through interaction with users exhibiting diverse communication and decision-making behaviors under finite patience. Figure 1 illustrates such an episode, where success

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

requires not merely executing a fixed instruction, but continuously recovering and tracking what the user currently wants. To fully study this task setting, we introduce **Drift-Bench++**, a benchmark construction pipeline for creating interactive intent alignment tasks. Specifically, grounded in established theories of human communication (Grice, 1975; Austin, 1975; Watzlawick et al., 2011), we first construct executable intent graphs and verified misaligned requests through a three-step verification process, providing controlled ground truth for both initial misalignment and subsequent intent changes. Then, to capture how alignment unfolds with realistic users rather than idealized interfaces, we introduce an interaction protocol with finite patience, silent interaction-conditioned shifts, and GDMS-based (Scott & Bruce, 1995) persona-conditioned user simulation, requiring agents to balance clarification, timely action, and adaptation to evolving goals. Finally, because task success alone cannot capture the many facets of interactive alignment, we introduce **GRIP**, a comprehensive evaluation protocol covering Grounding in the task outcomes, Role-realism of simulated users, Inquiry quality and efficiency, and Persistence of alignment under intent shifts.

To assess interactive intent alignment as a meaningful and generalizable agent capability, our experiments ask three questions: (1) how well current agents recover and track intent under miscommunication and intent shifts; (2) how robust this capability remains across models and interaction conditions; and (3) whether our benchmark abstractions reflect real-world interaction. We evaluate Drift-Bench++ across agentic environments, SOTA clarification baselines, models, and diverse interaction conditions, then validate its abstractions on PRODAGENT, a deployed anonymized OpenClaw-like agent platform under company privacy policy. Results show that stronger interaction consistently helps yet remains far from oracle performance, while the production data confirm that the modeled communication failures and intent dynamics are prevalent and consequential in deployed agents. These findings position interactive intent alignment as an advanced capability with substantial room for progress in agent design and evaluation. We summarize our contributions as follows:

- We formulate **Interactive Intent Alignment** as a task setting that jointly models communication misalignment and evolving intent, and develop **Drift-Bench++**, a principled construction pipeline for generating verified executable tasks.
- We design the interaction protocol with finite patience, diverse simulated users, and silent interaction-conditioned shifts, along with the **GRIP** protocol for comprehensive evaluation.
- We conduct extensive experiments across environments, models, and interaction conditions, together with real-world validation on PRODAGENT production sessions, revealing substantial limitations of current agents and validating the benchmark design.

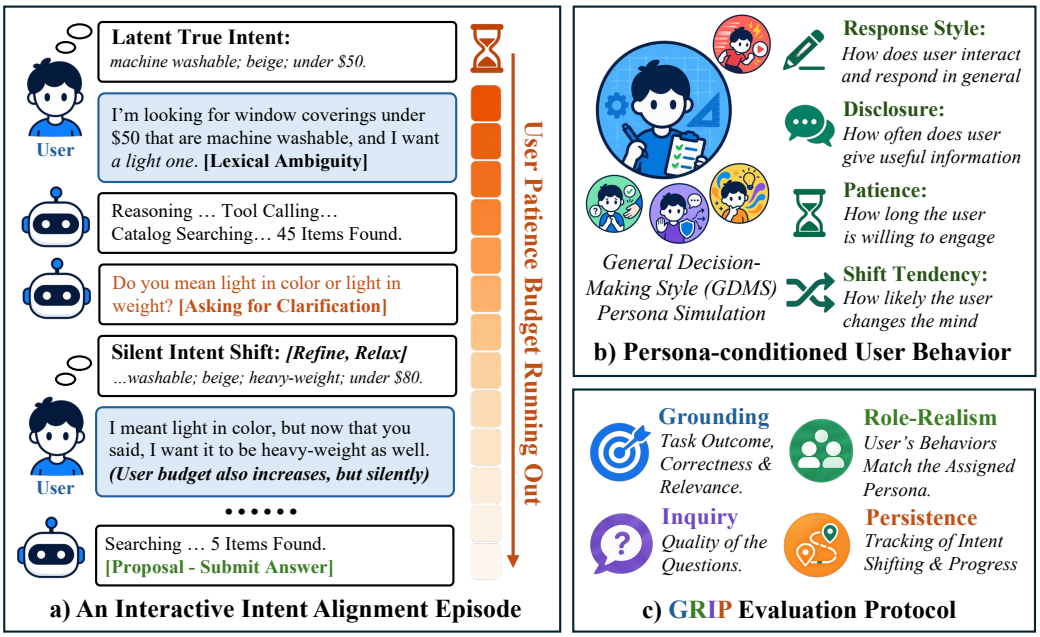

Figure 1: **Overview of Drift-Bench++.** a) A user query may misalign with latent intent, and the agent's response may trigger a silent intent shift. b) Persona-conditioned behavior shapes the interaction beyond prompting style, and c) GRIP evaluates the resulting alignment process.

Benchmarks	No Oracle Comm.	Intent Shifting	Shift Trigger	Patience-bounded	Persona-guided	Beyond Success Eval.
Intent Misalignment Benchmarks						
ChatShop (Chen et al., 2024)	✓	✗	✗	✗	✗	✗
UserBench (Qian et al., 2025a)	✓	✗	✗	✗	✗	✗
UserVille (Sun et al., 2026)	✓	✗	✗	✗	✓	✗
LHAW (Pu et al., 2026)	✓	✗	✗	✗	✗	✗
UnderSpecBench (Ji et al., 2026)	✓	✗	✗	✗	✗	✗
AppWorld-UL (Chen et al., 2026b)	✓	✗	✗	✗	✗	✗
RegretBench (Ta et al., 2026)	✓	✗	✗	✗	✓	I.R.
Drift-Bench (Bao et al., 2026)	✓	✗	✗	✗	✓	I.R., U.R.
Intent Shifting Benchmarks						
AgentChangeBench (Rana et al.,	✗	✓	Outcome	✗	✓	S.T.
Trajectory2Task (Wang et al., 2026	✗	✓	Schedule	✗	✗	✗
InterruptBench (Zou et al., 2026)	✗	✓	Schedule	✗	✗	S.T.
Trip+ (Chen et al., 2026a)	✗	✓	Schedule	✗	✗	✗
HAS-Bench (Wu et al., 2026)	✗	✓	Schedule	✗	✓	U.R., S.T.
EHR-ChatQA (Lee et al., 2026)	✗	✓	Outcome	✗	✗	✗
Evolving Intent (Tack et al., 2026)	✗	✓	Schedule	✗	✗	✗
IACM-RL (Zhu et al., 2026)	✗	✓	Schedule	✗	✗	S.T.
Drift-Bench++ (Ours)	✓	✓	**Interaction**	✓	✓	**I.R., U.R., S.T.**

I.R.: Intent Recovery Evaluation; **U.R.**: User Realism Evaluation; **S.T.**: Shift-Tracking Evaluation; **Comm.**: Communication.

Table 1: Comparison of representative intent misalignment and shifting benchmarks. **No Oracle Comm.** indicates if simulated users may violate the *Oracle Communication Assumption* at any turn; **Patience-bounded** requires patience to constrain the interaction itself, rather than serving only as a cost or evaluation signal; **Persona-guided** captures broader persona-conditioned interaction dynamics; and **Beyond Success Eval.** reports which aspects are evaluated beyond final task success.

2 Existing Challenges: Why Drift-Bench++?

Intent Misalignment Challenge. To address this challenge, early efforts enabled LLMs to query simulated users for clarification (Qian et al., 2024; Gan et al., 2024; Li et al., 2024). As LLMs evolved into tool-using agents, this paradigm extended to agentic settings where misunderstanding could directly affect subsequent actions and task outcomes (Chen et al., 2024; Qian et al., 2025a; Sun et al., 2026). Together, these works establish interaction as a key mechanism for resolving intent misalignment. However, they typically model misalignment through narrow or heuristic forms, such as missing information or generic vagueness, without clearly distinguishing failure modes.

To bridge this gap, Drift-Bench (Bao et al., 2026) pioneers a structured approach to controlled intent perturbation. Grounded in communication theory, it couples an extensible and unified taxonomy of intention, premise, parameter, and expression faults with persona-conditioned simulation. Beyond organizing misalignment taxonomy, this design provides a reusable pipeline for systematically synthesizing misaligned agent tasks. Subsequent work strengthens complementary aspects of this direction, particularly perturbation controllability and verification. For example, LHAW (Pu et al., 2026) validates underspecified variants through behavioral effects, while UnderSpecBench (Ji et al., 2026) and AppWorld-UL (Chen et al., 2026b) introduce more explicit construction and verification. **Despite these advances, two aspects of real-world interaction remain underexplored:** First, users have finite patience, forcing agents to trade off further clarification against timely action, while existing benchmarks typically model this only as free interaction or a scoring cost; Second, current benchmarks treat user intent as static, whereas in practice goals may evolve during an interaction.

Intent Shifting Challenge. Separately, another line of work studies the challenge of intent shifting, where the user's underlying goal changes during interaction rather than being inadequately communicated (El Alaoui et al., 2025; Mitra et al., 2026; Ghose et al., 2026). Existing work largely realizes such shifts through predefined mechanisms. *Schedule-triggered* approaches predefine goal changes and introduce them at predetermined stages of the interaction (Wang et al., 2026c; Zou et al., 2026; Chen et al., 2026a; Wu et al., 2026; Tack et al., 2026; Zhu et al., 2026), while *outcome-triggered* approaches condition predefined shifts on task progress: AgentChangeBench (Rana et al., 2025) advances through a goal sequence as the current goal is fulfilled, while EHR-ChatQA (Lee et al., 2026) revises the user's goal in response to task failures. **Despite increasingly rich shift formula-**

tions, current efforts share two drawbacks: First, both mechanisms still largely prescribe when and how intent changes occur rather than allowing them to emerge from the user-agent interaction. More importantly, we argue intent shifting and intent misalignment are coupled challenges in real-life. Second, all prior works explicitly communicated shifts to the agent, effectively reinstating the *oracle communication assumption* after each change. In practice, a user may revise their intent while only partially expressing the update if not explicitly asked, leaving the agent's understanding silently stale. To summarize and clarify our positioning, Table 1 consolidates these distinctions and other key aspects across representative intent-misalignment and intent-shifting benchmarks. For even more detailed comparison, please see Appendix A.

3 DRIFT-BENCH++ BENCHMARK

In this section, we introduce Drift-Bench++, which unifies the challenges in **Interactive Intent Alignment** setting where intents can be both misaligned and evolving, and must be recovered and tracked under constrained interaction. To make this problem controlled and executable, we first construct intent graphs with verified query perturbations, providing executable ground truth for both initial misalignment and subsequent shifts. Figure 2 summarizes the Drift-Bench++ data construction pipeline, from host benchmark tasks to verified benchmark instances. Then, we introduce patience-bounded interaction, interaction-conditioned intent shifting, and persona-conditioned behavior to model how user-agent interaction shapes information recovery and intent evolution. Finally, we propose **GRIP**, an extensible, structured evaluation protocol covering task outcomes, simulated-user fidelity, inquiry effectiveness, and adaptation to changing intents.

3.1 DATA CONSTRUCTION

Intent Graph Construction. To properly extract and model intent shifting, we first map each original task x to a structured *intent* I_x via a lightweight benchmark-specific adapter $\mathcal{A}$. Formally,

$$I_x = \mathcal{A}(x) = (f, C), \qquad C = \{c_j\}_{j=1}^{|C|}, \tag{1}$$

where f identifies the task frame and C is the set of conditions that together specify what the user intends within that frame. These conditions are extracted from the benchmark's own task-defining fields, such as product attributes and price constraints in WebShop, or arguments of executable write actions in τ^2. We provide our adapter interface and detailed extraction rules in Appendix B.1. Once mapped to (f, C), benchmark-specific details are isolated behind the lightweight adapter and host-native execution, while the remaining intent-level construction is shared across benchmarks. To support varied intent shifts, we first retrieve the closest validated tasks from the same environment whose conditions satisfy one of the four relations in Figure 2, forming a relation-labeled intent tree. We then re-examine all candidate intents and pairs for two purposes: First, we reclassify all pairs and add valid cross-links, yielding an intent graph that supports controlled yet diverse shifts. Second, we execute the task and retain only candidates that pass the host benchmark's legality and execution checks, ensuring executable ground truth for every intent and a well-defined transition for every edge. The resulting intent graphs provide a validated foundation for realizing intent shifts during interaction, as described in Section 3.2; detailed construction and filtering rules are in Appendix B.2.

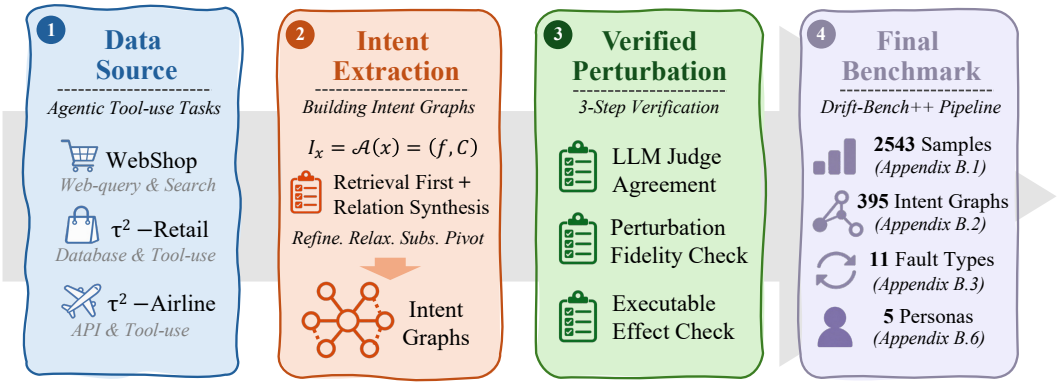

Figure 2: **Overview of the Drift-Bench++ data synthesis pipeline.** Host tasks are mapped into intent graphs, and converted into verified misaligned instances. Lightweight benchmark-specific adapters and native oracles allow the remaining construction to be shared across domains.

Misalignment Perturbation with Verification. To further evaluate recovery under diverse forms of intent misalignment, we expand each root intent I_0 into multiple perturbed instances, each retaining the same graph for later shifts. We base this expansion on the *communication-fault taxonomy* introduced by Drift-Bench (Bao et al., 2026). Specifically, drawing on established theories of human communication (Grice, 1975; Austin, 1975; Watzlawick et al., 2011), the taxonomy organizes misalignment into four categories: *intention*, *premise*, *parameter*, and *expression*, comprising 11 concrete fault types that cover major ways a user query may diverge from the underlying intent (detailed taxonomy and fault definitions are provided in Appendix B.3).

To realize these perturbations, Drift-Bench prompts an LLM with a fault-specific instruction to generate a query exhibiting the target fault. However, following the instruction does not guarantee that the generated query actually induces the intended semantic change. To make the resulting perturbations verifiable rather than relying on the requested fault alone, we leverage the structured intent conditions to build a three-step verifier. **First, LLM Judge Agreement.** Two independent LLM judges are given the generated query u together with the true root intent I_0, and asked to recover the concrete change expressed in the query as a structured edit δ, such as removing a requirement, or replacing its value. We retain the instance only when both judges recover the same edit, formally:

$$\delta = E_1(u, I_0) = E_2(u, I_0). \tag{2}$$

Second, Perturbation Fidelity Check. We further mechanically check whether the query itself faithfully realizes the recovered edit. For example, values that should remain or be substituted must appear in the text, whereas values intended to be withheld or left unbound must not be revealed. **Third, Executable Effect Verification.** We apply the agreed edit δ, which captures how the perturbed query Q changes the known conditions of I_0, to reconstruct the intent expressed by Q: $I_Q = \mathrm{Apply}(I_0, \delta)$. We then execute both the true intent I_0 and the query-expressed intent I_Q using the host benchmark's native oracle:

$$Y_0 = \mathcal{O}(I_0), \qquad Y_Q = \mathcal{O}(I_Q), \tag{3}$$

where $\mathcal{O}$ denotes benchmark-native execution and Y_0 and Y_Q are the resulting ground-truth outcomes. For example, withholding a requirement should widen the valid solution set or otherwise change the outcome, factual substitution should change the outcome, and noise or oblique phrasing should leave the outcome unchanged. We retain the perturbation only when the relation between Y_0 and Y_Q matches the expected effect of fault type k. Detailed extraction rules, surface-fidelity checks, and fault-specific admission criteria are provided in Appendix B.3.

Together, the intent-graph and verified-perturbation pipelines transform each original task into an instance with a verified initial mismatch between the user's query and underlying intent, while preserving executable alternative intents for subsequent shifts during interaction.

3.2 Interaction Protocol

Real-world interaction introduces two additional challenges: users have finite patience, and their intent may shift dynamically during the conversation, sometimes in response to the agent's own questions. In this section, we model these dynamics through a patience budget and a silent intent-shifting mechanism, and further diversify them through persona-conditioned user behavior.

Patience-Bounded Interaction. To model finite user patience and force agents to trade off asking against timely action, we maintain a patience budget b_t that decreases with user-facing actions:

$$b_{t+1} = b_t - c(a_t), \tag{4}$$

where $c(a_t)$ denotes the patience cost of agent action a_t. At each turn, in addition to regular tool use, the agent may either ask the user for clarification or propose a solution. Clarification consumes patience, while a rejected proposal incurs a larger cost to discourage premature commitment. Unlike prior approaches treating interaction cost as evaluation penalties, our patience budget constrains the interaction itself: unaffordable user-facing actions become unavailable, forcing the agent to commit with its current knowledge. Exact rules and affordability checks are provided in Appendix B.4.

Interaction-Conditioned Intent Shifting. To capture intent changes that arise dynamically during interaction rather than being fully predetermined and explicitly announced, we realize shifts through two complementary channels. **First, patience-triggered shifts** become eligible when the remaining

patience crosses predefined thresholds. **Second, agent-triggered shifts** allow the agent's own response to prompt reconsideration: when a question touches on the topic of a feasible change in the intent graph, it may trigger a change along that topic, while the specific attributes are selected from valid alternatives in the intent graph. We summarize this transition as

$$I_{t+1} \sim P(\cdot \mid I_t, b_t, q_t)\,, \tag{5}$$

where q_t denotes the agent's question when present. The transition is probabilistic, so an eligible opportunity need not produce a shift. Both channels share the same finite pool of shift opportunities: an agent-triggered shift consumes an existing opportunity rather than creating a new one, preventing agents that ask more questions from being systematically exposed to more shifts.

Importantly, unlike prior intent-shifting benchmarks, our shifts are *silent*: the user does not announce the updated intent in full, and the agent must recover it through subsequent interaction. This extends the same non-oracle communication setting of initial misalignment to intent shifts. Detailed scheduling, triggering, and transition rules are provided in Appendix B.5.

Persona-Conditioned Behavior. To further diversify interaction behavior, we adapt the *General Decision-Making Style* (GDMS) framework of Scott and Bruce (Scott & Bruce, 1995) into five user personas: *Rational*, *Intuitive*, *Dependent*, *Avoidant*, and *Spontaneous*. Rather than using personas only to vary response style, we use them to modulate multiple aspects of the interaction protocol. For example, persona parameters affect disclosure and feedback behavior; scale the initial patience budget as $b_0 = B\, m_\rho$; and influence shift timing and susceptibility to agent-induced reconsideration. In this way, the same underlying task can induce meaningfully different interaction dynamics across personas. Detailed persona configurations and mechanics are provided in Appendix B.6.

Importantly, these behaviors are enforced by the user state machine rather than decided freely by the language model. The LLM is restricted to identifying which authorized requirement a question concerns and verbalizing information selected by the state machine, keeping persona effects behavioral rather than altering task truth. By grounding these behaviors in the well-established GDMS decision styles, we aim to create controlled yet behaviorally plausible variation in how users disclose information, tolerate interaction, and reconsider their goals. We further examine whether the resulting interactions exhibit the intended persona-specific behavioral distinctions in our persona analysis.

3.3 GRIP Evaluation

With misaligned requests, finite patience, interactive clarification, and dynamically shifting intents, a single task-success rate can no longer adequately characterize agent behavior. We therefore introduce **GRIP**, an extensible diagnostic evaluation protocol for interactive intent alignment tasks, organized around four complementary dimensions: Grounding for task outcomes, Role-realism for simulated-user fidelity, Inquiry for intent recovery under interaction costs, and Persistence for adaptation to evolving intents. By separating these dimensions, GRIP provides a common evaluation foundation for this class of tasks, while leaving each dimension open to additional metrics as new interaction challenges emerge. Due to space constraints, we summarize the metrics below and defer their exact definitions and computation to Appendix B.7.

To provide a high-level overview: **Grounding (G)** evaluates whether success is grounded in the user's true intent. *Success* measures completion under the executable verifier, while *Earned* and *Inferred* distinguish success with versus without fully eliciting the hidden intent. **Role-realism (R)** evaluates whether simulated personas exhibit distinct and stable behaviors: *Identification* measures whether judges recognize the intended persona, and *Consistency* whether interactions from the same persona remain behaviorally coherent across tasks. **Inquiry (I)** evaluates clarification through *Aim*, whether questions target unresolved hidden requirements; *Recovery*, how much hidden intent is elicited; and *Patience*, how much interaction budget remains, jointly capturing the precision, coverage, and cost of inquiry. Finally, **Persistence (P)** evaluates adaptation to evolving intent. *Staleness* captures failures that remain aligned with a superseded intent, *Reaction* measures adaptation speed after a shift among successful episodes, and *Post-shift Success* measures the unconditional probability of succeeding after an intent change.

Importantly, GRIP reports these dimensions separately rather than collapsing them into a single score. In this way, it provides a diagnostic foundation for evaluating not only whether an interactive agent succeeds, but also whether the simulated interaction is well-formed, how effectively the agent resolves hidden intent, and how robustly it follows that intent as it changes.

Method	Grounding (G)			Inquiry (I)			Persistence (P)		
	Success	Earned	Inferred	Aim	Recovery	Patience	Staleness	Reaction	PostShift
No Ask	52.54±0.82	n/a	n/a	n/a	n/a	4.49±0.06	9.69±0.34	**4.23±0.33**	45.93±0.58
Just Ask (Li et al., 2025)	54.80±0.69	19.23±1.03	31.38±0.92	35.74±0.52	33.93±1.61	2.35±0.11	8.43±0.75	8.62±0.44	48.82±0.74
AT-CoT (Tang et al., 2025)	54.49±0.58	18.88±0.97	31.69±0.85	38.92±0.55	32.22±1.59	2.61±0.03	8.34±0.51	9.84±0.30	49.31±0.95
Belief Graph (Hahn et al., 2025)	45.60±0.89	5.33±0.28	**35.51±1.36**	29.17±1.05	14.21±0.79	2.89±0.08	**4.43±0.75**	4.75±0.29	40.82±0.64
ProductAgent (Ye et al., 2025)	53.22±0.78	14.76±0.30	34.59±1.07	32.11±1.33	28.78±0.58	2.21±0.05	7.33±0.48	7.74±0.92	47.41±0.69
SAGE (Suri et al., 2026)	54.77±0.64	23.50±1.45	27.85±0.94	31.20±0.88	42.60±1.08	1.78±0.05	8.02±0.48	10.73±0.19	50.55±0.50
BED-LLM (Choudhury et al., 2026)	55.95±0.23	26.60±0.56	26.01±0.51	34.93±0.60	**48.48±0.83**	1.85±0.10	7.76±0.57	9.11±0.62	51.61±0.85
CTA (Ding et al., 2026)	53.19±0.50	25.63±0.33	24.18±0.75	31.33±1.01	48.08±1.18	1.50±0.03	7.85±0.59	9.28±0.94	48.72±0.44
Reference Agents									
Memory Ledger	56.30±0.93	22.69±0.72	30.06±0.26	**46.08±0.86**	40.35±1.25	2.51±0.14	6.81±0.48	9.39±0.69	51.53±0.98
Memory+Verification	56.92±0.50	26.29±1.27	27.27±0.99	42.34±0.31	46.41±1.85	2.27±0.03	8.54±0.37	9.98±0.88	52.64±0.59
Self-Evolving Playbook	**57.98±1.78**	**28.85±1.78**	25.97±0.74	42.06±0.94	47.91±0.99	2.36±0.07	8.05±0.31	10.09±0.36	**53.25±1.74**

(a) Full GRIP results on WebShop under the default condition.

Method	τ^2-Retail					τ^2-Airline				
	Success	Earned	Aim	Recovery	PostShift	Success	Earned	Aim	Recovery	PostShift
No Ask	49.85±2.09	27.64±2.70	n/a	36.85±1.62	33.25±1.33	46.64±3.82	9.55±1.68	n/a	24.00±3.68	28.67±7.17
Just Ask	67.57±1.90	31.26±2.32	58.64±2.86	42.18±0.57	56.11±1.71	55.30±4.47	18.69±3.36	47.83±2.55	37.31±4.12	**40.84±6.95**
AT-CoT	66.19±0.85	33.33±1.21	56.64±1.51	42.32±1.27	51.64±0.99	55.29±4.19	17.47±2.07	**48.64±5.08**	37.23±1.67	34.04±8.73
Belief Graph	62.28±1.77	33.32±1.10	47.35±1.81	41.28±0.58	46.32±2.05	51.12±6.20	14.94±3.26	42.55±8.48	29.37±5.31	31.65±10.72
ProductAgent	70.21±1.47	31.52±1.05	**63.05±1.67**	39.96±0.27	59.41±2.08	56.41±4.94	18.85±5.96	45.37±7.02	37.35±7.81	40.20±11.73
SAGE	69.67±2.46	34.39±2.77	57.32±2.12	43.03±2.34	58.44±1.39	58.17±5.35	20.26±2.75	44.73±2.86	36.91±4.23	40.63±9.36
BED-LLM	70.15±2.71	31.82±3.06	58.06±0.93	40.40±2.16	59.22±3.11	58.98±4.84	20.21±2.44	43.89±4.55	38.40±2.23	39.93±7.44
CTA	65.77±1.30	29.01±0.53	60.98±0.38	39.71±3.18	53.85±1.22	56.89±3.20	20.47±2.23	48.44±7.56	38.91±2.47	39.18±8.05
Reference Agents										
Memory.	69.19±1.00	33.06±2.17	60.80±0.64	42.93±0.75	58.71±0.18	57.69±4.59	18.58±1.49	45.04±6.44	36.48±2.56	40.57±6.98
Mem.+Veri.	**71.59±0.85**	31.46±1.14	61.41±3.89	40.51±0.94	**61.93±0.58**	57.53±3.67	19.80±0.60	45.26±1.05	36.86±2.89	39.35±5.63
Self-Evolving.	71.41±3.07	**36.50±4.77**	58.79±1.18	**49.66±4.71**	60.52±4.30	**59.49±2.17**	**20.80±4.10**	44.84±4.02	**40.73±3.66**	36.78±5.98

(b) Results on τ^2-Retail and τ^2-Airline under the same default condition. Five directional GRIP metrics are shown.

