Concept Paper

Recommender Systems Should Now Be Designed Towards Agents

Chaoyue He *, Xin Zhou, Di Wang, Hong Xu, Wei Liu, Chunyan Miao

Posted Date: 2 April 2026

doi: 10.20944/preprints202604.0201.v1

Keywords: recommender systems; autonomous agents; recommender systems towards agents (RSTA); large language models (LLMs); multi-agent systems; agent-facing interfaces; tool use

 doi:10.20944/preprints202604.0201.v1

Article

Recommender Systems Should Now Be Designed Towards AGENTS

Chaoyue He [1], Xin Zhou [1], Di Wang[1], Hong Xu [1], Wei Liu [2] and Chunyan Miao [1]

[1] Alibaba–NTU Global e-Sustainability CorpLab (ANGEL), Singapore
[2] Alibaba Group, Hangzhou, China
* Correspondence: cyhe@ntu.edu.sg

Abstract

This position paper argues that **recommender systems should now be designed towards agents**. We use **recommender systems towards agents (RSTA)** to denote systems whose immediate consumer is an acting agent, an orchestration layer, or a multi-agent system; whose ranked objects are actionable interventions rather than human-viewable items; and whose success is measured by downstream trajectory utility under preference, cost, policy, and safety constraints. We advance three falsifiable claims: (1) *priority-sensitive ranking* can improve trajectory utility even when candidate sets are small, (2) *service-side information* can create value that local planning alone cannot fully recover, and (3) *oversight actions* such as verify, ask, defer, and escalate should be treated as recommendables rather than post-hoc filters. We sharpen the exclusion boundary against planning, routing, and human-facing recommendation; recast a WorkArena-style hardware-order task family as a full RSTA worked example with an explicit candidate inventory, ranked intervention slate, and trajectory-level objective; and outline an agenda spanning candidate-set reconstruction, oversight-aware ranking, service-to-agent interfaces, multi-agent orchestration, interface-robust evaluation, and governance. The goal is not to relabel all agentic decision making. It is to identify a critical layer: when agents face massive action spaces or bounded compute, ranking dictates which trajectories they can reach—and which catastrophic failures they avoid.

Keywords: recommender systems; autonomous agents; recommender systems towards agents (RSTA); large language models (LLMs); multi-agent systems; agent-facing interfaces; tool use

1. Introduction

Recommender systems have historically been designed around people. Their canonical outputs are products, ads, songs, creators, news, and movies; their canonical objectives are relevance, utility, conversion, retention, and satisfaction [1–4]. Even as the methodology evolved from matrix factorization to sequential, contextual, conversational, causal, and reinforcement-learning formulations, the final recommendation usually still targeted a human chooser [5–13]. We call this long-standing paradigm *recommender systems towards humans* (RSTH).

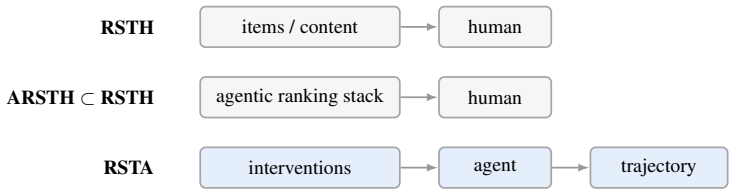

Figure 1. Three recommendation paradigms that are easy to conflate. Classical RSTH allocate human attention; ARSTH use agentic modules inside a still human-facing stack; RSTA target an acting agent and are judged by downstream trajectory outcomes.

 doi:10.20944/preprints202604.0201.v1

A nearby but different line of work uses LLMs and agents *inside* the recommendation pipeline while keeping the final recipient human. We refer to this family as *agentic recommender systems towards humans* (ARSTH). Recent surveys of foundation-model and LLM-powered recommender systems make this trend explicit [14–16]. ARSTH are important, but they are still a subset of RSTH: their ranked outputs remain human-facing. Figure 1 illustrates the distinctions among these paradigms.

This position paper argues that recommender systems should now be designed towards agents. We refer to this third destination as *recommender systems towards agents* (RSTA). Modern agents do not merely answer questions. They browse, retrieve, compare, call APIs, inspect files, manipulate software, invoke verifiers, and coordinate with other agents. Tool-use, web, app, and multi-agent environments already expose repeated decision points where ranking changes what trajectories an agent can reach [17–26]. This matters not only to recommender systems, but also to LLM agents, tool-use evaluation, multi-agent orchestration, AI systems interfaces, and market design and governance.

Services must also increasingly face agents directly rather than only human users. As industry leaders point out, the software ecosystem will need to be fundamentally re-engineered to keep up with the execution speed and scale of autonomous AI agents [27,28]. Open interoperability and tool protocols already make agent-facing capability surfaces more legible [29–31], while analyst reports forecast massive growth in agentic features, task-specific agents, and digital labor inside ordinary applications (see Figure 2) [32–36]. In a world where software is re-architected for machine consumption, recommendation does not disappear. It becomes the essential routing and filtering layer of the agent-facing interface.

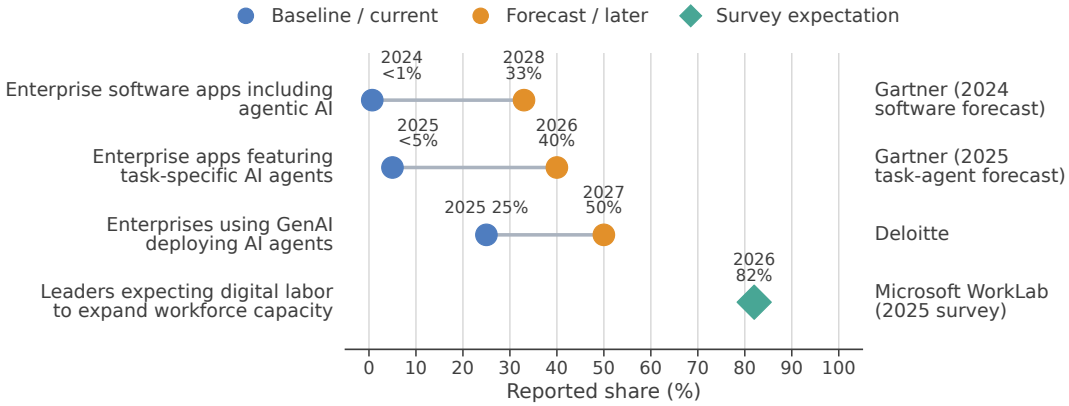

Figure 2. Quantitative signals of growing agent-facing software and digital labor.

A natural objection follows immediately: if agents become stronger reasoning engines, should they not simply ingest the environment and plan everything internally? Our answer is no, for two foundational reasons. First, **context limits and compute costs**. Modern digital environments genuinely expose massive candidate spaces—an agent cannot hold tens of thousands of API schemas, product catalogs, or enterprise approval paths in its working memory without suffering severe latency, cost, and attention-degradation. Filtering and ranking remain existential systems requirements. Second, even when the immediate candidate space is modest, **priority order dictates execution**. Because internal planning is compute-bounded and path-dependent, an agent can only deeply evaluate a bounded prefix of actions. If an oversight action (like *verify*) or an optimal API call is not ranked highly, the agent will never consider it, leading to brittle or unauthorized trajectories. The RSTA layer is therefore not merely a rebranding of offline reinforcement learning or contextual bandits; it is a distinct architectural layer that brings the scalability, candidate-generation, and multi-stakeholder optimizations of classical recommender systems directly into the agent's action space. Section 3 formalizes both regimes.

We make four concrete contributions. (1) We provide a clean boundary for what does and does not count as RSTA. (2) We turn the framing into three falsifiable claims. (3) We add a worked example by recasting a WorkArena-style hardware-order task family as a RSTA problem over grounded interventions. (4) We outline a focused agenda plus a lightweight evaluation template in Appendix I.

 doi:10.20944/preprints202604.0201.v1

The goal is not to rename all agentic decision making. The goal is to isolate a layer that is empirically testable, methodologically useful, and increasingly important.

2. What RSTA is—and What It Is Not

Our key distinction is not whether agents appear *somewhere* in the pipeline. It is whether the final ranked output is consumed by a human or by an acting agent. We therefore reserve RSTH for human-targeted ranking, reserve ARSTH for agentic modules inside human-targeted recommendation, and use RSTA only when the ranking is directed toward an acting agent, an orchestration layer, or a multi-agent system.

Definition.

A *recommender system towards agents* (RSTA) is a system that, given an agent's task, state, and feasible intervention inventory, produces a ranking over candidate interventions for that agent, with the explicit objective of improving downstream trajectory utility under preference, cost, policy, and safety constraints. The ranked objects can include tools, documents, offers, parameters, workflow branches, delegates, verifiers, and ask/act/defer/escalate options. We use *authorized agency* to denote the portion of the action surface the agent is both able and permitted to exercise.

