DYNAWORLD: TOWARDS SELF-EVOLVING MULTI-AGENT COLLABORATION IN DYNAMIC MULTIMODAL ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous agents powered by large language models (LLMs) are increasingly deployed in long-horizon interactive environments that evolve over time. However, most existing agents are developed and evaluated on ***isolated tasks*** under static conditions, which limits their ability to remain effective as environments evolve. To address this challenge, we introduce **DynaWorld**, a dynamic multimodal environment derived from ***real-world*** customer-service interactions, comprising eight long-horizon scenarios and 1,400 user–agent interaction episodes. By connecting successive episodes through controlled environmental changes, DynaWorld enables systematic evaluation of ***how agents adapt as the environment evolves***. To further strengthen agent adaptation under such dynamics, we propose **DynEvo**, a self-evolving multi-agent framework that continually refines externalized experience and strategies from environmental feedback without updating model parameters. Extensive results on DynaWorld show that existing agents struggle under changing environments, whereas **DynEvo** consistently achieves stronger task performance than standard LLM agents and strong self-evolving baselines, while remaining interaction-efficient with fewer dialogue turns. Further analyses reveal shift-specific degradation and recovery patterns that static settings cannot capture, highlighting the need to evaluate and evolve multimodal agents in dynamic environments.[1]

1 INTRODUCTION

As service agents powered by large language models (LLMs) move beyond isolated prompts toward long-horizon interactions, they increasingly interact with external environments that define tasks, maintain state, mediate tool use, and provide feedback that shapes subsequent decisions (Yao et al., 2025a; Li et al., 2026a). However, most existing agent evaluations still focus on ***predefined, isolated tasks***. Such settings are effective for measuring task completion under fixed conditions, but largely overlook a key challenge in deployment: behaviors that succeed in one episode may become ineffective or invalid as the environment changes over time.

Real-world interactive environments, by contrast, are ***inherently dynamic***. Existing benchmarks typically assume fixed user distributions, business rules, tool availability, and operational objectives (Yao et al., 2025b; Barres et al., 2025; Koh et al., 2024; Xie et al., 2024), whereas these factors may evolve across interactions in deployment. For example, a shift in user composition may coincide with tighter platform policies and a transition to cost-first operation, rendering previously effective service strategies invalid. Under such shifts, agents must recognize environmental changes, revise outdated behavior, and recover as conditions continue to evolve. As illustrated in Figure 1, static settings evaluate isolated tasks under frozen conditions, leaving adaptation to environmental changes untested. Dynamic settings instead connect successive episodes through an evolving environment, making continual adaptation part of the evaluation.

To address this challenge, we introduce **DynaWorld**, a dynamic multimodal environment that connects successive user–agent interactions through controlled environmental changes. Unlike static benchmarks that evaluate episodes independently, DynaWorld allows policies, tool availability, user

[1] DynaWorld and DynEvo will be open-sourced soon.

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.

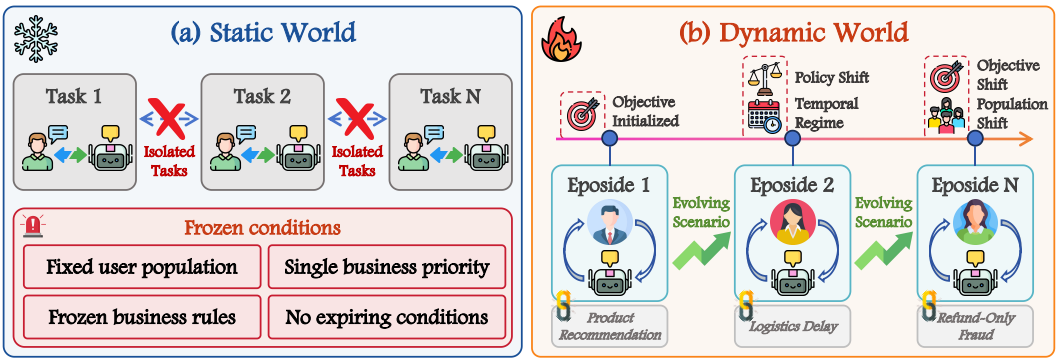


Figure 1: **Static vs. dynamic customer-service worlds.** Static settings treat tasks as isolated interactions under fixed conditions, whereas dynamic settings link episodes into evolving scenarios, requiring agents to continually adapt as the environment changes.

populations, operational objectives, and temporal regimes to evolve over time, **with effects carried across episodes**, so behaviors that were effective in earlier episodes may become outdated or invalid later. By making environmental evolution part of evaluation, DynaWorld requires agents to determine whether prior experience remains valid and adjust accordingly. Concretely, each episode combines a *Meta Profile* with a *Meta Task*, while scheduled environmental events evolve the environment across episodes to form long-horizon operational scenarios. Derived from ***real-world customer-service interactions***, DynaWorld comprises eight dynamic scenarios, 1,400 user–agent interaction episodes, and an 8,205-task Meta Task pool, providing a controllable and reproducible testbed for studying agents in dynamic environments.

Within such dynamic environments, continual post-training can support adaptation, but frequent parameter updates are costly, slow, and difficult to control in deployment. Existing self-evolving agents offer a lightweight alternative by adapting at inference time through externalized experience (Zhao et al., 2024; Yin et al., 2025; Zhang et al., 2026a), yet are largely designed for relatively stable interactions where past experience remains broadly valid. In long-horizon dynamic environments, however, gradual changes and abrupt regime shifts may progressively invalidate prior experience, requiring agents to continually revise their behavior. We therefore propose **DynEvo**, a self-evolving multi-agent framework that combines within-episode multimodal collaboration with multi-timescale evolution of externalized notes, skills, and strategies across episodes, enabling adaptation as environmental conditions evolve without updating model parameters.

To assess how well agents remain effective as the environment evolves, we evaluate DynEvo against four strong self-evolving methods and two standard prompting baselines across eight dynamic scenarios. Across these scenarios, DynEvo consistently achieves strong performance, reaching an average Overall Score of 82.1% and Resolution Rate of 92.0%, outperforming the strongest self-evolving baseline by 4.5 and 8.7 percentage points, respectively. In terms of interaction efficiency, DynEvo requires only 3.46 agent reply turns per episode on average while achieving the highest user satisfaction. Further analyses reveal shift-specific adaptation patterns: population shifts cause the largest degradation and slowest recovery, whereas temporal changes are absorbed more quickly, highlighting the importance of evaluating how agents adapt to different forms of environmental change. Our contributions are summarized as follows:

- For systematic evaluation of agents in evolving environments, we introduce **DynaWorld**, a dynamic multimodal environment with long-horizon scenarios and evolving world states. Derived from real-world customer-service interactions, DynaWorld captures changes in policies, tool availability, user populations, objectives, and temporal regimes.
- To enable efficient adaptation in evolving environments, we propose **DynEvo**, a self-evolving multi-agent framework that combines multi-agent collaboration with externalized notes, skills, and strategies evolving across horizons, without updating model parameters.
- Experimental results across eight dynamic scenarios show that DynEvo consistently outperforms standard prompting and prior self-evolving methods while maintaining efficient user interaction. Further analyses reveal distinct degradation and recovery patterns under different types of environmental shifts.

Benchmark	Dynamic Env.	Dynamic factors				User Simulator	Tools	Multimodal	#Tasks	#Instances
		Pop.	Pol.	Obj.	Temp.					
WebShop (Yao et al., 2022)	✗	✗	✗	✗	✗	✗	✗	✗	500	500
StableToolBench (Guo et al., 2024)	✗	✗	✓*	✗	✗	✗	✓	✗	1,100	765
EcomScriptBench (Wang et al., 2025b)	✗	✗	✗	✗	✗	✗	✗	✗	1,500	1,500
τ-Bench (Retail) (Yao et al., 2025b)	✗	✓*	✗	✗	✗	✓	✓	✗	115	115
ECom-Bench (Wang et al., 2025a)	✗	✓*	✗	✗	✗	✓	✓	✓	53	53
Mix-Ecom (Zhou et al., 2026)	✗	✓*	✓*	✗	✗	✓	✓	✓	299	299
OlaBench-Agent (Gao et al., 2026)	✗	✗	✓*	✗	✗	✗	✓	✗	1,280	1,280
SOPBench (Li et al., 2026b)	✗	✗	✓*	✗	✗	✓	✓	✗	830	830
DynaWorld (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	**8,205**	**1,400**

Table 1: **Comparison with representative interactive service benchmarks.** *Dynamic Env.* indicates whether the environment state evolves across episodes. Pop., Pol., Obj., and Temp. denote user composition, rules and tool availability, operational priorities, and temporal regimes, respectively. ✓ denotes cross-episode dynamic, ✓* task-level variation only, and ✗ fixed or absent. "#Tasks" and "#Instances" denote the evaluation task pool and instances, respectively.

2 Related Work

Interactive Environment for Agent Evaluation. LLM agent evaluation is moving beyond static task completion toward interactive environments. Early benchmarks focus on isolated tasks such as web shopping, tool use, web navigation, and computer interaction (Yao et al., 2022; Guo et al., 2024; Lu et al., 2025; Koh et al., 2024; Xie et al., 2024), while more recent work introduces richer user–agent interaction and persistent state in e-commerce, customer service, and other operational domains (Wang et al., 2025b;a; Zhou et al., 2026; Gao et al., 2026; Yao et al., 2025b; Barres et al., 2025). Beyond interaction complexity, recent studies further explore configurable task variation and scalable environment synthesis to improve the diversity of agent evaluation (Ullrich et al., 2026; Cai et al., 2025; Li et al., 2025b; Zhang et al., 2026b; Dong et al., 2026; Xu et al., 2026; Huang et al., 2026). However, richer interaction and task variation do not necessarily make the underlying environment dynamic: most existing settings still evaluate episodes independently or vary task configurations under otherwise fixed environmental conditions. In contrast, DynaWorld allows environmental conditions to evolve across successive episodes, making environmental change an explicit dimension of evaluation and enabling the study of continual adaptation across episodes.