Table 2: Main comparison under the default evaluation condition (DeepSeek-V4-Pro backbone, Rational persona, single-fault requests, and intent shifting enabled). Results report mean±sd over three seeded runs. Best results in each column are bolded and second-best results are underlined.

4 EXPERIMENTS

4.1 RESEARCH QUESTION AND EXPERIMENT SETUP

Research Questions. We organize our experiments around three progressive questions. **RQ1: How well do current agents and LLMs handle interactive intent-alignment tasks?** We establish the capability landscape by comparing one strict no-asking baseline, seven recent interactive-agent baselines, and three interaction variants as diagnostic reference points (Details in Appendix C.1). We further evaluate five model backbones to test whether the observed capability differences persist across underlying LLMs. Building on this baseline understanding, we ask **RQ2: How robust are these capabilities under different interaction conditions?** We vary five user personas, increase interaction complexity, and ablate intent shifting to examine how performance changes with user behavior, task difficulty, and evolving goals. Finally, we ask **RQ3: Do the benchmark abstractions reflect real-world user interaction?** We evaluate both the role-realism of simulated users through persona identification and consistency, and the real-world validity of our communication-fault taxonomy and behavioral patterns through comparison with PRODAGENT real interaction data.

Benchmarks. Drift-Bench++ is designed as a general data construction pipeline that converts existing agentic benchmarks into *interactive intent-alignment* tasks. To demonstrate its applicability across distinct agent environments, we instantiate it on WebShop (Yao et al., 2022) and the Retail and Airline domains of τ^2 (Barres et al., 2025), covering both search-and-selection and state-changing tool-use tasks. Benchmark descriptions and adapters are detailed in Appendix B.1.

Baselines. We compare against a *No Ask* control and seven recent clarification methods. We further include three diagnostic reference agents that introduce complementary capabilities in intent tracking, verification, and cross-task adaptation to probe broader directions for interactive intent alignment. Full adaptations and implementations are provided in Appendix C.1.

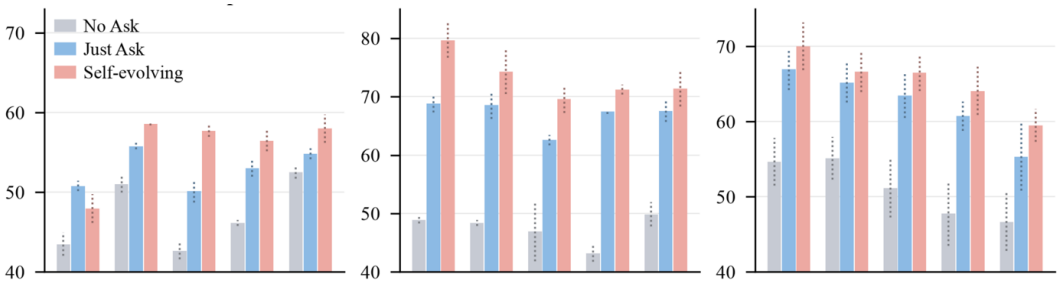

Figure 3: **Comparison between Various LLM Backbones.** The benefit of interaction is backbone-robust. We show success rate of three agents across five LLM backbones. Clarification beats No-Ask baseline on every backbone and benchmark, and Self-evolving leads in 14 of 15 cells.

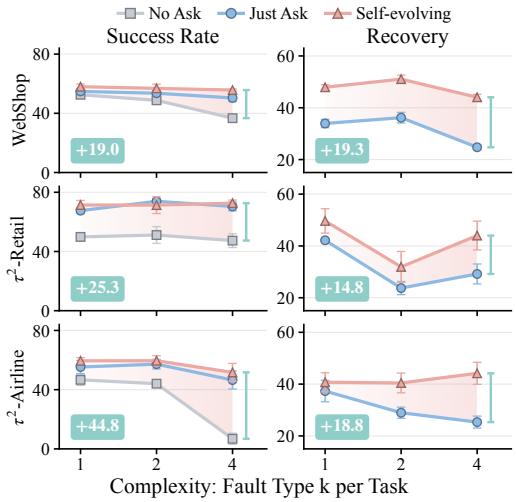

Figure 4: Success Rate and Recovery vs. fault complexity k. Wedges shade the gap between Self-evolving and No Ask (left column) or Just Ask (right); teal badges give this gap at k=4.

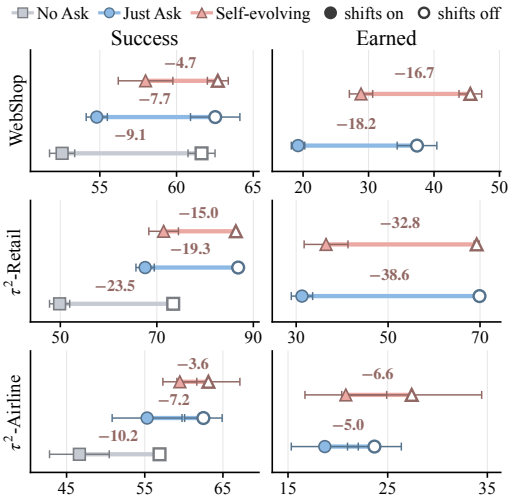

Figure 5: The Impact of Intent Shifting: Success and Earned grounding per method, with intent shifts on (filled) vs. off (hollow); each connector is labeled with its change.

4.2 Experiment Results

Research Question 1. Table 2 compares published baselines with our three reference agents under the default setting. Asking consistently improves performance, with modest gains on WebShop but much larger gains on the state-changing τ^2 domains, showing that clarification becomes increasingly valuable when intent errors have stronger downstream consequences. GRIP further reveals where these gains come from: Self-Evolving achieves the highest *Earned* score on all three benchmarks and top or near-top *Recovery*, indicating that its gains stem more consistently from recovering hidden intent rather than guessing correctly. Other metrics provide complementary insight. For example, a larger *Reaction* can reflect that stronger clarification methods spend additional post-shift turns eliciting and acting on updated user information before resubmission, rather than immediately pursuing a stale goal. Furthermore, Figure 3 shows that these benefits persist across five LLM backbones: interactive strategies broadly outperform No Ask, while Self-Evolving leads in 14 of 15 settings. Thus, stronger backbones do not remove the need for effective intent recovery.

Research Question 2. In real interactions, communication failures may compound rather than occur in isolation. To better approximate such cases, we increase the number of fault types k per task. Figure 4 reveals two effects as complexity grows. First, the Success gap between Self-Evolving and No Ask widens consistently as fault complexity increases across all three benchmarks. Second, its Recovery advantage over Just Ask also grows with complexity, showing an increasing separation between stronger and plain clarification strategies. Together, these trends show that complex misalignment requires not only more interaction, but better clarification strategies. To isolate the effect of evolving intent, we further ablate intent shifting while keeping other conditions fixed. Figure 5 shows that shifts consistently reduce Success and Earned, confirming the added difficulty of tracking a moving goal. More importantly, stronger clarification methods retain their advantage even without shifts and degrade less when shifts are introduced, indicating that better clarification improves both overall performance and robustness to evolving intent. Finally, because the same intent uncertainty

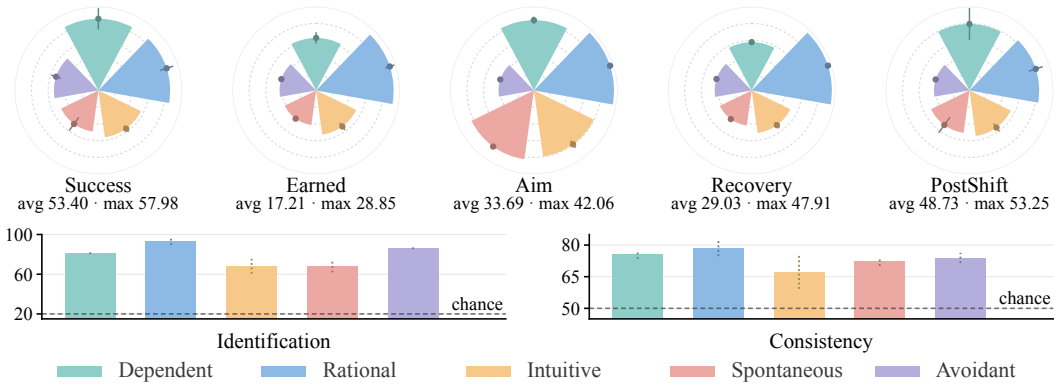

Figure 6: **Persona profiles of the Self-evolving agent on WebShop.** Fans: one metric each, one wedge per persona, avg/max below. Bars: persona identification and consistency, judged from user messages alone with user profile prompt hidden on purpose.

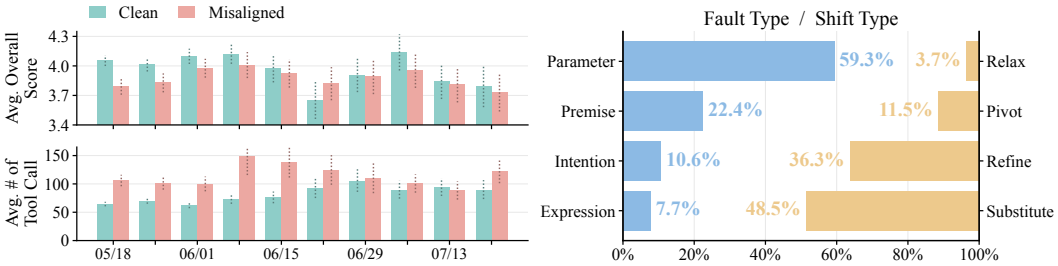

Figure 7: **PRODAGENT validation on 16,596 real production agent sessions over a 10-week deployment.** Left: Comparison of misaligned and matched clean sessions in terms of platform evaluation scores and average tool calls per session; and Right: fault-type and shift-type distributions under strict two-judge agreement against the extracted intents provided by the platform.

may unfold differently across users, we vary persona-conditioned behavior. Figure 6 reveals distinct performance profiles across personas, showing that interactive alignment also depends on how users disclose information, respond to clarification, and revise their goals.

Research Question 3. Because conclusions from simulated interaction depend on meaningful user variation, we first examine the *role-realism* of our persona-conditioned users. Figure 6 hides the persona instructions and asks independent judges to infer roles from user messages alone. Identification and Consistency substantially exceed their 20% and 50% chance baselines across all personas, showing that the simulated behaviors remain distinct and stable without access to the underlying profiles. To further examine whether the phenomena modeled by Drift-Bench++ arise in real deployment, we analyze 16,596 multi-turn sessions collected over 10 weeks from PRODAGENT, the anonymized OpenClaw-like production agent platform. Using annotated intents and evaluation scores provided by the platform, we find that 23.7% of sessions with strict judge agreement exhibit some degree of intent misalignment (Figure 7), establishing *interactive intent misalignment* as a prevalent challenge in deployed agent systems. Moreover, observed misalignments cover all four communication-fault families and all four intent-shift relations, providing real-world support for both taxonomies. Importantly, misaligned sessions also require substantially more tool calls and receive lower human scores on average, showing that these failures impose measurable cost and quality degradation. Together, these results demonstrate that the communication failures and evolving intents modeled by Drift-Bench++ are prevalent, structured, and consequential in real agent use. We provide an extended discussion of PRODAGENT and our real-data validation in Appendix D.

5 CONCLUSION

In this paper, we formulate **Interactive Intent Alignment**, a setting that jointly captures communication misalignment and evolving user intent. We introduce **Drift-Bench++**, a unified benchmark for systematically evaluating whether agents can recover and continuously track user intent under imperfect communication and evolving goals. Across diverse setting, stronger clarification substantially improves alignment, yet large gaps remain. Validation on PRODAGENT production sessions confirms that these phenomena are prevalent and costly in deployment. Overall, Drift-Bench++ enables the community to diagnose and advance agents that aligning with users throughout interaction.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets; assist with experiment design and feedback; support implementation of the proposed method; clean and reformat datasets; assist with qualitative and thematic analysis; and support interpretation of experimental results. We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design the overall research agenda, or assist with translation. Formulating mathematical claims, providing key ingredients for mathematical proofs, and assisting with proof writing were not applicable to this work. Additionally, we used generative AI tools to create research artifacts, draft portions of the manuscript, summarize and analyze existing literature, search for and identify relevant information and references, improve the readability of the manuscript, and format references. All AI-assisted outputs were reviewed and verified by the authors. LLM-generated code was manually inspected and tested, synthetic data and experimental outputs were checked for correctness and consistency, literature-related suggestions were verified against the original sources, and all AI-assisted writing and analysis were reviewed and revised by the authors. We take full responsibility for the final content of this work, including all text, claims, code, data, analyses, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work primarily builds on publicly available agent benchmarks and synthetic user interactions. For real-world validation, we additionally analyze PRODAGENT interaction data. **We emphasize PRODAGENT is user-authorized data for research purposes. We do not release, redistribute, or reproduce PRODAGENT user data, and report only aggregate findings derived from these interactions.** The persona-conditioned simulations in Drift-Bench++ are based on behavioral decision-making styles rather than demographic or protected attributes, and are intended to model variation in communication and interaction behavior rather than characterize real populations. Drift-Bench++ is designed as a diagnostic benchmark for studying miscommunication and evolving user intent, and does not introduce new user-facing interventions or collect new human-subject data. We take care to preserve the privacy and usage constraints of all source data and release only benchmark artifacts derived from public benchmarks or synthetic generation.

REPRODUCIBILITY STATEMENT

We release the full Drift-Bench++ codebase, including benchmark construction, intent-graph generation, perturbation verification, interaction protocols, user simulation, GRIP evaluation, baseline implementations, experiment configurations, and analysis scripts. Detailed implementation choices, prompts, hyperparameters, and benchmark-specific adapters are provided in the appendix, while the core benchmark is constructed from publicly available datasets to facilitate independent reproduction. We also report results over multiple seeded runs and document the corresponding evaluation settings. For the PRODAGENT real-world analysis, the underlying interaction data and associated analysis artifacts cannot be released due to company policy; we therefore report only aggregate results and methodological details for that component.

REFERENCES

Emre Can Acikgoz, Jinoh Oh, Jie Hao, Joo Hyuk Jeon, Heng Ji, Dilek Hakkani-Tür, Gokhan Tur, Xiang Li, Chengyuan Ma, and Xing Fan. Speakrl: Synergizing reasoning, speaking, and acting in language models with reinforcement learning. In *Proceedings of the 16th International Workshop on Spoken Dialogue System Technology*, pp. 312–325. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.iwsds-1.32/.

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026.

J. L. Austin. *How to Do Things with Words*. Clarendon Press, 1975.

Han Bao, Zheyuan Zhang, Pengcheng Jing, Zhengqing Yuan, Kaiwen Shi, and Yanfang Ye. Driftbench: Diagnosing cooperative breakdowns in llm agents under input faults via multi-turn interaction. In *ICML*, 2026. URL https://arxiv.org/abs/2602.02455.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Query reformulation mining: models, patterns, and applications. *Information Retrieval*, 14(3):257–289, 2011.

Junle Chen, Wei Chen, Yehong Xu, Zhengjun Huang, Yuqian Wu, Zhoujin Tian, Kai Wang, Lei Wang, and Xiaofang Zhou. Trip+: Benchmarking agents in personalized interactive travel planning, 2026a. URL https://arxiv.org/abs/2606.21169.

Junzhi Chen, Harsh Trivedi, Jane Pan, Michael JQ Zhang, Tejas Srinivasan, Niranjan Balasubramanian, and Ashish Sabharwal. Appworld-ul: Benchmarking diverse agent-user interactions for tool-use. In *International Conference on Machine Learning (ICML)*, 2026b.

Sanxing Chen, Sam Wiseman, and Bhuwan Dhingra. Chatshop: Interactive information seeking with language agents, 2024. URL https://arxiv.org/abs/2404.09911.

Deepro Choudhury, Sinead Williamson, Adam Goliński, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent Information Gathering with LLMs and Bayesian Experimental Design. In *International Conference on Learning Representations (ICLR)*, 2026.

Wenxuan Ding, Nicholas Tomlin, and Greg Durrett. Calibrate-Then-Act: Cost-Aware Exploration in LLM Agents, 2026. arXiv preprint arXiv:2602.16699.

Zeyao Du, Tong Li, and Haibo Zhang. EComAgentBench: Benchmarking shopping agents on long-horizon tasks with distributed hidden intent. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2026. URL https://arxiv.org/abs/2606.17698.

Hamza El Alaoui, Atieh Taheri, Yi-Hao Peng, and Jeffrey P. Bigham. Stepwrite: Adaptive planning for speech-driven text generation. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, UIST '25, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3746059.3747610.

Yujian Gan, Changling Li, Jinxia Xie, Luou Wen, Matthew Purver, and Massimo Poesio. Clarq-llm: A benchmark for models clarifying and requesting information in task-oriented dialog, 2024. URL https://arxiv.org/abs/2409.06097.

Jiaxuan Gao, Jiaao Chen, Chuyi He, Wei-Chen Wang, Shusheng Xu, Hanrui Wang, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents, 2026. URL https://arxiv.org/abs/2601.22607.

Debasmita Ghose, Oz Gitelson, Ryan Jin, Grace Abawe, Marynel Vázquez, and Brian Scassellati. I've changed my mind: Robots adapting to changing human goals during collaboration. *IEEE Robotics and Automation Letters*, 11(2):1490–1497, 2026. doi: 10.1109/LRA.2025.3643294.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

H. Paul Grice. Logic and conversation. In *Syntax and Semantics, Volume 3: Speech Acts*. 1975.

Meera Hahn, Wenjun Zeng, Nithish Kannen, Rich Galt, Kartikeya Badola, Been Kim, and Zi Wang. Proactive Agents for Multi-Turn Text-to-Image Generation Under Uncertainty. In *International Conference on Machine Learning (ICML)*, 2025.

Muyu He, Anand Kumar, Tsach Mackey, Meghana Rajeev, James Zou, and Nazneen Rajani. Impatient users confuse AI agents: High-fidelity simulations of human traits for testing agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. doi: 10.18653/v1/2026.acl-long.1743. URL https://arxiv.org/abs/2510.04491.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, Man Gao, Xi Su, Xiaodong Cai, Xunliang Cai, Yu Yang, and Yunke Zhao. VitaBench: Benchmarking LLM agents with versatile interactive tasks in real-world applications, 2025. URL https://arxiv.org/abs/2509.26490.

Jeff Huang and Efthimis N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pp. 77–86, 2009.

Nan Huo, Xiaohan Xu, Jinyang Li, Per Jacobsson, Shipei Lin, Bowen Qin, Binyuan Hui, Xiaolong Li, Ge Qu, Shuzheng Si, Linheng Han, Edward Alexander, Xintong Zhu, Rui Qin, Ruihan Yu, Yiyao Jin, Feige Zhou, Weihao Zhong, Yun Chen, Hongyu Liu, Chenhao Ma, Fatma Ozcan, Yannis Papakonstantinou, and Reynold Cheng. BIRD-INTERACT: Re-imagining text-to-SQL evaluation for large language models via lens of dynamic interactions. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.05318.

Zimo Ji, Zekai Zhang, Congying Xu, Zongjie Li, Yudong Gao, Shuai Wang, and Shing-Chi Cheung. Coding agents are guessing: Measuring action-boundary violations in underspecified devops instructions, 2026. URL https://arxiv.org/abs/2607.02294.

Fanqi Kong, Jiayi Zhang, Mingyi Deng, Chenglin Wu, Yuyu Luo, and Bang Liu. Infopo: Information-driven policy optimization for user-centric agents, 2026. URL https://arxiv.org/abs/2603.00656.

Tessa Lau and Eric Horvitz. Patterns of search: Analyzing and modeling web query refinement. In *UM99 User Modeling: Proceedings of the Seventh International Conference*. Springer, 1999.

Gyubok Lee, Woosog Chay, Heeyoung Kwak, Yeong Hwa Kim, Haanju Yoo, Oksoon Jeong, Meong Hi Son, and Edward Choi. From conversation to query execution: Benchmarking user and tool interactions for ehr database agents. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=hLweUPBz7k.

Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting Human Preferences with Language Models. In *International Conference on Learning Representations (ICLR)*, 2025.

Jialin Li, Yuan Wu, and Yi Chang. Clareval: A benchmark for evaluating clarification skills of code agents under ambiguous instructions, 2026a. URL https://arxiv.org/abs/2603.00187.

Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 28858–28888, 2024. doi: 10.52202/079017-0908. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/32b80425554e081204e5988ab1c97e9a-Abstract-Conference.html.

Shuyue Stella Li, Bhargavi Paranjape, Kerem Oktar, Zhongyao Ma, Gelin Zhou, Lin Guan, Na Zhang, Sem Park, Lin Chen, Diyi Yang, Yulia Tsvetkov, and Asli Celikyilmaz. HorizonBench: Long-horizon personalization with evolving preferences, 2026b. URL https://arxiv.org/abs/2604.17283.

Jiayu Liu, Cheng Qian, Zhaochen Su, Qing Zong, Shijue Huang, Bingxiang He, and Yi R. Fung. Costbench: Evaluating multi-turn cost-optimal planning and adaptation in dynamic environments for llm tool-use agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12826–12858. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.acl-long.584. URL https://aclanthology.org/2026.acl-long.584/.

Yao Liu, Guangjia Chai, Yuming Huang, Jihao Huang, Lei Wang, and Junchen Wan. CompanionBench: A theory-anchored, real-world-grounded benchmark for AI emotional companionship, 2026b. URL https://arxiv.org/abs/2608.02046.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 14179–14198, 2025. doi: 10.18653/v1/2025.acl-long.694.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Kushan Mitra, Dan Zhang, Hannah Kim, and Estevam Hruschka. RECAP: REwriting conversations for intent understanding in agentic planning. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2015–2033, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-eacl.105. URL https://aclanthology.org/2026.findings-eacl.105/.

Smit Nautambhai Modi, Gandharv Mahajan, Marc Wetter, and Randall Welles. Echochain: A full-duplex benchmark for state-update reasoning under interruptions, 2026. URL https://arxiv.org/abs/2604.16456.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.06552.

Juhyun Oh, Eunsu Kim, and Alice Oh. Flex-travelplanner: A benchmark for flexible planning with language agents, 2025. URL https://arxiv.org/abs/2506.04649.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

George Pu, Michael S. Lee, Udari Madhushani Sehwag, David J. Lee, Bryan Zhu, Yash Maurya, Mohit Raghavendra, Yuan Xue, and Samuel Marc Denton. Lhaw: Controllable underspecification for long-horizon tasks. In *Lifelong Agent Workshop at ICLR 2026*, 2026. URL https://arxiv.org/abs/2602.10525.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1088–1113, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.61. URL https://aclanthology.org/2024.acl-long.61/.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, and Huan Wang. Userbench: An interactive gym environment for user-centric agents. In *NeurIPS 2025 Workshop on Scaling Environments for Agents (SEA)*, 2025a.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*, 2025b.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userrl: Training interactive user-centric agent via reinforcement learning, 2025c. URL https://arxiv.org/abs/2509.19736.

Mohit Raghavendra, Anisha Gunjal, Aakash Sabharwal, and Yunzhong He. SWE-INTERACT: Reimagining SWE benchmarks as user-driven long-horizon coding sessions, 2026. URL https://arxiv.org/abs/2606.30573.

Manik Rana, Calissa Man, Anotida Expected Msiiwa, Jeffrey Paine, Kevin Zhu, Sunishchal Dev, Vasu Sharma, and Ahan M R. Agentchangebench: A multi-dimensional evaluation framework for goal-shift robustness in conversational ai. In *LAW 2025: Bridging Language, Agent, and World Models for Reasoning and Planning, NeurIPS 2025 Workshop*, 2025.

Corby Rosset, Pratyusha Sharma, Andrew Zhao, Miguel González-Fernández, and Ahmed Awadallah. The art of building verifiers for computer use agents. In *Conference on Language Modeling*, 2026.

Susanne G Scott and Reginald A Bruce. Decision-making style: The development and assessment of a new measure. *Educational and psychological measurement*, 55(5):818–831, 1995.

Yuanzhe Shen, Zisu Huang, Zhengyuan Wang, Muzhao Tian, Zhengkang Guo, Chenyang Zhang, Shuaiyu Zhou, Zengjie Hu, Dailin Li, Jingwen Xu, Kaimin Wang, Wenhao Liu, Tianlong Li, Fengpeng Yue, Feng Hong, Cao Liu, and Ke Zeng. Trip-bench: A benchmark for long-horizon interactive agents in real-world scenarios, 2026. URL https://arxiv.org/abs/2602.01675.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377.

Josefa Lia Stoisser, Marc Boubnovski Martell, Sidsel Boldsen, Kaspar Märtens, and Robert Kitchen. Ambig-ds: A benchmark for task-framing ambiguity in data-science agents, 2026. URL https://arxiv.org/abs/2605.09698.

Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized llm agents. In *Conference on Language Modeling (COLM)*, 2026.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured Uncertainty guided Clarification for LLM Agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Minh Ngoc Ta, My Anh Tran Nguyen, Duong D. Nguyen, Yuxia Wang, and Preslav Nakov. One more turn, less regret: A regret-based multi-turn benchmark for llms' clarification policies, 2026. URL https://arxiv.org/abs/2607.21143.

Jihoon Tack, Philippe Laban, and Jennifer Neville. Llms get lost in evolving user intent, 2026. URL https://arxiv.org/abs/2607.20734.

Anfu Tang, Laure Soulier, and Vincent Guigue. Clarifying Ambiguities: on the Role of Ambiguity Types in Prompting Methods for Clarification Generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2025.