A practical boundary test follows immediately. A problem counts as RSTA only when four conditions all hold: (1) the immediate recipient of the ranking is an acting agent or orchestration layer; (2) the ranked objects are explicit or reconstructable candidate interventions; (3) the ranking can change the downstream trajectory rather than only the surface presentation; and (4) success is measured by trajectory outcomes, possibly with multiple stakeholders behind the agent. Table 1 outlines clear negative examples to sharpen this boundary.

Table 1. What is not RSTA. The concept is narrow on purpose. It is not "anything with an agent," not "all planning," and not "all routing."

Negative example	Why it is *not* RSTA	What would make it RSTA
A movie or news recommender that uses an LLM to write better explanations for a human	The final recipient is still a person, and the ranking allocates human attention rather than authorized agency	The ranking would need to be consumed by an acting agent whose trajectory changes as a result
A planner over primitive UI actions with no explicit or reconstructable intervention inventory	This is just planning over low-level actions; there is no state-conditioned ranking interface over candidate interventions	Expose grounded candidates such as workflow fragments, approval paths, verifiers, or delegates and evaluate their trajectory effects
A backend model router that silently swaps models for cost reasons	This is only routing unless an acting agent explicitly consumes a ranked slate of alternatives tied to trajectory utility	Make the alternatives machine-actionable interventions for the agent, such as model + verifier + abstain choices
A post-hoc safety veto applied after the productive action has already been chosen	Oversight is treated as an external filter rather than as a ranked alternative inside the action slate	Include ask, verify, sandbox, defer, and escalate as first-class recommendables
A static API directory sorted alphabetically or by popularity	The ranking is not task-conditioned and does not target downstream trajectory utility for a live agent state	Condition on the task, current state, available authority, and execution consequences

The exclusion boundary matters because it prevents the concept from becoming vacuous. A problem is not RSTA merely because an LLM or an agent appears in the stack. Conversely, a problem can be RSTA even when the menu is small, because the core object is not menu size alone; it is the ranking of grounded interventions for an acting agent under delegated and distributed agency.

Operationally, what separates RSTA from "just planning," offline RL, or routing is a specific architectural interface. It is defined by an explicit or reconstructable candidate inventory, a ranked slate exposed directly to the acting agent, a bounded top-K inspection or deliberation budget that forces priority, the integration of possible service-private signals, and a diagnostic framework that separates failure attribution into candidate generation, ranking, and execution. While these pieces can exist in isolation, consolidating them creates a distinct, testable layer that dictates how agents navigate massive or compute-bounded action spaces.

 doi:10.20944/preprints202604.0201.v1

3. Formalization: lArge Surfaces, Priority-Sensitive Sets, and Implications

At the most general level, an RSTA answers one question: among the feasible next-step interventions available at step t, which ones should be placed first for the agent? We use four ingredients: current state s_t, principal-side context ρ_t, optional service-side context σ_t, and feasible candidate set Ω_t. An RSTA outputs an ordered slate

$$R_t = (r_{t,1}, \ldots, r_{t,L_t}) = f(s_t, \rho_t, \sigma_t, \Omega_t), \tag{1}$$

where each $r_{t,i} \in \Omega_t$ and order is part of the output. For agentic systems, the useful object is rarely only a tiny verb inventory such as {click, type, call, stop}. The useful object is the **effective intervention space**, the grounded and typed set of feasible next-step choices:

$$\Omega_t = \{x = (o, \theta, m, d) \mid o \in O_t,\ \theta \in \Theta_t,\ m \in M_t,\ d \in D_t,\ \text{and } x \text{ is feasible at } t\}. \tag{2}$$

Here O_t contains candidate objects such as items, tools, files, DOM elements, pages, sellers, or delegates; Θ_t contains feasible parameter settings; M_t contains execution modes such as *act*, *ask*, *defer*, *verify*, *sandbox*, or *escalate*; and D_t contains coordination choices such as collaborator, protocol, or aggregation mode. In a single-agent deployment, d can be a null delegate.

Let $\tau = (s_t, x_t, s_{t+1}, x_{t+1}, \ldots)$ denote the realized future trajectory after step t. We evaluate a ranking through the scalar utility of the trajectory it helps induce:

$$U(\tau) = u(\tau \mid \rho_t, \sigma_t). \tag{3}$$

Depending on the application, u may reward task completion and preference satisfaction, and penalize cost, delay, risk, or policy violations.

Large-surface regime.

In the first regime, recommendation matters because $|\Omega_t|$ is large or combinatorial. Shopping, tool use, web navigation, and enterprise workflows all produce this pattern. WebShop contains 1.18 million products, ToolLLM is built from 16,464 real-world APIs, AppWorld exposes 457 APIs across nine apps, and WorkArena-style service workflows contain rich grounded action surfaces [19,21,23–25]. In such settings recommendation still performs filtering and ranking, but over actionable interventions rather than only items.

Priority-sensitive regime.

The second regime is smaller but central to our framework. Even when $|\Omega_t|$ is modest, *ranking order can still matter* because internal planning is compute-bounded, latency-bounded, order-sensitive, and often path-dependent. Under bounded deliberation, the agent can only afford to inspect a top-K prefix of the ranked slate, which we denote as $P_t \subset \Omega_t$. The agent then selects its next action $a_t \in P_t$ based on its internal evaluation. If the ranking is poor and the globally optimal intervention $x^\star$ falls outside this prefix ($x^\star \notin P_t$), the agent is forced to choose a suboptimal action. This leads to a strict drop in the expected downstream trajectory utility:

$$\max_{x \in P_t} U(\tau \mid x) < U(\tau \mid x^\star). \tag{4}$$

This is the elementary but load-bearing reason recommendation survives even when menus are not large: ranking determines which interventions receive serious internal evaluation. Table 2 contrasts these two regimes.

 doi:10.20944/preprints202604.0201.v1

Table 2. Two regimes in which recommendation survives strong agents. The first resembles classical large-candidate recommendation; the second is the additional regime emphasized by RSTA.

Regime	Representative cases	Why an RSTA layer remains useful
Large-surface filtering	shopping catalogs, API hubs, dynamic DOMs, workflow fragments	the agent still needs filtering and re-ranking over grounded objects, parameters, bundles, and verifiers
Priority-sensitive small sets	ask/act/verify, choose among a few tools or models, pick a verifier or delegate	bounded deliberation, heterogeneous evaluation cost, and path dependence make order matter even when the menu is modest

Testable implications.

Claim 1: Priority-sensitive ranking improves trajectory utility even when candidate sets are small. Test by fixing the candidate inventory, varying slate order only, and measuring downstream trajectory utility rather than only next-step accuracy.

Claim 2: Service-side information creates value that local planning alone cannot fully recover. Test by comparing an agent-local ranker with and without access to service-private signals such as permissions, inventory, compatibility, reliability, or policy state.

Claim 3: Oversight actions should be modeled as recommendables, not post-hoc constraints. Test by allowing verify, ask, defer, sandbox, or escalate to compete in the slate and comparing trajectory-level safety and compliance against systems that apply guardrails only after ranking productive actions.

4. Worked Example: Recasting a WorkArena-Style Task as RSTA

To make the framing concrete without overclaiming a specific environment instance, we recast a *WorkArena-style hardware-order task* as a recommendation problem over grounded interventions [24,25]. WorkArena-style environments include service-catalog tasks in which an agent must order the right hardware item with the right specifications. The RSTA question is not only whether the final order is eventually correct. It is which next-step interventions should be surfaced first so that the trajectory stays accurate, efficient, and safe.

Consider a decision point at which the agent has reached the hardware catalog for a replacement laptop charger and sees several matches. One explicit candidate inventory is shown in Table 3. The example is faithful at the task-family level rather than a claim about one verbatim environment instance; the point is to make the candidate inventory, slate, and failure analysis fully explicit.

Table 3. An explicit inventory for a WorkArena-style hardware-order decision point. The key object is not a primitive browser verb alone, but a slate of grounded productive and oversight interventions.

ID	Candidate intervention $x \in \Omega_t$	Type	Why it belongs in the slate
x_1	open the most promising approved charger SKU and inspect its configuration fields	act	this is the shortest high-information path when the top match is likely correct
x_2	run a compatibility check against the employee's exact laptop model before purchase	verify	this reduces wrong-order risk even when the catalog title looks plausible
x_3	open a second near-match from the approved catalog for comparison	act	comparison is useful when several items share similar names but differ in wattage or connector type
x_4	ask the principal whether a travel charger is acceptable as a substitute if the exact SKU is unavailable	ask	this resolves a missing preference that changes which products are admissible
x_5	request manager approval for expedited shipping above the delegated threshold	escalate	this turns an otherwise unauthorized fast purchase into an authorized one
x_6	submit the first plausible item immediately without verification	act	this may save steps, but it is brittle when similar catalog items differ in important fields
x_7	purchase a third-party express-shipping charger from a non-approved seller	act	this can appear efficient, but it risks policy breach, reliability problems, and human cleanup

To see why this requires an RSTA layer rather than just local LLM planning, we map this inventory directly to our three claims.