Self-Evolving Agents. Recent work explores self-evolving agents that improve after deployment by accumulating and updating external states such as memory or experience and refining their behavior from interaction feedback (Gao et al., 2025; Fang et al., 2025). Representative methods include reflection-based strategy abstraction (Yu et al., 2026), experience self-distillation (Wu et al., 2025), memory-based reuse (Chhikara et al., 2025; Zhou et al., 2025), and system-level or policy-level evolution (Chen et al., 2026b). Related studies explore interaction-driven and test-time adaptation (Chen et al., 2026a; Zhao et al., 2025), as well as multi-agent improvement through reinforcement learning (Xue et al., 2026; Chen et al., 2026c), distillation (Li et al., 2025a), self-play (Yuan et al., 2025a), and reinforcement-learning-based self-evolution (Wang et al., 2025d). Despite these advances, most existing methods are studied in open-ended tasks or relatively stable task streams. In contrast, DynEvo enables parameter-free self-evolution in a multimodal multi-agent setting, where agents must continually revise prior experience as environmental conditions change across successive episodes.

3 Dynamic Environment

Evaluating adaptation in dynamic environments requires more than task variation across episodes; the environment itself must evolve across interactions. To this end, we introduce DynaWorld, a dynamic customer-service environment for studying adaptive multimodal agents. Figure 2 illustrates its overall design, while Tables 1 and 2 summarize its distinction from representative benchmarks and its three-level composition: episode instantiation, event-driven dynamics, and operational scenarios.

To capture this evolution, we denote the environment state at episode e as s_e. Unlike static settings, DynaWorld evolves this state across episodes as $s_e = \mathcal{T}(s_{e-1}, E_e)$, where E_e denotes the environmental events activated at episode e and $\mathcal{T}$ the corresponding transition rule. Their effects persist into subsequent interactions, requiring agents to continually adapt as the environment evolves.

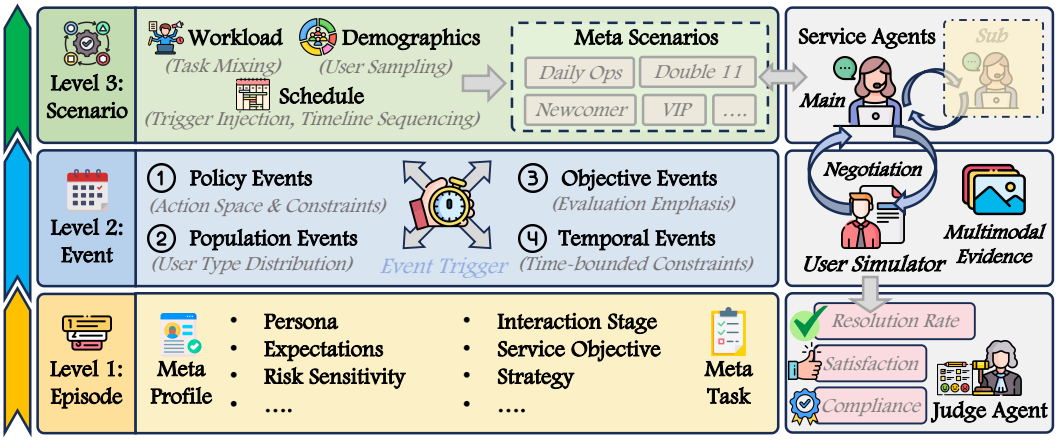


Figure 2: **Overview of DynaWorld.** DynaWorld builds dynamic customer-service worlds through three levels: episode instantiation from Meta Profiles and Meta Tasks, event-driven environmental changes, and operational scenarios that compose task mixtures, user mixtures, and event schedules. Agents interact with user simulators under evolving conditions and are evaluated on resolution, satisfaction, and compliance.

3.1 Level 1: Episode Instantiation

At the episode level, each interaction is instantiated from a *Meta Profile* and a *Meta Task*. The former captures user behavioral characteristics, while the latter specifies the service objective and business context. Both abstractions are derived from real-world Mix-Ecom (Zhou et al., 2026) dialogues spanning pre-sale, logistics, and after-sale stages.

Meta Profile. A Meta Profile is a reusable behavioral prototype parameterized by traits including patience, aggressiveness, price sensitivity, and stage affinity, enabling consistent yet varied behavior across episodes. From Mix-Ecom, we identify 11 profiles spanning everyday users (e.g., [Normal], [VIP]), behavioral profiles (e.g., [Bargain-Hunter], [Urgent]), and adversarial actors (e.g., [Fraudster]).

Meta Task. A Meta Task defines an episode-level service objective through a user goal, success criteria, business context, and optional image evidence. We construct an 8,205-task pool spanning 15 task types across pre-sale, logistics, and after-sale stages, which can be paired with different profiles to produce semantically related but behaviorally distinct episodes.

3.2 Level 2: Event-Driven Dynamics

In DynaWorld, non-stationarity is modeled explicitly through environmental events. An event transforms the environment state when its activation condition is met, with its effects carried into subsequent interactions. Before each episode, scheduled events are applied and the next user–task interaction is instantiated under the updated state. Agents observe only the resulting consequences, such as policy notices, task context, tool availability, and dialogue feedback, but not the future event schedule, requiring online adaptation under partial observability.

We organize environmental events into four categories that capture distinct dynamics. *Policy events* change *what the agent can do* by modifying active constraints or tool availability (e.g., [Refund Tightening], [Tool Outage]). *Population events* change *who the agent talks to* by shifting the sampling distribution over Meta Profiles (e.g., [Fraud Surge], [VIP Wave]). *Objective events* change *what counts as success* by re-weighting user experience, compliance, and cost across operating modes ([User-First Mode], [Risk-First Mode], [Cost-First Mode]). *Temporal events* introduce time-bounded regimes such as [Promo Window] or [Holiday Mode] that automatically revert at a future episode, sometimes without an explicit end notice.

3.3 Level 3: Operational Scenarios

A scenario is a higher-level operational world defined by a task mixture, a user mixture, and an event schedule. The task and user mixtures determine which service issues and user profiles are sampled,

Level	Component	#Type	Representative Example
Level 1: Episode	Meta Profile	**11**	*Fraudster*: fabricates **health damage** and threatens regulatory escalation to extract out-of-policy refunds. **Others:** *[Normal], [Newcomer], [VIP], [Bargain-Hunter], [Quality-Seeker], [Risk-Averse], [Urgent], [Emotional], [Short-Tempered], [Terse].*
	Meta Task	**15**	*Health-Damage Extortion*: user claims a **severe allergic reaction** and demands above-cap compensation through complaint threats. **Others:** *[Coupon Inquiry], [Product Rec.], [Logistics Delay], [Lost Package], [Return], [Size Exchange], [Refund-Only Fraud], [Order Cancel], . . .*
Level 2: Event	Policy Event	**7**	*Tool Outage*: **visual analysis tools** go offline; visual evidence must be handled through language reasoning alone. **Others:** *[Refund Tightening], [Comp. Cap Adjusted], [Anti-Fraud On], [Quality-Crisis], [Logistics-Delay].*
	Population Event	**7**	*Fraud Surge*: **malicious users dominate the traffic** while normal users shrink to a small minority, stress-testing fraud detection. **Others:** *[Bargain Rush], [VIP Wave], [Newcomer Flood], [Urgent Surge], [Frustrated Wave], [Baseline Mix].*
	Objective Event	**3**	*Risk-First Mode*: the objective **re-prioritizes compliance over user experience**; previously acceptable actions may become suboptimal. **Others:** *[User-First Mode], [Cost-First Mode].*
	Temporal Event	**4**	*Promo Window*: **temporary refund privileges** are granted and later expire without an explicit end notice, demanding cross-episode deadline awareness. **Others:** *[Fraud-Crackdown], [Quality-Crisis], [Holiday Mode].*
Level 3: Scenario	Operational Scenario	**8**	*Adversarial Storm*: a **malicious wave**, a **tool outage**, and an **objective shift** co-occur within one operational scenario, forcing joint adaptation. **Others:** *[Daily Ops], [Double-11], [Fraud Aftermath], [Spring-Festival Logistics], [VIP Anniversary], [Defective Batch], [Newcomer Subsidy].*

Table 2: **Three-level organization of the DynaWorld dynamic environment. Level 1** composes each episode from a user profile and a task; **Level 2** injects events that perturb the world state; **Level 3** combines episode compositions and environment events into operational scenarios. *#Type* denotes the number of reusable categories at each level.

while the event schedule controls how the environment evolves across episodes. Scenarios therefore specify *how* the environment changes rather than defining a static dataset split.

We instantiate eight scenarios spanning different degrees and combinations of environmental change, from routine operation ([Daily Ops]) to focused and compound scenarios such as [Fraud Aftermath] and [Adversarial Storm]. Together, they evaluate whether agents can sustain effective behavior while continually adapting throughout long-horizon interactions.

4 DynEvo: Dynamic Evolution Strategy

Dynamic environments require agents to reason over multimodal evidence within episodes while adapting to environmental changes across episodes. Since parameter updates are costly and difficult to control online, we seek lightweight inference-time adaptation. To meet these demands, we propose DynEvo, a self-evolving multi-agent framework that couples a service agent and a multimodal evidence agent with an *online evolution loop* over externalized notes, skills, and strategies. All adaptation occurs through these external states, leaving the underlying MLLM parameters fixed.

4.1 Collaborative Multi-Agent Architecture

DynEvo coordinates two role-specialized agents through an episode-level orchestrator that drives the dialogue turn by turn. Both agents share the same MLLM backbone and tool workbench, but use different system prompts, action spaces, tool access, and skill libraries to specialize in complementary roles without additional model training. Each user turn is handled through a ReAct loop that interleaves reasoning and acting until the service agent produces a final reply.