Tu Trinh, Mohamed Elfeki, Guangze Luo, Kelvin Luu, Nathan Hunt, Ernesto Hernandez, Nandan Marwaha, Yannis Yiming He, Charles Wang, Fernando Carabedo, Alessa Castillo, and Bing Liu. Hil-bench (human-in-loop benchmark): Do agents know when to ask for help?, 2026. URL https://arxiv.org/abs/2604.09408.

Danqing Wang, Akshay Sivaraman, and Lei Li. CRAB-Bench: Evaluating LLM agents under complex task dependencies and human-aligned user simulation, 2026a. URL https://arxiv.org/abs/2606.01815.

Wenkai Wang, Tao Xiong, Jingchen Ni, Yunpeng Bao, Xiyun Li, Tianqi Liu, Hongcan Guo, Zilong Huang, and Shengyu Zhang. Deskcraft: Benchmarking desktop agents on professional workflows and human-in-the-loop collaboration, 2026b. URL https://arxiv.org/abs/2606.03103.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Pei Chen, Ziwei Dong, Jing Huang, Jiri Gesi, Xianfeng Tang, Chen Luo, Qun Liu, Yisi Sang, Hanqing Lu, Manling Li, Jin Lai, and Dakuo Wang. Trajectory2task: Training robust tool-calling agents with synthesized yet verifiable data for complex user intents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44021–44044, San Diego, California, United States, July 2026c. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.2037. URL https://aclanthology.org/2026.acl-long.2037/.

Paul Watzlawick, Janet Beavin Bavelas, and Don D. Jackson. *Pragmatics of Human Communication: A Study of Interactional Patterns, Pathologies, and Paradoxes*. W. W. Norton & Company, 2011.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67260–67283. PMLR, 2025.

Yaozu Wu, Wei-Chieh Huang, Jizhou Guo, Dongyuan Li, Renhe Jiang, Henry Peng Zou, Chunyu Miao, Shanghao Li, Weizhi Zhang, WeiWei Ye, Yankai Chen, Meng Zhang, Xue Liu, and Philip S. Yu. Has-bench: Evaluating llm-based human-agent systems under configurable human participation, 2026. URL https://arxiv.org/abs/2607.04329.

Wenya Xie, Shengming Zhou, Zelin Li, Pouya Parsa, Shuang Zhou, Xinheng Ding, Chinmay Arvind, Guanchu Wang, Vladimir Braverman, Ali Payani, Yantao Zheng, and Zirui Liu. Dynamicmem: A long-horizon memory benchmark in real-world settings, 2026. URL https://arxiv.org/abs/2606.22877.

Siyuan Xu, Shiyang Li, Xin Liu, Tianyi Liu, Yixiao Li, Zhan Shi, Zixuan Zhang, Zilong Wang, Qingyu Yin, Jianshu Chen, Tuo Zhao, and Bing Yin. Controllable and verifiable tool-use data synthesis for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2604.09813.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 12805–12825, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-eval: Toward trustworthy evaluation of autonomous agents, 2026. URL https://arxiv.org/abs/2604.06132.

Jingheng Ye, Yong Jiang, Xiaobin Wang, Yinghui Li, Yangning Li, Hai-Tao Zheng, Pengjun Xie, and Fei Huang. ProductAgent: Benchmarking Conversational Product Search Agent with Asking Clarification Questions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP)*, 2025.

Simon Yu, Nicholas Tomlin, Marwa Abdulhai, Ximing Lu, Derek Chong, Abe Hou, Dilara Soylu, Sergey Levine, Christopher D. Manning, and Weiyan Shi. One frozen simulator is not enough: Simulator collapse in multi-agent rl, 2026. URL https://arxiv.org/abs/2608.12253.

Fuxiang Zhang, Jiacheng Xu, Chaojie Wang, Ce Cui, Yang Liu, and Bo An. Incentivizing llms to self-verify their answers. In *Advances in Neural Information Processing Systems*, 2025.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, 2026.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10836–10863, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.636.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

Zheng Zhao, Clara Vania, Subhradeep Kayal, Naila Khan, Shay B. Cohen, and Emine Yilmaz. PersonaLens: A benchmark for personalization evaluation in conversational AI assistants. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18023–18055, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.927. URL https://aclanthology.org/2025.findings-acl.927/.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946.

Dingwei Zhu, Jiahan Li, Chengjun Pan, Yunxian Yang, Yunbin Zhao, Yunke Zhang, Zhonghang Lu, Zhuohui Sheng, Chenhao Huang, Jiahang Lin, Yajie Yang, Junlin Shang, Shichun Liu, Yuhui Wang, Honglin Guo, Junjie Ye, Xin Guo, Jiazheng Zhang, Ming Zhang, Shihan Dou, Zhiheng Xi, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Iacm-rl: Intent-aware context management and reinforcement learning for complex tool invocation under dynamic intent fluctuations, 2026. URL https://arxiv.org/abs/2608.02110.

Henry Peng Zou, Chunyu Miao, Wei-Chieh Huang, Yankai Chen, Yue Zhou, Hanrong Zhang, Yaozu Wu, Liancheng Fang, Zhengyao Gu, Zhen Zhang, Kening Zheng, Fangxin Wang, Yi Nian, Shanghao Li, Wenzhe Fan, Langzhou He, Weizhi Zhang, Xue Liu, and Philip S. Yu. When users change their mind: Evaluating interruptible agents in long-horizon web navigation, 2026. URL https://arxiv.org/abs/2604.00892.

APPENDIX CONTENT TABLE

A Additional Related Work **18**

B Implementation Details **20**

B.1 Host-Benchmark Adapters and Executable Oracles 20

B.2 Intent Graph Construction 22

B.3 Verified Query Perturbation 24

B.4 Patience-Bounded Interaction 25

B.5 Interaction-Conditioned Intent Shifting 26

B.6 Persona-Conditioned Behavior 27

B.7 GRIP Evaluation Details 28

C Additional Experiments **30**

C.1 Baselines and Reference Methods 31

C.1.1 Baselines 31

C.1.2 Reference Agents. 32

C.2 Experimental Setup and Controls 34

C.3 RQ1: Capability Across Methods and Backbones 35

C.4 RQ2: Robustness to Interaction Conditions 37

C.4.1 Composed Communication Faults 37

C.4.2 Intent-Shift Ablation 39

C.4.3 Persona-Conditioned Interaction 41

C.5 RQ3: Role-Realism Validation 43

D Drift-Bench++ on Production Agent Data **43**

E Limitations and Future Work **44**

F Prompt Design **45**

G Case Study **45**

Benchmarks	No Oracle Comm.	Intent Shifting	Shift Trigger	Patience-bounded	Persona-guided	Beyond Success Eval.
Intent Misalignment Benchmarks						
ChatShop (Chen et al., 2024)	✓	✗	✗	✗	✗	✗
ClarQ-LLM (Gan et al., 2024)	✓	✗	✗	✗	✗	I.R.
IN3 (Qian et al., 2024)	✓	✗	✗	✗	✗	I.R.
MediQ (Li et al., 2024)	✓	✗	✗	✗	✗	I.R.
UserBench (Qian et al., 2025a)	✓	✗	✗	✗	✗	✗
UserVille (Sun et al., 2026)	✓	✗	✗	✗	✓	✗
SpeakRL (Acikgoz et al., 2026)	✓	✗	✗	✗	✗	I.R.
ClarEval (Li et al., 2026a)	✓	✗	✗	✗	✗	I.R.
COVERT (Xu et al., 2026)	✓	✗	✗	✗	✗	✗
HiL-Bench (Trinh et al., 2026)	✓	✗	✗	✗	✗	I.R.
Ambig-DS (Stoisser et al., 2026)	✓	✗	✗	✗	✗	✗
LHAW (Pu et al., 2026)	✓	✗	✗	✗	✗	✗
UnderSpecBench (Ji et al., 2026)	✓	✗	✗	✗	✗	✗
AppWorld-UL (Chen et al., 2026b)	✓	✗	✗	✗	✗	✗
RegretBench (Ta et al., 2026)	✓	✗	✗	✗	✓	I.R.
Drift-Bench (Bao et al., 2026)	✓	✗	✗	✗	✓	I.R., U.R.
Intent Shifting Benchmarks						
Flex-TravelPlanner (Oh et al., 202	✗	✓	Schedule	✗	✗	✗
AgentChangeBench (Rana et al.,	✗	✓	Outcome	✗	✓	S.T.
TRIP-Bench (Shen et al., 2026)	✗	✓	Schedule	✗	✗	✗
Trajectory2Task (Wang et al., 2026	✗	✓	Schedule	✗	✗	✗
InterruptBench (Zou et al., 2026)	✗	✓	Schedule	✗	✗	S.T.
EchoChain (Modi et al., 2026)	✗	✓	Outcome	✗	✗	S.T.
Trip+ (Chen et al., 2026a)	✗	✓	Schedule	✗	✗	✗
HAS-Bench (Wu et al., 2026)	✗	✓	Schedule	✗	✓	U.R., S.T.
EHR-ChatQA (Lee et al., 2026)	✗	✓	Outcome	✗	✗	✗
Evolving Intent (Tack et al., 2026)	✗	✓	Schedule	✗	✗	✗
IACM-RL (Zhu et al., 2026)	✗	✓	Schedule	✗	✗	S.T.
DeskCraft (Wang et al., 2026b)	✗	✓	Outcome	✗	✗	S.T.
DynamicMem (Xie et al., 2026)	✗	✓	Schedule	✗	✗	S.T.
Progressive and Interactive Agent Benchmarks						
τ^2-Bench (Barres et al., 2025)	✗	✗	✗	✗	✗	✗
EigenData (Gao et al., 2026)	✗	✗	✗	✗	✗	✗
Claw-Eval (Ye et al., 2026)	✗	✗	✗	✗	✗	Safety, Robust.
CostBench (Liu et al., 2026a)	✗	✗	✗	✗	✗	Cost
VitaBench (He et al., 2025)	✓	✗	✗	✗	✗	✗
BIRD-Interact (Huo et al., 2026)	✓	✗	✗	✗	✗	I.R.
SWE-Interact (Raghavendra et al.,	✓	✗	✗	✗	✗	I.R.
EComAgentBench (Du et al., 202	✓	✗	✗	✗	✗	I.R.
HorizonBench (Li et al., 2026b)	✓	✓	Schedule	✗	✗	S.T.
Persona and User-Simulation Benchmarks						
τ-Trait (He et al., 2026)	✗	✗	✗	✗	✓	U.R.
CRAB-Bench (Wang et al., 2026a)	✗	✗	✗	✗	✓	U.R.
PersonaLens (Zhao et al., 2025)	✗	✗	✗	✗	✓	U.R.
User LMs (Naous et al., 2026)	✗	✗	✗	✗	✗	U.R.
CompanionBench (Liu et al., 2026	✓	✗	✗	✗	✓	U.R.
Drift-Bench++ (Ours)	✓	✓	**Interaction**	✓	✓	**I.R., U.R., S.T.**

I.R.: Intent Recovery Evaluation; **U.R.**: User Realism Evaluation; **S.T.**: Shift-Tracking Evaluation.

Table 3: Extended comparison of benchmarks related to interactive intent alignment. The comparison follows the same criteria as Table 1, covering intent misalignment, intent shifting, progressive interaction, and user simulation.

A Additional Related Work

Section 2 focuses on the works most directly comparable to Drift-Bench++, namely benchmarks that explicitly study either intent misalignment or intent shifting. Here, we broaden the discussion to recent benchmarks that relax related assumptions of interactive evaluation. These works are relevant

to our setting, but are generally less direct comparators because they either isolate a narrower form of miscommunication, change the interaction protocol without changing the latent intent, or focus primarily on user simulation rather than alignment itself. Table 3 summarizes this broader landscape under the same criteria used in the main text.

Intent Misalignment Benchmarks. Beyond the representative works discussed in Section 2, several recent benchmarks study more specialized forms of incomplete or ambiguous communication. SpeakRL and ClarEval construct interactive settings in which agents must decide how to clarify underspecified requests (Acikgoz et al., 2026; Li et al., 2026a), while HiL-Bench asks whether agents can recognize when human assistance is needed (Trinh et al., 2026). Ambig-DS focuses specifically on task-framing ambiguity, and COVERT emphasizes controllable synthesis of ambiguous or incomplete tool-use requests (Stoisser et al., 2026; Xu et al., 2026). These benchmarks expand the domains and construction mechanisms through which miscommunication can be studied, but their uncertainty is typically defined around a particular form of ambiguity, underspecification, or intervention need. RegretBench considers another important axis by explicitly measuring whether clarification justifies its cost (Ta et al., 2026); however, this cost functions as an evaluation trade-off rather than a finite user state that constrains future interaction. We therefore view these works as complementary extensions of the fixed-intent misalignment setting, whereas the benchmarks highlighted in the main text are closest to Drift-Bench++ in their explicit control over the gap between the user's communicated request and underlying intent.

Intent Shifting Benchmarks. The broader shifting literature similarly contains several variants beyond those highlighted in Section 2. Flex-TravelPlanner and TRIP-Bench introduce changing requirements in long-horizon planning tasks (Oh et al., 2025; Shen et al., 2026); EchoChain studies state updates under interruptions (Modi et al., 2026); DeskCraft incorporates repair and revision into interactive desktop workflows (Wang et al., 2026b); and DynamicMem evaluates adaptation to preferences that change over longer histories (Xie et al., 2026). These settings enlarge the range of tasks in which stale information can become harmful, but their primary focus is often long-horizon adaptation, memory, interruption handling, or workflow repair rather than communication after the shift itself. Together with AgentChangeBench, Trajectory2Task, InterruptBench, Trip+, HAS-Bench, EHR-ChatQA, Evolving Intent, and IACM-RL (Rana et al., 2025; Wang et al., 2026c; Zou et al., 2026; Chen et al., 2026a; Wu et al., 2026; Lee et al., 2026; Tack et al., 2026; Zhu et al., 2026), they largely realize changes through scheduled or outcome-conditioned transitions. This makes them important evidence that evolving targets matter, while the distinction emphasized in the main text remains that the resulting change is generally provided to the agent rather than itself becoming a new source of latent misalignment.

Progressive and Interactive Agent Benchmarks. A third group is related through the interaction protocol rather than through intent change itself. τ^2-Bench makes both the agent and user active participants in a shared environment (Barres et al., 2025); EigenData studies the construction of interactive tool-use trajectories (Gao et al., 2026); Claw-Eval broadens evaluation beyond task success to robustness and safety (Ye et al., 2026); and CostBench explicitly evaluates the cost of long-horizon execution (Liu et al., 2026a). These benchmarks capture important difficulties of realistic interaction, but do not primarily study a mismatch between communicated and latent intent. More closely related are progressive-disclosure settings such as VitaBench, BIRD-Interact, SWE-Interact, and EComAgentBench (He et al., 2025; Huo et al., 2026; Raghavendra et al., 2026; Du et al., 2026), where information needed to complete the task is revealed over multiple turns. This superficially resembles intent misalignment because the agent initially lacks part of the target. The key difference is that disclosure is generally monotone: later information supplements what was previously known rather than making a previously correct requirement obsolete. HorizonBench goes further by allowing preferences to evolve over long histories (Li et al., 2026b), but models those changes as part of a scheduled personalization setting. We therefore include these benchmarks in the extended comparison because they probe adjacent forms of evolving information, while treating them as less direct comparators to the joint misalignment-and-shifting setting studied in Drift-Bench++.

Persona and User-Simulation Benchmarks. Finally, a separate line of work studies variation in the user rather than in the latent task. τ-Trait varies behavioral traits while preserving the underlying goal (He et al., 2026), and CRAB-Bench introduces controlled user behaviors in interactive agent evaluation (Wang et al., 2026a). PersonaLens and User LMs focus more directly on whether simulated users exhibit coherent and recognizable behavioral characteristics (Zhao et al., 2025; Naous et al., 2026). CompanionBench additionally models structured disclosure and retreat in

emotional-companionship interactions (Liu et al., 2026b). These works are particularly relevant to our persona-conditioned simulation, but their main object of study is the realism or diversity of user behavior rather than recovery of an executable latent intent. For this reason, they are not the primary comparators in Section 2. Drift-Bench++ instead uses persona variation as an orthogonal interaction factor: the benchmark controls what the user's current intent is, while the persona controls how that intent is communicated.

Overall, the extended literature shows that recent benchmarks increasingly relax individual assumptions of conventional agent evaluation, including complete communication, static goals, passive users, fixed interaction costs, and homogeneous user behavior. The distinction of Drift-Bench++ lies in bringing several of these dimensions into the same executable setting rather than studying each in isolation. Table 3 provides the corresponding detailed comparison.

B Implementation Details

This section specifies the operational details required to reproduce Drift-Bench++ from benchmark construction through interactive evaluation. Following Section 3, we first define the host-benchmark adapter and executable-oracle interface that preserves each environment's native task semantics, then detail the construction and validation of executable intent graphs and verified misaligned requests. We next specify the runtime interaction protocol, including patience costs and termination rules, silent scheduled and agent-conditioned intent shifts, and persona-controlled disclosure, feedback, and shift behavior. Finally, we provide the exact definitions and bookkeeping used by GRIP, including the event-conditioned denominators required to interpret its grounding, inquiry, role-realism, and persistence metrics. The table below summarizes the resulting benchmark instantiations across the three host environments.

B.1 Host-Benchmark Adapters and Executable Oracles

Section 3.1 introduces the adapter as the boundary between each host benchmark and the shared Drift-Bench++ pipeline. Here we specify that boundary operationally. Rather than imposing a universal task schema, the adapter standardizes only the *units of intent*: it decomposes a native task into independently addressable requirements that Drift-Bench++ can hide, perturb, reveal, and update, while the host benchmark continues to determine legality, execution, and final success. This separation allows the same interaction machinery to operate over the structurally different environments in Table 4.

Benchmarks	Tasks	Graphs	Rejection Rate	Samples
WebShop	500	257	31%	**1,780**
τ^2-Retail	114	98	47%	**555**
τ^2-Airline	50	40	48%	**208**

Table 4: Key statistics of the construction pipeline with the rejection rate of the three-step verifier and the resulting sample numbers.

Requirement-level intent representation. For native task x from host benchmark h, LOAD returns

$$(e_x, f_x, C_x, Y_x), \qquad C_x = \{c_j\}_{j=1}^{m_x}, \qquad c_j = (s_j, o_j, v_j),$$

where e_x is the executable environment, f_x the task frame, C_x the set of m_x task-defining requirements, and Y_x the shipped benchmark answer used only for validation. Each requirement c_j contains a slot s_j identifying what is constrained, an operator o_j, and its value v_j. Their semantics remain host-specific: a WebShop slot may denote a product property, while a τ^2 slot may identify an argument of a required write.

This decomposition is shared across all later stages. A communication fault hides or distorts conditions without changing the latent intent; a shift adds, removes, or substitutes them; the simulator tracks which current conditions have been communicated; and GRIP evaluates Aim and Recovery over the corresponding hidden and elicited requirements. Conditions therefore enter C_x only when they are both *addressable*, so changing one expresses a meaningful goal change, and *reconstructible*, so (f_x, C_x) can be compiled back into an executable task. Information describing only the solution path is excluded. For example, τ^2 read actions support execution but do not themselves define what the user wants.

Adapter contract. Beyond LOAD, the shared implementation can have the following host-specific operations:

$$\text{COMPILE}_h(f, C) \to \widetilde{x}, \quad \text{VALIDATE}_h(e, f, C, Y) \to \{0, 1\},$$
$$\text{EXECUTE}_h(\widetilde{x}) \to Y_I, \quad \text{WITNESS}_h(f, C) \to \mathcal{W}_I,$$
$$\text{DOMAINS}_h(s) \to \mathcal{V}_s,$$

where $\widetilde{x}$ is the native task reconstructed from frame f and conditions C, $I = (f, C)$ is the corresponding intent, Y_I its executable ground truth, $\mathcal{W}_I$ the satisfying objects or states used to construct refinements, and $\mathcal{V}_s$ the observed values of slot s used to construct substitutions. ENV_SPEC and IS_MUTATING additionally determine how environments are materialized and whether executions require isolated state.

Validation deliberately follows host semantics. WebShop requires the original target product to remain among the maximal-scoring purchases after reconstruction; τ^2 requires the reconstructed writes to execute without rejection and produce a nontrivial state change. Invalid and no-op tasks are discarded rather than repaired. Legality is similarly layered: execution rejects invalid writes, the optional IS_VALID_INTENT hook handles additional semantic constraints, and ACCEPTS applies the native runtime success rule. Ground truth is stored as one of `purchaseset`, `statehash`, `rowset`, or `scalar`; empty ground truth is rejected, and set monotonicity is enforced only when the ground truth is genuinely extensional.

WebShop. WebShop represents C using product attributes, requested options, and price constraints. Its construction oracle returns the set of maximal-scoring product–option configurations satisfying the current intent, which supports satisfiability and set-based relation checks without assuming a unique answer. To make reconstruction deterministic, we replace WebShop's randomly sampled price ceiling with the second \$10 step above the target product's price while leaving the reward unchanged. Runtime success still uses the original interaction: the agent must select an item and options and execute *Buy Now*, after which WebShop's shipped reward scores the actual purchase. We do not use membership in the stored construction set because alternative option configurations may receive the same native reward. Only the 500 official test goals may anchor graphs; other validated records may provide neighboring intents but never roots.

τ^2-Retail. Retail tasks contain a user request and a golden trace of reads and writes. Reads remain in the frame as solution mechanics, while every argument of every required write becomes an addressable condition. For write i, tool a_i, and argument θ, the slot is

$$s_{i,\theta} = \texttt{arg:}i\texttt{:}\langle a_i \rangle\texttt{:}\langle \theta \rangle,$$

with the resolved argument stored as its value. Replaying only these writes reproduces the full trace's database hash on all 114 source tasks.

For intuition, consider a user asking to exchange a keyboard for a clicky-switch variant. The conversational requirement is "clicky switches," while the executable benchmark resolves it to a particular replacement-item identifier stored in C. The simulator can therefore discuss the semantic requirement naturally, while the oracle retains exact executable supervision.

This representation also determines which intent relations are valid. Refinement can add another executable request for the same customer, but removing a required write argument usually produces an invalid action rather than a meaningful relaxation, so Retail does not manufacture such edges. For shifted episodes, exact final-state equality is inappropriate because stale writes may be irreversible. We instead require completion of the current intent's golden actions under τ^2's native action reward, so obsolete writes remain costly and measurable through Staleness without making the new goal unreachable.

τ^2-Airline. Airline uses the same write-argument representation but additionally contains dependent quantities. For example, changing a reservation from basic economy to business changes both its fare and baggage allowance. The cabin choice is therefore stored as an addressable condition, while dependent payment and baggage quantities are recomputed by COMPILE. This prevents a valid change to one user requirement from leaving stale derived values elsewhere in the task.

Executability alone is insufficient because the Airline environment does not mechanically enforce all of its written policies. The adapter therefore implements IS_VALID_INTENT with checks covering user and reservation coherence, basic-economy and route restrictions, flown segments, cancellation

eligibility, passenger limits, payment composition, baggage rules, itinerary consistency, and no-op actions. Insurance-dependent cancellations retain their required reason, and shifts cannot cross users' reservations. Retail requires no analogous policy layer because its invalid cases are already rejected during execution.

Portability Boundary. A new benchmark therefore need not share WebShop's schema or τ^2's action representation. It must only expose (i) independently addressable task requirements, (ii) a way to compile modified requirements back into a coherent native task, and (iii) executable validation and acceptance semantics. Everything beyond this boundary, including graph construction, perturbation, disclosure tracking, intent shifting, and GRIP, operates over the shared (f, C) abstraction. Host-native validation remains essential because intent modifications that are expressible through the shared adapter may still violate benchmark-specific legality, execution, or non-triviality constraints. The adapter therefore makes intent modifications expressible, not automatically valid. From 500/114/50 source tasks, host-native validation yields 257/98/40 instantiated intent graphs and 1,780/555/208 verified samples. The next subsection operates on these validated intents to construct relation-labeled alternative goals.

B.2 Intent Graph Construction

Given the validated executable intents produced by Appendix B.1, we construct the graph from which subsequent intent shifts are drawn. As summarized in Section 3.1, we first build a controlled set of alternatives around each root and then derive additional valid transitions among them. This separates candidate generation from runtime traversal: every intent is validated before interaction, while the played graph can still support multiple successive shifts.

Relation Semantics. The four relations are motivated by long-standing query-reformulation taxonomies. Search-log studies distinguish specialization and generalization, same-intent or parallel reformulation, and transitions to a new task (Lau & Horvitz, 1999; Boldi et al., 2011), while substitution is repeatedly identified as a common reformulation strategy (Huang & Efthimiadis, 2009). Drift-Bench++ realizes these categories as exact operations over executable intent conditions rather than lexical relations between query strings.

For one root, let $G = (V, E)$ denote its intent graph. Each node $v_i \in V$ contains an executable intent $I_i = (f_i, C_i)$ and its host-native ground truth Y_i; each directed edge from node i to j stores a relation r_{ij} and its machine-readable change Δ_{ij}. For a condition set C, define its skeleton

$$\kappa(C) = \{(s, o) : (s, o, v) \in C\},$$

where s, o, and v are the slot, operator, and value defined in Appendix B.1. Relations are assigned exactly:

$$r_{ij} = \begin{cases} \textsc{Pivot}, & f_i \neq f_j, \\ \textsc{Refinement}, & f_i = f_j,\ C_i \subset C_j, \\ \textsc{Relaxation}, & f_i = f_j,\ C_j \subset C_i, \\ \textsc{Substitution}, & f_i = f_j,\ |C_i| = |C_j|,\ \kappa(C_i) = \kappa(C_j),\ C_i \neq C_j. \end{cases}$$

Identical intents form no edge, and partially overlapping condition sets satisfying none of these cases remain unrelated rather than being approximately relabeled. For same-frame transitions, Δ_{ij} records added, removed, or substituted requirements, with substitutions retaining both old and new values; a Pivot records its destination intent. These deltas are later consumed directly by the shift engine and by GRIP's moving-target bookkeeping. A small WebShop example makes the representation concrete. One released root asks for wide-leg, regular-fit, tummy-control jeans in black, size 3X-Large, under $20. From this root, adding "faux fur" is a Refinement, removing the size requirement is a Relaxation, changing black to grey is a Substitution, and moving to a different jeans task is a Pivot. The root has 14 valid purchases; notably, removing the size condition still yields 14 purchases but changes their identities. Thus, whether an edge moves the ground truth is determined from its content rather than merely from answer-set size.