First, consider **Claim 1 (priority-sensitive small sets)**. Even though this menu contains only seven interventions—easily fitting into an LLM's context window—an agent's internal deliberation is bounded by token budgets, latency requirements, and API rate limits. If the agent can only afford to

 doi:10.20944/preprints202604.0201.v1

deeply evaluate or simulate a top-K prefix (e.g., $K = 2$), the ranking strictly dictates the trajectory. If the premature submit (x_6) is ranked above the compatibility check (x_2), the agent executes a brittle shortcut before it ever considers verification.

Second, consider **Claim 2 (service-side information asymmetry)**. Why should the enterprise catalog platform determine this ranking rather than the local agent? Because the platform holds private signals. Suppose the platform's backend analytics reveal a 40% historical return rate for this specific charger SKU due to a recent manufacturer defect, a fact absent from the public product page. The local agent cannot know this. The service-side RSTA must therefore elevate the compatibility check (x_2) and the alternative comparison (x_3) to the top of the slate to protect downstream utility.

Finally, this inventory embodies **Claim 3 (oversight as recommendables)** by forcing productive actions (x_1, x_6, x_7) to compete directly against verification, clarification, and escalation (x_2, x_4, x_5) within the exact same slate.

A plausible service-aware ranked slate is therefore:

$$R_t = [x_2,\ x_3,\ x_1,\ x_5,\ x_4,\ x_6,\ x_7]. \tag{5}$$

This order says: verify compatibility first (due to hidden return risks); compare the near-match next; inspect the best approved candidate only after verification; escalate for faster shipping if necessary; ask about substitutes if the exact item is not available; keep premature submit and non-approved purchase safely near the bottom.

The natural objective is trajectory-level rather than action-level:

$$U(\tau) = \alpha T(\tau) + \beta M(\tau) + \lambda G(\tau) - \gamma C(\tau) - \delta L(\tau) - \eta V(\tau), \tag{6}$$

where $T(\tau)$ is binary task completion, $M(\tau)$ measures how well the item matches the instruction, $G(\tau)$ is governance and authorization conformity, $C(\tau)$ is the spend, $L(\tau)$ represents latency or delay, and $V(\tau)$ penalizes policy or safety violations. The coefficients $\alpha, \beta, \lambda, \gamma, \delta, \eta \geq 0$ are weighting parameters that balance task efficiency against cost and operational risk.

This recast exposes a failure that coarse end-to-end evaluation can miss. Standard end-to-end evaluation may focus on whether an order request was eventually created and roughly matched the instruction. That can fail to distinguish two different trajectories: one that ranked verification early and completed the task through the intended service-catalog path, and another that ranked premature submit or non-approved purchase above verification. The latter may still look superficially successful while being brittle, unauthorized, or costly to clean up. From an RSTA perspective, the diagnosis is therefore not only "the agent failed" or "the agent succeeded." It is whether the ranking over grounded interventions surfaced the right actions in the right order. Table 4 highlights why this RSTA recast is methodologically different from ordinary end-to-end agent evaluation.

Table 4. Why a RSTA recast is methodologically different from ordinary end-to-end agent evaluation.

Evaluation axis	Typical end-to-end agent view	RSTA view
Choice set	hidden inside prompts, tool specs, or environment code	explicit or reconstructable candidate inventory Ω_t
Actions	productive actions dominate; oversight appears as a guardrail	productive and oversight interventions compete in the same slate
Metric	task success or exact-match proxy	trajectory utility with cost, safety, policy, and stakeholder outcomes
Failure attribution	one scalar failure label	separate diagnosis for candidate generation, ranking, inspection, and execution

5. Alternative Views

We highlight five strong objections to the RSTA framing, spanning agent capabilities, machine learning paradigms, and systems architecture.

 doi:10.20944/preprints202604.0201.v1

Counterargument 1: Stronger agents make recommendation obsolete.

The most immediate objection is that if foundation models become sufficiently capable planners, an explicit recommendation layer becomes unnecessary; the agent should simply ingest the environment and plan its own path.

Response: We disagree because stronger reasoning does not erase physical and operational constraints. First, *autonomy does not remove candidate scale*. A web agent still faces millions of domains; a buyer still faces massive product catalogs. Better planning helps reason *over* these choices, but it does not bypass the need for an efficient retrieval and filtering layer before the LLM's context window is invoked. Second, *autonomy raises governance stakes*. In delegated settings, the central question is often not whether the agent *can* act, but whether it *should*. The RSTA layer provides a structural choke-point to enforce shared autonomy and intelligent delegation [37–39], treating oversight actions (verify, ask, defer) as recommendables that compete directly with productive actions under constrained computational budgets.

Counterargument 2: RSTA is just offline RL or contextual bandits.

A classical machine learning objection is that if the "user" is the agent state, the "item" is an action, and the "objective" is trajectory utility, the problem is mathematically identical to a Markov Decision Process. Under this view, RSTA is merely a rebranding of offline reinforcement learning or contextual bandits [9,40].

Response: While the mathematical DNA overlaps, RSTA provides a fundamentally distinct *systems architecture* and *inductive bias*. RL agents typically assume control over the policy and treat the environment as a passive transition function. In contrast, modern digital ecosystems are multi-stakeholder. A travel platform or enterprise SaaS actively wants to shape the agent's behavior to respect inventory limits, load balancing, and platform safety. Recommender systems bring mature architectural solutions to this exact problem: decoupled two-tower models, efficient candidate generation pipelines, and multi-objective ranking functions that balance the agent's utility against the service's operational constraints [12].

Counterargument 3: RSTA is simply Retrieval-Augmented Generation (RAG) for tools.

A common systems objection is that finding the right tool is just a vector search problem: embed the agent's state, retrieve the top-K API schemas, and append them to the prompt.

Response: RAG optimizes for semantic similarity; RSTA optimizes for downstream trajectory utility. A pure semantic retriever cannot balance dynamic service-side constraints (e.g., rate limits, real-time inventory) against the agent's budget. Furthermore, RAG retrieves passive documents, whereas RSTA ranks a heterogeneous space of executable interventions—including meta-actions like *verify*, *ask*, and *escalate*—that fundamentally alter the execution graph rather than just augmenting the context.

Counterargument 4: Infinite context windows will eliminate the need for filtering.

As foundation models scale to millions of tokens, one might argue that an agent can simply ingest the entire candidate space (e.g., a whole product catalog or enterprise API directory) and evaluate it jointly.

Response: Even if context limits vanish, reasoning over massive action spaces introduces severe latency, cost, and attention degradation. More critically, the "infinite context" argument assumes the service provider is willing to transmit its entire dynamic backend state to the client agent. In reality, proprietary databases, dynamic pricing, and live inventory require a federated bottleneck. RSTA provides this necessary privacy-preserving interface, whether integrated locally with structured API responses or deployed explicitly as a service-side ranker.

 doi:10.20944/preprints202604.0201.v1

Counterargument 5: Agent-to-service interaction should be deterministic, not probabilistic.

Software engineering principles suggest that machine-to-machine communication should rely on deterministic routing and precise API contracts, not probabilistic recommendation slates.

Response: While the *execution* of an API must be deterministic, the *discovery and selection* of that API in an open ecosystem is inherently probabilistic. When a principal's intent is ambiguous, or when thousands of overlapping services compete (e.g., booking a flight across multiple vendors), hard-coded deterministic logic is brittle. RSTA acts as the crucial probabilistic bridge between fuzzy human intent and deterministic machine execution.

6. Service-to-Agent Interfaces and Governance

It is important to clarify that the RSTA framework does not strictly require a complex, federated two-layer architecture. At its core, RSTA is simply the ranking of grounded interventions for an acting agent. However, as deployments scale, a natural *service-facing* pattern emerges to solve the information asymmetry problem. In many domains, the service knows something the local agent does not: inventory state, compatibility graphs, permission structure, reliability priors, pricing, or latent policy constraints. An agent-facing capability surface—acting as a service-side ranker—can therefore create value that local planning alone cannot reconstruct. The output need not be a passive list; it can be a machine-readable action slate with attached evidence, constraints, and executable affordances.

This is also why RSTA is governance-relevant. RSTH allocate scarce human attention. RSTA allocate scarce *authorized agency*. A weak RSTH wastes a click; a weak RSTA can overspend money, violate policy, execute the wrong configuration, or prioritize a brittle shortcut over a safe and intended workflow. Table 5 maps these ranking failures to concrete harms.

Table 5. From ranking errors to concrete harms. RSTA failures matter because the slate controls what forms of agency are surfaced first to an acting system.