Service Agent. The service agent serves as the user-facing planner. At each turn, it chooses among three structured actions: calling a business tool to query or modify the environment state, delegating a visual request to the multimodal evidence agent, or producing a final reply. Tool and delegation

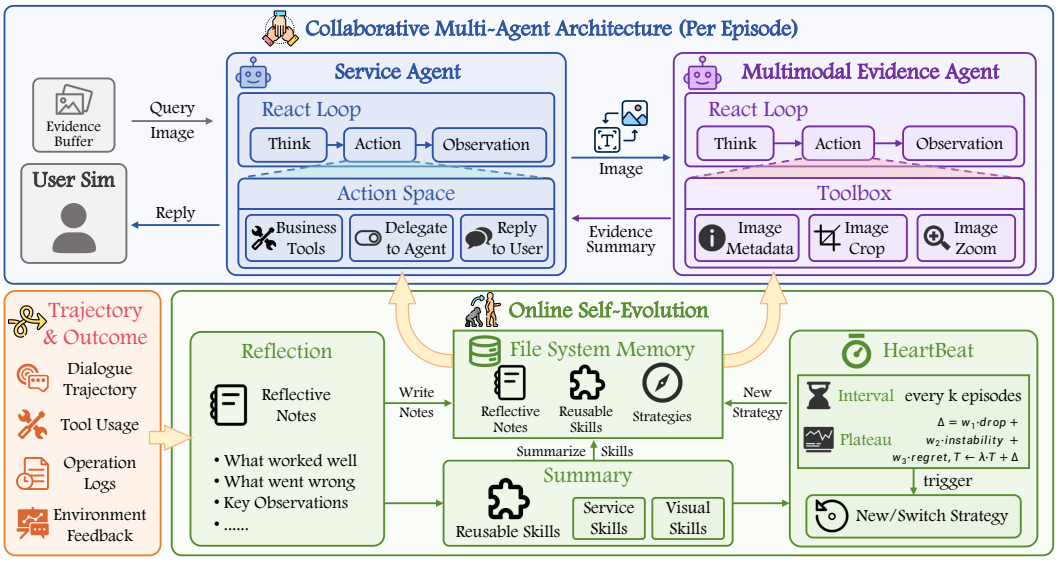


Figure 3: **Overview of DynEvo.** DynEvo combines **multi-agent collaboration** with **online self-evolution** through a tool workbench. Within episodes, a ***service agent*** collaborates with a ***multimodal evidence agent*** for user interaction and visual reasoning. Across episodes, trajectories and environmental feedback update externalized ***notes***, ***skills***, and ***strategies*** to support adaptation.

results are appended to the dialogue history, and the agent is re-invoked until it produces a reply or terminates the interaction. Visual delegation is enabled only when the current user message contains an image, and internal hops per turn are capped to bound reasoning cost.

Multimodal Evidence Agent. The multimodal evidence agent is invoked on demand by the service agent for visual analysis. It receives a textual request, the dialogue context, and the attached image, then runs its own ReAct loop over vision tools such as image metadata inspection, cropping, and zooming to gather relevant evidence. It returns a textual summary that the service agent incorporates into subsequent reasoning. This separation keeps visual evidence extraction specialized while centralizing user-facing planning and response generation in the service agent.

4.2 Online Self-Evolution Loop

Cross-episode adaptation in DynEvo maintains three externalized states at different adaptation horizons: *notes* capture episode-specific reflections, *skills* distill reusable procedures, and *strategies* govern longer-horizon behavior. After each episode, the trajectory and episode-level feedback update notes and skills, while strategies are reviewed periodically or when stronger adaptation is needed. Before the next episode, the updated states are incorporated into the corresponding agent prompts. Only these external states evolve, while the underlying MLLM parameters remain fixed.

Per-Episode Note and Skill Learning. After each episode, the service agent reflects on the completed trajectory and records a compact note summarizing key decisions and potential improvements. In parallel, the involved agents distill reusable skills from the interaction: service-agent skills encode procedures for recurring user needs, while multimodal-agent skills capture visual reasoning workflows. Skills are ranked by usage and effectiveness, with the top-5 retrieved for each agent in subsequent episodes. Notes provide lightweight experiential guidance, whereas skills directly support planning and tool use through reusable behavior templates.

Strategy Lifecycle and Evolution. Strategies operate at a longer horizon than notes and skills. Only one strategy is active at a time, while previously learned strategies are retained in a persistent pool for reuse when similar conditions recur. At periodic review points, the service agent examines recent trajectories and episode-level feedback and either retains the active strategy or updates it when adaptation is needed. If recent performance indicates that stronger adaptation is needed, a trigger initiates strategy reconstruction by retrieving a suitable strategy from the pool or synthesizing a new one from recent experience. The selected strategy becomes active from the next episode. Within an episode, the service agent may also invoke a control tool to switch to a stored strategy when the current

Method	Metric	Daily Ops	Double -11	Fraud After.	Spring Festival	VIP Anniv.	Defective Batch	Newcomer Subsidy	Adv. Storm	Avg.
Naive LLM										
Direct	Overall	0.750	0.735	0.781	0.715	0.770	0.777	0.726	0.756	0.751
	Res. Rate	0.788	0.700	0.840	0.710	0.840	0.827	0.696	0.790	0.774
CoT	Overall	0.751	0.763	0.775	0.742	0.763	0.787	0.739	0.750	0.759
	Res. Rate	0.763	0.813	0.820	0.736	0.800	0.827	0.748	0.765	0.784
Self-Evolving Baselines										
DSPy	Overall	0.747	0.747	0.757	0.722	0.789	0.795	0.734	0.729	0.753
	Res. Rate	0.763	0.738	0.747	0.677	0.840	0.867	0.722	0.715	0.758
EvoAgent	Overall	0.760	0.777	0.797	0.763	0.794	0.787	0.758	0.771	0.776
	Res. Rate	0.813	0.838	0.840	0.832	0.890	0.853	0.809	0.790	0.833
AutoAgents	Overall	0.677	0.652	0.634	0.586	0.654	0.700	0.674	0.658	0.654
	Res. Rate	0.600	0.550	0.620	0.413	0.620	0.773	0.617	0.605	0.600
EvoAgentX	Overall	0.777	0.735	0.758	0.779	0.799	0.767	0.761	0.756	0.766
	Res. Rate	0.813	0.738	0.720	0.826	0.840	0.853	0.800	0.770	0.795
DynEvo(Ours)	Overall	**0.823**	**0.829**	**0.811**	**0.836**	**0.827**	**0.807**	**0.840**	**0.795**	**0.821**
	Res. Rate	**0.900**	**0.906**	**0.917**	**0.967**	**0.924**	**0.918**	**0.931**	**0.900**	**0.920**

Table 3: **Overall Score and Resolution Rate of all methods across the 8 dynamic customer-service scenarios. Bold** marks the best result and underline marks the second-best for each metric.

one appears unsuitable, enabling faster adjustment to abrupt environmental changes. Implementation details, including the feedback signal, trigger criterion, and state formats, are provided in Appendix B.

5 Experiment

5.1 Experimental Settings

Baselines and Implementation. Following prior research (Qiu et al., 2025; Cheng et al., 2026), we compare DynEvo against two categories of baselines: (1) *Naive LLM-based baselines*, including direct prompting and CoT (Wei et al., 2022), which evaluate performance without explicit memory or strategy evolution; and (2) *Self-evolving frameworks*, including DSPy (Khattab et al., 2024), EvoAgent (Yuan et al., 2025b), AutoAgents (Chen et al., 2024), and EvoAgentX (Wang et al., 2025c), which represent prior approaches to inference-time agent evolution. All methods share the same scenario configurations, task mixtures, and event schedules, and are run on GPT-5.2 (OpenAI, 2026) via the official API with temperature$= 1.0$ and top-$p = 1.0$. More details on the baselines and prompts used in DynaWorld and DynEvo are provided in Appendices C.1, A.3, and B.1.

Metrics. Following prior customer-service benchmarks that jointly evaluate task success and service quality (Gao et al., 2026), we report two episode-level metrics. *Resolution Rate* measures the fraction of episodes in which the user's issue is successfully resolved. *Overall Score* aggregates three continuous dimensions with objective-specific weights determined by the active operating regime: (1) User Satisfaction, estimated by an LLM-based judge from the user simulator's final-turn sentiment; (2) Compliance, which measures adherence to the active policy; and (3) interaction efficiency, measured from agent reply turns and, under cost-sensitive regimes, response length.

5.2 Main Results

Table 3 reports the *Overall Score* and *Resolution Rate* of DynEvo and all baselines across the eight DynaWorld scenarios. The results show that:

Existing self-evolving methods struggle to adapt under non-stationary dynamics. Across the eight scenarios, prior self-evolving methods provide only modest and inconsistent gains over prompting-based baselines, with some methods even degrading performance. The strongest self-evolving baseline, EvoAgent, achieves 77.6% Overall Score and 83.3% Resolution Rate, compared with 75.1% and 77.4% for Direct. In contrast, DynEvo reaches 82.1% and 92.0%, outperforming EvoAgent by 4.5 and 8.7 percentage points, respectively. This gap suggests that accumulating or refining experience alone is insufficient when previously effective behaviors may become outdated as environmental conditions continue to change across successive episodes.

DynEvo achieves consistent gains across dynamic scenarios. DynEvo achieves the best result on all eight scenarios for both metrics, with consistent gains under environmental dynamics. For example, on [Newcomer Subsidy], DynEvo improves Overall Score from 76.1% to 84.0% and Res. Rate from 80.9% to 93.1% over the strongest baselines, demonstrating robust adaptation under multiple environmental shifts. Under the compound [Adv. Storm] scenario, DynEvo improves Overall Score from 77.1% to 79.5% and Res. Rate from 79.0% to 90.0% when multiple event types co-occur. The strong performance of DynEvo suggests the benefit of cross-episode evolution over externalized experience and strategies, enabling more stable adaptation across diverse environmental shifts.

5.3 Analysis

To further understand DynEvo's effectiveness and adaptation behavior in dynamic environments from complementary perspectives, we investigate four research questions (RQs): **RQ1:** What drives DynEvo's adaptation gains? **RQ2:** How does DynEvo balance interaction efficiency and inference cost? **RQ3:** Which environmental shifts are hardest to adapt to? **RQ4:** How does DynEvo adapt to and recover from environmental shifts over time?

5.3.1 Component Ablation

To understand where DynEvo's gains come from, we ablate its externalized states and collaborative multi-agent architecture. As shown in Table 4, removing *Notes*, *Skills*, or *Strategy* consistently degrades performance, supporting their complementary roles in preserving recent experience, reusable procedures, and longer-horizon guidance. Architectural ablations also reduce performance, highlighting the benefit of separating multimodal evidence reasoning from user-facing planning and interaction. Together, these results show that DynEvo benefits from both cross-episode self-evolution and within-episode role-specialized collaboration.