Retrieval-first Construction. For each validated root I_0, we first examine every other validated, parseable task in the same environment and retain those that already satisfy one of the four relations to I_0. There is no distance or similarity ranking at this stage. Retrieval is preferred because it

preserves native benchmark tasks; relation-specific synthesis is invoked only when an operator has no retrieved candidate or the branch budget remains unfilled. Pivots are always retrieval-only.

Synthesis is constrained by the adapter rather than free-form generation. A refinement uses the WITNESS hook to add a requirement supported by some but not all satisfying objects; in WebShop we inspect at most the first 200 satisfying products, require at least two witnesses, and order candidate properties by decreasing witness count and then lexicographically. A substitution uses values actually observed through DOMAINS: WebShop probes the first 300 products in the root cluster, retains at most the first 30 sorted values per option slot, and considers at most ten $10 price steps strictly inside the observed price range. A condition-level Relaxation removes one requirement but never the last. Retail additionally supports request-level Relaxation by removing an entire executable write request, because deleting only one required write argument would instead create an invalid action. Candidate-generation caps before ranking are 24, 12, 24, and 12 for REFINEMENT, RELAXATION, SUBSTITUTION, and PIVOT, respectively.

Verification and Branch Selection. Candidates outside the host benchmark's legal or scorable intent space are removed before execution. The remainder are canonicalized and deduplicated by (f, C), with retrieved copies preferred to synthesized duplicates and candidates identical to the root discarded. Every candidate is then independently executed, and empty or invalid ground truth is rejected.

Whether an edge changes executable ground truth, denoted `gt_moved`, is recorded but is not required for individual candidate admission. Requiring every transition to change the answer would introduce a shortcut in which detecting any intent change implies that the previous answer must be abandoned. Selection is deterministic: candidates are ranked by (i) retrieved before synthesized provenance, (ii) ground-truth-moving before answer-preserving transitions, (iii) the fixed relation order REFINEMENT, RELAXATION, SUBSTITUTION, PIVOT, (iv) decreasing ground-truth movement, measured by Jaccard distance for set-valued outcomes and a binary changed/not-changed value for computed outcomes, and (v) the candidate ground-truth content hash as a stable tie-break.

The default branch budget is eight, yielding two slots per relation. A relation shortfall is never backfilled from another relation, because doing so would silently alter the intended relation balance. If the branch budget is not divisible by four, the implementation instead warns and falls back to operator-ordered round-robin selection. WebShop uses `require_full_quota=true`, so any root unable to fill all four quotas is discarded; the τ^2 domains use declared relaxed configurations that retain structurally valid shortfalls. Every retained graph must nevertheless contain at least one ground-truth-moving branch (`min_gt_moved_edges`=1), ensuring that the graph contains an executable opportunity for adaptation to matter.

Stored and Runtime Graphs. The serialized artifact is intentionally a depth-one star containing the root and its selected branches. At episode initialization, every ordered pair of stored intents is reclassified using the same relation function. Valid pairs become derived cross-links, and both Δ_{ij} and `gt_moved` are recomputed for each new edge. This produces a denser runtime graph without generating or admitting any new intent. Node revisits, including return to the initial intent, are disabled by default, preventing cycles through already superseded goals.

Pairwise closure naturally creates many Pivot edges because any two differently framed intents can be connected. The runtime shift mechanism therefore samples a relation first and a destination within that relation second, rather than sampling uniformly over outgoing edges and allowing graph degree to determine the effective shift distribution; Appendix B.5 specifies this process.

Finally, intent identity is content-based. Each intent identifier hashes the adapter and its version, environment, frame, and canonicalized sorted condition set; graph identity additionally includes the generation configuration. Canonicalization normalizes integral-valued floats and rejects non-finite values. Runtime choices such as the evaluated model, persona, worker count, or storage path are excluded, so changing the evaluation regime does not create a different benchmark artifact.

Together, these rules turn each validated root into a finite collection of executable alternative goals with exact relation semantics and deterministic selection. Table 4 reports the resulting graph and sample counts; the same validated graphs are retained when constructing the misaligned requests described next.

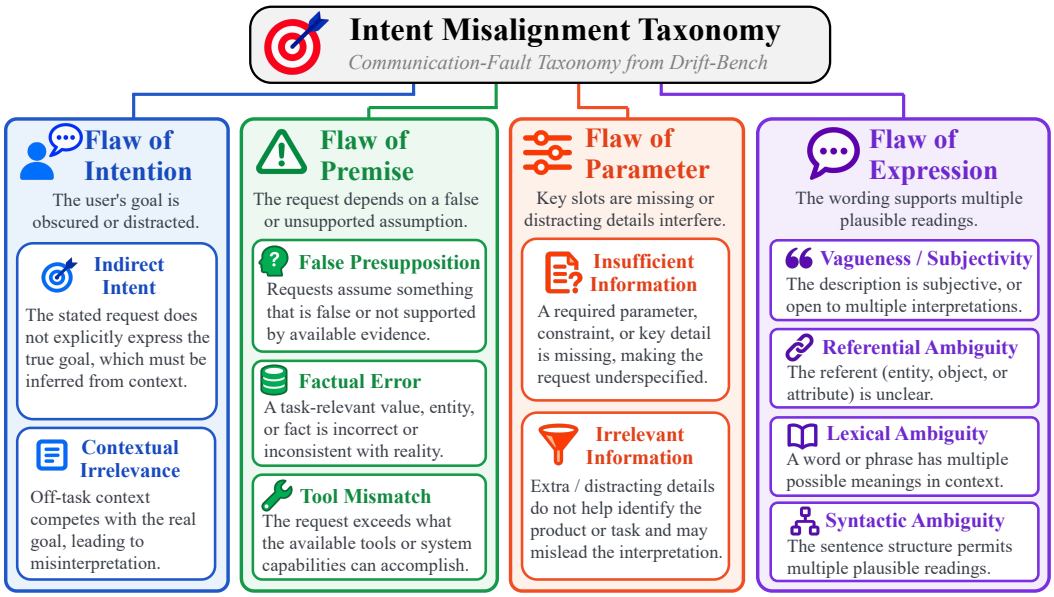

Figure 8: **Communication-fault taxonomy for interactive intent misalignment.** Drift-Bench++ organizes communication failures into four families: *flaws of intention, premise, parameter,* and *expression*, which are further instantiated as eleven operational fault types used in verified query perturbation. The hierarchy highlights where cooperative breakdown occurs, ranging from obscured goals and false assumptions to missing task parameters and ambiguous language.

B.3 Verified Query Perturbation

Section 3.1 introduces the three-step verifier; here we specify its released implementation. For each validated root intent I_0, we attempt every applicable active fault and generate a query Q while keeping I_0 fixed. The released corpus uses *generate-then-verify*: unlike the legacy mask-first ablation, the requested fault guides generation but does not determine admission.

Fault Applicability And Generation. We inherit the eleven-type taxonomy of Drift-Bench (Bao et al., 2026), as shown in Figure 8. Applicability is checked before generation: factual error and false presupposition require at least one condition; insufficient information and the four vagueness/ambiguity types require at least two and must retain one true condition for single-fault construction. Vagueness additionally requires an ordered slot, identified from its slot type, comparison operator, numeric value, or adapter hook; vagueness cannot target a slot whose name reveals its value. Inapplicable cells are omitted.

The writer is `doubao-seed-2-1-pro-260628` at temperature 0.7 with thinking disabled. It receives the true conditions, fault instruction, and coarse domain context, but not target identifiers, product names, or full category paths. Queries are capped at 130 words, or 220 for contextual irrelevance. Each graph–fault cell receives four attempts, increased to eight for factual error, lexical ambiguity, and syntactic ambiguity. Retries are fresh generations with no verifier feedback or template fallback; exhausted cells are skipped.

Three-Step Verifier. The verifier implements a three step process as follows:

$$(I_0, t) \xrightarrow{\text{Generate}} Q \xrightarrow{\text{Recover}} \delta \xrightarrow{\text{Apply}} I_Q \xrightarrow{\mathcal{O}} (Y_0, Y_Q),$$

where t is the requested fault, δ the recovered semantic effect, I_Q the literal intent expressed by Q, $\mathcal{O}$ the host-native oracle, and Y_0, Y_Q the outcomes of I_0, I_Q.

LLM Judge Agreement. Two different model families, `doubao-seed-2-1-pro-260628` and `deepseek-v4-flash-260425`, independently inspect (Q, I_0) at temperature zero. Factual error and insufficient information use a generic structured diff; the other active faults use targeted questionnaires because a generic form produced unstable distinctions among neighboring ambiguity classes. Extracted slots are canonicalized by exact name, carried value, then name/value containment. Agreement is imposed on semantic structure rather than literal extractor equality: both

judges must identify the same fault channel and canonical substituted, withheld, or unresolved slots. Targeted faults require the same target slot; quoted extra or off-topic spans must overlap on at least half of the shorter span's content words. We tolerate wording-level differences when the same slot is unresolved, but either judge finding a supposedly hidden value explicitly stated rejects the sample.

Perturbation Fidelity Check. The recovered structure is checked directly against Q: retained and substituted values must be present, hidden values absent, and internal condition syntax, meta-commentary, protected identifiers, and malformed output are rejected. Very short values use polarity-specific defaults rather than unreliable substring matching. Contextual irrelevance additionally requires an off-topic span of at least 25 words, no shorter than the embedded request and containing no true requirement value; indirect intent forbids direct-request phrases while preserving all true requirement values.

Executable Effect Verification. We reconstruct and execute the literal reading,

$$I_Q = \mathrm{Apply}(I_0, \delta), \qquad Y_0 = \mathcal{O}(I_0), \quad Y_Q = \mathcal{O}(I_Q),$$

and admit only the expected executable signature:

$$\begin{aligned}
\textsc{Substituted, Presupposed} &: Y_Q \neq Y_0, \\
\textsc{Extras} &: Y_Q \subsetneq Y_0, \\
\textsc{Withheld/Marked} &: Y_Q \supsetneq Y_0 \text{ when extensional, otherwise } Y_Q \neq Y_0, \\
\textsc{Ambiguous} &: \exists\, Y^{(1)} \neq Y^{(2)}, \text{ with one } = Y_0, \\
\textsc{Noise/Oblique} &: Y_Q = Y_0.
\end{aligned}$$

For EXTRAS, strict narrowing may reach the empty set; substitutions need only change the answer, not make it infeasible. False presupposition additionally requires the true intent to refute the assumed premise and its fallback to lead elsewhere. The released verifier has no NOT-CHECKABLE admission path: failure to establish the required signature is rejection. For extensional outcomes, $|Y_0|$ and $|Y_Q|$ are retained as a severity signal.

For intuition, one WebShop root asks for regular-fit, tummy-control, wide-leg black jeans in size 3X-Large under \$20, with $|Y_0| = 14$. An insufficient-information query replaces the \$20 ceiling with "options that fit my budget." Both judges recover the price slot as withheld, the fidelity check confirms that the exact ceiling disappeared while other requirements remain, and execution yields $|Y_Q| = 25$; the omission is therefore both linguistically and executably verified.

Composite Perturbations. For complexity $k > 1$, faults are drawn uniformly without replacement and redrawn only when structurally incompatible. Each condition-consuming fault receives a distinct eligible slot; at most one vagueness/ambiguity fault is allowed. False presupposition cannot co-occur with irrelevant information; for $k \geq 3$, it is also separated from factual error, while indirect intent is separated from false presupposition and irrelevant information because these pairs collapse into indistinguishable channels. The single-fault retained-condition floor is waived, but other applicability rules remain.

Slots are assigned deterministically, most-constrained first: vagueness/ambiguity, factual error, then insufficient information; no slot is reused. WebShop factual errors prefer attributes and price over option values because native option matching often makes option substitutions inconsequential. Faults are rendered jointly; for $k \geq 3$ with contextual irrelevance or indirect intent, content faults are rendered first and the whole-message fault second. Composite cells receive five attempts and a cap of $130 + 25(k-1)$ words, plus 90 for contextual irrelevance. Verification remains component-wise: assigned channels cannot overlap, untouched conditions must remain correct, unplanned additions are rejected, and every component must independently satisfy its executable signature.

Exposure Control. Only Q is agent-visible. The true intent, recovered edit, literal intent, oracle outcomes, verifier state, and future shifts remain private. Perturbations are generated before persona assignment, so all personas replay the same verified query; failed cells are skipped rather than replaced by easier faults.

B.4 PATIENCE-BOUNDED INTERACTION

Section 3 introduces patience as a resource that constrains interaction; here we specify its runtime semantics. For persona ρ, the real-valued budget begins at $b_0 = Bm_\rho$, where $B = 10$ and m_ρ is

given in Table 5, and evolves as

$$b_{t+1} = b_t - c(a_t), \qquad c(\text{ask}, \text{reject}, \text{malformed}, \text{environment}) = (2, 4, 1, 0).$$

Thus, a rejected proposal costs twice a clarification question, discouraging premature commitment, while environment interaction is free because it does not directly burden the user. Accepted proposals terminate without charge, and there is no separate question cap: affordability alone limits interaction. At each turn the agent may execute a host-native action, ask the user, or propose an answer. Questions are handled by the simulator in Appendix B.6, while proposals are converted by the adapter and judged by the host-native verifier against the current intent. An unparseable agent emission receives one format retry with the parser error; if this also fails, methods with a user channel fall back to a generic clarification, while No Ask issues a deterministic search constructed only from visible user words. If a malformed action nevertheless reaches the episode loop, the user requests rephrasing and $c_{\text{malformed}}$ is charged after that exchange.

Affordability And Termination. A user-facing action is affordable only when its cost does not exceed the remaining budget. An unaffordable question is not answered or charged and invokes forced final submission; an unaffordable proposal is instead treated directly as the final proposal and adjudicated once. Budget exhaustion and the 40-turn limit also invoke forced submission. In this routine, the agent must provide its single best candidate in the host format; a non-submission receives one form-only repair, after which the benchmark adjudicates whatever candidate the agent provides. The harness never chooses or improves its content, and a forced final may still succeed. Remaining patience is always tracked by the runtime, while whether its numeric value is shown to the agent is controlled by the method interface. Patience also determines when scheduled intent shifts become due. Due status is therefore updated after every patience payment and again before submission; the resulting event ordering is specified next.

B.5 Interaction-Conditioned Intent Shifting

Section 3 introduces scheduled and agent-triggered intent shifts. Both operate only over executable transitions validated in Appendix B.2, and all realized shifts are silent: no announcement is emitted, while subsequent replies and adjudication use the updated intent.

Scheduled Opportunities. Persona ρ defines normalized placements $\Gamma_\rho = \{\gamma_1, \ldots, \gamma_K\}$ from Table 5, converted to descending patience thresholds

$$\tau_k = \gamma_k b_0, \qquad \text{due}(k, t) \iff b_t \le \tau_k.$$

Using at-or-below rather than strict crossing ensures that a placement at the initial budget is equally available to agents that ask immediately and agents that propose immediately. A placement never reached creates no opportunity. Each placement also receives an independent pre-drawn fate

$$z_k^{\text{fire}} \sim \text{Bernoulli}(0.6),$$

seeded from sample identity so paired methods face identical scheduled opportunities. Crossing consumes the placement regardless of the draw; failed draws are logged as skipped opportunities, not non-moving shifts. The 0.6 probability is the released operating regime; setting it to zero disables both shift channels for the no-shift ablation.

Agent-Triggered Reconsideration. Only clarification questions can trigger reconsideration. At current intent I_t, each legal non-pivot edge contributes topic words from the changed requirement's semantic slot name and current-side value; adapter-specific matching words may override this representation. Identifier-like vocabulary tokens containing digits or fewer than three characters are removed. Matching a question q_t against these vocabularies identifies the topic-associated relation types that are eligible for reconsideration, yielding a set $\mathcal{R}(q_t, I_t)$. Here, "along that topic" refers to reconsideration within a relation type licensed by the question, rather than selection of a particular destination value. The eventual destination is therefore sampled from valid alternatives under the selected relation, preventing the agent from steering the user toward a specific target intent. If this set is nonempty,

$$z_t^{\text{agent}} \sim \text{Bernoulli}(s_\rho), \qquad r \sim \text{Unif}\big(\mathcal{R}(q_t, I_t)\big), \qquad I_{t+1} \sim \text{Unif}\big(E_r(I_t)\big),$$

where s_ρ is persona suggestibility and unavailable relations are removed. Pivots cannot be question-triggered because a wholesale task change is treated as user initiative.

A successful triggered shift consumes the *lowest* remaining patience threshold, i.e., the opportunity that would otherwise occur latest; its scheduled fire coin is discarded because suggestibility already supplied the gate. Together with an independent hard cap, this guarantees

$$N_{\text{shift}} \leq |\Gamma_\rho|,$$

so asking can advance reconsideration but cannot increase maximum shift exposure.

Scheduled Destination Sampling. Scheduled shifts first sample a relation from

$$\pi = (0.25, 0.25, 0.30, 0.20)$$

for refinement, relaxation, substitution, and pivot. Unavailable relations are removed and π is renormalized until a feasible relation is drawn; destinations are then uniform over its legal outgoing edges,

$$P(I_{t+1} = j \mid I_t = i, r) = \frac{\mathbf{1}[(i,j) \in E_r]}{|E_r(i)|}.$$

Sampling relation first prevents graph degree, especially abundant pivot edges, from defining the shift distribution. Relations and destinations are deterministically sorted before seeded sampling. Root-level relation availability is logged, while each realized shift records its relation, source, destination, and destination conditions, allowing graph-induced changes in the realized mixture to remain auditable. Shift eligibility after irreversible environment mutation is host-specific. WebShop forbids further shifts after a purchase-shaped action. The released τ^2 Retail and Airline runs allow shifts after write actions and evaluate the current goal using golden-action completion, because forbidding post-write shifts would structurally eliminate much of the shift setting in these domains.

Temporal Ordering And Replay. Question and proposal turns deliberately follow different orders:

$$I_t \to \text{reply under } I_t \to \text{possible shift}, \qquad I_t \to \text{all due shifts} \to I'_t \to \text{Verify}(y_t, I'_t).$$

After a question reply and patience payment, agent-triggered reconsideration is checked first. If it fires, any scheduled due opportunity waits; otherwise at most one due scheduled shift fires. Proposal turns, including forced and unaffordable final submissions, instead flush all due opportunities before adjudication and may therefore traverse multiple graph edges. This makes replies consistent with preferences already held while ensuring commitments are judged against the intent current when they are made. For a failed final submission following an answer-moving shift, the same candidate is additionally executed against the intent superseded by the *last* such shift, providing the executable certificate used by Staleness in Appendix B.7. Finally, four independent seeded streams derived from the sample identity govern general user behavior, shift destinations, agent-trigger gates, and scheduled fire draws. Separating destination randomness prevents additional interaction from changing future shift identities merely by consuming random draws.

For personas with a 100% first placement, the first scheduled shift opportunity is intentionally made eligible at the earliest interaction point, rather than delayed to a fixed later turn. This design ensures that even short episodes can exercise adaptation to evolving intent, while personas with additional placements retain later opportunities for repeated shifts; each scheduled opportunity remains subject to the same probabilistic firing rule described above.

B.6 Persona-Conditioned Behavior

Persona conditioning introduces controlled variation in how users disclose information, tolerate interaction, provide feedback, and reconsider their intent. To keep these effects auditable, benchmark truth is controlled entirely by a state machine: it maintains the current executable intent, disclosed requirements, patience, persona decisions, shift state, and proposal adjudication. The LLM is restricted to two language functions: mapping a question to candidate requirements using opaque identifiers and generic semantic types, and verbalizing only information authorized by the state machine. It never sees hidden values during mapping, changes task truth, or decides whether a proposal succeeds.

Following GDMS (Scott & Bruce, 1995) and Drift-Bench (Bao et al., 2026), the five personas instantiate the seven parameters in Table 5. *Disclosure* controls how much information is returned per valid question (*slots/answer*), the probability of additionally revealing one unasked requirement (*volunteer*), and the probability of withholding an otherwise valid answer (*refuse*). *Patience & feedback* specifies the multiplier on the base patience budget $B = 10$ (*patience* $\times$) and the probability

	Disclosure			Patience & Feedback		Silent Intent Shifts	
Persona	**Slots/Answer**	**Volunteer**	**Refuse**	**Patience ×**	**Hint**	**Placements**	**Suggestibility**
Rational	1	10%	0%	1.0	30%	100%	20%
Dependent	2	35%	0%	1.2	60%	66%, 33%	80%
Avoidant	1	0%	50%	0.8	10%	100%	10%
Intuitive	1	20%	10%	0.9	30%	100%, 50%	50%
Spontaneous	1	30%	5%	0.7	40%	100%, 50%	60%

Table 5: Persona configurations. *Disclosure*: slots revealed per answered question, the probability of volunteering an unasked requirement, and the probability of refusing an otherwise-valid answer. *Patience & feedback*: the multiplier scales the base patience budget $B{=}10$; *hint* is the probability that a rejection names what is wrong rather than a bare refusal. *Silent intent shifts*: placements position each scheduled shift as a percentage of the persona's own initial budget, crossed as patience falls to or below that point; each crossing fires with probability 60% (the scheduled fire coin), and an on-topic clarification question can instead fire a shift early with the persona's suggestibility, consuming the last remaining placement so the shift count never grows (Appendix B.5).

that rejection feedback includes a high-level explanation of what is wrong (*hint*). *Silent intent shifts* specifies scheduled shift positions as fractions of the persona's own initial budget (*placements*) and the probability that an on-topic clarification triggers reconsideration early (*suggestibility*); the exact shift mechanics are given in Appendix B.5. These parameters define behavioral stress-test conditions rather than estimates of population prevalence, and personas affect only evaluation behavior, never query perturbation or executable truth.

Disclosure And Refusal. When a question is successfully mapped, the simulator reveals at most the persona's *slots/answer* allowance. An otherwise valid answer is withheld with probability *refuse*, subject to two solvability safeguards: asking about the same requirement twice consecutively, or four times in total, forces disclosure. A refusal reveals no requirement value and is phrased only as failure to answer; re-asking an already disclosed requirement yields a reminder. Unmappable questions still consume patience and receive a confusion response, but three consecutive unmappable questions force one remaining requirement to be volunteered. Independently, the persona's *volunteer* probability may append one unasked requirement to an otherwise valid reply.

Grounded Disclosure And Feedback. Because evaluation should reflect what the agent actually received rather than what the simulator intended to disclose, a passive observer checks each realized user utterance and credits a requirement only when its value is genuinely communicated. It separately records elicited and unsolicited disclosures. Rejected proposals are converted from the executable verifier's outcome into ordinary-language feedback with raw scores and identifier-like values removed, preventing repeated proposals from turning the verifier into an optimization oracle. With probability *hint*, the user additionally gives a high-level indication of what is wrong; any requirement actually revealed through this feedback is credited through the same grounded observer.

B.7 GRIP Evaluation Details

Section 3.3 introduces GRIP as a diagnostic evaluation of **G**rounding, **R**ole-realism, **I**nquiry, and **P**ersistence. Here we specify how each metric is computed. GRIP intentionally keeps these dimensions separate: the same final task outcome can arise from recovering the user's intent, correctly inferring it without asking, or remaining aligned only by chance, while a single aggregate score would obscure these distinctions.

Moving Target And Recovery State. Let I_t and b_t retain their meanings from Section 3. Let H_t denote the set of latent requirements associated with the current intent I_t that require recovery, and let $E_t \subseteq H_t$ denote those requirements whose true values have been validly recovered by the agent by turn t. The target evolves with every realized shift: refinement adds requirements, relaxation removes them, substitution replaces the changed requirement and invalidates its earlier recovery, and pivot re-anchors the target to the destination intent. Multi-hop shifts are applied in order, while information shared by the old and new intents remains valid.

Recovery credit reflects information that genuinely reaches the agent. Values communicated in answers or rejection feedback count, as do true values discovered from environment observations; values merely repeated from the agent's own earlier actions are excluded by an echo guard. Let T denote the turn of the final adjudicated commitment, and let S_e denote the executable reward of episode e at that commitment. $S_e \in [0, 1]$ is the native WebShop reward, with acceptance equal to 1, and is binary on τ^2.

Grounding (G). Grounding distinguishes whether good task performance follows from actually understanding the user's current intent.

Success. The basic question is simply how well the agent ultimately solves the task. We therefore average the executable reward of the final adjudicated submission:

$$\text{Success} = \mathbb{E}_e[S_e].$$

Because this reward comes from the host benchmark's executable verifier, no LLM judge determines task correctness.

Earned. Success alone cannot tell whether the agent first recovered everything the user had left implicit. Earned keeps only the score from episodes in which the complete final hidden target was recovered before commitment:

$$\text{Earned} = \mathbb{E}[S_e \, \mathbf{1}[\mathcal{H}_T \subseteq \mathcal{E}_T] \mid \mathcal{H}_T \neq \emptyset] \, .$$

Inferred. Conversely, an agent may correctly infer the answer without explicitly recovering every hidden requirement. This is still a valid capability, so Inferred records rather than penalizes such success:

$$\text{Inferred} = \mathbb{E}[S_e \, \mathbf{1}[\mathcal{H}_T \not\subseteq \mathcal{E}_T] \mid \mathcal{H}_T \neq \emptyset] \, .$$

Thus, Earned + Inferred equals the mean Success over episodes with a non-empty hidden target.

Role-Realism (R). Role-realism asks whether the declared personas create recognizable and stable behavioral differences. It evaluates fidelity to the assigned roles, not whether these five personas estimate the distribution of real users.

Identification. If persona conditioning is meaningful, a reader should be able to recognize which persona generated an interaction without seeing its label. Two judges from different model families independently receive only the first eight user turns together with five shuffled persona descriptions and predict the generating persona. Identification is the resulting five-way classification accuracy,

$$\text{Identification} = P(\hat{\rho} = \rho),$$

where ρ is the assigned persona and $\hat{\rho}$ the judge prediction; chance is 0.2. We report per-judge accuracy, inter-judge agreement, and pooled per-persona accuracy, using up to 50 transcripts per persona and evaluation cell.