RSTA failure	Concrete harm	Why the ranking layer matters
Act outranks verify or ask	wrong execution or overspending	the system allocates delegated authority too aggressively
Submit outranks policy or compatibility checks	compliance breach or misconfiguration	authorization and verification are treated as afterthoughts rather than ranked alternatives
Execution outranks sandbox or test	unsafe execution	the slate ignores asymmetry between productive and irreversible actions
Delegate or protocol misranking	coordination failure or protocol gaming	orchestration choices are themselves recommendables with real downstream cost

Some deployments may eventually separate agent-side and service-side rankers, while others may keep a single integrated ranking layer. That architectural choice should be treated as a future systems agenda rather than part of the core definition. The core definition is simpler: rank grounded interventions for an acting agent, and evaluate the resulting trajectory.

7. A Focused Research Agenda

The research agenda does not require an entirely new environment literally named "RSTA." Existing agent environments already contain recommendation-shaped decisions. The near-term opportunity is to expose candidate inventories, rank productive and oversight actions jointly, and evaluate trajectory outcomes with the correct stakeholders in mind.

We emphasize five priorities, as outlined in Figure 3.

 doi:10.20944/preprints202604.0201.v1

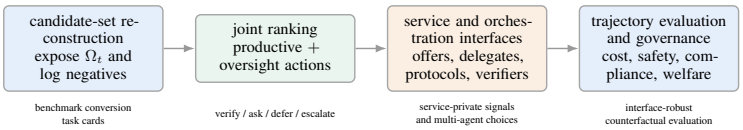

Figure 3. A compact RSTA agenda. We argue for moving left-to-right: reconstruct candidate inventories, rank productive and oversight actions jointly, expose service and orchestration choices as recommendables, and evaluate the resulting trajectories with governance in view.

1. Candidate-set reconstruction.

End-to-end agent benchmarks often hide feasible alternatives inside prompts, tool specifications, or environment code. RSTA work should make Ω_t explicit or deterministically reconstructable. Without that, it is hard to distinguish ranking quality from candidate-generation quality.

2. Oversight as a recommendable.

Verify, ask, defer, sandbox, and escalate should compete directly against productive actions. This is the clearest methodological implication of Claim 3.

3. Service-to-agent interface design.

If services increasingly face agents directly, then ranking becomes partly an interface problem: which objects, schemas, evidence packets, and executable affordances should be exposed, in what order, and with what guarantees?

4. Multi-agent orchestration as recommendation.

Delegate choice, protocol choice, topology choice, and critic selection are ranking problems whenever they shape the trajectory of an agent team rather than merely backend plumbing [26,41–45].

5. Interface-robust evaluation and governance.

Agentic performance is often sensitive to the harness rather than only the base model. RSTA evaluation should therefore report interface perturbations, side effects, approval events, and stakeholder outcomes, not only task success [46–50]. Appendix I provides a task card that makes candidate sets, recommendables, stakeholders, and trajectory metrics explicit.

8. Optimizing RSTA: From Clicks to Trajectories

Moving the immediate target of recommendation from humans to agents forces a fundamental shift in how systems learn and evaluate, transitioning from optimizing immediate, point-wise engagement signals (e.g., clicks, dwell time) over static offline datasets to optimizing delayed, multi-step trajectory utility within executable environments. The most profound methodological shift lies in supervision: whereas classical RSTH treat a skipped item as a soft negative and a click as a positive, RSTA must leverage grounded execution traces—where hallucinated tool calls, rollbacks, or unauthorized state mutations serve as severe hard penalties, and successful oversight actions (like *verify*) serve as highly positive signals even if they temporarily delay completion. This connects RSTA directly to modern preference learning via Direct Preference Optimization (DPO) [51–55] and offline reinforcement learning [56–58], where the objective is to reliably induce safer, cheaper, or faster task completion trajectories rather than satisfying human aesthetics. Finally, RSTA offer an evaluation advantage through *executable counterfactuals*; unlike human recommendation where unseen slates cannot be genuinely simulated, simulable RSTA environments (e.g., WorkArena-style tasks or OSWorld) allow researchers to execute alternative slates and strictly verify the causal impact of off-policy ranking interventions on downstream trajectory utility.

 doi:10.20944/preprints202604.0201.v1

9. Broader Implications: Interfaces, Data, and Market Structure

The implication of this shift is conceptual as much as technical. Recommendation is moving from the screen into the loop. Once services and platforms interact directly with agents, the unit of optimization is no longer only *what captures a person's attention*. It is also *what productively and safely shapes an agent's next reachable trajectory*. That shift changes interfaces, supervision, and competition.

First, it changes interfaces. Agent-facing services need not merely wrap old APIs with natural-language endpoints. They may need structured, evidence-bearing, executable recommendation surfaces that expose bundles, constraints, and affordances in forms agents can consume efficiently. In many domains, the interface question and the ranking question become inseparable.

Second, it changes data flywheels. Human-facing recommenders learn from impressions, clicks, dwell time, and conversion. Agent-facing systems can learn from tool calls, arguments, retries, approval events, overrides, verifications, side effects, and task outcomes. That creates richer supervision, but it also raises the cost of ranking errors because the feedback loop is closer to real action.

Third, it changes market structure. If a growing share of digital interaction is mediated by agents, then services compete not only on human UX but also on being machine-readable, composable, verifiable, and trustworthy to agentic consumers. Metadata quality, provenance, policy transparency, and protocol participation become competitive dimensions. Recommendation research should engage with that transition directly rather than treating it as someone else's infrastructure problem.

10. How the Claims can be Tested

To move beyond definitions, the most useful next step is a small experimental program that could falsify our framework. The three claims in Section 3 are designed to be tested with existing benchmark families plus modest additional instrumentation.

Testing Claim 1.

Start from an environment with a modest action menu, such as tool selection or approval-choice settings, and hold the candidate inventory fixed. Then vary only the order of the slate delivered to the agent, while constraining the agent's evaluation budget so that it can only inspect a top-K prefix or a bounded number of expensive checks. If trajectory utility changes materially under these interventions, then priority-sensitive ranking is doing real work even when menus are small.

Testing Claim 2.

Compare two rankers on the same tasks and with the same principal-side context: one receives only agent-local state, while the other additionally receives service-private signals such as permissions, inventory, compatibility, reliability, or latent policy state. The relevant outcome is not merely next-step preference accuracy. It is whether access to service-private signals improves downstream utility in ways the local agent could not reconstruct alone.

Testing Claim 3.

Evaluate two otherwise matched systems: one ranks productive and oversight actions in a common slate, while the other ranks productive actions first and applies guardrails only afterward. Then measure not only completion, but also approval efficiency, compliance, side effects, and human cleanup burden. If oversight-as-recommendable improves usable autonomy or reduces harmful interventions at comparable completion rates, then the post-hoc-guardrail view is incomplete.

11. Conclusions

Recommendation is moving from the screen to the execution loop, shifting the optimization unit from human attention to authorized machine agency. The central question is no longer what humans should see, but what acting agents should prioritize, verify, or delegate. As digital services expose agent-facing interfaces and planning remains constrained by compute and governance, an explicit

 doi:10.20944/preprints202604.0201.v1

RSTA layer becomes indispensable for safe, efficient operation. By ranking productive and oversight actions jointly, RSTA provides a robust foundation for next-generation interactive systems where trajectory utility, robustness, and strict governance are engineered directly into the slate.

Acknowledgments: This research is supported by the RIE2025 Industry Alignment Fund (Award I2301E0026) and the Alibaba–NTU Global e-Sustainability CorpLab.

Appendix A. Extended Related Work and Market Context

This appendix maps the adjacent literatures that make the RSTA framing timely, extending the foundational concepts introduced in Section 1.

Recommendation and interaction paradigms.

Before agents became direct consumers of ranked interventions, recommender systems had already expanded well beyond static collaborative filtering to encompass deep learning [59]. Hybrid recommenders [60], matrix-factorization and pairwise-ranking formulations [3,4], sequential and transformer-based recommendation [5,7,61–63], conversational recommendation [10,11,64], multi-stakeholder recommendation [12], and causal or reinforcement-learning views of recommendation, including counterfactual estimators and slate optimization [9,13,40,65–70] all broadened what a recommender system can optimize. In parallel, interface-agent and mixed-initiative work treated personalization, delegation, and control transfer as first-class design problems [37–39,71–76]. Furthermore, early work on agent-mediated electronic commerce and shopbots [77–80] laid the economic groundwork for machine consumers, while classical principal-agent theory [81] formalizes the delegation risks. These literatures matter because RSTA inherit ranking questions from the former and autonomy questions from the latter.

Agent behavior, tool use, and evaluation.

Recent agent work shifted from prompt-only interaction to tool use, reflection, web navigation, and long-horizon work tasks [17,18,82–91]. The evaluation ecosystem now includes generalist agent benchmarks such as AgentBench [92], workplace-style digital-worker environments such as TheAgentCompany [93], and broader surveys and infrastructure analyses of agent evaluation methodology [50,94–97]. For this paper, the key point is not benchmark performance per se; it is that these environments repeatedly expose explicit or recoverable choice points where ranking over grounded interventions affects downstream trajectories.

Interoperability, runtimes, and public ecosystems.