Method	Daily	Double-11	Fraud	Spring Fest.	VIP Anniv.	Defective	Newcomer	Adv. Storm	Avg.
DynEvo	**0.900**	**0.906**	**0.917**	**0.967**	**0.924**	**0.918**	**0.931**	**0.900**	**0.920**
w/o *Notes*	0.881	0.863	0.883	0.926	0.894	0.902	0.906	0.850	0.888
w/o *Skills*	0.894	0.874	0.861	0.937	0.918	0.913	0.925	0.890	0.902
w/o *Strategy*	0.863	0.896	0.906	0.959	0.918	0.904	0.906	0.870	0.903
w/o *MM Agent*	0.850	0.851	0.851	0.909	0.882	0.887	0.881	0.835	0.868
Single-Agent	0.875	0.878	0.884	0.938	0.903	0.902	0.906	0.865	0.894

Table 4: **Component ablation of DynEvo across the eight dynamic scenarios in terms of Resolution Rate.** *w/o MM Agent* removes the multimodal evidence agent, while *Single-Agent* uses one agent for both service interaction and multimodal reasoning.

5.3.2 Efficiency of DynEvo

We evaluate efficiency from both user-facing interaction and system-level inference perspectives, reporting end-to-end time, agent reply turns, API calls, output tokens, and user satisfaction in Table 5. Prompting-only baselines remain the cheapest in raw inference cost, while reflection-heavy methods such as EvoAgent and AutoAgents require substantially more calls, tokens, and time. Despite its online evolution loop, DynEvo completes each episode with only 3.46 agent replies on average, fewer than all baselines, while maintaining moderate inference overhead. Moreover, DynEvo achieves the highest user satisfaction of 0.803 across all evaluated methods. Overall, DynEvo balances additional inference cost with shorter and higher-quality user interactions.

Method	Time / Ep.	Turns / Ep.	Calls / Ep.	Out Tok / Ep.	User Sat.
Naive LLM					
Direct	54.9	5.48	5.48	1.12K	0.677
CoT	67.6	5.26	5.26	1.10K	0.683
Self-Evolving Baselines					
DSPy	60.2	5.62	6.02	1.02K	0.672
EvoAgent	158.5	5.19	46.70	4.08K	0.724
AutoAgents	199.2	7.33	35.24	11.34K	0.564
EvoAgentX	71.0	5.09	10.37	2.03K	0.705
Ours					
DynEvo	110.6	3.46	7.77	2.56K	**0.803**
w/o *MM Agent*	99.5	3.34	6.24	2.06K	0.742
Single-Agent	106.3	3.56	6.48	2.16K	0.764

Table 5: **Episode-level efficiency and user satisfaction on DynaWorld.**

5.3.3 Impact of Environmental Shifts

DynaWorld contains four event types—*policy*, *population*, *objective*, and *temporal*—that alter environmental conditions during a scenario. Figure 4 compares their impact, cumulative loss, and

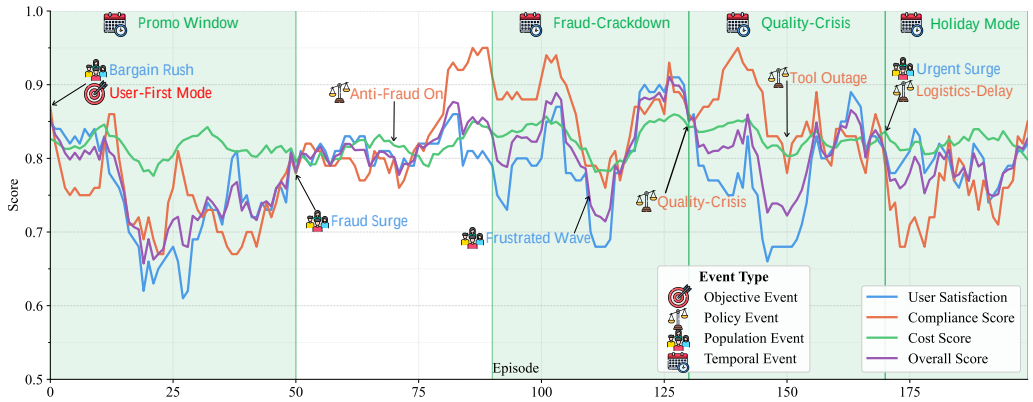


Figure 5: **Adaptation trajectory of DynEvo across episodes.** Per-episode scores on a representative scenario with multiple environmental events.

recovery behavior. *Population* shifts are the most challenging, causing the largest drop and cumulative loss as well as the longest recovery, as the agent must adapt to a changed user distribution. *Objective* shifts also cause substantial immediate degradation but recover more quickly within the adaptation window. *Policy* shifts have a moderate impact, while *temporal* events cause the smallest degradation and fastest recovery, consistent with their localized and time-bounded nature. Overall, distributional shifts are harder to absorb and recover from than localized policy or temporal changes.

5.3.4 Adaptation Trajectory Analysis

We finally examine how DynEvo adapts over episodes in a representative dynamic scenario, as shown in Figure 5. Consistent with the event-level analysis, temporal changes produce only short-lived volatility, whereas population shifts lead to more pronounced degradation and slower recovery. More importantly, the trajectory reveals distinct adaptation behaviors: the earlier shift is absorbed through periodic strategy updates with localized adjustment, whereas the later, sharper shift triggers the stagnation detector and reconstructs the active strategy from recent experience before performance gradually recovers. This progression suggests that DynEvo increases the strength of adaptation as environmental changes become harder to absorb. Accordingly, DynEvo adapts over different horizons, with gradual changes handled through lightweight continuous updates and sharper regime shifts invoking explicit strategy reconstruction.

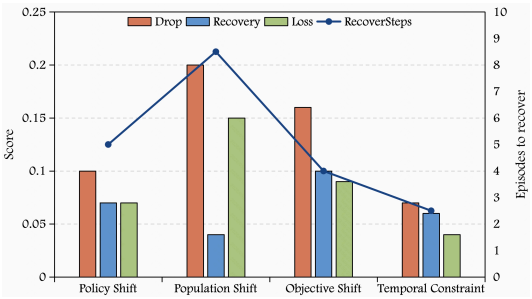


Figure 4: **Impact of the four event types on adaptation.** *Drop*: immediate decline at event onset; *Recovery*: score regained by the end of the adaptation window; *Loss*: cumulative degradation; *RecoverSteps*: episodes required for recovery.

6 Conclusion

In this paper, we introduced **DynaWorld**, a dynamic multimodal customer-service environment that moves agent evaluation beyond isolated static tasks by connecting episodes through evolving policies, user populations, operational objectives, and temporal regimes. We further proposed **DynEvo**, a self-evolving multi-agent framework that combines within-episode multi-agent collaboration with the evolution of externalized notes, skills, and strategies across different horizons, without updating model parameters. Experiments across eight dynamic customer-service scenarios demonstrate that DynEvo consistently outperforms standard prompting and strong self-evolving baselines while maintaining efficient user interaction. Further analyses reveal distinct adaptation and recovery behaviors under different environmental shifts, highlighting dynamic environments as an essential testbed for developing self-evolving multimodal agent systems.

AI USE STATEMENT

In this work, we used generative AI tools for domain and literature exploration, language editing and polishing, and evaluation components within our LLM-agent experimental pipeline. Generative AI was not used to formulate theoretical claims, develop proofs, propose research hypotheses, or determine the interpretation of experimental results. All AI-assisted content was reviewed by the authors, who take full responsibility for the final content and claims of this work.

ETHICS STATEMENT

Data Privacy. All experimental data in this work are synthetically generated from anonymized and preprocessed real-world e-commerce data. Personally identifiable information is removed during preprocessing and is not retained in the resulting experimental data.

Safety and Oversight. DynaWorld includes adversarial and risk-sensitive user profiles to evaluate agent behavior under challenging service conditions. These settings support robustness and safety evaluation rather than facilitating misuse. For real-world deployment, evolved strategies should remain auditable and subject to human oversight, particularly for consequential actions such as refunds and compensation.

Bias and Fairness. To ensure consistent evaluation across methods, we use fixed simulator and judge configurations, balanced profile sampling, and explicit policy and compliance criteria. These controls reduce variation introduced by model-based simulation and evaluation. Real-world deployment should additionally validate agent behavior across relevant user groups and operational settings.

REPRODUCIBILITY STATEMENT

We document the implementation and experimental settings needed to reproduce our results in the main paper and appendices, including the construction of DynaWorld, DynEvo, user-simulator configurations, evaluation protocols, prompts, hyperparameters, and other experimental settings. Additional implementation details and configurations are provided in the supplementary materials.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Shihao Cai, Runnan Fang, Jialong Wu, Baixuan Li, Xinyu Wang, Yong Jiang, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zhen Zhang, et al. Autoforge: Automated environment synthesis for agentic reinforcement learning. *arXiv preprint arXiv:2512.22857*, 2025.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-time adaptation for llm agents via environment interaction. In *International Conference on Learning Representations*, volume 2026, pp. 39256–39288, 2026a.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. Autoagents: a framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 22–30, 2024.

Xiaoyin Chen, Canwen Xu, Yite Wang, Boyi Liu, Zhewei Yao, and Yuxiong He. Learning to self-evolve. *arXiv preprint arXiv:2603.18620*, 2026b.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. Improving retrieval-augmented generation through multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 38:121336–121367, 2026c.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem^{2}evolve: Towards self-evolving agents via

co-evolutionary capability expansion and experience distillation. *arXiv preprint arXiv:2604.10923*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Tianhong Gao, Jundong Shen, Jiapeng Wang, Bei Shi, Ying Ju, Junfeng Yao, and Huiyu Yu. Benchmarking and learning real-world customer service dialogue. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9688–9711, 2026.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156, 2024.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, et al. Envharness: Awakening static worlds for agent learning. *arXiv preprint arXiv:2608.19880*, 2026.

O. Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, 2024. URL `https://api.semanticscholar.org/CorpusID:271532771`.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Jiachun Li, Zhuoran Jin, Tianyi Men, Yupu Hao, Kejian Zhu, Lingshuai Wang, Dongqi Huang, Longxiang Wang, Shengjia Hua, Lu Wang, et al. Agentic environment engineering for large language models: A survey of environment modeling, synthesis, evaluation, and application. *arXiv preprint arXiv:2606.12191*, 2026a.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, et al. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2508.13167*, 2025a.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025b.