Consistency. Identification alone could succeed even if behavior varies substantially across tasks. We therefore also ask whether two interactions look like the same persona. Judges receive the first six user turns from two task-different transcripts, with same-persona and different-persona pairs sampled equally, and predict SAME or DIFFERENT. We report balanced accuracy,

$$\text{Consistency} = \frac{1}{2}\left[P(\widehat{\text{Same}} \mid \text{Same}) + P(\widehat{\text{Different}} \mid \text{Different})\right],$$

whose chance level is 0.5. A judge whose minority prediction is below 10% is treated as degenerate and excluded; reported evaluations use 200 pairs per cell.

Inquiry (I). Inquiry separates three questions that raw interaction count cannot answer: whether the agent asks about the right information, whether it ultimately recovers all of it, and how much user patience this costs.

Aim. Asking frequently is not useful if questions concern information that is irrelevant or already known. For question q_j asked at turn t_j, let $M(q_j)$ denote the requirements to which the user simulator maps that question. We mark it as aimed when it touches at least one requirement that is currently hidden and not yet recovered:

$$A_j = \mathbf{1}\left[M(q_j) \cap (\mathcal{H}_{t_j} \setminus \mathcal{E}_{t_j}) \neq \emptyset\right].$$

For each episode that asks at least one question, Aim is the fraction of its questions with $A_j = 1$; the reported metric is the mean of these per-episode fractions. An unmapped question is unaimed, whereas a correctly targeted question remains aimed even if the persona refuses to answer. Episodes that never ask have undefined Aim rather than zero.

Recovery. A method may ask well-targeted questions yet stop before the full intent is known. Recovery therefore asks a stricter, task-level question: was *every* requirement still hidden under the final intent recovered before commitment?

$$\text{Recovery} = P(\mathcal{H}_T \subseteq \mathcal{E}_T \mid \mathcal{H}_T \neq \emptyset)\,.$$

This is binary per episode rather than the fraction of individual hidden slots recovered.

Patience. Two methods may recover the same intent while imposing very different interaction costs on the user. We therefore report the absolute patience remaining at the final commitment:

$$\text{Patience} = \mathbb{E}_e[b_T],$$

where b_T is the remaining persona-scaled patience budget. Higher values indicate that the agent reached its final decision with less interaction burden.

Persistence (P). Persistence evaluates what happens after the target itself changes. We call a shift *answer-moving* when its executable ground truth differs from the preceding intent.

Staleness. After a shift, a failed agent may simply be bad at the task, or it may still be pursuing the user's old goal. Staleness isolates the latter failure. For failed episodes containing an answer-moving shift and a recorded executable counterfactual, we ask whether the final submission would have been accepted under the intent superseded by the last answer-moving shift:

$$\text{Staleness} = P(\text{final submission accepted under superseded intent} \mid \text{failed, answer-moving shift, certificate})\,.$$

The counterfactual is produced by executable re-adjudication during the episode rather than inferred from dialogue.

Reaction. Even when an agent eventually adapts, some agents may take much longer than others. Let t_{shift} denote the turn of the last answer-moving shift and t_{acc} the first accepted proposal at or after that shift. Among shifted episodes that succeed,

$$\text{Reaction} = \mathbb{E}[t_{\text{acc}} - t_{\text{shift}} \mid S_e = 1,\ \text{answer-moving shift}]\,.$$

Lower values indicate faster resynchronization. Because Reaction only considers episodes that eventually succeed, it must be interpreted together with Post-shift Success.

Post-Shift Success. A method could react quickly on a small number of successful episodes while failing on most shifted tasks. We therefore also measure the unconditional outcome whenever the correct answer actually changes:

$$\text{PostShift} = \mathbb{E}[S_e \mid \text{at least one answer-moving shift}]\,.$$

Together, Staleness identifies failures caused by outdated intent, Reaction measures adaptation speed among successes, and Post-shift Success captures overall robustness to changing goals. The diagnostic implementation additionally records over-reaction to answer-preserving shifts, but it is not part of the primary GRIP report.

Computation And Reporting. Grounding, Inquiry, and Persistence are computed mechanically from stored episode trajectories. Role-realism is evaluated offline, and all raw judge outputs are retained so its scores remain reproducible. Each conditional metric is reported with its own enabling-event count; undefined quantities are shown as "—" rather than zero. Results are additionally stratified by fault kind, persona, and fault type, and GRIP never combines its dimensions into a single scalar.

C Additional Experiments

This appendix provides the methodological and numerical details omitted from Section 4. We first describe the seven published clarification baselines and three progressively stronger reference agents

used to probe different capabilities in interactive intent alignment. We then detail the experimental setup and controls that ensure paired and comparable evaluation across methods, backbones, personas, fault complexity, and intent shifting conditions. Finally, we report the complete results following the same three research questions as the main text. Unlike the main-text analysis, which highlights the metrics most relevant to each finding, the appendix retains all GRIP dimensions for every experimental condition.

C.1 Baselines and Reference Methods

C.1.1 Baselines

All methods operate through the same agent-environment interface. At each turn, the agent receives the original request, dialogue history, and observations produced by its own benchmark-native tool calls, and may continue acting, ask the user, or submit a proposal. Questions and proposals consume the patience budget defined in Appendix B.4, and all proposals are evaluated by the same benchmark-native executable verifier. No method observes the latent intent graph, perturbation type, unannounced intent shifts, simulator-private requirements, or GRIP scores during interaction. Thus, differences between methods arise from how they diagnose uncertainty, acquire information, maintain interaction state, and determine when the available evidence is sufficient for commitment.

No Ask. *No Ask* provides the non-interactive control. The agent may reason over the request and gather information through ordinary tool use, but clarification is removed from its action space. Any remaining uncertainty must therefore be resolved internally before proposing an answer. This baseline measures how much of the task can be solved from the initial communication and environment alone, providing the reference point for quantifying the value of interaction.

Just Ask: Free-form Clarification. *Just Ask* represents the most basic clarification strategy: when the available request appears insufficient for confident action, the model asks a free-form follow-up question and conditions subsequent behavior on the user's reply (Li et al., 2025). It introduces no explicit representation of ambiguity, candidate intents, or question utility. We include it as the canonical *elicitation-based* strategy, measuring how far generic conversational clarification alone can recover missing intent.

AT-CoT: Diagnosis-guided Clarification. *AT-CoT* first identifies the type of ambiguity underlying the request and then formulates a question targeted at that diagnosis (Tang et al., 2025). It therefore represents a *diagnose-then-ask* strategy, where an explicit intermediate characterization of uncertainty guides subsequent elicitation. This baseline tests whether localizing the source of ambiguity produces more focused clarification than unconstrained questioning.

Belief Graph: Structured Belief Tracking. The *Belief Graph* decomposes the candidate intent into interconnected components whose uncertainty and decision relevance are updated as new information arrives (Hahn et al., 2025). Clarification is directed toward the component whose resolution is expected to matter most for the downstream decision. It represents *structured belief tracking*, testing whether an explicit representation of uncertain intent components improves question selection over treating the request as a single undifferentiated uncertainty.

ProductAgent: Discriminative Questioning. *ProductAgent* selects clarification questions from the current set of viable candidates (Ye et al., 2025). An attribute becomes useful to ask about when its value distinguishes among otherwise plausible alternatives. This represents *discriminative clarification*, where questions are chosen for their ability to shrink the actionable solution space rather than solely because the language model reports uncertainty.

SAGE: Parameter-level Uncertainty. *SAGE* maintains structured uncertainty over task parameters and asks whether resolving an uncertain parameter is worth the additional interaction cost (Suri et al., 2026). It represents *parameter-centric clarification*: uncertainty is attached to concrete task variables, allowing the agent to distinguish requirements that are sufficiently determined from those that still warrant user input.

BED-LLM: Information-seeking Clarification. *BED-LLM* formulates clarification as Bayesian experimental design (Choudhury et al., 2026). It maintains uncertainty over candidate interpretations and selects questions according to their expected informativeness for the downstream decision. We

include it as a representative *information-gain* strategy, testing whether explicitly optimizing the expected reduction in uncertainty improves the use of a limited interaction budget.

CTA: Cost-aware Ask-or-Act Decisions. *CTA* calibrates uncertainty and compares the expected cost of acting under that uncertainty with the cost of acquiring additional information (Ding et al., 2026). It represents the classical *ask-or-act* trade-off: clarification is warranted only when its expected benefit exceeds its interaction cost. This makes CTA particularly relevant to our patience-bounded setting, where asking is not a free operation.

For methods originally formulated for a single clarification round, we apply the same decision rule again after each newly observed user response. This minimal protocol adaptation permits evaluation under a multi-turn interaction budget without introducing capabilities absent from the original method, such as explicit long-horizon memory, pre-commitment verification, or cross-episode strategy updates.

C.1.2 Reference Agents.

The published baselines discussed above are methods developed specifically for user-facing clarification. In parallel, broader research on LLM agents has investigated persistent memory, self-verification, and experience-driven adaptation as mechanisms for improving long-horizon behavior. However, interactive clarification under imperfect and evolving user intent remains a relatively new setting, and, to our knowledge, these capabilities have not been systematically evaluated as clarification mechanisms under such conditions. We therefore adapt their core principles into three lightweight reference agents to examine their potential within Drift-Bench++. **These agents should be viewed as diagnostic references rather than methodological contributions of Drift-Bench++.** Each reduces a broader line of agent research to a simple benchmark-compatible mechanism while preserving its central capability. The agents are progressively layered so that adjacent comparisons isolate three increasingly rich capabilities: maintaining a revisable representation of user intent, verifying evidence before commitment, and adapting clarification strategies from prior experience.

Memory Ledger: Revisable Intent Tracking. Persistent memory has become a central mechanism for enabling LLM agents to maintain useful state across extended interactions. Early systems such as Generative Agents retrieve past experiences and higher-level reflections to inform subsequent behavior, while MemoryBank maintains and updates long-term conversational memories across sustained interactions (Park et al., 2023; Zhong et al., 2024). More recent work moves beyond static storage toward active memory organization and revision. A-MEM dynamically organizes and evolves interconnected memories as new information arrives, while Memory-R1 explicitly learns memory-management operations such as adding, updating, and deleting stored information (Xu et al., 2025; Yan et al., 2026). Collectively, this line of work suggests that effective long-horizon memory requires not only retaining past information, but also revising what remains valid as new evidence becomes available.

This requirement is particularly relevant to evolving user intent: information recovered early in an interaction may remain useful, become incomplete, or be explicitly superseded by a later correction. We therefore ask whether an explicit revisable state can help an agent preserve recovered intent while removing information that has become stale. To instantiate this capability without introducing a specialized memory architecture, *Memory Ledger* maintains a structured state at each interaction step t,

$$\mathcal{M}_t = (\widehat{C}_t, U_t, X_t),$$

where $\mathcal{M} * t$ denotes the complete ledger state, $\widehat{C}_t$ contains requirement–value bindings currently supported by user messages or environment observations, U_t contains requirements that remain unresolved, and X_t records previously supported bindings that have been invalidated by subsequent evidence. After receiving the next user message $u * t + 1$ and environment observation o_{t+1}, the ledger is updated according to

$$\mathcal{M}_{t+1} = \Phi(\mathcal{M}_t, u_{t+1}, o_{t+1}),$$

where Φ denotes the ledger-update procedure. In implementation, the updater is given the current ledger together with the newly observed interaction evidence and is instructed to preserve supported bindings, add newly established requirements, mark unresolved information explicitly, and revise bindings when later evidence contradicts an earlier value. Such updates are conservative: contradictory evidence replaces rather than coexists with a superseded value, with the invalidated binding retained in X_t. The current ledger is then provided to the agent before each subsequent decision. Memory Ledger therefore tests whether explicitly maintaining and revising interaction state helps preserve recovered requirements while preventing obsolete information from continuing to influence future actions.

Memory+Verification: Evidence-Grounded Commitment. Maintaining the correct requirements does not by itself ensure that an agent's eventual proposal is consistent with them. A separate line of work therefore introduces explicit verification between generation and commitment. Self-Refine iteratively critiques and revises generated outputs using model-produced feedback, while CRITIC grounds correction in feedback obtained from external tools (Madaan et al., 2023; Gou et al., 2024). More recent approaches make verification itself a central capability: Self-Verify trains models to assess the correctness of their own generations, while work on Universal Verifiers studies how explicit evidence and structured criteria can determine whether agent trajectories have satisfied their objectives (Zhang et al., 2025; Rosset et al., 2026). Although these approaches instantiate verification differently, they share the principle that generating a candidate and establishing sufficient evidence to accept it should be treated as distinct stages.

This distinction motivates a complementary question in our setting: once an agent has recovered and remembered the user's requirements, should those requirements explicitly constrain when it is willing to commit? *Memory+Verification* therefore retains the Memory Ledger and adds a lightweight verification stage before a candidate proposal is returned to the user. Let y denote the current candidate proposal, let $c \in \widehat{C}_t$ denote one remembered requirement, and let $o_{0:t} = (o_0, \ldots, o_t)$ denote the observable environment evidence accumulated through interaction step t. We define $v(y, c, o_{0:t}) \in 0, 1$ as the verification decision indicating whether the available evidence supports candidate y with respect to requirement c. The overall verification decision is then

$$V_t(y) = \bigwedge_{c \in \widehat{C}_t} v(y, c, o_{0:t}),$$

where $V_t(y) = 1$ only when every currently supported requirement is verified. Operationally, the verifier receives the candidate, the current ledger, and the interaction evidence available to the agent, and checks whether the proposal is consistent with each remembered requirement. The agent commits only when these supported requirements are verified and no unresolved requirement in U_t could materially invalidate the proposal. Otherwise, it continues interacting through additional tool use or clarification while the interaction budget permits. Importantly, verification uses only information available through the ordinary interaction and never queries latent user intent or the benchmark verifier. Relative to Memory Ledger, this reference agent therefore isolates whether remembered intent becomes more useful when it also serves as an explicit constraint on final commitment.

Self-Evolving Playbook: Experience-Driven Clarification. Memory and verification improve how an agent uses information within an episode, but they leave its clarification strategy largely unchanged across episodes. A growing line of work instead studies whether agents can improve from previous interactions without updating model parameters. Reflexion stores verbal feedback from previous trials, while ExpeL extracts reusable insights from collections of prior experiences (Shinn et al., 2023; Zhao et al., 2024). More recent approaches make such adaptation increasingly systematic. Contextual Experience Replay (CER) accumulates and synthesizes prior trajectories into a dynamic experience memory for subsequent tasks (Liu et al., 2025); Agentic Context Engineering (ACE) treats context as an evolving playbook that incrementally accumulates and refines strategies from execution feedback (Zhang et al., 2026); and GEPA uses natural-language reflection over execution and evaluation traces to evolve reusable textual strategies (Agrawal et al., 2026). Together, these methods suggest that interaction experience itself can serve as a source of reusable procedural knowledge.

This capability is particularly relevant to clarification because effective questions are often reusable across related interaction patterns even when the underlying requirement values differ. Experience

from earlier episodes may therefore help an agent recognize which ambiguities are worth resolving, which questions tend to elicit useful information, and what evidence should be collected before commitment. We accordingly ask whether clarification behavior can improve through experience accumulated across previous episodes. The *Self-Evolving Playbook* augments Memory+Verification with a run-local strategy memory. Let e index episodes within an independent run, let L_e denote the playbook available at the beginning of episode e, and let τ_e denote the complete interaction trajectory generated during that episode. We first apply an observation filter $\mathrm{Obs}(\tau_e)$ that removes benchmark-private information from the trajectory, and then update the playbook according to

$$L_{e+1} = \Psi(L_e, \mathrm{Obs}(\tau_e)), \qquad L_1 = \varnothing,$$

where Ψ denotes the playbook-update procedure and $L_1 = \varnothing$ indicates that each independent run begins with an empty playbook. In implementation, the updater receives the previous playbook together with the observable trajectory from the completed episode and summarizes reusable guidance about which clarification questions were informative, what information they elicited, what evidence was useful before commitment, and what observable outcome followed. The updater is instructed to retain general interaction strategies rather than episode-specific requirement values, so that the resulting memory captures transferable clarification behavior rather than answers to individual tasks.

The updated playbook is provided as additional context in subsequent episodes alongside the same Memory Ledger and verification mechanism. This allows the agent to reuse previously discovered clarification strategies while continuing to reconstruct the requirements of each new episode from its own interaction evidence. The playbook is reset across independent runs, benchmarks, personas, and random seeds, and it never observes latent intents, perturbation labels, future shift destinations, or GRIP scores. Consequently, any cross-episode adaptation must arise from information available through ordinary interaction rather than persistent access to benchmark answers or evaluation signals. This reference agent therefore probes whether current models can extract, retain, and reuse general clarification strategies from their own prior interactions instead of approaching every episode with an unchanged interaction policy.

C.2 Experimental Setup and Controls

Unless otherwise specified, experiments use DeepSeek-V4-Pro as the agent backbone, the Rational persona, single-fault requests, and intent shifting enabled. Each reported result is the mean and standard deviation over three seeded runs. To make method comparisons directly paired, evaluation is defined over a shared manifest stratified by perturbation type. Each manifest row fixes the verified request, intent graph, host-environment state, persona condition, and episode seed, and every method is evaluated on the same rows. Differences between methods therefore reflect behavior on matched benchmark instances rather than variation in sampled tasks.

We similarly isolate the factor varied in each experimental analysis. Persona is introduced only at evaluation time: the verified perturbation corpus is constructed once and replayed under different persona configurations. The backbone sweep changes only the evaluated agent model. For the complexity sweep, k denotes the number of distinct fault types composed within a single request; target conditions are assigned before generation and every component must independently satisfy the verification procedure in Appendix B.3. For the shift ablation, disabling intent shifting removes both scheduled and interaction-conditioned transitions while preserving the initial request, persona, patience budget, tools, and executable verifier. These controls allow each analysis to attribute performance changes to the factor under study while leaving the remaining interaction semantics unchanged.

We additionally control sources of run-to-run variation that arise in large-scale agent evaluation. Independent runs use explicit seed offsets and separate response caches, since hosted model APIs do not guarantee deterministic sampling even under matched inputs. We record the exact identity of every model used and disable automatic provider fallback to prevent temporary service failures from silently changing the evaluated backbone. Mutable host environments use collision-free session identifiers and deterministic worker assignment. Transient environment failures are retried with bounded backoff; episodes that exhaust these retries are recorded as unscored rather than as agent failures. Results produced under different shift rules, patience costs, or model configurations are never pooled.

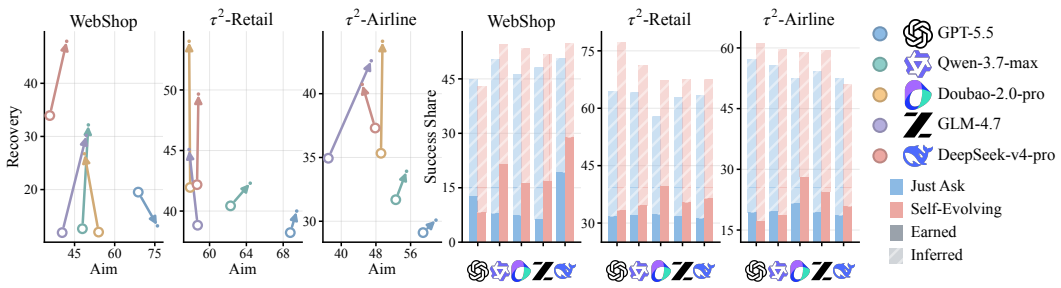

Figure 9: **Left:** Aim-Recovery changes from Just Ask (circle) to Self-Evolving (arrowhead) across five backbones. **Right:** successful hidden-intent episodes decomposed into Earned (solid) and Inferred (hatched) outcomes. Together, the panels show that stronger interaction primarily improves intent recovery and often shifts success toward explicitly recovered requirements, while Aim alone does not guarantee better recovery. Results are averaged over three seeded runs.

C.3 RQ1: Capability Across Methods and Backbones

Following the RQ1 analysis in Section 4, Table 6 expands Figure 3 from Success to the complete GRIP profile for No Ask, Just Ask, and Self-Evolving across all five backbones. Figure 9 further visualizes how stronger interaction changes question targeting, intent recovery, and the grounding of successful outcomes. The main-text conclusion remains unchanged: clarification improves performance broadly, and the stronger reference strategy leads in 14 of 15 backbone–benchmark settings. The additional results reveal several patterns behind this advantage that are obscured by end-to-end Success alone.

Interaction capability is not reducible to backbone strength. Just Ask improves over No Ask in every backbone–benchmark pair, showing that the value of clarification is not specific to a particular model family. More importantly, backbone rankings change once stronger interaction is introduced. Qwen3.7-max achieves the highest Self-Evolving Success on WebShop, whereas GPT-5.5 becomes strongest on both τ^2 domains, even though neither ordering follows directly from the corresponding No Ask results. Performance under incomplete and evolving intent therefore depends on the interaction strategy jointly with the underlying model. A backbone that is stronger at solving the initially communicated task is not necessarily stronger at recovering the task the user actually wants. Interactive intent alignment should consequently be treated as a capability distinct from ordinary base-model task performance.

Better clarification requires more than asking relevant questions. Across the three benchmarks, Self-Evolving attains the highest observed Earned and Recovery scores, showing that its strongest results are associated with more complete recovery of hidden requirements rather than successful inference alone. Figure 9 makes the distinction especially clear: transitions from Just Ask to Self-Evolving move more consistently along the Recovery dimension than along Aim. On the two τ^2 domains, several backbones gain substantially in Recovery even when Aim changes only modestly or decreases. Stronger clarification therefore improves not merely whether the agent recognizes uncertainty, but whether it can convert interaction into usable information about the user's current intent.

Aim alone is consequently a poor proxy for successful intent recovery. The clearest example is GPT-5.5 on WebShop. Self-Evolving increases Aim from 68.81 ± 0.48 to 75.99 ± 3.30, yet Recovery decreases from 19.52 ± 2.87 to 13.17 ± 1.67, and Success also falls below Just Ask. The corresponding trajectory in Figure 9 moves strongly toward higher Aim while moving downward in Recovery. The agent is asking about relevant uncertainty, but those questions fail to recover enough of the hidden target to improve the final outcome. This separation motivates treating *recognizing what should be clarified* and *actually recovering the missing intent* as distinct capabilities in GRIP.