Open protocols and runtime layers are making agent-facing interfaces more legible. Beyond the MCP and A2A announcements themselves [29,30], recent surveys synthesize emerging protocol families and their design trade-offs [31]. This matters because service-to-agent recommendation is partly an interface-design problem: what the service exposes, and how it exposes it, can affect both ranking quality and downstream execution.

Market and deployment context.

Industry reports increasingly describe agents as a software and workflow layer rather than an isolated model feature. McKinsey's state-of-AI and agentic-organization reports, Gartner's enterprise-agent forecasts, Deloitte's enterprise-adoption forecast, and Microsoft WorkLab's digital-labor framing all point in that direction [27,28,32–36,98,99]. We also anticipate this extending to synthetic media and autonomous creative generation [100,101]. We use these sources only as directional timing signals, not as proof that any one interface design or market structure will dominate.

 doi:10.20944/preprints202604.0201.v1

Appendix B. Boundaries Against Adjacent Literatures

We introduced a narrow boundary on purpose in Section 2. Table A1 expands this comparison and shows why RSTA is adjacent to, but not reducible to, planning, routing, shared autonomy, or learning to defer.

Table A1. Comparison with adjacent literatures. Every row is related to RSTA; none is sufficient to replace the concept.

Concept	Immediate recipient	Typical objects	Why it is not identical to RSTA
RSTH	human	items, content, ads	recipient and beneficiary typically coincide; ranking is exposed directly to a person
ARSTH	human	still human-facing recommendations	agentic modules improve a human-facing recommender system, but the final consumer remains human [14–16]
Planning or policy learning	agent	full action policies	broader umbrella; may omit explicit candidate-set semantics or ranking interfaces
Tool use or routing	agent	tools, models, documents	important RSTA subcases, but narrower than the full space of bundles, modes, and oversight choices [102–106]
Mixed initiative or shared autonomy	human and/or agent	control transfer	authority sharing matters, but the literature is not organized around ranking grounded intervention candidates for agent trajectories [37,75,76]
Learning to defer or delegation	model or human	ask, abstain, handoff	an important subset of recommendables inside RSTA, not the whole problem [39,107–110]

Appendix C. Additional Formal Details and Derivation

This appendix supports Section 3. We keep the notation deliberately light, so four clarifications matter.

First, the tuple $x = (o, \theta, m, d)$ in Equation 2 should be read as a *typed intervention template*, not as a claim that every application has four equally rich coordinates. In a single-agent deployment, d can be a null value. In a fixed-parameter action, θ can be trivial. Some environments also collapse mode and object into one choice. The point of Equation 2 is simply to make explicit that agentic recommendation acts over grounded choices rather than over abstract verbs only.

Second, Equation 3 is intentionally general. A trajectory utility can always be written as a task-level scoring function

$$U(\tau) = u(\tau \mid \rho_t, \sigma_t),$$

where u is application-dependent. When a concrete decomposition is useful, one simple special case is

$$U(\tau) = \alpha T(\tau) + \beta P(\tau) + \lambda S(\tau) - \gamma C(\tau) - \delta R(\tau) - \eta V(\tau), \tag{A1}$$

where T denotes task completion, P principal utility or preference realization, S service-side utility, C cost, R risk, and V policy or safety violations. Our core argument does not depend on this additive form; it is only one convenient instantiation.

 doi:10.20944/preprints202604.0201.v1

Third, bounded deliberation does not require literal prefix scanning in every implementation. It is enough that recommendation order changes which candidates receive substantive internal evaluation. Let

$$V^{\pi_a}(R_t, s_t) = \mathbb{E}[U(\tau) \mid s_t,\ \pi_a \text{ receives } R_t] \tag{A2}$$

denote the value of ranking R_t for an internal planner π_a. If π_a can only deeply inspect a subset induced by the ranking and its budget, then $V^{\pi_a}(R_t, s_t)$ is generally not invariant to permutations of R_t. This is the operational sense in which recommendation matters even when the menu is modest.

Fourth, path dependence strengthens the case for a recommendation layer. Suppose that choosing candidate x_t changes not only immediate utility but also the next feasible candidate set:

$$\Omega_{t+1} \sim p(\Omega_{t+1} \mid s_t, x_t). \tag{A3}$$

Then an early ranking mistake can send the agent into a low-value branch whose later candidate sets are themselves worse. This is common in shopping, web navigation, and enterprise workflows.

Finally, service-facing RSTA is not reducible to local planning when the service has private signals z_t^σ unavailable to the local agent. In that case a service-side ranking function can be written as

$$R_t^{\sigma \to a} = f_\sigma(s_t, \rho_t, z_t^\sigma, C_t), \tag{A4}$$

where C_t is the current offer or action surface exposed by the service. If z_t^σ includes compatibility, fraud, inventory, or policy signals, then a purely local planner cannot reconstruct the same ranking from the user's state alone. This is one formal route by which agent-facing services create a genuinely distinct recommendation layer.

Appendix D. Domain Case Studies for Service-To-Agent Recommendation

Section 6 argued that services can rank structured action opportunities for agents using information that is not always locally available to the principal-side agent. The same structural pattern recurs in shopping, travel, enterprise software, and developer platforms. The subsections below make that pattern concrete one case at a time.

Appendix D.1. Shopping

Shopping makes preference elicitation, approval thresholds, and structured offers explicit. A buyer agent may need to clarify soft attributes such as acceptable substitutes, return-risk tolerance, or delivery urgency before acting [111,112]. It may also need to request human confirmation at salient thresholds, which aligns with broader human-AI interaction guidance on confidence, control transfer, and user trust [113,114]. Figure A1 visualizes this two-layer architecture, and Table A2 details a shopping-specific decomposition of the intervention space.

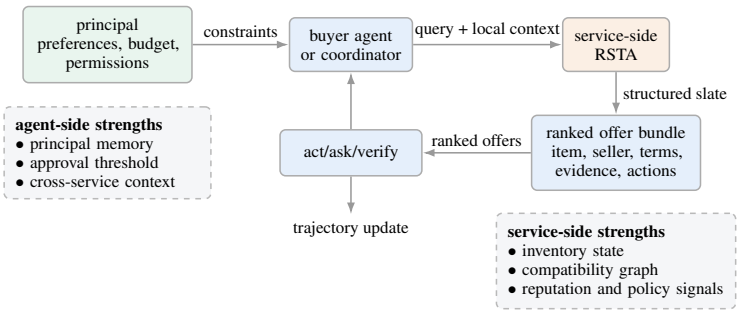

Figure A1. A shopping stack with an agent-side ranking layer and a service-side ranking layer. The agent-side layer contributes principal context and approval logic; the service-side layer contributes inventory, compatibility, and policy signals. This figure shows one important deployment pattern, not a claim that every RSTA deployment must contain both layers.

 doi:10.20944/preprints202604.0201.v1

Table A2. A shopping-specific decomposition of Equation 2. The table makes the effective intervention space concrete for one service-facing domain.

Layer	Typical contents	Why it matters	Where it is often known best
Object layer O_t	products, sellers, substitutes, accessories, bundles	determines the feasible commercial options	service and cross-shop aggregators
Parameter layer Θ_t	size, color, shipping, coupon, warranty, payment, return terms	changes utility even when the underlying product is fixed	service and principal jointly
Mode layer M_t	shortlist, compare, ask, reserve, cart, buy, wait, verify	determines whether the agent should act or seek more evidence	principal and policy layer
Delegation layer D_t	verifier, budget checker, comparison agent, human approval path	changes the reliability and governance of the trajectory	agent orchestration stack

Appendix D.2. Travel

Travel adds high-cost commitment points, corporate policy constraints, and rebooking uncertainty. A travel agent often knows the principal's calendar, layover tolerance, loyalty preferences, and approval rules, while a platform can rank fare bundles using private availability, fare rules, and partner-inventory signals. Figure A2 illustrates this dynamic.

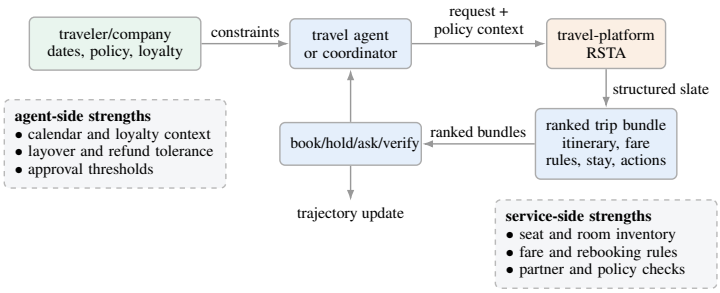

Figure A2. Travel as service-to-agent recommendation. The platform ranks structured itinerary bundles for an agent that already knows dates, preferences, and authorization rules.