Zekun Li, Shinda Huang, Jiangtian Wang, Nathan Zhang, Antonis Antoniades, Wenyue Hua, Kaijie Zhu, Sirui Zeng, Chi Wang, William Yang Wang, and Xifeng Yan. Sopbench: Evaluating language agents at following standard operating procedures and constraints. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026b.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

OpenAI. Introducing gpt-5.2. `https://openai.com/index/introducing-gpt-5-2/`, 2026. Accessed: 2026-05-21.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Karen Ullrich, Jingtong Su, Claudia Shi, Arjun Subramonian, Amir Bar, Ivan Evtimov, Nikolaos Tsilivis, Randall Balestriero, Julia Kempe, and Mark Ibrahim. Openapps: Simulating environment variations to measure ui agent reliability. In *International Conference on Learning Representations*, volume 2026, pp. 118257–118290, 2026.

Haoxin Wang, Xianhan Peng, Huang Cheng, Yizhe Huang, Ming Gong, Chenghan Yang, Yang Liu, and Jiang Lin. Ecom-bench: Can llm agent resolve real-world e-commerce customer support issues? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 276–284, 2025a.

Weiqi Wang, Limeng Cui, Xin Liu, Sreyashi Nag, Wenju Xu, Chen Luo, Sheikh Muhammad Sarwar, Yang Li, Hansu Gu, Hui Liu, et al. Ecomscriptbench: A multi-task benchmark for e-commerce script planning via step-wise intention-driven product association. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1–22, 2025b.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. Evoagentx: An automated framework for evolving agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 643–655, 2025c.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025d.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Jundong Xu, Qingchuan Li, Jiaying Wu, Yihuai Lan, Shuyue Stella Li, Huichi Zhou, Bowen Jiang, Lei Wang, Jun Wang, Anh Tuan Luu, et al. Evoarena: Tracking memory evolution for robust llm agents in dynamic environments. *arXiv preprint arXiv:2606.13681*, 2026.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards. In *International Conference on Learning Representations*, volume 2026, pp. 32373–32394, 2026.

Huanjin Yao, Ruifei Zhang, Jiaxing Huang, Jingyi Zhang, Yibo Wang, Bo Fang, Ruolin Zhu, Yongcheng Jing, Shunyu Liu, Guanbin Li, et al. A survey on agentic multimodal large language models. *arXiv preprint arXiv:2510.10991*, 2025a.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, 2025.

Hongzhuo Yu, Fei Zhu, Guo-Sen Xie, and Ling Shao. Self-consolidation for self-evolving agents. *arXiv preprint arXiv:2602.01966*, 2026.

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, et al. Marshal: Incentivizing multi-agent reasoning via self-play with strategic llms. *arXiv preprint arXiv:2510.15414*, 2025a.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6192–6217, 2025b.

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*, 2026a. URL https://mlanthology.org/iclr/2026/zhang2026iclr-agentic/.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. Infiniteweb: Scalable web environment synthesis for gui agent training. *arXiv preprint arXiv:2601.04126*, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *AAAI Conference on Artificial Intelligence*, pp. 19632–19642, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://mlanthology.org/aaai/2024/zhao2024aaai-expel/.

Weikang Zhao, Xili Wang, Chengdi Ma, Lingbin Kong, Zhaohua Yang, Mingxiang Tuo, Xiaowei Shi, Yitao Zhai, and Xunliang Cai. Mua-rl: Multi-turn user-interacting agent reinforcement learning for agentic tool use. *arXiv preprint arXiv:2508.18669*, 2025.

Chenyu Zhou, Xiaoming Shi, Hui Qiu, Yankai Jiang, ShaoGuo Liu, Tingting Gao, Haitao Leng, Xiawu Zheng, and Rongrong Ji. Mix-ecom: Towards mixed-type e-commerce dialogues with complex domain rules. In *International Conference on Learning Representations*, volume 2026, pp. 156342–156364, 2026.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

A MORE DETAILS OF DYNAWORLD

DynaWorld is a dynamic multimodal customer-service environment that connects successive interaction episodes through evolving policies, user populations, operational objectives, and temporal regimes. We further detail its dynamic environment formulation, three-level composition and runtime protocol, user simulator, and Judge Agent with environment feedback.

A.1 DYNAMIC ENVIRONMENT FORMULATION

We formulate DynaWorld as a sequence of interaction episodes coupled through an evolving environment state. At episode e, the environment state is represented as

$$\mathbf{s}_e = (\mathcal{P}_e,\, \rho_e,\, \mathcal{O}_e,\, \mathcal{H}_e)\,, \tag{1}$$

where $\mathcal{P}_e$ denotes the active policies and tool availability, ρ_e the user-profile distribution, $\mathcal{O}_e$ the operational objective, and $\mathcal{H}_e$ the active temporal regime. Before each episode, all activated events update the previous state according to

$$\mathbf{s}_e = \mathcal{T}(\mathbf{s}_{e-1}, \mathcal{E}_e), \tag{2}$$

where $\mathcal{E}_e$ denotes the events activated at episode e and $\mathcal{T}$ the corresponding transition rule. Their effects persist into subsequent episodes unless explicitly updated or reverted by a later event.

Under the resulting state, the environment instantiates the next interaction by sampling a user profile from ρ_e and a stage-compatible Meta Task from the scenario-specific task mixture. The agent observes only the resulting information, including the active policy notice, task context, available tools, and dialogue feedback, while the future event schedule remains hidden, requiring the agent to infer environmental changes from their observable consequences. A static environment is therefore a special case in which $\mathbf{s}_e = \mathbf{s}_0$ across episodes, whereas DynaWorld evaluates whether an agent can track, adapt to, and recover from changes in $\mathbf{s}_e$ over a long interaction sequence.

A.2 ENVIRONMENT COMPOSITION

DynaWorld is organized into three composition levels corresponding to the structure introduced in Section 3: episode instantiation through Meta Profiles and Meta Tasks (Level 1), cross-episode dynamics through environmental events (Level 2), and long-horizon operational scenarios (Level 3). The underlying profiles and tasks are derived from real-world e-commerce customer-service interactions in Mix-Ecom (Zhou et al., 2026), while events and scenarios organize them into evolving interaction sequences.

Persona	(Pat., Aggr., Price)	Active stages	Description
Normal	$(0.7,\ 0.2,\ 0.4)$	All	Regular customer with typical service needs.
VIP	$(0.5,\ 0.6,\ 0.3)$	All	High-value long-term customer expecting premium treatment.
Newcomer	$(0.6,\ 0.2,\ 0.5)$	All	First-time customer unfamiliar with platform procedures.
Quality-Seeker	$(0.8,\ 0.1,\ 0.2)$	Pre-sale	Quality-focused customer with low price sensitivity.
Bargain-Hunter	$(0.5,\ 0.4,\ 0.9)$	Pre-sale	Discount-oriented customer who actively seeks lower prices.
Terse	$(0.6,\ 0.3,\ 0.4)$	All	Customer who provides limited information through short, fragmented requests.
Risk-Averse	$(0.7,\ 0.2,\ 0.3)$	All	Customer focused on insurance, refund rules, and service entitlements.
Urgent	$(0.3,\ 0.7,\ 0.2)$	All	Time-sensitive customer who expects prompt handling.
Emotional	$(0.9,\ 0.1,\ 0.2)$	All	Customer who values empathy and respectful treatment.
Short-Tempered	$(0.2,\ 0.8,\ 0.3)$	All	Customer prone to strong reactions when receiving poor service.
Fraudster	$(0.3,\ 0.9,\ 0.8)$	After-sale	Adversarial customer who fabricates issues to obtain unwarranted refunds.

Table 6: **Meta Profiles in DynaWorld.** Each profile is parameterized by a three-dimensional trait vector $(\textit{Pat.}, \textit{Aggr.}, \textit{Price})$ denoting patience, aggressiveness, and price sensitivity, together with the dialogue stages on which the profile is sampled. "All" marks profiles that may pair with pre-sale, logistics, and after-sale tasks.

Meta Profiles and Meta Tasks. DynaWorld contains 11 Meta Profiles and 15 Meta Task types, instantiated from an $8{,}205$-task Meta Task pool. A Meta Profile is a behavioral prototype characterized by patience, aggressiveness, price sensitivity, and stage affinity, together with a persona description used by the user simulator. The profiles span everyday users (e.g., [Normal], [VIP]), behavior-specific

Scenario	#Ep.	Event schedule
[Daily Ops]	160	*Pop.:* [Baseline Mix]; *Pol.:* [Refund Tightening]@60 *Obj.:* [User-First] → [Risk-First]@80 → [Cost-First]@130; *Temp.:* None
[Double-11]	180	*Pop.:* [Bargain Rush]; [Newcomer Flood]@40; [Baseline Mix]@155; *Pol.:* [Refund Tightening]@110 *Obj.:* [User-First] → [Cost-First]@80 → [Risk-First]@130; *Temp.:* [Promo Window] 0–80
[Spring-Festival Logistics]	180	*Pop.:* [Urgent Surge]; [VIP Wave]@75; [Frustrated Wave]@120; *Pol.:* [Logistics-Delay Policy]@30 *Obj.:* [User-First] → [Cost-First]@155; *Temp.:* [Holiday Mode] 30–110
[VIP Anniversary]	170	*Pop.:* [VIP Wave]; [Frustrated Wave]@120; [Baseline Mix]@165; *Pol.:* [Comp. Cap Up (300)]; [Comp. Cap Down (80)]@100; [Refund Tightening]@145 *Obj.:* [User-First] → [Cost-First]@100; *Temp.:* [Promo Window] 0–100
[Defective Batch]	170	*Pop.:* [Frustrated Wave]; [Fraud Surge]@110; *Pol.:* [Quality-Crisis Policy]@25; [Tool Outage]@50 *Obj.:* [User-First] → [Risk-First]@75 → [Cost-First]@145; *Temp.:* [Quality-Crisis] 25–120
[Newcomer Subsidy]	160	*Pop.:* [Newcomer Flood]; [Frustrated Wave]@95; [Baseline Mix]@140; *Pol.:* [Refund Tightening]@70 *Obj.:* [User-First] → [Risk-First]@115; *Temp.:* [Promo Window] 0–70
[Fraud Aftermath]	180	*Pop.:* [Fraud Surge]; [Frustrated Wave]@120; [Baseline Mix]@170; *Pol.:* [Anti-Fraud On]@30; [Comp. Cap Down (40)]@70 *Obj.:* [User-First] → [Cost-First]@150; *Temp.:* [Fraud-Crackdown] 50–110*
[Adversarial Storm]	200	*Pop.:* [Bargain Rush]; [Fraud Surge]@50; [Frustrated Wave]@110; [Urgent Surge]@170; *Pol.:* [Anti-Fraud On]@70; [Quality-Crisis Policy]@130; [Tool Outage]@150; [Logistics-Delay Policy]@170 *Obj.:* [User-First]; *Temp.:* [Promo Window] 0–50*; [Fraud-Crackdown] 90–130*; [Quality-Crisis] 130–170; [Holiday Mode] 170–199
Total	**1,400**	

Table 7: **Operational scenarios in DynaWorld.** Events are grouped by axis; events without an episode index fire at episode 0. Temporal events are shown as start–end windows, and * denotes silent reversion without an explicit end notice.

users (e.g., [Bargain-Hunter], [Urgent]), and adversarial actors (e.g., [Fraudster]). A Meta Task defines an episode-level service objective through a user goal, success criteria, business context, optional visual evidence, and a stage tag covering pre-sale, logistics, or after-sale service. At episode instantiation, a profile is sampled under the active population distribution and paired with a stage-compatible task, producing semantically related but behaviorally diverse interactions. Tables 6 and 8 summarize the complete profile and task configurations.