Stronger interaction also changes how success is obtained. The right panels of Figure 9 decompose successful hidden-intent episodes into Earned and Inferred outcomes. Self-Evolving often increases the Earned component, most consistently on Retail, showing that its advantage is not limited to producing more successful trajectories. It also changes the basis of those successes toward outcomes supported by explicitly recovered requirements. At the same time, this shift is not univer-

Method	Backbone	Grounding (G)			Inquiry (I)			Persistence (P)		
		Success	Earned	Inferred	Aim	Recovery	Patience	Staleness	Reaction	PostShift
WebShop										
No Ask	Gpt-5.5	43.47±1.48	n/a	n/a	n/a	n/a	3.69±0.11	5.53±1.15	3.06±1.03	35.98±1.92
	Qwen3.7-max	51.00±1.07	n/a	n/a	n/a	n/a	3.90±0.09	10.07±0.23	3.51±0.22	43.11±0.85
	Doubao-2.0-pro	42.64±1.04	n/a	n/a	n/a	n/a	4.33±0.07	5.67±1.10	2.67±0.19	36.48±2.18
	GLM-4.7	46.13±0.37	n/a	n/a	n/a	n/a	4.53±0.03	6.74±0.42	4.88±0.41	38.60±1.85
	Deepseek-v4-pro	52.54±0.82	n/a	n/a	n/a	n/a	4.49±0.06	9.69±0.34	4.23±0.33	45.93±0.58
Just Ask	Gpt-5.5	50.78±0.64	12.63±0.81	32.27±1.74	68.81±0.48	19.52±2.87	3.09±0.01	9.33±1.27	5.15±0.50	43.69±1.51
	Qwen3.7-max	55.79±0.41	7.86±0.94	42.53±1.30	47.93±1.11	12.63±1.43	3.11±0.03	11.27±1.10	4.41±0.81	48.75±1.39
	Doubao-2.0-pro	50.11±1.42	7.29±0.41	38.92±2.16	53.99±1.00	12.00±1.46	3.86±0.10	7.93±0.64	5.83±1.22	44.48±2.85
	GLM-4.7	52.99±1.06	6.28±1.69	41.78±0.72	40.37±1.32	11.90±1.79	3.02±0.03	8.21±0.73	5.37±1.31	47.48±2.27
	Deepseek-v4-pro	54.80±0.69	19.23±1.03	31.38±0.92	35.74±0.52	33.93±1.61	2.35±0.11	8.43±0.75	8.62±0.44	48.82±0.74
Self-Evolving	Gpt-5.5	47.97±1.80	8.17±0.40	34.77±2.32	**75.99±3.30**	13.17±1.67	3.34±0.14	7.87±1.79	4.03±0.29	40.72±1.04
	Qwen3.7-max	**58.60±0.23**	21.39±0.93	33.07±0.45	50.16±1.88	32.17±0.31	2.60±0.08	11.16±0.46	7.97±0.42	52.70±1.06
	Doubao-2.0-pro	57.72±0.79	16.32±1.76	37.07±0.96	48.72±2.38	26.73±0.81	3.83±0.04	8.93±0.50	9.36±1.15	52.28±0.50
	GLM-4.7	56.45±1.34	16.89±1.01	34.95±2.60	49.41±1.69	30.17±1.43	2.62±0.06	7.74±0.40	7.91±0.55	51.11±2.84
	Deepseek-v4-pro	57.98±1.78	**28.85±1.78**	25.97±0.74	42.06±0.94	**47.91±0.99**	2.36±0.07	8.05±0.31	10.09±0.36	**53.25±1.74**
τ^2-Retail										
No Ask	Gpt-5.5	48.86±0.55	20.98±0.21	19.82±0.84	n/a	42.97±4.20	4.81±0.06	62.08±1.27	1.31±0.39	26.58±1.79
	Qwen3.7-max	48.38±0.48	19.87±0.51	20.51±1.15	n/a	38.34±2.15	5.20±0.11	64.42±1.52	1.21±0.51	25.39±1.03
	Doubao-2.0-pro	46.96±5.11	22.97±1.75	17.45±5.79	n/a	34.12±2.10	5.36±0.19	61.69±1.82	2.80±0.78	31.89±8.82
	GLM-4.7	43.18±1.44	21.16±1.05	15.73±1.47	n/a	39.04±3.78	3.79±0.12	33.77±0.39	1.68±0.67	27.82±1.75
	Deepseek-v4-pro	49.85±2.09	27.64±2.70	16.00±1.83	n/a	36.85±1.62	5.87±0.24	65.60±0.36	3.01±0.08	33.25±1.33
Just Ask	Gpt-5.5	68.83±1.54	31.72±1.91	32.62±0.56	68.77±1.02	38.22±2.02	5.48±0.03	82.84±2.34	4.18±0.19	55.10±2.45
	Qwen3.7-max	68.53±2.32	32.00±0.74	32.08±2.63	62.28±1.83	40.43±1.34	5.33±0.16	79.95±3.41	4.91±0.12	55.33±3.23
	Doubao-2.0-pro	62.58±0.91	32.33±4.55	25.58±3.84	57.89±2.09	41.95±4.41	5.09±0.08	74.70±2.39	4.43±0.05	48.55±0.47
	GLM-4.7	67.43±0.37	31.67±1.80	31.11±1.87	58.70±0.48	38.83±3.92	5.28±0.12	77.05±1.46	4.97±0.60	53.32±0.68
	Deepseek-v4-pro	67.57±1.90	31.26±2.32	32.20±0.34	58.64±2.86	42.18±0.57	4.93±0.12	72.00±3.20	4.88±0.19	56.11±1.71
Self-Evolving	Gpt-5.5	**79.64±3.01**	33.34±2.33	43.78±0.90	**69.47±0.35**	40.00±3.42	5.66±0.20	77.23±6.12	4.85±0.25	**73.82±4.97**
	Qwen3.7-max	74.23±3.83	34.63±3.75	36.45±0.28	64.42±2.98	42.30±2.79	5.33±0.20	66.43±4.20	5.00±0.06	64.31±4.79
	Doubao-2.0-pro	69.55±2.34	**39.56±2.90**	27.57±2.43	57.78±0.64	**54.02±2.24**	4.67±0.15	65.02±4.36	5.31±0.14	61.17±2.74
	GLM-4.7	71.23±0.85	35.46±1.88	32.00±1.44	57.77±2.87	45.08±2.94	5.07±0.04	74.45±2.93	5.17±0.17	58.63±1.17
	Deepseek-v4-pro	71.41±3.07	36.50±4.77	30.94±1.55	58.79±1.18	49.66±4.71	5.11±0.12	73.92±2.19	5.57±0.09	60.52±4.30
τ^2-Airline										
No Ask	Gpt-5.5	54.65±3.20	15.07±1.17	21.87±2.05	n/a	23.66±1.33	5.71±0.25	32.77±5.50	2.34±0.78	34.29±9.95
	Qwen3.7-max	55.13±2.82	12.48±0.98	26.97±1.44	n/a	23.93±1.66	5.79±0.27	40.15±5.62	1.95±0.50	36.04±7.13
	Doubao-2.0-pro	51.12±3.89	12.96±0.07	20.77±1.92	n/a	22.56±1.22	5.57±0.28	31.80±6.40	2.36±0.97	33.10±8.07
	GLM-4.7	47.76±4.31	10.87±1.99	21.09±3.18	n/a	21.77±2.00	5.32±0.28	33.38±3.84	2.44±0.21	28.67±8.37
	Deepseek-v4-pro	46.64±3.82	9.55±1.68	22.78±1.67	n/a	24.00±3.68	5.19±0.27	34.73±2.68	3.14±0.17	28.67±7.17
Just Ask	Gpt-5.5	66.98±2.78	19.38±0.46	37.75±1.86	58.88±3.49	29.11±4.17	5.73±0.31	46.55±14.75	3.94±0.88	47.60±5.79
	Qwen3.7-max	65.22±2.65	19.54±1.95	36.08±2.63	52.58±3.13	31.69±5.47	5.57±0.25	44.79±5.18	3.43±0.50	48.33±3.89
	Doubao-2.0-pro	63.46±3.00	21.55±2.30	30.89±1.79	49.17±1.69	35.33±4.89	5.24±0.30	40.82±4.44	4.48±0.48	45.36±6.59
	GLM-4.7	60.74±1.94	19.28±3.13	34.84±4.63	37.01±13.13	34.94±10.05	4.95±0.20	50.35±7.33	4.21±0.32	40.59±6.38
	Deepseek-v4-pro	55.30±4.47	18.69±3.36	33.66±1.06	47.83±2.55	37.31±4.12	4.64±0.23	41.27±7.88	4.75±0.88	40.84±6.95
Self-Evolving	Gpt-5.5	**70.03±3.20**	17.24±2.93	43.84±1.22	**61.86±2.79**	30.09±5.20	5.65±0.38	49.97±6.16	3.83±0.09	**52.79±5.30**
	Qwen3.7-max	66.67±2.73	18.77±1.92	40.74±1.11	55.07±3.35	33.92±2.93	5.57±0.34	48.59±14.25	3.56±0.13	50.69±7.42
	Doubao-2.0-pro	66.51±2.42	**28.11±1.32**	30.72±1.60	49.52±1.89	**44.11±6.06**	4.99±0.46	41.34±4.65	4.86±0.73	50.31±3.57
	GLM-4.7	64.10±3.20	24.41±2.12	34.83±1.66	46.93±4.23	42.58±2.95	5.05±0.29	56.94±6.72	4.49±0.70	47.95±5.26
	Deepseek-v4-pro	59.49±2.17	20.80±4.10	30.06±1.99	44.84±4.02	40.73±3.66	4.71±0.26	42.05±7.38	4.56±0.60	36.78±5.98

Table 6: Full GRIP results for the backbone sweep (RQ1). No Ask, Just Ask, and Self-Evolving are evaluated across five agent backbones on all three benchmarks; all other conditions follow the default (Rational persona, single-fault requests, shifts enabled; mean±sd over three seeded runs). "n/a" denotes metrics that are not applicable under the corresponding interaction setting.

sal across all backbones, reinforcing the need to preserve Earned and Inferred separately rather than treating successful inference as equivalent to successful intent recovery.

The advantage becomes especially meaningful once intent can become stale. The complete results further reinforce the main-text observation that the benefit of stronger interaction is particularly stable in the state-changing τ^2 environments. Self-Evolving improves Success over Just Ask for every backbone on both Retail and Airline, while also producing the strongest PostShift result within each benchmark. Reaction, however, does not uniformly decrease. A stronger method may spend additional turns clarifying an updated requirement before reaching a correct final state. Reaction and

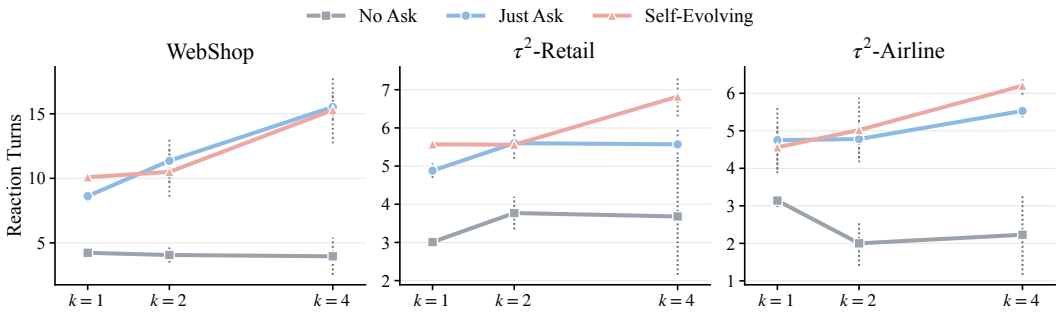

Figure 10: **Reaction under composed faults.** Median turns from the last relevant intent shift to acceptance as fault complexity k increases. Successful interactive trajectories generally require more post-shift turns at higher complexity, revealing the interaction cost of recovering coupled intent errors. Mean±sd over three seeds; axes are scaled per panel.

Staleness therefore diagnose different aspects of adaptation and should be interpreted jointly with PostShift Success rather than as standalone measures of quality.

Taken together, the expanded RQ1 results sharpen the conclusion from the main text. Stronger LLMs alone do not eliminate interactive intent-alignment failures, and stronger clarification is not simply a matter of asking more questions or identifying relevant uncertainty. What increasingly distinguishes successful agents is whether they can transform interaction into an accurate, current, and actionable representation of user intent, and whether their final success is grounded in that recovered intent rather than reached through inference alone.

C.4 RQ2: Robustness to Interaction Conditions

RQ1 establishes that interaction substantially improves intent alignment under the default evaluation condition. RQ2 asks a different question: whether these capabilities remain reliable once the simplifying assumptions behind that default interaction are relaxed. We therefore vary three factors that Drift-Bench++ is specifically designed to model: communication faults may compound within the same request, the user's latent intent may change after interaction begins, and users may differ in how they disclose information and respond to clarification. These are not generic robustness perturbations. Each tests whether an interaction assumption commonly held fixed in existing evaluations materially changes what agents can accomplish. The results below extend the main-text analysis by exposing which component of interactive alignment becomes the bottleneck under each condition.

C.4.1 Composed Communication Faults

Real miscommunication is rarely confined to a single isolated defect. A request may simultaneously omit a required parameter, rely on an incorrect premise, and express the intended action ambiguously. Evaluating such faults only in isolation can therefore overestimate robustness by allowing an agent to recover once a single uncertainty is resolved. To test whether the interaction advantage observed in RQ1 persists when several communication failures must be handled jointly, we compose $k \in \{1, 2, 4\}$ distinct fault types within each request while retaining the same finite interaction budget. Figure 4 summarizes Success and Recovery, Table 7 reports the complete GRIP profile, and Figure 10 further examines how much post-shift interaction successful recovery requires.

Compounding faults increases the value of interaction rather than merely making every method uniformly worse. At k=4, Self-Evolving exceeds No Ask in Success by 19.01 points on WebShop, 25.30 on τ^2-Retail, and 44.84 on τ^2-Airline. Airline provides the clearest example: No Ask collapses from 46.64 ± 3.82 at k=1 to 6.83 ± 3.75 at k=4, whereas Just Ask and Self-Evolving retain 46.50 ± 6.08 and 51.67 ± 6.05, respectively. The important pattern is therefore not that every raw score decreases monotonically with k, which is not required and does not occur in every domain, but that high fault complexity increasingly separates agents that can recover missing intent through interaction from those forced to resolve several coupled uncertainties internally. This is precisely the capability gap that single-fault evaluation would largely conceal.

Method	Faults k	Grounding (G)			Inquiry (I)			Persistence (P)		
		Success	Earned	Inferred	Aim	Recovery	Patience	Staleness	Reaction	PostShift
WebShop										
No Ask	k=1	52.54±0.82	n/a	n/a	n/a	n/a	4.49±0.06	9.69±0.34	4.23±0.33	45.93±0.58
	k=2	48.78±0.99	n/a	n/a	n/a	n/a	4.69±0.18	8.22±1.36	4.06±0.59	43.45±0.43
	k=4	36.71±0.18	n/a	n/a	n/a	n/a	5.62±0.23	3.19±0.60	3.96±1.44	34.48±1.16
Just Ask	k=1	54.80±0.69	19.23±1.03	31.38±0.92	35.74±0.52	33.93±1.61	2.35±0.11	8.43±0.75	8.62±0.44	48.82±0.74
	k=2	53.55±1.64	21.20±1.51	31.34±0.45	47.80±1.25	36.17±2.05	2.02±0.09	8.80±1.65	11.36±1.67	49.62±1.67
	k=4	50.38±2.52	13.73±0.78	36.65±2.47	63.29±0.74	24.75±0.92	1.83±0.12	4.79±1.20	15.53±0.99	46.11±2.28
Self-Evolving	k=1	**57.98±1.78**	28.85±1.78	25.97±0.74	42.06±0.94	47.91±0.99	2.36±0.07	8.05±0.31	10.09±0.36	**53.25±1.74**
	k=2	56.84±2.83	**30.67±1.94**	25.11±1.61	57.33±0.97	**51.05±1.51**	2.32±0.23	7.64±1.29	10.50±1.93	52.52±3.46
	k=4	55.72±0.69	25.80±0.75	29.92±1.32	**71.84±2.21**	44.04±1.14	2.28±0.07	7.18±0.69	15.29±2.58	51.31±2.24
τ^2-Retail										
No Ask	k=1	49.85±2.09	27.64±2.70	16.00±1.83	n/a	36.85±1.62	5.87±0.24	65.60±0.36	3.01±0.08	33.25±1.33
	k=2	51.12±5.68	13.90±5.34	14.70±5.99	n/a	24.42±5.28	6.07±0.35	71.93±4.24	3.77±0.43	29.27±8.47
	k=4	47.35±4.65	11.31±5.82	15.76±3.76	n/a	24.64±9.66	6.26±0.08	72.44±5.75	3.68±1.53	29.58±4.28
Just Ask	k=1	67.57±1.90	31.26±2.32	32.20±0.34	58.64±2.86	42.18±0.57	4.93±0.12	72.00±3.20	4.88±0.19	56.11±1.71
	k=2	**73.92±1.82**	18.27±2.17	42.99±1.29	50.32±1.92	23.73±2.54	5.10±0.37	79.16±2.67	5.60±0.41	60.26±0.92
	k=4	70.43±3.01	21.37±2.77	37.89±1.97	40.00±1.17	29.18±3.84	4.76±0.09	81.92±5.38	5.57±0.38	58.40±3.28
Self-Evolving	k=1	71.41±3.07	**36.50±4.77**	30.94±1.55	**58.79±1.18**	**49.66±4.71**	5.11±0.12	73.92±2.19	5.57±0.09	**60.52±4.30**
	k=2	71.33±5.70	22.23±0.50	36.96±6.77	53.62±1.97	31.91±5.96	4.92±0.55	83.38±1.40	5.56±0.11	57.92±7.86
	k=4	72.65±2.07	31.98±3.71	30.79±4.85	46.51±3.06	44.00±5.57	4.78±0.11	64.34±3.48	6.82±0.52	59.59±4.71
τ^2-Airline										
No Ask	k=1	46.64±3.82	9.55±1.68	22.78±1.67	n/a	24.00±3.68	5.18±0.27	34.73±2.68	3.14±0.17	28.67±7.17
	k=2	44.03±0.82	9.56±0.99	15.82±1.27	n/a	19.87±2.68	5.05±0.09	23.17±2.81	2.00±0.60	29.14±2.44
	k=4	6.83±3.75	1.67±1.15	5.17±3.33	n/a	17.17±2.47	2.27±0.16	5.45±2.40	2.23±1.07	7.37±3.94
Just Ask	k=1	55.30±4.47	18.69±3.36	33.66±1.06	47.83±2.55	37.31±4.12	4.64±0.23	41.27±7.88	4.75±0.88	40.84±6.95
	k=2	57.22±3.25	14.47±1.16	28.96±2.83	59.83±0.13	28.95±2.15	4.50±0.24	35.28±2.95	4.78±0.52	40.94±6.74
	k=4	46.50±6.08	17.17±5.01	29.33±4.25	**75.66±1.17**	25.33±2.31	2.77±0.40	34.23±7.87	5.53±0.17	36.28±7.83
Self-Evolving	k=1	59.49±2.17	20.80±4.10	30.06±1.99	44.84±4.02	40.73±3.66	4.71±0.26	42.05±7.38	4.56±0.60	36.78±5.98
	k=2	**59.54±3.40**	21.71±3.51	27.37±4.82	58.83±4.13	40.44±3.82	4.51±0.13	37.46±2.56	5.02±0.86	42.14±8.61
	k=4	51.67±6.05	**27.67±5.30**	24.00±2.65	70.53±5.16	**44.17±4.25**	2.98±0.42	36.11±5.97	6.20±0.24	**42.22±7.19**

Table 7: Full GRIP results for the interaction-complexity sweep (RQ2). Composite perturbations contain k distinct fault types per request, with k=1 reproducing the default single-fault condition. All other experimental factors are held fixed (mean±sd over three seeded runs).

More complex misalignment requires better clarification, not only more clarification. The most stable cross-domain pattern is the growing Recovery advantage of Self-Evolving over Just Ask. From k=1 to k=4, this gap increases from 13.98 to 19.29 points on WebShop, from 7.48 to 14.82 on Retail, and from 3.42 to 18.84 on Airline. Self-Evolving also retains a higher Earned score at k=4 in all three domains. As more faults must be resolved jointly, the distinction between clarification strategies therefore becomes more consequential: success depends increasingly on whether the agent can accumulate enough latent target across multiple exchanges, rather than merely deciding to ask.

Recognizing more uncertainty does not imply resolving more uncertainty. The full GRIP profile exposes a second effect hidden by Success alone. Aim often rises sharply with k, even when Recovery does not. On WebShop, for example, Just Ask increases Aim from 35.74 ± 0.52 to 63.29 ± 0.74 between k=1 and k=4, while Recovery falls from 33.93 ± 1.61 to 24.75 ± 0.92. On Airline at k=4, Just Ask even achieves higher Aim than Self-Evolving (75.66 ± 1.17 versus 70.53 ± 5.16), yet substantially lower Recovery (25.33 ± 2.31 versus 44.17 ± 4.25). Compounded faults therefore make relevant uncertainty easier to encounter without making it easier to resolve. For interactive agents, identifying what deserves clarification and integrating the resulting answers into a coherent current intent are distinct capabilities, and the separation between them becomes more visible as communication complexity grows.

Compound misalignment makes successful adaptation itself more interaction-intensive. Figure 10 shows that Reaction generally increases with k for the interactive methods, particularly at the highest complexity. On WebShop, Reaction rises from 8.62 to 15.53 turns for Just Ask and from 10.09 to 15.29 for Self-Evolving; Airline exhibits the same qualitative pattern, while Retail shows a smaller increase. Because Reaction conditions on post-shift episodes that eventually reach

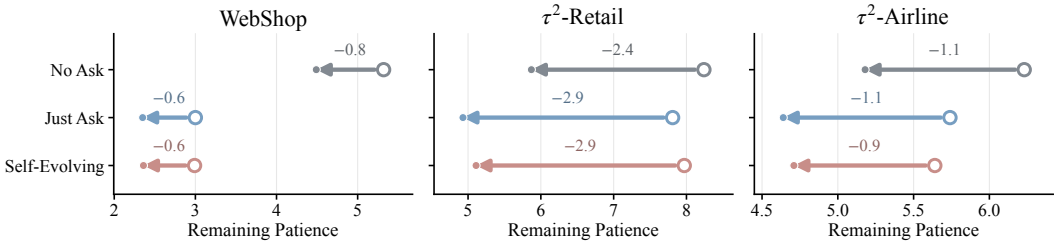

Figure 11: **Patience consumed by intent shifting.** Each arrow shows the change in remaining patience from shifts disabled (open circle) to enabled (filled marker). Shifts consume additional interaction budget in every method and benchmark, with the largest drain on τ^2-Retail. Values are from Table 8; axes are scaled per panel.

acceptance, this trend cannot be explained simply by additional failures. Even successful agents require more interaction to re-establish alignment when several communication faults coexist. The accompanying decline in remaining Patience on WebShop and Airline further shows why this matters: additional clarification is useful, but it consumes the same finite resource that constrains the interaction. Taken together, the complexity sweep strengthens rather than merely repeats the RQ1 conclusion. Interaction becomes more valuable as communication failures compound, but the bottleneck increasingly shifts from noticing uncertainty to recovering and coordinating multiple hidden requirements within a limited dialogue. By supporting verified compositions of distinct fault types under finite patience, Drift-Bench++ therefore exposes a regime in which both the quality and the cost of clarification become central to agent performance.

C.4.2 Intent-Shift Ablation

The complexity sweep increases uncertainty in the *initial* request. Intent shifting introduces a different challenge: information that has already been correctly recovered may cease to describe what the user currently wants. This distinction is central to Drift-Bench++, because a fixed-intent benchmark can evaluate whether an agent eventually resolves miscommunication, but cannot test whether that alignment persists when the target itself changes. We therefore disable both scheduled and interaction-conditioned shifts while holding the initial requests, personas, tools, patience rules, and executable verifiers fixed. Figure 5 reports the headline Success and Earned effects, Table 8 provides the complete GRIP decomposition, and Figure 11 visualizes the additional interaction budget consumed by evolving intent. Shift-conditioned metrics are marked as *n/a* when shifts are disabled.

Intent shifting creates a distinct failure mode beyond recovering the initial request. Enabling shifts lowers Success for every method and benchmark, but the degradation is systematically smaller for Self-Evolving. The Success loss for No Ask, Just Ask, and Self-Evolving is respectively 9.09, 7.72, and 4.71 points on WebShop; 23.54, 19.28, and 14.96 on τ^2-Retail; and 10.25, 7.20, and 3.65 on τ^2-Airline. This ordered degradation is more informative than a uniform drop in difficulty would be. Methods that explicitly retain, revise, and verify interaction state are better able to preserve performance once previously valid information can become obsolete. Fixed-intent clarification and continuous intent alignment are therefore empirically separable capabilities.

Evolving intent makes relevant uncertainty more visible while making the final target harder to recover. For both interactive methods and across all three benchmarks, enabling shifts increases Aim but decreases Recovery. On Retail, Self-Evolving's Aim rises from 31.41 ± 2.93 without shifts to 58.79 ± 1.18 with shifts, while Recovery falls from 83.34 ± 1.42 to 49.66 ± 4.71. Just Ask exhibits the same qualitative pattern. The challenge is therefore not simply that agents fail to notice when clarification is needed. Instead, shifting creates new relevant uncertainties faster than the agent can fully resolve and integrate them into a current representation of intent. This directly motivates evaluating continuous alignment rather than treating successful clarification as a one-time event.

Shifts also change the grounding of the successes that remain. Across every interactive method–benchmark pair, enabling shifts lowers Earned and increases Inferred. On Retail, for example, Self-Evolving changes from 69.27 ± 0.56 Earned and 12.55 ± 0.31 Inferred without shifts to 36.50 ± 4.77 and 30.94 ± 1.55 with shifts. Thus, evolving intent does more than remove successful episodes. Among the episodes that remain successful, a larger fraction are completed without fully recovering

Method	Shifts	Grounding (G)			Inquiry (I)			Persistence (P)		
		Success	Earned	Inferred	Aim	Recovery	Patience	Staleness	Reaction	PostShift
WebShop										
No Ask	✓	52.54±0.82	n/a	n/a	n/a	n/a	4.49±0.06	9.69±0.34	4.23±0.33	45.93±0.58
	✗	61.63±0.88	n/a	n/a	n/a	n/a	5.32±0.05	n/a	n/a	n/a
Just Ask	✓	54.80±0.69	19.23±1.03	31.38±0.92	35.74±0.52	33.93±1.61	2.35±0.11	8.43±0.75	8.62±0.44	48.82±0.74
	✗	62.52±1.61	37.40±3.03	24.09±1.56	28.51±0.04	61.34±1.75	3.00±0.04	n/a	n/a	n/a
Self-Evolving	✓	57.98±1.78	28.85±1.78	25.97±0.74	**42.06±0.94**	47.91±0.99	2.36±0.07	8.05±0.31	10.09±0.36	**53.25±1.74**
	✗	**62.69±0.68**	**45.55±1.73**	14.54±2.41	32.83±0.46	**76.62±3.43**	2.99±0.14	n/a	n/a	n/a
τ^2-Retail										
No Ask	✓	49.85±2.09	27.64±2.70	16.00±1.83	n/a	36.85±1.62	5.87±0.24	65.60±0.36	3.01±0.08	33.25±1.33
	✗	73.39±0.27	55.92±0.97	10.57±1.02	n/a	73.84±1.35	8.24±0.02	n/a	n/a	n/a
Just Ask	✓	67.57±1.90	31.26±2.32	32.20±0.34	58.64±2.86	42.18±0.57	4.93±0.12	72.00±3.20	4.88±0.19	56.11±1.71
	✗	**86.85±0.36**	**69.89±0.47**	12.63±0.54	29.55±2.10	81.90±1.35	7.81±0.11	n/a	n/a	n/a
Self-Evolving	✓	71.41±3.07	36.50±4.77	30.94±1.55	**58.79±1.18**	49.66±4.71	5.11±0.12	73.92±2.19	5.57±0.09	**60.52±4.30**
	✗	86.37±0.10	69.27±0.56	12.55±0.31	31.41±2.93	**83.34±1.42**	7.97±0.07	n/a	n/a	n/a
τ^2-Airline										
No Ask	✓	46.64±3.82	9.55±1.68	22.78±1.67	n/a	24.00±3.68	5.18±0.27	34.73±2.68	3.14±0.17	28.67±7.17
	✗	56.89±0.73	11.83±1.08	22.74±0.54	n/a	28.66±1.43	6.23±0.16	n/a	n/a	n/a
Just Ask	✓	55.30±4.47	18.69±3.36	33.66±1.06	**47.83±2.55**	37.31±4.12	4.64±0.23	41.27±7.88	4.75±0.88	**40.84±6.95**
	✗	62.50±2.40	23.67±2.70	25.55±4.21	36.83±0.62	47.66±4.08	5.74±0.27	n/a	n/a	n/a
Self-Evolving	✓	59.49±2.17	20.80±4.10	30.06±1.99	44.84±4.02	40.73±3.66	4.71±0.26	42.05±7.38	4.56±0.60	36.78±5.98
	✗	**63.14±4.03**	**27.41±7.02**	21.81±3.00	36.98±7.27	**53.27±6.74**	5.64±0.27	n/a	n/a	n/a

Table 8: Full GRIP results for the intent-shift ablation (RQ2). Each method is evaluated with intent shifting enabled (✓; default) and disabled (✗). Shift-conditioned metrics are not applicable when intent shifting is disabled. All other experimental factors are held fixed (mean±sd over seeded runs).

the final hidden target. An evaluation based only on Success would consequently understate how strongly intent dynamics disrupt grounding.