Appendix D.3. Enterprise SaaS

Enterprise assistants add workflow state, permission structure, audit requirements, and approval routing. Here the recommendables are often not end-user-visible "items" at all, but workflow fragments, templates, connector choices, escalation paths, and approval actions. Figure A3 demonstrates this pattern.

 doi:10.20944/preprints202604.0201.v1

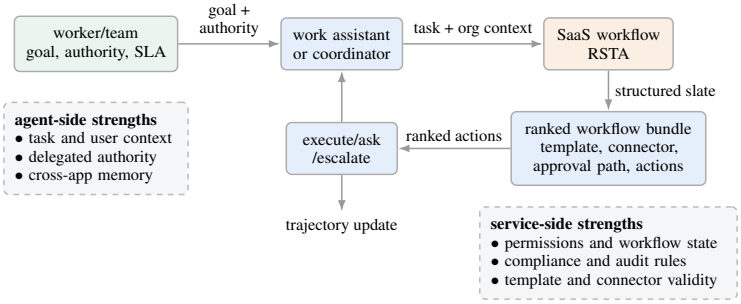

Figure A3. Enterprise SaaS as service-to-agent recommendation. The service ranks executable workflow choices rather than merely presenting UI elements for a human operator.

Appendix D.4. Developer platforms

Developer platforms expose another high-value case. A coding agent may hold local code intent and repository context, while the platform can rank tools, test bundles, environments, reviewers, and rollback paths using CI state, compatibility information, and deployment-risk signals. Figure A4 maps this architecture, and Table A3 summarizes the parallel case studies across all these domains.

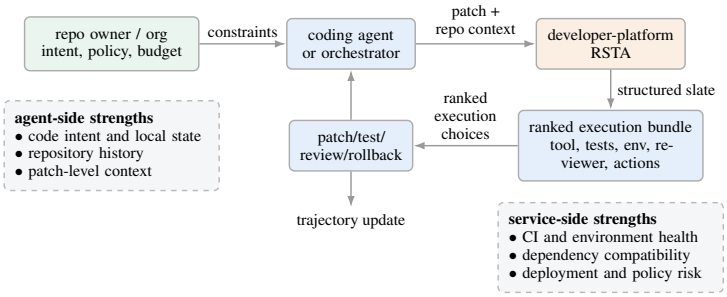

Figure A4. Developer platforms as service-to-agent recommendation. The platform can rank tool, verification, and rollout choices using signals that a local coding agent does not fully possess.

Table A3. Parallel domain case studies for service-to-agent recommendation. Each row instantiates the same core idea: the service-side layer can rank structured action opportunities for an acting agent using information that the principal-side agent may not fully possess locally.

Domain	What the service can rank for the agent	Important service-private or platform signals	Principal-side constraints	Representative resources
Shopping	products, sellers, bundles, coupons, shipping windows, return terms, compatibility checks, buy or reserve actions	inventory volatility, seller reliability, counterfeit risk, compatibility graphs, fulfillment likelihood	budget, brand exclusions, urgency, risk tolerance, approval threshold	WebShop, DeepShop, WebMall, AgenticShop [21,115–117]
Travel	itineraries, fare families, baggage bundles, hotel combinations, refund options, booking actions	seat availability, fare rules, rebooking frictions, partner inventory, corporate policy checks	dates, layover tolerance, loyalty preferences, travel policy, refund sensitivity	general web-agent environments provide partial travel analogues, especially for booking and itinerary-style web interaction [22,118]
Enterprise SaaS	workflow fragments, approval paths, document templates, connectors, escalation options	permission structure, compliance rules, org workflow state, template validity, connector health	delegated authority, SLA targets, audit requirements, team policy	WorkArena, WorkArena++, AssistantBench, and AppWorld provide close enterprise-workflow analogues [23–25,119]
Developer platforms	tools, test bundles, deployment targets, rollback actions, reviewer assignments, verifier workflows	CI state, dependency compatibility, environment health, repo policy, deployment risk	reliability threshold, code ownership, review rules, budget, sandbox policy	SWE-bench, SWE-agent, and OSWorld are the closest direct resources; AppWorld is a useful workflow analogue [23,120–122]

 doi:10.20944/preprints202604.0201.v1

Appendix E. Evaluation conversion cookbook

Section 7 argued that RSTA can be studied now. Table A4 makes that claim operational by mapping environment families to candidate inventories and evaluation targets.

Table A4. An evaluation conversion cookbook for RSTA. The core recipe is to expose Ω_t, log the chosen intervention, and evaluate downstream trajectories rather than only next-step accuracy.

Family	Existing resources	Recommendables to expose	Good evaluation targets
Tool and function selection	ToolLLM, API-Bank, ToolTalk, ToolSandbox, BFCL, Gorilla [19,20,85,86,89,90]	tools, schemas, arguments, retries, verifiers, abstentions	call validity, task completion, cost, latency, safe abstention
Shopping and browsing	WebShop, Mind2Web, WebArena, DeepShop, WebMall, AgenticShop, BrowseComp [21,22,87,115–118]	products, sellers, sources, filters, page actions, ask/buy/stop options	success, attribute match, browsing efficiency, source fidelity
Enterprise web and apps	AppWorld, WorkArena, WorkArena++, AssistantBench, TheAgentCompany [23–25,93,119]	forms, pages, templates, APIs, workflow fragments, approvals, clarification actions	task completion, workflow correctness, approval efficiency, context grounding
Computer use and software engineering	OSWorld, SWE-bench, SWE-agent, GAIA [88,120–122]	files, commands, tests, patches, rollback paths, reviewers	execution correctness, test pass rate, collateral damage, cost
Protocolized AI-native systems	AI-NativeBench and similar white-box protocolized suites [49]	agentic spans, handoffs, protocol choices, exposed affordances, recovery paths	failure attribution, protocol adherence, latency, diagnosability
Multi-agent coordination	AgentBench, MultiAgentBench, AutoSLURP, protocol-selection work [26,41–43,92,123–126]	delegatee, protocol, topology, verifier, aggregation strategy	team success, handoff quality, latency, message cost, robustness

These newer resources also broaden the methodological agenda. AgentBench and TheAgentCompany highlight generalist and workplace-style agent evaluation [92,93]; AI-NativeBench shows the value of white-box tracing over MCP/A2A-style systems [49]; and recent agentic-commerce auditing isolates the choice step itself, making exposure effects and model-dependent choice behavior measurable for AI buyers [127]. ASTRA-bench further shows the need for context-rich assistant evaluation rather than only context-free tool calls [128].

A second useful appendix perspective is scale. Table A5 records concrete evidence that agent candidate spaces are already nontrivial in practice.

 doi:10.20944/preprints202604.0201.v1

Table A5. Concrete scale evidence for the large-surface regime discussed in Section 3.

Environment family	Concrete scale evidence	Primitive verbs may still look small	Why recommendation remains central
Tool use	ToolLLM built from 16,464 APIs [19]	call, stop, retry	choose tool, schema, arguments, order, verifier
Shopping	WebShop contains 1.18M products [21]	search, click, buy	choose product, seller, bundle, shipping, ask/buy/wait
Web navigation	Mind2Web spans 2,350 tasks and 137 websites [22]	click, type, stop	choose element, source, query, continuation path
App ecosystems	AppWorld exposes 457 APIs across 9 apps [23]	call app action, edit, stop	choose app, API, parameters, plan structure
Enterprise workflows	WorkArena++ grows to 682 tasks; OSWorld has 369 tasks [25,122]	browse, edit, run, stop	choose forms, files, tests, approvals, escalation steps

Appendix F. Infrastructure and Quantitative Signals for Agent-Facing Services

The broader background in Sections 1 and 6 depends on more than environment progress. Table A6 gathers protocol and market signals that support the claim that many services may need agent-facing surfaces.

Table A6. Protocol, interface, and timing signals supporting the broader service-facing background.

Signal	What the source says	Why it matters for RSTA
MCP (Anthropic)	an open standard for connecting AI systems with external tools and data sources [29]	services are increasingly being exposed as machine-consumable capability surfaces rather than only human UIs
A2A (Google)	an open protocol for agents to communicate, exchange information, and coordinate actions across enterprise platforms [30]	recommendation increasingly happens inside a multi-service, multi-agent ecosystem rather than inside one app
Enterprise software forecasts (Gartner)	33% of enterprise software applications will include agentic AI by 2028, up from less than 1% in 2024 [32]	agent-facing capability surfaces are likely to become part of ordinary software, not just experimental demos
Task-specific agents (Gartner)	40% of enterprise applications will feature task-specific AI agents by the end of 2026, up from less than 5% in 2025 [33]	agent-targeted interaction is becoming a mainstream application-layer concern
Digital labor and agentic organizations	leaders expect digital labor to expand workforce capacity, and firms are moving toward humans and AI agents working together at scale [34–36]	once interaction is increasingly mediated by agents, services need ways to rank structured actions for machine consumers

 doi:10.20944/preprints202604.0201.v1

Appendix G. Expanded Research Agenda and Open Questions

Section 7 outlined the main priorities. Table A7 expands them into concrete questions, immediate tasks, and longer-term stakes.

Table A7. A more comprehensive research-agenda map for RSTA to support the diversity of future tasks and questions emphasized in Section 7.