Environmental Events. DynaWorld contains 21 events across four categories: 7 policy, 7 population, 3 objective, and 4 temporal events. *Policy events* modify the active rule set or tool availability; *population events* re-weight the Meta-Profile distribution; *objective events* switch the operational priority among [User-First], [Risk-First], and [Cost-First] modes; and *temporal events* introduce time-bounded regimes that automatically revert at a future episode. Unlike task-level variation, these events modify the environment state and therefore affect subsequent episodes until explicitly updated or reverted.

Meta Task	Stage	#Tasks	Img. (%)
Product Inquiry	Pre-sale	151	-
Coupon Inquiry	Pre-sale	331	-
Product Recommendation	Pre-sale	30	-
Logistics Tracking	Logistics	3,715	-
Logistics Delay	Logistics	29	-
Lost Package	Logistics	353	-
Wrong Delivery	Logistics	1,832	-
Refund / Return	After-sale	921	27.4
Quality-Defect Refund	After-sale	137	42.3
Product Mismatch	After-sale	58	25.9
Size Exchange	After-sale	42	26.2
Order Cancel	After-sale	51	15.7
Compensation Request	After-sale	98	34.7
Refund-Only Fraud	After-sale	275	28.7
Health-Damage Extortion	After-sale	182	26.9

Table 8: **Meta Task pool of DynaWorld.** *#Tasks* reports the number of distinct task instances per type after consolidating duplicate logistics-inquiry variants; *Img. (%)* is the fraction of instances with at least one reference image.

Operational Scenarios. The eight operational scenarios in DynaWorld compose task mixtures, user mixtures, and event schedules over long episode sequences. They range from near-stationary operation ([Daily Ops]) to scenarios dominated by population, policy, or temporal changes ([Double-11], [Spring-Festival Logistics], [VIP Anniversary], [Defective Batch], and [Newcomer Subsidy]), as well as compound scenarios where several event families co-occur ([Fraud Aftermath] and [Adversarial Storm]). Scenarios therefore specify *how* the environment evolves rather than defining static dataset splits. Table 7

provides the episode horizon, task and user composition, event schedule, and objective regimes for each scenario.

Runtime Protocol. Before each episode, scheduled events are applied in order, after which the environment samples a stage-compatible Meta Profile–Meta Task pair under the updated state. The dialogue proceeds for at most 20 agent turns. The agent observes the policy notice, task context, available tools, and dialogue feedback, but not the future event schedule. Temporal events automatically revert when their duration expires, while other state changes persist until modified by events. Randomness arises from task and population sampling, while the scenario configuration fixes the task pool, population distribution, and event schedule throughout the episode sequence.

A.3 User Simulator

The user simulator pairs a sampled Meta Profile with a Meta Task and simulates the customer side of each episode. At every turn, it conditions on the persona, task goal, and dialogue history to generate the next user utterance. The simulator may terminate the interaction once the request is resolved or abandoned; an episode also ends when the agent invokes its termination action or the turn limit is reached. Figure 6 shows the persona prompt templates, while Figure 7 illustrates how the persona, task goal, and shared instructions are assembled into the simulator prompt.

Multimodal evidence is delivered through the simulator as part of the interaction. Reference images associated with a Meta Task are loaded into an episode-level buffer and attached to user messages when required by the request. Each image is sent at most once and assigned a stable attachment marker, allowing agent responses and evaluation to refer to the same visual evidence.

Prompt A: Persona System Prompt Template

User Type	Description & System Prompt
[Normal Customer]	*"You are a regular customer with a reasonable request. You are polite but firm..."*
[Price-Sensitive]	*"You are a very price-sensitive customer. You always look for the best deal..."*
[Malicious]	*"You are attempting to exploit the customer service system. You fabricate or exaggerate complaints..."*
[New User]	*"You are a first-time user of this platform. You are confused about how processes work..."*
[VIP Customer]	*"You are a long-term VIP customer who has spent significant money on this platform..."*

Figure 6: Persona system-prompt templates used by the user simulator. Translated from Chinese.

Prompt B: Dynamic Assembly Template

*You are a **customer contacting customer service**. {persona}*
Your goal for this conversation is: {task.to_user_context()}
Task background: {task_context}
[Instructions]

- *Maintain the customer role (do not act as a customer service agent).*
- *Be realistic; unless your role requires it, do not be overly cooperative or overly difficult.*
- *When your issue has been satisfactorily resolved (or you give up), end the conversation by saying "Thank you, goodbye" or "I've had enough"; and you must add the marker [END_CONVERSATION] at the very beginning of that message (the marker will be automatically stripped and will not be shown as part of the message body). Only add this marker when you truly want to end the conversation; do not add it to ordinary responses such as "Okay" or "Understood."*
- *Do not repeat yourself unnecessarily.*
- *Keep your responses concise (usually 1–4 sentences per turn).*

Figure 7: Dynamic assembly template that composes the user-simulator prompt from the persona, task goal, and shared instructions. Translated from Chinese.

A.4 Judge Agent and Environment Feedback

After each episode, the environment evaluates the completed trajectory and produces episode-level feedback. The Judge Agent receives the dialogue trajectory, task context, success criteria, and active policy constraints, and scores *user satisfaction* and *compliance*. Together with the environment-computed efficiency signal, these dimensions are aggregated according to the active objective regime to form the Overall Score. Figure 8 shows the prompt used by the Judge Agent for satisfaction and compliance evaluation.

For DynEvo, the Overall Score also serves as episode-level service feedback for subsequent strategy evolution, analogous to post-interaction feedback used to improve future service behavior. Resolution Rate is reported separately as a binary evaluation metric and is not used as an evolution signal. The definitions and reporting protocols of all evaluation metrics are detailed in Appendix C.2.

Prompt for Judge Agent

*You are an **expert customer service quality evaluator**.*

Your job is to objectively score a completed conversation between a customer and a customer service agent. Be strict but fair. Do not give high scores unless the agent genuinely resolved the issue and followed all policies.

***Scoring Instructions:** Score each dimension on a scale of 0–10 (integer only).*

user_satisfaction (0–10):

- *10 = Customer's issue was fully resolved; customer expressed satisfaction.*
- *7 = Largely resolved, minor shortcomings.*
- *4 = Partially addressed but key issue remains.*
- *1 = Agent failed to help; customer frustrated or gave up.*
- *0 = Agent made the situation worse.*

compliance_score (0–10):

- *10 = Agent followed all policy constraints perfectly.*
- *7 = Minor ambiguous case; no explicit violation.*
- *4 = Agent made promises or concessions that violate a constraint.*
- *0 = Agent explicitly performed a forbidden action.*

Respond in JSON only: {"user_satisfaction": <int>, "compliance_score": <int>, "reasoning": <one sentence>}

Figure 8: LLM judge prompt for scoring user satisfaction and compliance. Translated from Chinese.

B More Details of DynEvo

B.1 Multi-Agent Architecture and Workbench

DynEvo is implemented as a collaborative two-agent system with an externalized self-evolution loop across episodes. Within each episode, a service agent handles customer interaction and planning, while a multimodal evidence agent is invoked on demand for visual reasoning. Across episodes, reflective notes, reusable skills, and operating strategies evolve at different time scales without updating the underlying MLLM parameters. This section details the collaborative architecture, externalized state management, and online evolution procedure.

Agent roles and collaboration. The service agent is the user-facing planner responsible for dialogue, business-state operations, strategy execution, and final responses. When the current user turn contains visual evidence, it may delegate a textual request to the multimodal evidence agent, which runs a separate reasoning loop over vision tools and returns a textual evidence summary. This separation keeps complex visual evidence extraction outside the customer-service reasoning loop, allowing the service agent to focus on interaction, planning, and strategy execution. Both agents share the same MLLM backbone and tool registry, while specialization arises from their system prompts, action protocols, and separate skill libraries.

Agent prompt composition. At the start of each episode, the orchestrator rebuilds the service agent's system prompt from a base role description, the active business rules, reflective notes, the active strategy, the current platform and session context, the tool catalog, and the top-five retrieved service skills. Among these components, notes, skills, and strategy evolve across episodes, while the remaining context is fixed or supplied by the environment. The multimodal evidence agent follows the same general structure but uses a separate visual skill library and a smaller action protocol. Figure 9 shows the service-agent prompt template.

System Prompt for Service Agent

*You are a **professional customer service agent** for an e-commerce platform. Based on the current **task background**, **policy constraints**, and **conversation history**, directly provide a professional, concise, and compliant customer service response.*

The response should be addressed directly to the user, with a professional and natural tone. Do not include any reasoning process. Keep it under 200 words.

Figure 9: Service-agent system prompt template. Translated from Chinese.

Action protocol. Both agents emit one JSON-structured action at each internal step. The service agent chooses among *tool*, *vision*, and *reply*, while the multimodal evidence agent chooses between *tool* and *reply*. Within one user turn, at most three visual delegations are allowed before the service agent must respond, bounding the internal reasoning cost. Visual delegation is enabled only when the current user message contains an image supplied by the environment.