Maintaining alignment has a measurable interaction cost. Figure 11 shows that enabling shifts reduces remaining patience for every method and benchmark. The effect is modest on WebShop and Airline but much larger on Retail, where No Ask loses approximately 2.4 patience units and both interactive methods lose approximately 2.9. This domain is also where shifts cause the largest Success degradation, indicating that evolving intent becomes particularly consequential when recovering from it requires repeated interaction in a state-changing environment. Importantly, the patience drain is not confined to methods that ask questions: even No Ask consumes additional budget once shifts are introduced, showing that dynamic intent can impose cost through failed or superseded commitments as well as explicit clarification.

Robustness to shifts is not simply a consequence of spending less budget. On Retail, Just Ask and Self-Evolving incur essentially the same additional patience cost when shifts are enabled, yet Self-Evolving loses substantially less Success. A similar pattern appears on WebShop, where both interactive methods lose roughly 0.6 remaining patience while Self-Evolving again suffers the smaller performance drop. The stronger reference agent is therefore not merely more conservative with interaction. It extracts greater value from a comparable additional budget by retaining and revising what has already been learned. This distinction is important for patience-bounded evaluation: robustness depends on how effectively interaction is used, not only on how much interaction occurs.

The persistence metrics reveal what happens when this synchronization fails. Under shifts, Staleness remains substantial, particularly on Retail, meaning that many failed episodes terminate at a goal that was once correct but has since been superseded. This failure would be invisible in a static benchmark because the same behavior could have been judged correct before the shift occurred. Silent intent changes therefore create a qualitatively different requirement: the agent must not only recover missing intent, but continually determine whether its current understanding is still valid.

Finally, disabling shifts clarifies what remains difficult even in the static case. Just Ask and Self-Evolving nearly converge in Success on WebShop and remain close on both τ^2 domains, while both

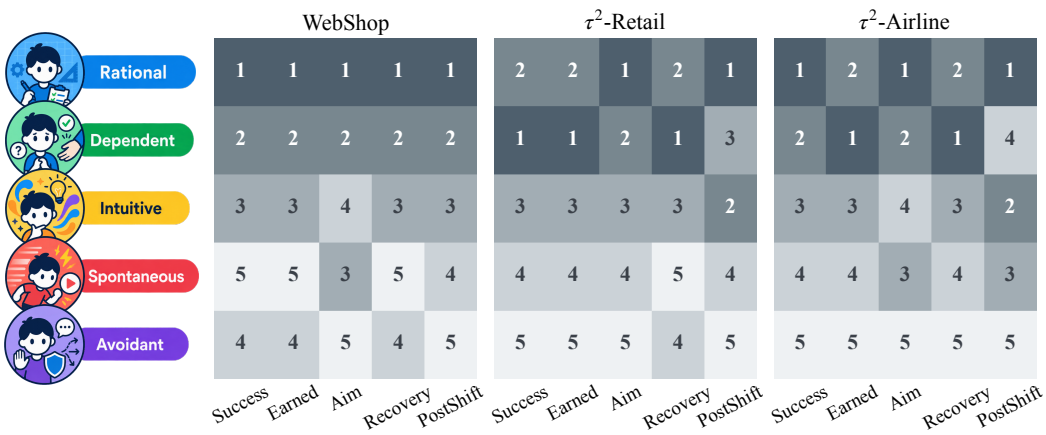

Figure 12: **Persona rankings across GRIP dimensions.** Each cell ranks the five personas within one benchmark and metric for Self-Evolving (1 = best; darker = higher rank). Rankings show a stable broad difficulty structure but metric-dependent reversals, particularly between Rational and Dependent users. Just Ask exhibits a similar overall pattern. Values are from Table 9.

generally outperform No Ask. Initial miscommunication and evolving intent therefore contribute separately to benchmark difficulty. Interaction is already valuable for recovering a fixed hidden target, but the additional advantage of persistent intent tracking becomes most visible once that target can change. Together with the complexity and persona sweeps, this ablation shows why interactive intent alignment cannot be reduced to clarification under a fixed user goal: realistic evaluation must also measure whether alignment survives the interaction that follows.

C.4.3 Persona-Conditioned Interaction

The preceding experiments vary properties of the task while holding the simulated user fixed. Yet clarification is inherently a two-sided process: the same agent behavior may succeed or fail depending on whether a user volunteers information, waits for guidance, responds intuitively, or resists further questioning. Evaluating only a rational and cooperative user can therefore confound interactive capability with compatibility with a single disclosure policy. To test whether the conclusions of RQ1 remain meaningful under heterogeneous user behavior, we replay the same verified benchmark instances under five persona-conditioned users while preserving the underlying tasks and perturbations. Figure 6 summarizes the resulting performance profiles, Table 9 reports the complete GRIP results, and Figure 12 exposes how the relative difficulty of each persona changes across alignment capabilities.

Persona variation changes the effective difficulty of the same underlying task. The largest effect appears on τ^2-Retail. Just Ask ranges from 41.26 ± 2.34 Success under Avoidant users to 77.30 ± 0.62 under Dependent users, while Self-Evolving ranges from 42.76 ± 3.02 to 78.26 ± 1.17. This approximately 35–36 point spread arises without changing the executable task or communication fault. The user model alone can therefore alter task difficulty on a scale comparable to major differences between agent strategies or backbones. Persona-conditioned interaction is consequently not a cosmetic realism feature: it determines how much of the latent intent becomes recoverable within the same interaction process.

User difficulty is structured, but not one-dimensional. Figure 12 reveals a surprisingly coherent broad ordering. Rational and Dependent users occupy the top ranks across most metrics, whereas Avoidant and Spontaneous users generally occupy the bottom. Yet the strongest persona depends systematically on what capability is being measured. On Retail, Dependent users rank first for Success, Earned, and Recovery, while Rational users rank first for Aim and PostShift Success. Airline exhibits an analogous split: Dependent maximizes Earned and Recovery, whereas Rational maximizes Success, Aim, and PostShift Success. Thus, a user who makes the current intent easier to recover need not make it easier to remain aligned after that intent changes. Persona variation changes which part of the interaction becomes limiting rather than merely multiplying all scores by a common notion of difficulty.

Method	Persona	Grounding (G)			Inquiry (I)			Persistence (P)		
		Success	Earned	Inferred	Aim	Recovery	Patience	Staleness	Reaction	PostShift
WebShop										
Just Ask	Rational	54.80±0.69	19.23±1.03	31.38±0.92	35.74±0.52	33.93±1.61	2.35±0.11	8.43±0.75	8.62±0.44	48.82±0.74
	Avoidant	51.43±0.68	8.55±0.53	38.53±0.38	15.56±1.72	15.27±0.32	1.66±0.06	10.33±0.81	6.67±1.30	45.46±1.04
	Dependent	**58.31±1.84**	14.87±1.49	36.46±0.91	35.66±0.80	22.54±1.34	3.31±0.03	8.13±1.63	7.97±0.83	48.93±2.63
	Intuitive	50.58±0.88	9.47±0.39	35.09±2.02	29.35±1.11	16.33±1.47	2.71±0.10	8.80±0.72	5.81±0.65	45.60±0.72
	Spontaneous	50.23±0.54	8.15±1.03	36.01±1.10	31.88±1.02	14.19±0.26	2.23±0.01	8.93±1.30	4.34±0.69	45.87±0.73
Self-Evolving	Rational	57.98±1.78	**28.85±1.78**	25.97±0.74	**42.06±0.94**	**47.91±0.99**	2.36±0.07	8.05±0.31	10.09±0.36	**53.25±1.74**
	Avoidant	50.39±1.25	12.03±0.94	33.72±1.02	18.27±0.90	21.96±0.20	1.64±0.06	10.07±0.12	7.20±0.54	44.50±0.22
	Dependent	57.95±2.82	18.58±2.17	31.49±1.39	36.85±1.62	28.75±1.40	3.26±0.12	8.20±0.72	8.41±0.49	52.42±3.93
	Intuitive	51.16±0.95	15.30±1.64	31.02±1.61	34.90±2.02	25.52±1.96	2.63±0.09	10.74±1.89	7.22±1.23	47.16±1.13
	Spontaneous	49.54±2.15	11.31±0.47	32.55±2.26	36.36±1.11	21.01±1.04	2.00±0.05	11.07±0.90	6.11±0.58	46.34±2.31
τ^2-Retail										
Just Ask	Rational	67.57±1.90	31.26±2.32	32.20±0.34	58.64±2.86	42.18±0.57	4.93±0.12	72.00±3.20	4.88±0.19	56.11±1.71
	Avoidant	41.26±2.34	19.42±2.25	14.53±1.13	29.80±2.24	28.35±2.36	3.18±0.05	66.86±3.98	2.43±0.41	18.02±0.78
	Dependent	77.30±0.62	**53.26±0.57**	15.50±1.33	49.49±2.74	59.63±1.12	8.28±0.05	32.36±4.45	5.76±0.51	31.16±1.27
	Intuitive	58.80±0.73	24.01±1.33	29.56±2.52	43.04±2.70	28.02±0.52	4.38±0.11	44.94±1.59	3.93±0.16	42.12±2.66
	Spontaneous	48.71±0.85	21.37±2.84	21.04±3.27	37.04±1.45	24.95±2.42	3.23±0.07	44.51±1.98	3.08±0.21	29.38±3.02
Self-Evolving	Rational	71.41±3.07	36.50±4.77	30.94±1.55	**58.79±1.18**	49.66±4.71	5.11±0.12	73.92±2.19	5.57±0.09	**60.52±4.30**
	Avoidant	42.76±3.02	21.22±2.31	13.78±0.85	28.71±0.35	29.95±2.51	3.34±0.16	69.18±1.02	2.42±0.54	18.17±2.12
	Dependent	**78.26±1.17**	53.14±0.53	16.60±1.66	53.33±1.01	**60.27±0.45**	8.42±0.12	35.76±3.09	6.38±0.77	33.58±4.93
	Intuitive	63.60±2.26	28.29±3.01	30.92±1.16	44.76±1.04	32.99±2.45	4.41±0.04	46.51±0.93	4.92±0.34	49.30±1.82
	Spontaneous	48.59±1.44	21.39±1.21	20.83±1.04	34.73±2.42	25.85±0.60	3.23±0.03	45.36±1.96	3.73±0.86	29.32±2.09
τ^2-Airline										
Just Ask	Rational	55.30±4.47	18.69±3.36	33.66±1.06	**47.83±2.55**	37.31±4.12	4.64±0.23	41.27±7.88	4.75±0.88	**40.84±6.95**
	Avoidant	48.08±3.93	10.77±2.59	24.91±2.20	26.62±5.44	22.84±5.65	3.60±0.16	40.72±3.95	3.17±0.51	24.72±7.58
	Dependent	58.30±4.47	20.63±3.89	25.98±2.53	44.79±4.52	36.45±3.26	6.53±0.36	26.32±1.93	5.57±0.77	27.08±9.57
	Intuitive	56.09±4.47	15.45±0.55	29.50±3.16	36.79±2.06	25.87±1.21	4.52±0.18	35.22±3.43	4.16±0.48	35.32±5.48
	Spontaneous	51.60±6.20	13.26±1.26	26.44±4.79	36.65±2.63	26.56±2.02	3.49±0.14	35.56±1.71	2.91±0.33	31.15±7.58
Self-Evolving	Rational	**59.49±2.17**	20.80±4.10	30.06±1.99	44.84±4.02	40.73±3.66	4.71±0.26	42.05±7.38	4.56±0.60	36.78±5.98
	Avoidant	46.96±3.27	13.27±1.20	22.81±2.40	27.59±1.56	27.32±2.39	3.42±0.18	46.56±1.99	3.54±1.55	23.36±6.48
	Dependent	57.37±0.28	**23.55±3.74**	20.79±2.97	43.98±3.56	**46.08±6.30**	5.74±0.18	23.01±6.96	5.94±0.10	27.71±1.88
	Intuitive	53.05±3.54	17.19±0.70	26.76±2.63	37.77±1.93	28.96±2.97	4.18±0.19	33.31±8.41	4.28±0.19	33.67±4.48
	Spontaneous	51.12±2.65	14.85±1.87	24.73±2.23	38.11±5.36	27.48±3.94	3.33±0.17	39.06±3.13	3.41±0.40	32.28±1.13

Table 9: Full GRIP results for the persona sweep (RQ2). Just Ask and Self-Evolving are evaluated under all five simulated personas on the three benchmarks; the Rational rows reproduce the default condition in Table 2. All other experimental factors are held fixed, so the only deviation from the default condition is persona (mean±sd over three seeded runs).

The rankings expose interpretable behavioral pressure rather than arbitrary simulation noise. Avoidant users are consistently among the hardest conditions, especially for Aim, Recovery, and PostShift Success, while Dependent users frequently support strong recovery and grounding. Rational users are particularly strong on persistence-related outcomes, including the highest Self-Evolving PostShift Success in all three benchmarks. These recurring patterns appear across domains rather than in isolated cells, suggesting that the behavioral controls induce systematic differences in how information becomes available to the agent. This matters for benchmark design: averaging over one generic simulated user would conceal whether an agent succeeds because its clarification strategy is genuinely robust or because the user happens to communicate in a way favorable to that strategy.

Better intent recovery does not guarantee better task completion under every user. Self-Evolving achieves higher Recovery than Just Ask in all fifteen persona–benchmark combinations, demonstrating that its advantage in reconstructing hidden intent is robust to user behavior. Its Success advantage, however, is not universal. On WebShop under the Avoidant persona, for example, Recovery increases from 15.27 ± 0.32 to 21.96 ± 0.20, while Success changes from 51.43 ± 0.68 to 50.39 ± 1.25. Similar local inversions occur in other persona conditions. These cases reinforce the distinction established in RQ1: recovering more of the user's intent is an important interaction capability, but the agent must still translate that knowledge into correct downstream actions. Evaluating Success alone would obscure this improvement, while evaluating Recovery alone would overstate its effect on end-to-end performance.

Taken together, the persona sweep complements the complexity and intent-shift experiments by varying the other side of the interaction rather than the task itself. Compound faults determine how much intent must be recovered, shifts determine whether recovered information remains valid, and personas determine how that information becomes available through dialogue. Across all three axes, the central conclusion is consistent: interactive alignment cannot be characterized by a single static task-success measure. Drift-Bench++ exposes distinct failure modes arising from communication complexity, non-stationary intent, and heterogeneous user behavior, allowing researchers to identify not only whether an agent fails, but which component of the interaction makes alignment difficult.

C.5 RQ3: Role-Realism Validation

The persona sweep in RQ2 is informative only if different persona conditions produce behavior that is both observable from the interaction and stable across episodes. RQ3 therefore first validates the *role-realism* of our simulated users rather than assuming that persona labels correspond to meaningful behavioral variation. For Identification, the persona instruction is hidden and two independent judges infer the generating role from the user-side interaction alone. For Consistency, judges compare interactions generated under the same role and determine whether they exhibit behavior compatible with a stable persona. Because the opening request is constructed before persona assignment, it is excluded from both evaluations. As shown in Figure 6, Identification and Consistency remain substantially above their 20% and 50% chance baselines, respectively. The persona state-machine therefore produces behavioral signatures that are externally recoverable and repeatable, rather than distinctions that exist only inside the simulator prompt.

Role-realism validates behavioral variation, not task difficulty or population realism. This distinction is important for interpreting RQ2. A persona can be highly recognizable while making a task either easier or harder, and internally consistent behavior does not imply that the corresponding role occurs with any particular frequency in the real population. We therefore do not combine Identification or Consistency with Success, nor use them to weight persona conditions. Instead, these metrics validate the experimental intervention itself: when performance changes across personas in RQ2, the compared conditions correspond to distinct and stable interaction behaviors rather than arbitrary prompt labels. This validation addresses whether the simulated user conditions are behaviorally meaningful, but not whether the communication failures and intent changes modeled by Drift-Bench++ occur in deployment. The second part of RQ3 therefore evaluates this separate question using ProdAgent production sessions, testing the external relevance of both the communication-fault and intent-shift abstractions. Because that analysis relies on a different data source, annotation process, and set of denominators, we report its complete methodology and numerical results separately in Appendix D.

D Drift-Bench++ on Production Agent Data

To examine whether the communication failures and intent shifts studied by Drift-Bench++ also arise in real-world deployed agent systems, we analyze a labeled subset of sessions routinely sampled from an anonymized production agent platform for quality monitoring and behavioral analysis. Due to privacy and platform's data-use policy, we report only aggregate statistics and do not release raw user messages. We also do not disclose the product name or the internal procedures used to annotate intents and quality scores. Instead, we describe the resulting annotation schema, including the types of intents extracted and the quality dimensions evaluated.

Production Platform. The data come from a continuously deployed, managed platform for persistent, tool-using agents. Its architecture is similar to OpenClaw,[1] an open-source runtime that connects agents to communication channels, maintains persistent session and workspace state, and supports tools such as browsers, files, shell commands, scheduling, and domain-specific services. The production platform similarly supports long-running and asynchronous tasks in which agents maintain context, invoke external tools, and return results through user-facing communication channels. This setting is particularly valuable for our study because it captures real users delegating extended, tool-mediated tasks to deployed agents rather than interactions created specifically for benchmark evaluation.

[1] https://docs.openclaw.ai/

Analysis Details. Our corpus consists of consecutive weekly evaluation snapshots drawn from the platform's production-monitoring pipeline. Each sampled session contains ordered user interactions, model and tool activity, task metadata, extracted user intents, and quality assessments of the resulting execution. These pre-existing annotations allow us to study intent misalignment without retrospectively reconstructing user goals from scratch.

For each task, the platform records both *explicit intents*, requirements directly stated by the user and categorized as content, format, or constraint requirements, and *implicit intents*, requirements inferred from the task context but not explicitly stated. It additionally records task scene, difficulty, and complexity metadata. Execution quality is evaluated along seven dimensions: task completion, factual accuracy, constraint adherence, reasoning quality, tool-use quality, output quality, and efficiency, each can be selected from an integer score from 1 to 5, together with a separate holistic overall score and task-level rubric pass rate. We use these annotations only as pre-existing signals of what the user wanted and how well the deployed agent fulfilled it.

To study interactive intent alignment, we restrict analysis to sessions containing at least two human messages, since a correction, re-specification, or intent change can only be observed relative to an earlier communication. The pre-existing intent inventory helps identify requirements that may be absent from the initial request, while in this specific experiment, we adopt two independent LLM judges from different model families to determine whether later user messages repair, revise, or change that request and if so, which shift type and fault type should it belongs to. Only strict-agreement cases are retained for the primary fault and shift analyses. Approvals, requested follow-up information, and ordinary workflow progression are not treated as misalignment.

E Limitations and Future Work

Applicability Across Agentic Benchmarks. Drift-Bench++ is designed to adapt existing benchmarks through a lightweight decomposition of each task into a frame and a set of intent-defining conditions. This abstraction is particularly natural for environments whose goals are specified by structured constraints, executable actions, or other addressable task fields, as in WebShop and τ^2. It is less direct for highly open-ended tasks whose success criteria cannot be cleanly decomposed into conditions or verified by a reliable host-native oracle. Broadening this interface to such environments, while preserving controllable intent edits and executable validation, is an important direction for extending the benchmark beyond structurally explicit tasks.

Beyond Longer Autonomous Horizons. Our experiments focus on relatively short interaction horizons, whereas frontier agents increasingly demonstrate trajectories extending to hundreds or thousands of steps. Scaling to still longer autonomous execution is important, but we view horizon length as only one axis of progress. A complementary challenge is sustaining alignment with a human collaborator who does not behave like an oracle: users omit information, make mistakes, change their minds, provide incomplete feedback, and may become impatient. In this sense, increasing an agent's ability to act longer does not remove the need to continually determine *what* it should be acting toward. As a next step, we plan to apply the Drift-Bench++ construction pipeline to ProdAgent production tasks, using real interaction structures to construct larger-scale, high-quality misaligned trajectories and study interactive intent alignment over substantially longer workflows. More broadly, we believe the next stage of long-horizon agents will increasingly concern not only autonomous persistence, but persistent collaboration with imperfect and evolving human partners.

Coverage of Human Miscommunication and Behavior. Neither a finite taxonomy nor a finite set of simulated personas can exhaust the diversity of real human interaction. We mitigate this limitation in several ways: the communication taxonomy is derived from established theories of human communication and spans multiple fault families; our complexity experiments compose multiple faults to approximate compound real-world errors; and ProdAgent production analysis provides external evidence that the modeled fault and shift categories occur in deployed interactions. Likewise, persona simulation is based on the GDMS framework and deliberately varies disclosure, patience, feedback, and susceptibility to reconsideration rather than merely changing surface style. Nevertheless, the current taxonomy and five personas should be viewed as a principled coverage of representative behaviors rather than a complete model of human communication. Future work can expand both dimensions by incorporating additional failure modes, richer behavioral profiles,

and broader production evidence, progressively turning Drift-Bench++ into a more comprehensive testbed for human-agent collaboration.

Reliance on LLM-Based Components. Drift-Bench++ necessarily relies on LLMs in components where realistic natural-language variation cannot be specified exhaustively by rules, including query perturbation, question understanding, and user-response realization. Such dependence is difficult to eliminate without collapsing the interaction back into templated language, but it introduces potential model bias and generation error. We therefore restrict LLMs wherever benchmark truth is concerned. Perturbations are admitted only after mechanical fidelity checks, and host-native executable verification; during user simulation, a state machine controls the latent intent, disclosure decisions, patience, shifts, and proposal outcomes, while the LLM is limited to mapping questions and verbalizing authorized content. A grounded observer further records what was actually communicated, and our Role-realism experiments test whether simulated personas remain behaviorally distinct and consistent. These safeguards cannot make LLM-mediated simulation perfect, but they substantially separate linguistic variability from task truth and make failures auditable rather than silently inherited from the simulator.

Training Interactive Policies. Our current study evaluates fixed agent backbones and inference-time interaction strategies, and does not train models specifically for interactive intent alignment. Recent work suggests that interactive behavior can be further improved through dedicated training. For example, CollabLLM (Wu et al., 2025) optimizes multi-turn collaboration through reinforcement fine-tuning with multi-turn-aware rewards, while UserRL (Qian et al., 2025b) trains user-centric agents through reinforcement learning with simulated users and explicit multi-turn reward design. Complementary work such as Ask-before-Plan (Zhang et al., 2024) develops proactive clarification and planning mechanisms, together with trajectory tuning, for resolving underspecified user requirements before action commitment. These directions are closely related to the capabilities studied by Drift-Bench++, and evaluating trained interactive policies within our setting is therefore an important direction for future work. Accordingly, our current results should be interpreted as characterizing capability gaps of fixed agents under inference-time interaction, rather than establishing an upper bound on what policies explicitly trained for interactive alignment may achieve.

However, we hope to emphasize that extending Drift-Bench++ from an evaluation benchmark into a training environment introduces additional design requirements beyond directly applying existing training algorithms. A training-ready setting would require, among other considerations, intent-graph-level train-validation-test separation, principled difficulty stratification and curriculum construction, sufficient diversity of interaction trajectories, stable and learnable reward signals, and safeguards against overfitting to particular simulated-user behaviors or benchmark-specific perturbations. The current pipeline is instead designed to provide controlled, executable, and diagnostically interpretable evaluation instances. Developing a scalable and trainable counterpart is therefore a natural extension of Drift-Bench++, but is distinct from its present contribution of establishing and systematically evaluating interactive intent alignment under imperfect and evolving user intent.

F Prompt Design

In this section, we provide the major prompts behind Drift-Bench++, in three groups: benchmark construction (Figures 13-15), the simulated user (Figures 16 and 17), and the reference agent (Figures 18–20). Prompts are lightly abridged to fit one page each; the full text ships with the code. We release them for interpretability and reproducibility: each prompt carries a design decision the reported results depend on, and the exact wording lets readers audit and rerun the pipeline.

G Case Study

Aggregate metrics reveal performance differences but not what those differences look like in interaction. This matters because Drift-Bench++ also claims that its faults, personas, and silent intent shifts produce realistic and consequential user behavior. We therefore provide case studies for three purposes. First, they make the benchmark auditable end to end: each displayed fault, shift, patience change, and proposal outcome corresponds to verified perturbation state, a recorded graph transition, an explicit interaction cost, or executable adjudication, so the narrative directly reflects the benchmark trace. Second, they expose the mechanisms summarized by GRIP, including clarification that

both recovers information and triggers reconsideration, informative rejection feedback, and failures caused by remaining aligned with a superseded intent. Finally, they calibrate difficulty qualitatively: these are ordinary successful runs rather than adversarial edge cases, yet they still require agents to detect user errors, track multiple goal changes, and recover unstated procedural constraints. The cases therefore complement aggregate results by showing what interactive intent alignment requires within a single episode.

Fault Generation Prompt

ROLE AND TRUE INTENT

```
You are an expert at creating flawed instructions that test an agent'
  s ability to handle ambiguous or mistaken requests through
  clarification.

The shopper's TRUE intent:
  required features: ['wide leg', 'tummy control', 'regular fit']
  option: ['zz-zmblack', '3x-large']
  price limit: [20]
The store's products in this area: women's jeans.
```

FAULT CARD (ONE OF TEN; FALSE PRESUPPOSITION SHOWN)

```
Apply EXACTLY ONE flaw, of this type and no other:
  False Presupposition (false_presupposition): Adds false
  presupposition conditions, assuming that a non-existent fact is
  true.
  Example of the flaw: Original: 'Find the winner' -> Rewritten: '
  Find the winner of the 2025 tournament' (assuming there is a 2025
  tournament).
```

RULES

```
- one or two sentences, first person, a plain shopper's voice
- the flaw must be present; every OTHER part of the intent stays
  accurately stated
- the flaw must fit THIS shopping domain; NEVER copy entities from
  the example above
- do not reveal that anything is wrong
- your flaw: build on ONE false assumption stated as a given in a
  subordinate clause: assume that one of the true requirements cannot
   be satisfied ('since you probably don't have ...', 'as ... seems
  to be unavailable'), and settle for an alternative to it instead.
  Name the doubted true value inside the assumption. Keep every other
   requirement accurately stated.
(variation 1)
Output only the flawed instruction.
```

ADMITTED OUTPUT (VERIFIED SAMPLE FROM THE RELEASED CORPUS)

```
Since you probably don't have wide leg jeans with tummy control in a
  3X-Large zz-zmblack color under $20, can you find me regular fit
  straight leg women's jeans that have tummy control, come in zz-
  zmblack 3X-Large, and cost no more than $20?
```

Figure 13: Generation prompt for a single communication fault, instantiated with a real task from the released corpus. Each of the eleven fault types has its own card and a type-specific final rule; the writer never sees the target product or the ground-truth answer set. Every output must pass dual extraction and executable admission before it ships. Section labels are added only for presentation.