Subdirection	Core questions	Near-term tasks	Longer-term value
Candidate-set construction	How should typed intervention inventories be represented, exposed, and reconstructed?	expose Ω_t, log negatives, separate candidate generation from ranking	cumulative science and better off-policy evaluation
Composite ranking	How should recommender systems score bundles, workflow fragments, and structured offers?	adapt slate and bundle methods to agent traces	better control over long-horizon execution quality
Autonomy calibration	When should the agent act, ask, abstain, verify, or escalate?	joint ask/act/defer training and approval logging	safer delegated automation with higher usable autonomy
Preference and authority elicitation	Which missing preferences, permissions, or thresholds should be clarified before action?	active clarification policies and approval-threshold modeling	better alignment between delegated authority and actual user intent
Multi-agent orchestration	Which delegate, protocol, topology, or verifier should be chosen?	convert coordination traces from MultiAgentBench and Auto-SLURP into ranking problems	more reliable agent teams and lower communication waste
Service-to-agent interface design	What objects, evidence packets, and affordances should services expose?	evaluate agent-facing slates in shopping, travel, and enterprise apps	a new interface layer for the agent economy
Interface-robust evaluation	Do results survive benign rewrites of tool names, schemas, or protocol surfaces?	report invariance scores and run interface perturbations	environment-invariant science rather than interface-specific shortcutting
Tracing, provenance, and diagnostics	What should be logged so errors can be attributed to candidate construction, ranking, orchestration, or execution?	trace agentic spans, decision boundaries, evidence packets, and recovery events	debuggable, auditable, cumulative RSTA systems
Markets, incentives, and governance	How do exposure, manipulation, fairness, audit, and protocol gaming change in agent-mediated markets?	stress tests for service-side incentives and authority boundaries	principled market design for agent-facing ecosystems

 doi:10.20944/preprints202604.0201.v1

These rows matter because delegated agents often act under incomplete principal specifications, ranking gains may otherwise reflect interface memorization rather than generalizable choice quality, and protocolized agent systems can fail in candidate construction, ranking, orchestration, or execution for very different reasons [48,49,111,112,129].

Appendix H. Failure Modes and Reporting Questions

Appendix Table A8 summarizes the failure modes most specific to RSTA. These support the argument in Section 6 that recommendation towards agents is higher stakes than recommendation towards humans.

Table A8. A failure taxonomy for RSTA to complement the research agenda in Section 7.

Failure mode	Description
Silent over-autonomy	the system ranks productive actions when it should have ranked ask, verify, or escalate first
Oversafe paralysis	abstention and escalation are over-ranked until delegation loses practical value
Preference hallucination	the agent acts as though a principal preference were known when it was not elicited or grounded
Candidate-set myopia	the recommender system performs well over a poor candidate inventory and repeatedly misses better trajectories
Authority inversion	technically feasible actions outrank normatively authorized ones
Verification neglect	productive actions systematically outrank tests, evidence checks, or policy review
Service-side manipulation	service objectives dominate principal utility when ranking directly to agents
Coordination collapse	multi-agent recommender systems over-decompose or under-verify, causing delay or failure

Appendix I. Minimal Artifact: An RSTA Evaluation Template

A lightweight public object can help distinguish an interesting perspective from a reusable research program. Table A9 is a minimal RSTA task card that can be attached to task conversions, ablations, and evaluation reports.

 doi:10.20944/preprints202604.0201.v1

Table A9. A one-page RSTA evaluation card. It is intentionally lightweight: enough structure to make the ranking layer explicit without overconstraining future task design.

Field	What to report
Immediate recipient	single acting agent, orchestration layer, or multi-agent system
Principal(s) and service stakeholder(s)	who benefits, who authorizes, who constrains, and who may have competing utility
State s_t	task state, memory, permissions, budgets, latency bounds, and visible environment context
Candidate inventory Ω_t	explicit or reconstructable interventions, including productive and oversight actions
Recommendable types	objects, parameters, modes, delegates, protocols, verifiers, approvals, abstentions
Service-private signals	compatibility, inventory, reliability, permissions, policy state, pricing, fraud or market signals
Ranking output	top-L slate, not only the final chosen action
Trajectory outcomes	task success, preference fit, cost, latency, safety, compliance, side effects, human cleanup
Failure attribution	whether errors arise from candidate generation, ranking, orchestration, or execution
Interface robustness	whether the result survives benign rewrites of schemas, tool names, or protocol surfaces

The task card can be used as a checklist:

1. Who is the immediate recipient of the ranking?
2. Who are the principal and service stakeholders, if any?
3. What exactly are the recommendable objects, and how is Ω_t defined?
4. Are ask, abstain, verify, sandbox, or escalate available as recommendables?
5. Are candidate sets explicit or deterministically reconstructable?
6. Are both task-centric and stakeholder-centric outcomes reported?
7. If the system is multi-agent, what team state, protocol state, and handoff events are observed?
8. Does the evaluation isolate ranking quality from candidate-generation quality?

References

1. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* **2005**, *17*, 734–749.
2. Ricci, F.; Rokach, L.; Shapira, B. Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook* **2021**, pp. 1–35.
3. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37.
4. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* **2012**.
5. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* **2015**.
6. Quadrana, M.; Cremonesi, P.; Jannach, D. Sequence-aware recommender systems. *ACM computing surveys (CSUR)* **2018**, *51*, 1–36.
7. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE international conference on data mining (ICDM). IEEE, 2018, pp. 197–206.

 doi:10.20944/preprints202604.0201.v1

8. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1419–1428.
9. Li, L.; Chu, W.; Langford, J.; Schapire, R.E. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the Proceedings of the 19th international conference on World wide web, 2010, pp. 661–670.
10. Christakopoulou, K.; Radlinski, F.; Hofmann, K. Towards conversational recommender systems. In Proceedings of the Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 815–824.
11. Gao, C.; Lei, W.; He, X.; De Rijke, M.; Chua, T.S. Advances and challenges in conversational recommender systems: A survey. *AI open* **2021**, *2*, 100–126.
12. Abdollahpouri, H.; Adomavicius, G.; Burke, R.; Guy, I.; Jannach, D.; Kamishima, T.; Krasnodebski, J.; Pizzato, L. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction* **2020**, *30*, 127–158.
13. Afsar, M.M.; Crump, T.; Far, B. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys* **2022**, *55*, 1–38.
14. Huang, C.; Huang, H.; Yu, T.; Xie, K.; Wu, J.; Zhang, S.; Mcauley, J.; Jannach, D.; Yao, L. A survey of foundation model-powered recommender systems: From feature-based, generative to agentic paradigms. *arXiv preprint arXiv:2504.16420* **2025**.
15. Peng, Q.; Liu, H.; Huang, H.; Yang, Q.; Shao, M. A survey on llm-powered agents for recommender systems. *arXiv preprint arXiv:2502.10050* **2025**.
16. Zhang, Y.; Qiao, S.; Zhang, J.; Lin, T.H.; Gao, C.; Li, Y. A survey of large language model empowered agents for recommendation and search: Towards next-generation information retrieval. *arXiv preprint arXiv:2503.05659* **2025**.
17. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems* **2023**, *36*, 68539–68551.
18. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.R.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the The eleventh international conference on learning representations, 2022.
19. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* **2023**.
20. Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; Li, Y. Api-bank: A comprehensive benchmark for tool-augmented llms. In Proceedings of the Proceedings of the 2023 conference on empirical methods in natural language processing, 2023, pp. 3102–3116.
21. Yao, S.; Chen, H.; Yang, J.; Narasimhan, K. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems* **2022**, *35*, 20744–20757.
22. Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; Su, Y. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems* **2023**, *36*, 28091–28114.
23. Trivedi, H.; Khot, T.; Hartmann, M.; Manku, R.; Dong, V.; Li, E.; Gupta, S.; Sabharwal, A.; Balasubramanian, N. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2024, pp. 16022–16076.
24. Drouin, A.; Gasse, M.; Caccia, M.; Laradji, I.H.; Del Verme, M.; Marty, T.; Boisvert, L.; Thakkar, M.; Cappart, Q.; Vazquez, D.; et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718* **2024**.
25. Boisvert, L.; Thakkar, M.; Gasse, M.; Caccia, M.; De Chezelles, T.L.; Cappart, Q.; Chapados, N.; Lacoste, A.; Drouin, A. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems* **2024**, *37*, 5996–6051.
26. Zhu, K.; Du, H.; Hong, Z.; Yang, X.; Guo, S.; Wang, D.Z.; Wang, Z.; Qian, C.; Tang, R.; Ji, H.; et al. Multiagentbench: Evaluating the collaboration and competition of llm agents. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 8580–8622.