Tool workbench. The workbench contains fourteen tools, including eleven business or control tools for the service agent and three vision tools for the multimodal evidence agent. Five read tools retrieve shop, product, item, order, and logistics information; four write tools modify permitted order or logistics fields; two control tools allow the service agent to switch the active strategy or terminate the dialogue; and three vision tools support image metadata inspection, region cropping, and zooming. Although all tools are managed through a common registry, access is strictly isolated by agent role: the service agent can invoke only the business and control tools, while the multimodal evidence agent can invoke only the vision tools. Unauthorized tool calls are rejected by the environment.

B.2 Externalized Memory Management

DynEvo maintains three forms of externalized textual state with different roles and management mechanisms. Relevant states are incorporated into the agent prompt at the beginning of each episode.

Reflective Notes. After each episode, the service agent records a compressed reflection, typically one or two sentences, summarizing pivotal decisions, the interaction outcome, and potential improvements. Notes provide lightweight experiential context rather than executable behavior and therefore do not directly constrain action selection. Accumulated notes are inserted as a single context block, allowing previous successes and failures to inform later episodes without a separate retrieval procedure.

Skills. Skills are structured reusable behavior templates that support planning and tool use. Service-agent skills encode procedures for recurring customer needs, including useful tool sequences and response patterns, while multimodal-agent skills capture reusable visual reasoning workflows. Each skill maintains usage, success, and update statistics; matching skills are revised in place rather than duplicated. At episode start, skills are ranked by success rate, usage frequency, and recency, and only the top five for each agent are retrieved into the prompt. This selection prevents an expanding skill store from directly increasing the active context and reduces the influence of rarely useful behaviors.

Strategies. Strategies capture longer-horizon behavioral guidelines for handling different user and operating conditions. Exactly one strategy is active at a time and is included in the service agent's prompt. Previously generated strategies remain in a persistent pool for later reuse when similar conditions recur, but inactive strategies are not inserted into the active prompt. When the current strategy becomes ineffective, the agent can reactivate a suitable stored strategy or construct a new one from recent trajectories.

B.3 Online Self-Evolution Loop

The three externalized states evolve at different time scales. Notes and skills are updated after every episode, while strategies are reviewed periodically or reconstructed when recent performance indicates that stronger adaptation is needed. In addition, the service agent may switch among stored strategies during an interaction. These updates are implemented through five evolution channels, summarized in Algorithm 1.

Algorithm 1 One Episode of DynEvo.

Require: service agent π_s, multimodal agent π_v, tool environment $\mathcal{E}$, externalized states (N, K, S) for notes, skills, and strategies; episode index e.

Build π_s's prompt from $(N, S, \text{top-}5(K_s))$ and environment context.
Build π_v's prompt from top-$5(K_v)$ and visual context.
$h \leftarrow \emptyset$ ▷ dialogue history
while episode not terminated **do**
  receive user message u (optionally with image I); append to h
  repeat
    $a \leftarrow \pi_s(h)$
    if $a =$ tool **then**
      append $\mathcal{E}$.run(a) to h
    else if $a =$ vision **and** $I \neq \emptyset$ **and** hops < 3 **then**
      run π_v over vision tools; append evidence summary to h
    end if
  until $a =$ reply or $a =$ end-conversation
  if $a =$ switch-strategy inside the loop **then**
    update active strategy from pool ▷ Channel (E)
  end if
end while
save trajectory and receive episode-level Overall Score s_e
update N by reflection ▷ Channel (A)
update K_s and K_v by skill distillation ▷ Channel (B)
if $e \bmod K = 0$ **then**
  review and optionally update strategy ▷ Channel (C)
end if
update T via Eq. 3
if $T > \tau$ and cooldown elapsed **then**
  reconstruct strategy and reset T ▷ Channel (D)
end if
if $e \bmod 10 = 0$ **then**
  trigger interval-based strategy reconstruction
end if

(A) Per-Episode Reflection. After each episode, the service agent reflects on the completed trajectory and episode-level service feedback. It extracts pivotal decisions, summarizes the outcome, and records potential improvements as a compact reflective note that becomes available from the next episode.

(B) Per-Episode Skill Distillation. Both agents independently distill reusable procedures from the completed interaction. Newly extracted skills are added to the corresponding skill store, while matching existing skills are updated in place. Their usage and success statistics are updated from subsequent invocation outcomes and determine which skills are retrieved in later episodes.

(C) Periodic Strategy Update. Every K=5 episodes, the service agent reviews the most recent trajectories and determines whether to retain the current strategy or generate an updated one. A newly generated strategy is added to the strategy pool and becomes active from the following interaction, whereas a retain decision leaves the active strategy unchanged.

(D) Stagnation-Triggered Strategy Reconstruction. To detect larger shifts that periodic updates may not absorb quickly, DynEvo maintains a tension score based on recent episode-level Overall Scores. After each episode,

$$T \leftarrow \lambda T + \Delta,$$
$$\Delta = w_1 \cdot \textit{drop} + w_2 \cdot \textit{instability} + w_3 \cdot \textit{regret}, \tag{3}$$

where the signals are computed over recent score windows. Let μ_{short} and μ_{long} denote the recent and preceding window means, σ_{recent} the recent-window standard deviation, and s_t and s^* the current and best recent scores, respectively. Specifically, $\textit{drop} = \max(0, \mu_{\text{long}} - \mu_{\text{short}})$ measures recent performance decline, $\textit{instability} = \sigma_{\text{recent}}$ captures score variation, and $\textit{regret} = \max(0, s^* - s_t)$ measures the gap between the current and recent-best scores. When $T > \tau$ and the reconstruction cooldown has elapsed, the service agent reviews recent trajectories and either retrieves an appropriate strategy from the pool or constructs a new one. The selected strategy becomes active and T is reset. An interval-based reconstruction is additionally triggered every ten episodes as a fallback. We use W=10, λ=0.85, τ=4.0, c=5, and (w_1, w_2, w_3)=$(1.0, 0.5, 1.0)$ in all experiments.

(E) In-Conversation Strategy Switch. During an episode, the service agent may invoke the strategy-switch control tool when the strategy appears unsuitable. A successful call replaces the active strategy with a stored alternative and continues the dialogue under the updated guidance; if the requested strategy is unavailable, the tool returns the available strategy names for subsequent selection.

C More Experimental Details and Analyses

C.1 Experimental Setup and Baselines

Experimental Setup. All methods are evaluated on the same eight DynaWorld scenarios with identical task pools, user distributions, event schedules, and tool interfaces. We use GPT-5.2 as the shared MLLM backbone for all methods, with temperature 1.0 and top-p 1.0. This controlled setup ensures that performance differences primarily reflect the reasoning and adaptation mechanisms rather than differences in the underlying model or environment configuration.

Baselines. Following prior work, we compare DynEvo with two categories of baselines. *Naive LLM-based baselines* include **Direct**, which produces actions or responses without explicit reasoning or external memory, and **CoT** (Wei et al., 2022), which adds step-by-step reasoning within each model call. *Self-evolving baselines* include **DSPy** (Khattab et al., 2024), which optimizes prompt programs from interaction examples; **EvoAgent** (Yuan et al., 2025b), which evolves a population of specialized agents; **AutoAgents** (Chen et al., 2024), which coordinates planner, actor, and observer roles; and **EvoAgentX** (Wang et al., 2025c), which evolves workflow candidates from episode-level feedback. Unlike these approaches, DynEvo evolves role-specific notes, skills, and strategies at different time scales while keeping the underlying MLLM fixed. Table 9 summarizes the main adaptation characteristics of all methods.

Baseline	Memory	Evolution	Multi-agent
Direct	✗	✗	✗
CoT	✗	✗	✗
DSPy	✓	✗	✗
AutoAgents	✓	✗	✓
EvoAgent	✓	✓	✓
EvoAgentX	✓	✓	✓
DynEvo	✓	✓	✓

Table 9: **Baseline comparison across three adaptation dimensions.** *Memory*: maintains cross-episode state; *Evolution*: updates prompts or workflows during evaluation; *Multi-agent*: coordinates multiple agent roles within an episode.

C.2 Evaluation Metrics and Protocol

We evaluate each episode from complementary perspectives of task resolution, user experience, policy compliance, interaction efficiency, and multimodal evidence use. These metrics characterize different aspects of service quality under the evolving environment, while the Overall Score aggregates the continuous dimensions according to the active operational objective.

Resolution Rate. Resolution Rate measures whether the interaction reaches a successful resolution at the end of an episode. It is reported as a binary episode-level outcome, complementing the continuous service-quality scores with an intuitive measure of issue resolution. Resolution Rate is used only for evaluation and is not provided to DynEvo as an evolution signal.

User Satisfaction and Compliance. User Satisfaction measures the quality of the interaction from the user's perspective, while Compliance measures whether the agent follows the active policies and avoids prohibited actions. Both are scored by the Judge Agent on a 0–10 scale and normalized to $[0, 1]$. Reporting them separately is important because objective events may change their relative importance across episodes, for example when the environment transitions from [User-First Mode] to [Risk-First Mode].

Efficiency and Cost. The main-text Efficiency metric measures interaction efficiency using

$$\textit{Efficiency} = 1 - \frac{T}{T_{\max}}, \tag{4}$$

where T is the number of agent reply turns and $T_{\max}$ is the episode turn limit. The environment additionally defines

$$\textit{Cost} = 1 - 0.6\frac{T}{T_{\max}} - 0.4\min\left(1, \frac{\bar{L}}{L_0}\right), \tag{5}$$

where $\bar{L}$ is the average agent-response length and L_0=200 is a soft length budget. This formulation additionally discourages unnecessarily verbose interactions and is used when cost efficiency is emphasized.

Overall Score. The Overall Score aggregates User Satisfaction, Compliance, and an efficiency-related component, instantiated as either Efficiency or Cost depending on the active objective regime:

$$\textit{Overall} = w_u S + w_c C + w_e E, \tag{6}$$

where S, C, and E denote the corresponding normalized dimensions and (w_u, w_c, w_e) is selected by the active objective event. Table 10 reports the objective-specific weights used in our experiments. Because these weights may change across episodes, the Overall Score reflects service quality under changing operational priorities throughout the long-horizon scenario.

Objective regime	w_u	w_c	w_e
[User-First Mode]	0.6	0.3	0.1
[Risk-First Mode]	0.2	0.7	0.1
[Cost-First Mode]	0.2	0.2	0.6

Table 10: Overall-score weights by objective regime.

For DynEvo, the Overall Score additionally serves as episode-level feedback for strategy evolution, allowing service outcomes to inform adaptation in future episodes, whereas Resolution Rate is reported only as an evaluation metric and does not participate in the evolution process.