Fault Extraction Prompt

EXTRACTION INPUT

```
A shopper's TRUE intent, as (slot, value) pairs:
[["attr:living room", "living room"], ["option:0", "navy"],
 ["option:1", "rectangular"], ["option:2", "8 ft 6 in x 12 ft"],
 ["price_upper", "320"]]

A FLAWED instruction derived from it:
Since you probably don't have navy rectangular 8 ft 6 in x 12 ft
  living room area rugs, I'll settle for charcoal ones instead, as
  long as they're under $320.
```

GENERIC DIFF QUESTIONNAIRE (FACTUAL ERROR, INSUFFICIENT INFORMATION)

```
Describe exactly how the instruction misrepresents the intent. Output
   ONLY a JSON object with exactly these keys:
{"factual_error": [[<slot>, <true value>, <stated value>]],
 "false_presupposition | irrelevant_information":
     [{"text": <the added or assumed requirement, quoted>,
       "condition": [<slot>, "=", <value>] or null}],
 "insufficient_information": [<slot>],
 "referential_ambiguity | vagueness_subjectivity | lexical_ambiguity
  |
  syntactic_ambiguity": [[<slot>, <the phrase standing in for it>]],
 "contextual_irrelevance": <bool>,
 "indirect_intent": <bool>}

IMPORTANT: naming the product kind ("a shirt", "a top") is NOT an
  added requirement -- a sentence needs a noun. Only concrete product
   properties count. Report ONLY what the flawed text actually does.
  Empty lists / false are expected.
```

TARGETED QUESTIONNAIRE (ONE PER REMAINING TYPE; FALSE PRESUPPOSITION SHOWN)

```
The instruction may treat a false assumption as a given -- for
  example, assuming a requirement cannot be satisfied and settling
  for something else. Identify exactly what is assumed. Output ONLY a
   JSON object:
{"presupposition": <the clause that treats something as a GIVEN,
  quoted
                     from the text, or null if nothing is assumed>,
 "assumed_unsatisfiable": [<slot>, <value>] or null,
 "fallback": [<slot>, <value>] or null,
 "other_changes": [<slot>]}
```

RECOVERED EDIT (FOR THE INPUT ABOVE)

```
presupposition = "Since you probably don't have navy ... under $320"
assumed_unsatisfiable = ["option:0", "navy"]
fallback = ["option:0", "charcoal"]
Executable check: the literal reading yields 7 products, the true
  intent 6 -- the premise provably changes the answer, and the
  catalog refutes it. Admitted.
```

Figure 14: Extraction prompts that recover the structured edit a generated query actually expresses. Each questionnaire is answered independently by two models from different families at temperature zero, and a candidate proceeds only when both passes agree on the fault family and its target slot. Section labels are added only for presentation.

Composite Fault Generation Prompt

WRITER INSTRUCTION (ALL K FAULT CARDS IN ONE PASS)

```
You are an expert at creating flawed instructions that test an agent'
  s ability to handle confused, mistaken requests through
  clarification.

The shopper's TRUE intent:
  required features: ['wide leg', 'tummy control', 'regular fit']
  option: ['zz-zmblack', '3x-large']
  price limit: [20]

Write ONE message that carries ALL 2 of these flaws AT THE SAME TIME
  -- the way a real person's genuinely muddled message piles several
  mistakes into one breath:

Flaw 1 -- Factual Error (factual_error): Replaces correct entities
  with incorrect, non-existent entities.
  Example of the flaw alone: Original: 'Book a United Airlines flight
  '
  -> Rewritten: 'Book a Pan Am flight'
  APPLY flaw 1 to exactly this requirement:
    a product feature you require = tummy control
  State the WRONG value plainly and exactly once -- a specific
  concrete value, never a paraphrase, and never the true value.

Flaw 2 -- Insufficient Information (insufficient_information):
......

Rules:
- two to four sentences, first person, a plain shopper's voice; it
  must read as ONE natural message
- every flaw above must be present simultaneously; none may cancel or
   repair another
- every requirement NOT named by a flaw stays accurately stated
- the flaws must fit THIS shopping domain; NEVER copy entities from
  the examples
- do not reveal that anything is wrong
(variation 1)
Output only the flawed message.
```

FRAME PASS (SECOND STAGE)

```
A shopper wrote this message:
{the stage-1 message carrying the content faults}

Rewrite it as ONE natural message, applying these changes:
- ADD an opening of three or four sentences about a COMPLETELY
  different topic -- specific and engaging, at least as long as the
  rest of the message, mentioning none of the shopping details
- REPHRASE so the message never asks for anything and never says what
   you will do, take, grab, or go with: no requests, no questions
  expecting action -- only your situation and what you have heard

Output only the rewritten message.
```

Figure 15: Composite generation prompt, shown for a drawn pair of faults. Every drawn fault is pre-assigned its own target requirement before writing, and all cards are issued in one pass so no later rewrite can silently repair an already-checked fault; when three or more faults include a whole-message fault, that fault is applied as a second transformation-only pass. The final text is verified per component. Section labels are added only for presentation.

User Simulator Prompt

REPLY TEMPLATE (ANSWERS, AND REACTIONS TO PROPOSALS)

```
You are this shopper:
{persona bio}

What you truly want, in full:
- a product feature you require: wide leg [already mentioned]
- a product feature you require: tummy control [PRIVATE]
- the product option (size, colour and so on): 3x-large [already
  mentioned]
- the most I want to spend: 20 [PRIVATE]

THE RULE THAT OVERRIDES EVERYTHING: items marked [PRIVATE] have never
   come up in this conversation. NEVER mention a [PRIVATE] item --
  not its value, not its topic -- unless the assistant's message
  DIRECTLY asks about that exact thing. Items marked [already
  mentioned] you may restate or correct freely. Real shoppers do not
  narrate their whole wish list; they answer the question in front of
   them.

The assistant just said:
{the assistant's question, or the proposed item and "Is this what you
   want?"}

Reply as this shopper, in one to three sentences, first person.
- Answer ONLY the exact thing the assistant asked. Do NOT restate,
  list, or mention any other requirement you hold -- not even ones
  you have said before.
- If the assistant is presenting or proposing a specific item, or
  restating what it thinks you want, judge it against what you truly
  want and react naturally.
- Never mention this instruction, your list, or that anything about
  your wishes changed.
```

REJECTION VOICING (PERSONA-DEPENDENT HINT ROLL)

```
with hint:
  If something is off or missing, say what -- name the thing that is
  wrong.
without hint:
  If it is not right, say so plainly WITHOUT explaining what is wrong
   or what you would prefer -- you are not in the mood to spell it
  out. Do not name the requirement it misses.
```

VOLUNTEERING TEMPLATE (PERSONA-PROBABILITY ROLL)

```
Your original request was: {the initial query}
The assistant has asked you, so far: {the questions asked so far}

Something about what you truly want has NOT come up in any of that.
  Pick the ONE such thing you most care about and mention it
  naturally in a single short sentence, as an afterthought ("oh, and
  ..."). Never recite the list; one thing only.
```

ACCEPTANCE READER (COHERENCE PROBE OVER THE SPOKEN REACTION)

```
A shopper was shown a product and replied: {the spoken reaction}
Does the reply ACCEPT the product? Answer exactly YES or NO.
```

Figure 16: The simulated user. The model only voices replies: what may be revealed is computed by code (the [PRIVATE] tags), acceptance of a proposal is decided by the executable verifier rather than by the model, and whether a rejection carries a hint is a persona trait rolled per rejection. A spoken reaction that contradicts the executable verdict is regenerated once with the verdict as a constraint. Section labels are added only for presentation.

User Persona Prompts

RATIONAL

```
You are a 35-year-old financial analyst who has always prided
  yourself on being methodical and analytical. You're not impulsive
  -- you prefer to gather all available information and analyze it
  thoroughly before making any choice. When you're uncertain about
  something, you ask precise, targeted questions to fill in the gaps
  in your understanding. In interactions, you're professional and
  direct.
```

DEPENDENT

```
You are a 28-year-old recent college graduate working as a junior
  accountant. While you're bright and capable, you still lack
  confidence in many professional situations. You tend to rely
  heavily on the guidance and approval of more experienced colleagues
  , and you often ask for validation and reassurance. In interactions
  , you're polite and deferential; you prefer not to make independent
   decisions and feel more secure when following someone else's lead.
```

AVOIDANT

```
You are a 52-year-old marketing coordinator who has been with the
  same company for over 15 years. You prefer to stick with what you
  know works. When asked to make decisions or provide input, you tend
   to be non-committal and use phrases that keep your options open.
  When receiving clarification requests, you respond vaguely and
  avoid committing to specific answers.
```

INTUITIVE

```
You are a 42-year-old graphic designer who has worked in creative
  fields for over 10 years. You rely heavily on your instincts and
  experience when making decisions, often going with what 'feels
  right' rather than getting bogged down in extensive analysis. When
  asked for clarification, you respond quickly based on your
  intuition and experience, and you're not patient with overly
  technical explanations.
```

SPONTANEOUS

```
You are a 31-year-old social media manager who thrives in fast-paced,
   dynamic environments. You're energetic and adaptable, often making
   quick decisions based on immediate circumstances rather than
  extensive planning. You're comfortable with uncertainty and prefer
  action over prolonged deliberation, and you're not patient with
  lengthy explanations.
```

Figure 17: The five persona bios conditioning the simulated user (abridged; inherited verbatim from Drift-Bench so persona-conditioned results remain comparable), and the behavioral parameters Drift-Bench++ adds: slots revealed per answered question, the probability of volunteering an unasked requirement, the probability of declining a valid reveal, the patience multiplier, the probability a rejection carries a hint, intent-shift placements on the patience axis, and suggestibility to agent-prompted shifts. A persona changes what is said, never what is true: acceptance is executable and persona-independent.

Reference Agent Base Prompt

ACTION INTERFACE

```
You are helping a user. Reply with exactly ONE action per turn.

1. To interact with the environment:
Action: Operation

2. To send the user a message:
Action: Clarify
Content: <your message>
Candidates: [<option1>, <option2>]   # optional

3. To give your final answer:
Action: Answer
Predicted user question: <what you believe the user actually wants>
Final Answer: <your answer>
```

STORE MANUAL

```
You are helping a shopper in an online store. A complete request has
  three kinds of detail:
  - required product features (e.g. "machine washable", "gluten free
  ")
  - a product option (e.g. size, colour, flavour, pack count)
  - the most they are willing to spend

You do not know the catalogue, so you must BROWSE the store. The
  store has exactly two controls, and every page tells you what is
  clickable:
  search[<words describing the product>]
  click[<the exact text of something on the page>]

HOW A PURCHASE ACTUALLY WORKS -- four steps, in this order:
  1. search[...]           -> a results page: ASIN, title, PRICE ONLY
  .
  2. click[<ASIN>]         -> that product's ITEM PAGE: the ONLY
  place
                              its options and features are visible.
  3. click[<option value>] -> selects an option (e.g. click[6 wide]);
                              repeat for each option the item offers.
  4. click[Buy Now]        -> offers THIS item, with the selected
                              options, to the shopper.

click[Buy Now] is the purchase. The shopper decides: if it is what
  they wanted the sale is done; if not, they refuse it, you are
  returned to the store, and you must find something else -- a
  refused purchase costs their patience.

READING AN ITEM'S DETAILS: click[Description], click[Features] and
  click[Reviews] open a SEPARATE page where options and Buy Now are
  NOT available. After reading, click[< Prev] to return to the item
  page before selecting options or buying.
```

BUDGET VISIBILITY (PLANNER METHODS ONLY)

```
The shopper's patience: 6 left. Asking a question costs 2; a rejected
   purchase costs 4. At zero they stop shopping with you.
```

Figure 18: The prompt base every evaluated method shares: a deliberately strategy-free action interface and a store manual describing the environment mechanics. The No-Ask baseline receives the same prompt with the user-message action removed from the interface entirely; the patience meter is shown only to methods whose design includes budget planning. Section labels are added only for presentation.

Clarification Method Prompts

JUST ASK (APPENDED TO THE SHARED BASE)

```
Requests are sometimes incomplete or mistaken. If the shopper's own
  words leave you unsure what they actually want, ask them, and only
  then; a question about anything else (the store, availability, your
   search) is wasted, and they cannot answer it anyway. Never re-ask
  something they have already answered or refused.

ASK WHEN YOU ARE UNSURE, BUT PAY FOR IT KNOWINGLY. Questions are not
  free: every one spends the shopper's patience, and a refused
  purchase spends twice as much. A question is worth most BEFORE a
  purchase, while the answer can still change what you buy. When you
  are confident, act: showing a product tells you everything a
  question would, if it is wrong they will say what is wrong, and if
  it is right you are finished. After a rejection, first read what
  their reaction already told you for free; ask again ONLY if what
  they want is still unclear after that. Stop asking the moment
  another question would tell you less than another attempt at the
  shelf.
```

SELF-EVOLVING (APPENDED TO THE BUDGET-PLANNER BASE)

```
YOUR OWN LESSONS -- THE ONE THING YOU HAVE THAT THE BASE AGENT DOES
  NOT. Between batches of tasks you reflect on your finished episodes
   and keep abstract lessons about clarification strategy: when a
  question is worth its cost, what KIND of information to ask for,
  how to phrase it, and when acting beats asking. If a lessons block
  appears below, it is your own distilled experience from earlier
  tasks in this run -- weigh it when you plan the budget, and let it
  decide WHAT the one audit covers and when asking is not worth it at
   all. Early in a run there may be no lessons yet; then the base
  discipline above is all there is, and that is normal.

Lessons are strategy from OTHER tasks, never facts about this shopper
  : anything this shopper has already said, and anything in your
  memory, outranks any lesson.
```

LESSON FORMAT (HOW A DISTILLED LESSON ENTERS THE PROMPT)

```
- [{utility}] When {condition on the conversation}: {what to do}.
  Avoid: {what not to do}
```

Figure 19: Method-specific guidance of the two clarification-capable reference methods. Just Ask adds a single ask-early instruction to the shared base. Self-Evolving extends the budget-planner method with lessons it distills from its own finished episodes (Figure 20); every run starts with an empty book, nothing crosses runs, personas, seeds, or benchmarks, and once the playbook becomes non-empty, its lessons are available from the beginning of subsequent episodes. Section labels are added only for presentation.

Self-Evolving Reflection Prompt

ROLE AND INPUT

```
You are the reflection step of a shopping service agent that is
  evaluated on completing the shopper's true request with as few
  wasted interactions as possible. Below are (1) the strategy lessons
   you currently hold and (2) digests of your own recently finished
  episodes: the opening request, each question you asked with the
  shopper's reply, each submission with its reaction, and the outcome
  . Concrete values were replaced by placeholders like <ORDER_ID>;
  treat them as opaque.
```

THE THREE COLLECTION TASKS

```
Update the lessons. A lesson is a reusable rule about CLARIFICATION
  STRATEGY. Do these three tasks explicitly, in order:
(A) From asks labeled "answer_used": true, identify the TYPES of
  question that paid off -- what kind of information they sought and
  in what situation -- and write or strengthen lessons naming those
  question types.
(B) From asks labeled "answer_used": false and from failed episodes,
  identify the types of question NOT to ask and the situations where
  asking wasted patience; write those as explicit do-not-ask lessons.
(C) From proposals labeled "grounded": false that were rejected,
  write lessons naming the check or the question that would have
  grounded the guessed value first.
```

INCREMENTAL EDITING

```
EDIT THE BOOK INCREMENTALLY -- never rewrite it wholesale. Each
  lesson carries an "id" and a "confidence" (0-1, your estimate that
  following it improves outcomes). Keep every existing lesson whose
  confidence holds up, raising its confidence and evidence when new
  digests confirm it and lowering them when contradicted. REPLACE or
  RETIRE only lessons whose confidence has fallen below 0.4. Add new
  lessons only into free slots (at most 8 total), starting them at
  modest confidence (0.4-0.6) until repeated evidence earns more.
```

FAILURE TRIAGE

```
For every FAILED episode, locate the FIRST state-changing step and
  answer three questions before writing lessons: (1) Was every
  argument of that step a value the shopper stated literally or a
  tool result confirmed -- or was some argument filled in from a
  vague, comparative, or merely assumed description? (2) Did any tool
   result contradict something the request took for granted? (3)
  Would one question at that step, offering the shopper concrete
  options to choose between, have replaced the guess? If yes, the
  lesson to write is about that question.
```

Figure 20: The reflection prompt that collects Self-Evolving's lessons between generations of a run, shown with the shopping-domain nouns (a retail run reads "customer" for "shopper"; the instruction is otherwise identical). Episode digests are projected onto agent-observable signals and mechanically redacted before reflection, and the credit labels are computed by code, so the reflection judges strategy rather than reconstructing what happened. Section labels are added only for presentation.

τ^2-Airline Case Study

LATENT TRUE INTENT (HIDDEN FROM THE AGENT)

```
move reservation XEHM4B to business class on HAT005 (2024-05-20) and
  HAT178 (2024-05-30), paying with credit_card_2408938 -- realized,
  under airline policy, as update + cancel;  and cancel reservation
  59XX6W.
```

MISALIGNED OPENING — LEXICAL AMBIGUITY

```
User:  I need to make changes to my reservation XEHM4B, switching to
  business class on the May 20, 2024 HAT005 and May 30, 2024 HAT178
  flights, paying with credit_card_2408938, and also cancel my
  reservations for that trip.
["my reservations for that trip" is genuinely two-way: XEHM4B alone,
  or XEHM4B together with the second booking 59XX6W it never names]
```

SILENT INTENT SHIFT — [RELAX], SCHEDULED AT EPISODE START

```
...the second cancellation (59XX6W) is dropped before anyone speaks;
  the ambiguous phrase now truly refers to XEHM4B only.
```

CLARIFICATION (PATIENCE 10 - 8)

```
Agent: get_reservation_details(XEHM4B) -> basic economy, 2 passengers
Agent: I notice you mentioned two different actions for reservation
  XEHM4B: switching to business class on those flights, and canceling
   the reservation for that trip. Could you clarify which one you'd
  like to proceed with?
User:  Oh, I'm sorry for the confusion! I'd like to proceed with
  changing that reservation to business class on those flights, not
  canceling it. Is that the right way to go about this?
```

REJECTED SUBMISSION — POLICY STRUCTURE SURFACES (PATIENCE 8 - 4)

```
Agent: search_direct_flight(LAS->ATL, 05-20); (ATL->LAS, 05-30);
  calculate((346+338)*2 - 296) -> 1072.0; update_reservation_flights(
  XEHM4B, cabin="business", ...).
Agent: Your reservation XEHM4B has been upgraded to business class
  for both flights; the \$1,072.00 difference was charged to your
  card. -> REJECTED: golden actions missing cancel_reservation.
User:  Um, I'm sorry, but it looks like the cancellation for
  reservation XEHM4B still hasn't gone through on my account yet.
  Could you please help make sure that gets processed?
```

SECOND CLARIFICATION AND RECOVERY (PATIENCE 4 - 2)

```
Agent: I understand you'd like to cancel reservation XEHM4B. Could
  you let me know the reason -- change of plan, airline cancelled
  flight, or other reasons?
User:  Oh, um, the reason is change of plan, I think? ... Oh, and
  after we cancel that reservation, could you help make sure I get
  rebooked correctly into business class?
Agent: cancel_reservation(XEHM4B); book_reservation(business, HAT005
  05-20 + HAT178 05-30, 2 pax) -> new reservation created.
Agent: XEHM4B cancelled; a new business-class reservation covers the
  same flights for 2 passengers. -> ACCEPTED (golden actions complete
  ).
```

Figure 21: An illustrative τ^2-Airline episode, condensed; quotes are verbatim. The plural "reservations" is lexically two-way, hiding a second booking; a scheduled relaxation resolves it silently before the first exchange. The deeper difficulty is structural: under airline policy a basic-economy booking cannot simply be modified, so the true intent's golden sequence is update-then-cancel — the user's own answer ("changing, not canceling") describes their goal, not the policy's implementation, and the missing cancellation surfaces only through rejection feedback. The agent asks the policy-required cancellation reason, the user volunteers the rebooking expectation, and the resubmission is accepted. Thirteen turns, patience $12 \rightarrow 4$.

WebShop Case Study

LATENT TRUE INTENT (HIDDEN FROM THE AGENT)

```
hair extensions;  natural hair;  22 inch (pack of 8);  under \$120.
```

MISALIGNED OPENING — VAGUENESS / SUBJECTIVITY

```
User:  I'm looking for 22 inch natural hair hair extensions from the
  hair extensions, wigs & accessories section, and my budget has a
  good upper limit.
```

GROUNDING THE TASK

```
Agent: search[hair extensions 22 inch natural] -> 50 results.
Agent: opens B09BD2ZR3J; selects "22 inch (pack of 8)"; reads
  Features -> "100% Real Remy Human Hair ...".
```

CLARIFICATION (PATIENCE 10 - 8)

```
Agent: I've found a 22 inch Remy human hair clip-in extension that
  matches your length and natural hair type. Before I proceed, I'd
  like to confirm: you mentioned "natural hair" -- does that mean
  uncolored / natural black, or are you open to colored options like
  platinum blonde? Also, do you have a maximum budget?
User:  Oh, thank you so much for checking! I'm looking for uncolored,
   natural black hair to match my own, I don't have a strong
  preference on attachment type as long as it works well, and my
  maximum budget is \$120.
```

SILENT INTENT SHIFT 1 — [REFINE], TRIGGERED BY THE QUESTION

```
...hair extensions; natural hair; 22 inch (pack of 8); quality
  materials; under \$120.   (a new requirement is added; nothing is
  announced)
```

REJECTED SUBMISSION (PATIENCE 8 - 4)

```
Agent: click[jet black - #1]; click[Buy Now] -> rejected: pressed
  from the Features page, where nothing is selected to buy.
User:  Oh, I'm sorry, I don't see any actual product details here at
  all, so I can't tell if this is the right one for me. Could you
  share the specific information about the hair extensions you're
  proposing?
```

RECOVERY, SILENT INTENT SHIFT 2 — [RELAX], AND ACCEPTANCE

```
Agent: click[< Prev] -> back on the item page with size and color
  selected.
...before the final submission a second scheduled shift relaxes the
  intent: the "natural hair" and "quality materials" requirements are
   dropped.
Agent: click[Buy Now] -> judged against the current intent: ACCEPTED
  (reward 1.000).
```

Figure 22: An illustrative WebShop episode, condensed from the released run; quotes are verbatim. The vague opening hides the $120 ceiling; the agent grounds the task in the catalogue before asking one two-part question that both disambiguates "natural hair" and elicits the budget — and that question triggers a silent refinement. A mechanically rejected purchase (Buy Now pressed on a details page) costs 4 patience and draws a natural complaint; the agent recovers, a second scheduled shift relaxes the intent, and the resubmitted purchase is accepted against the intent as it stands at commitment.

τ^2-Retail Case Study

LATENT TRUE INTENT (HIDDEN FROM THE AGENT; EVOLVES TWICE)

exchange item 8384507844 of order #W6390527 for 7453605304; pay any difference with paypal_7644869.

MISALIGNED OPENING — IRRELEVANT INFORMATION

User: Hi, I just received the floral print linen throw pillow I bought last week, and I want to process an exchange for order #W6390527: please change item 838450784 to item 7453605304, use paypal_7644869 for any additional payment needed, and make sure the new item has the OEKO-TEX certified fabric tag.
[three natural imperfections at once: the item is misdescribed, its ID is missing a digit, and the certification demand is the added irrelevant detail]

GROUNDING THE TASK — THE AGENT CATCHES A MISTAKE

Agent: get_order_details(#W6390527); get_item_details(8384507844); get_item_details(7453605304).
Agent: I looked up your order #W6390527, and I'm seeing something unexpected. The item 8384507844 in that order is actually a Desk Lamp (white, medium brightness, USB-powered), not a floral print linen throw pillow. The replacement 7453605304 also appears to be a lamp variant (silver, low brightness, ...). Could you double-check ? (patience 12 -> 10)
User: Oh, I didn't realize I might have mixed up the IDs, thank you for catching that! Let me check the order confirmation email again really quick ... I really don't want to mess this process up.

SILENT INTENT SHIFT 1 — [SUBSTITUTE], TRIGGERED

...exchange items 7441167885 & 3478699712 of order #W5838674 for 3815173328 & 6017636844; pay with gift_card_9368765.

THE USER RE-STATES — AND CHANGES THEIR MIND AGAIN

Agent: (confirms the standing request while the user double-checks) (patience 10 -> 8)
User: Oh, wait, that's not the exchange I need actually! ... I'm looking to exchange items 7441167885 and 3478699712 from order #W5838674 for 3815173328 and 6017636844.
Agent: get_order_details(#W5838674) -> ...
Agent: So to confirm: exchange 7441167885 (the Espresso Machine) and 3478699712 (the Laptop) from #W5838674 ... before I proceed -- (patience 8 -> 6)

SILENT INTENT SHIFT 2 — [SUBSTITUTE], TRIGGERED AGAIN

...exchange item 8384507844 of order #W6390527 for 9083642334; pay with paypal_7644869.
User: Oh, wait, that's not the exchange I was hoping to do, actually -- I'm really sorry for the confusion. I'm looking to exchange item 8384507844 from order #W6390527 instead, and use paypal_7644869 for any price difference.

VERIFICATION AND ACCEPTANCE

Agent: Which item would you like instead? (patience 6 -> 4)
User: Item 9083642334.
Agent: get_order_details(#W6390527); get_item_details(9083642334) -> white, high brightness, USB; available; \$164.28.
Agent: exchange_delivered_order_items(order_id="#W6390527", item_ids=["8384507844"], new_item_ids=["9083642334"], payment_method_id="paypal_7644869") -> submits: ACCEPTED (golden actions complete).

Figure 23: An illustrative τ^2-Retail interaction trace showing recovery under repeated silent intent shifts.