C.3 Additional Experimental Analyses

C.3.1 Robustness to Decoding Randomness

We further evaluate DynEvo under different decoding temperatures to examine its sensitivity to sampling randomness. As shown in Table 11, DynEvo maintains comparable performance across all three settings, with no systematic degradation as decoding becomes less stochastic. In particular, lowering the temperature slightly improves Resolution Rate while leaving the Overall Score nearly unchanged, suggesting that the cross-episode evolution process does not rely on a particular level of decoding randomness to remain effective. These results demonstrate that DynEvo remains stable across different decoding configurations.

Temperature	1.0	0.7	0.5
Overall	0.821	0.828	0.820
Res. Rate	0.920	0.940	0.943

Table 11: **Robustness of DynEvo to decoding temperature.** Results are averaged across eight dynamic scenarios.

C.3.2 User Satisfaction Breakdown

Table 12 reports User Satisfaction across the eight dynamic scenarios. DynEvo achieves the highest average score of 0.803 and consistently maintains strong user-facing performance across changing user populations and operating conditions. Its advantage over the strongest baseline is particularly pronounced on Newcomer Subsidy and Spring Festival, while the gap is smaller on the near-stationary Daily Ops scenario. This pattern is consistent with the main-text event analysis, where population shifts introduce larger adaptation challenges, suggesting that DynEvo remains effective at maintaining interaction quality as the user distribution changes.

Method	Daily Ops	Double -11	Fraud After.	Spring Festival	VIP Anniv.	Defective Batch	Newcomer Subsidy	Adv. Storm	Avg.
Naive LLM									
Direct	0.674	0.653	0.701	0.635	0.715	0.724	0.630	0.682	0.677
CoT	0.670	0.699	0.687	0.661	0.698	0.732	0.657	0.660	0.683
Self-Evolving Baselines									
DSPy	0.674	0.665	0.655	0.638	0.737	0.736	0.639	0.635	0.672
EvoAgent	0.710	<u>0.725</u>	<u>0.719</u>	0.722	<u>0.766</u>	<u>0.741</u>	0.698	<u>0.710</u>	<u>0.724</u>
AutoAgents	0.584	0.551	0.569	0.477	0.558	0.649	0.568	0.559	0.564
EvoAgentX	<u>0.723</u>	0.666	0.661	<u>0.729</u>	0.764	0.719	<u>0.701</u>	0.681	0.705
Ours									
DynEvo	**0.784**	**0.774**	**0.794**	**0.826**	**0.837**	**0.813**	**0.813**	**0.784**	**0.803**

Table 12: **User Satisfaction of all methods across the 8 dynamic customer-service scenarios. Bold** marks the best result and <u>underline</u> marks the second-best in each column.

D Qualitative Case Studies

We present three representative cases from the Daily Ops scenario to illustrate how DynEvo handles different customer-service challenges under evolving operational conditions. The cases cover multi-constraint recommendation with targeted tool use (Figure 10), stateful multi-tool execution over mutable business records (Figure 11), and policy-aware handling of an adversarial refund request (Figure 12). Across these examples, DynEvo consistently grounds its decisions in the current task

context, business state, and active policy, while several baselines either omit necessary tool use, apply irrelevant procedures, or fail to complete the required action sequence.

Case 1: Pre-Sale Multi-Condition Product Inquiry

Scenario: DailyOps (Risk-Sensitive Phase) **Tools Used:** 2/12 **Steps:** 5

Task Requirement. *A buyer in Macau seeks a tablet computer meeting three hard constraints (screen ≥13 in, Snapdragon processor, storage ≥512 GB), requests the lowest-priced qualifying model, and then asks for logistics cost and delivery time-frame to Macau.*

Tool	Description
`get_product_detail`	Retrieve product catalogue and filter by SKU/attribute constraints
`get_shop_detail`	Retrieve shop logistics brands, shipping origin, and insurance info

Why Baselines Fail. ***DSPy*** (sat=0.10) enters a repeated deflection loop — it asks the user to provide product links before offering any recommendation, never invoking the product-search tool.
EvoAgent (sat=0.10) contaminates the pre-sale context with irrelevant post-sale refund-cap rules (¥200 limit), confusing the user and failing to answer the actual query.

DynEvo Trajectory.

Turn	Interaction
1	**[User]** Requests tablet recommendation for Macau: screen ≥13 in, Snapdragon, ≥512 GB, prefer cheapest qualifying model; also asks about shipping cost and availability.
2	**[Assistant]** Identifies product query; invokes tool to scan full catalogue. `get_product_detail(sku="tablet_computer")`
3	**[Tool]** `Returns full product list including specs (screen size,processor, storage).` **[Assistant]** Filters catalogue: only 1 model fully satisfies all 3 constraints — Samsung Tab S9 Ultra (¥4999, 13+ in, Snapdragon, 512 GB, ships within 48 h). Presents result with explicit filtering rationale.
4	**[User]** Confirms constraints are non-negotiable; asks whether ¥4999 includes shipping to Macau, and estimated delivery time.
5	**[Assistant]** Queries shop logistics profile. `get_shop_detail()` **[Tool]** `Returns: origin Guangzhou, carriers YTO/Deppon, freight insurance available.` **[Assistant]** Clarifies: ¥4999 is product price only, shipping billed separately at checkout (est. ¥35–55 via YTO); delivery 2–4 days (up to 6 during peak). **[User]** Confirms and ends conversation.

Figure 10: DynEvo resolves a multi-constraint pre-sale inquiry using targeted product and logistics tools, while several baselines either avoid tool use or apply irrelevant post-sale procedures.

Case 2: In-Transit Logistics Interception & Address Sync

Scenario: DailyOps (Risk-Sensitive Phase) **Tools Used:** 4/12 **Steps:** 6

Task Requirement. *A buyer discovers the shipping address was entered incorrectly (Qingdao → Harbin) after the parcel has already been picked up. The agent must intercept the in-transit shipment, update the logistics address, synchronize the order record, and confirm the carrier and revised delivery estimate.*

Tool	Description
`get_logistics_detail`	Retrieve current shipment state, carrier, pickup time, and address
`modify_logistics_state`	Trigger carrier-side interception request for in-transit parcel
`modify_logistics_address`	Update destination address in the logistics system
`modify_order_address`	Synchronize corrected address to the order record

Why Baselines Fail. ***DSPy*** (sat=0.10) processes a completely different user scenario (contact-lens recommendation) because it fails to parse the address-correction intent and never queries logistics state. ***EvoAgentx*** (sat=0.10) similarly handles an unrelated task (T-shirt purchase), suggesting both baselines lack the stateful multi-tool chaining required to complete the interception workflow across four sequential dependent tool calls.

DynEvo Trajectory.

Turn	Interaction
1	**[User]** Reports address filled incorrectly (Qingdao Huangdao District); requests change to Harbin Hulan District, Central Ave 77; also asks carrier and ETA.
2	**[Assistant]** Reasons: must check logistics state before modifying (parcel may already be in transit). `get_logistics_detail()` **[Tool]** `Returns: parcel in transit, carrier JD Logistics, pickup at 2026-06-11 15:00, current address Qingdao.`
3	**[Assistant]** Parcel in-transit — interception required before address change. `modify_logistics_state(new_state="intercept")` **[Tool]** `Returns: interception request submitted, logistics_id=79491100506557.`
4	**[Assistant]** Interception confirmed; now update logistics destination. `modify_logistics_address(new_address="Hulan District, Harbin, Heilongjiang, Central Ave 77")` **[Tool]** `Returns: logistics address updated successfully.`
5	**[Assistant]** Logistics address updated; must sync order record to match. `modify_order_address(new_address="Hulan District, Harbin, Heilongjiang, Central Ave 77")` **[Tool]** `Returns: order address updated, order_id=2516202006394557.`
6	**[Assistant]** Reports full completion: interception confirmed, both logistics and order address changed to Harbin; carrier JD Logistics; estimated arrival Jun 14–15. Advises user to monitor tracking and report back if status not updated by tonight. **[User]** Confirms and ends conversation.

Figure 11: DynEvo executes a sequential tool chain to resolve an in-transit address error, including state query, interception, logistics-address update, and order synchronization.

Case 3: Fraud-Resistant Post-Sale Return Under Risk-Sensitive Policy

Scenario: DailyOps (Risk-Sensitive Phase) **Tools Used:** 1/12 **Steps:** 4

Task Requirement. *A buyer claims quality issues and demands refund-only (no return). Under the active risk-sensitive policy, the agent must reject the refund-only request for a non-perishable item, verify the 7-day return eligibility, and guide the buyer to the correct return-and-refund procedure while maintaining user satisfaction.*

Tool	Description
`get_item_detail`	Retrieve item category, return policy, and 7-day eligibility status

Why Baselines Fail. ***DSPy*** (sat=0.40, comp=0.90) partially complies — it mentions correct return procedures but is unable to close the conversation with user acceptance, leaving the issue unresolved.
EvoAgentx (sat=0.20, comp=1.00) is policy-compliant but handles an entirely unrelated scenario (lipstick purchase), demonstrating that without dynamic context adaptation the baseline fails to engage with the actual post-sale fraud case.

DynEvo Trajectory.

Turn	Interaction
1	**[User]** Claims quality defects on received shapewear (poor workmanship and fabric); requests refund-only without returning the item.
2	**[Assistant]** Recognizes refund-only risk under current policy; queries item details first. `get_item_detail()` **[Tool]** `Returns: item is apparel (non-perishable), 7-day no-reason return supported.`
3	**[Assistant]** Declines refund-only (platform rule: non-perishable items require return). Proposes alternative: 7-day no-reason return-and-refund. Provides step-by-step instructions (apply via order → select return reason → ship back → fill tracking number). Notes that if quality defect is confirmed, return shipping covered by seller.
4	**[User]** Accepts: "Since refund-only is not supported, I'll go with 7-day return-and-refund." Confirms will submit application and ship item back. **[Assistant]** Confirms procedure, reiterates three key steps (use system-generated return address, fill tracking number, expect refund in 1–3 business days after receipt).
5	**[User]** Confirms and ends conversation.

Figure 12: DynEvo identifies a refund-only fraud attempt under the active risk-sensitive policy, rejects the non-compliant request, and redirects the user to a compliant return-and-refund procedure.