 doi:10.20944/preprints202604.0201.v1

27. OfficeChai Team. Will Need To Re-engineer Tools So That They Can Keep Up With The Speeds Of AI Agents: Google's Jeff Dean. https://officechai.com/ai/will-need-to-re-engineer-tools-so-that-they-can-keep-up-with-the-speeds-of-ai-agents-googles-jeff-dean/, 2026. Published March 29, 2026; accessed 2026-03-31.
28. Tan, A. Advancing to the next frontier of AI. https://www.computerweekly.com/news/366640859/Advancing-to-the-next-frontier-of-AI, 2026. Published March 29, 2026; accessed 2026-03-31.
29. Anthropic. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol, 2024. Published November 25, 2024; accessed 2026-03-30.
30. Google Developers Blog. Announcing the Agent2Agent Protocol (A2A). https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/, 2025. Published April 9, 2025; accessed 2026-03-30.
31. Ehtesham, A.; Singh, A.; Gupta, G.K.; Kumar, S. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279* **2025**.
32. Gartner. Gartner IT Symposium/Xpo 2024 Barcelona: Day 1 Highlights. https://www.gartner.com/en/newsroom/press-releases/2024-11-04-gartner-it-symposium-xpo-2024-barcelona-day-1-highlights, 2024. Published November 4, 2024; accessed 2026-03-30.
33. Gartner. Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025. https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025, 2025. Published August 26, 2025; updated September 5, 2025; accessed 2026-03-30.
34. Deloitte. Deloitte Global's 2025 Predictions Report: Generative AI: Paving the Way for a Transformative Future in Technology, Media, and Telecommunications. https://www.deloitte.com/global/en/about/press-room/deloitte-globals-2025-predictions-report.html, 2024. Published November 19, 2024; accessed 2026-03-30.
35. Microsoft WorkLab. 2025: The Year the Frontier Firm Is Born. https://www.microsoft.com/en-us/worklab/work-trend-index/2025-the-year-the-frontier-firm-is-born, 2025. Published April 23, 2025; accessed 2026-03-30.
36. Sukharevsky, A.; Krivkovich, A.; Gast, A.; Storozhev, A.; Maor, D.; Mahadevan, D.; Hämäläinen, L.; Durth, S. The Agentic Organization: Contours of the Next Paradigm for the AI Era. https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/the-agentic-organization-contours-of-the-next-paradigm-for-the-ai-era, 2025. Accessed 2026-03-30.
37. Javdani, S.; Srinivasa, S.S.; Bagnell, J.A. Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings* **2015**, *2015*, 10–15607.
38. Scerri, P.; Pynadath, D.V.; Tambe, M. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research* **2002**, *17*, 171–228.
39. Tomašev, N.; Franklin, M.; Osindero, S. Intelligent AI delegation. *arXiv preprint arXiv:2602.11865* **2026**.
40. Levine, S.; Kumar, A.; Tucker, G.; Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* **2020**.
41. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In Proceedings of the First conference on language modeling, 2024.
42. Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; Ghanem, B. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems* **2023**, *36*, 51991–52008.
43. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In Proceedings of the The twelfth international conference on learning representations, 2023.
44. Du, Y.; Li, S.; Torralba, A.; Tenenbaum, J.B.; Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. In Proceedings of the Forty-first international conference on machine learning, 2024.
45. Du, H.; Su, J.; Li, J.; Ding, L.; Yang, Y.; Han, P.; Tang, X.; Zhu, K.; You, J. Which LLM Multi-Agent Protocol to Choose? *arXiv preprint arXiv:2510.17149* **2025**.
46. Yao, S.; Shinn, N.; Razavi, P.; Narasimhan, K. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045* **2024**.

 doi:10.20944/preprints202604.0201.v1

47. Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L.; Fischer, M.; Tramèr, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems* **2024**, *37*, 82895–82920.
48. Gu, W.; Li, C.; Yu, Z.; Sun, M.; Yang, Z.; Wang, W.; Jia, H.; Zhang, S.; Ye, W. What Do Agents Learn from Trajectory-SFT: Semantics or Interfaces? *arXiv preprint arXiv:2602.01611* **2026**.
49. Wang, Z.; Yu, G.; Lyu, M.R. AI-NativeBench: An Open-Source White-Box Agentic Benchmark Suite for AI-Native Systems. *arXiv preprint arXiv:2601.09393* **2026**.
50. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. Harness Engineering for Language Agents: The Harness Layer as Control, Agency, and Runtime. *Preprints.org* **2026**.
51. Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C.D.; Ermon, S.; Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems* **2023**, *36*, 53728–53741.
52. Wu, J.; Xie, Y.; Yang, Z.; Wu, J.; Gao, J.; Ding, B.; Wang, X.; He, X. β-DPO: Direct Preference Optimization with Dynamic β. *Advances in Neural Information Processing Systems* **2024**, *37*, 129944–129966.
53. Meng, Y.; Xia, M.; Chen, D. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems* **2024**, *37*, 124198–124235.
54. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. SPˆ2DPO: An LLM-assisted Semantic Per-Pair DPO Generalization. *arXiv preprint arXiv:2601.22385* **2026**.
55. Liu, S.; Fang, W.; Hu, Z.; Zhang, J.; Zhou, Y.; Zhang, K.; Tu, R.; Lin, T.E.; Huang, F.; Song, M.; et al. A survey of direct preference optimization. *arXiv preprint arXiv:2503.11701* **2025**.
56. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* **2017**.
57. Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* **2024**.
58. Liu, Z.; Chen, C.; Li, W.; Qi, P.; Pang, T.; Du, C.; Lee, W.S.; Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783* **2025**.
59. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* **2019**, *52*, 1–38.
60. Burke, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* **2002**, *12*, 331–370.
61. He, C.; Liu, Y.; Guo, Q.; Miao, C. Multi-scale quasi-RNN for next item recommendation. *arXiv preprint arXiv:1902.09849* **2019**.
62. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1441–1450.
63. He, C. Neural inductive biases for sequential recommendation. PhD thesis, Nanyang Technological University, 2025.
64. Jannach, D.; Manzoor, A.; Cai, W.; Chen, L. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)* **2021**, *54*, 1–36.
65. Swaminathan, A.; Joachims, T. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems* **2015**, *28*.
66. Schnabel, T.; Swaminathan, A.; Singh, A.; Chandak, N.; Joachims, T. Recommendations as treatments: Debiasing learning and evaluation. In Proceedings of the international conference on machine learning. PMLR, 2016, pp. 1670–1679.
67. Joachims, T.; Swaminathan, A.; Schnabel, T. Unbiased learning-to-rank with biased feedback. In Proceedings of the Proceedings of the tenth ACM international conference on web search and data mining, 2017, pp. 781–789.
68. Ie, E.; Jain, V.; Wang, J.; Narvekar, S.; Agarwal, R.; Wu, R.; Cheng, H.T.; Chandra, T.; Boutilier, C. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In Proceedings of the IJCAI, 2019, Vol. 19, pp. 2592–2599.
69. Jiang, N.; Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In Proceedings of the International conference on machine learning. PMLR, 2016, pp. 652–661.
70. Wang, W.; Zhang, Y.; Li, H.; Wu, P.; Feng, F.; He, X. Causal recommendation: Progresses and future directions. In Proceedings of the Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval, 2023, pp. 3432–3435.

 doi:10.20944/preprints202604.0201.v1

71. Maes, P. Agents that reduce work and information overload. *Communications of the ACM* **1994**, *37*, 30–40.
72. Lashkari, Y.; Metral, M.; Maes, P. Collaborative interface agents. *Readings in agents* **1997**, pp. 111–116.
73. Schiaffino, S.; Amandi, A. User–interface agent interaction: personalization issues. *International Journal of Human-Computer Studies* **2004**, *60*, 129–148.
74. Armentano, M.; Godoy, D.; Amandi, A. Personal assistants: Direct manipulation vs. mixed initiative interfaces. *International Journal of Human-Computer Studies* **2006**, *64*, 27–35.
75. Horvitz, E. Principles of mixed-initiative user interfaces. In Proceedings of the Proceedings of the SIGCHI conference on Human Factors in Computing Systems, 1999, pp. 159–166.
76. Allen, J.E.; Guinn, C.I.; Horvtz, E. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications* **1999**, *14*, 14–23.
77. Guttman, R.H.; Moukas, A.G.; Maes, P. Agent-mediated electronic commerce: A survey. *The Knowledge Engineering Review* **1998**, *13*, 147–159.
78. Maes, P.; Guttman, R.H.; Moukas, A.G. Agents that buy and sell. *Communications of the ACM* **1999**, *42*, 81–ff.
79. Kephart, J.O.; Hanson, J.E.; Greenwald, A.R. Dynamic pricing by software agents. *Computer Networks* **2000**, *32*, 731–752.
80. Kephart, J.O.; Greenwald, A.R. Shopbot economics. *Autonomous Agents and Multi-Agent Systems* **2002**, *5*, 255–287.
81. Laffont, J.J.; Martimort, D. *The theory of incentives: the principal-agent model*; Princeton university press, 2002.
82. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems* **2023**, *36*, 8634–8652.
83. Shim, J.; Seo, G.; Lim, C.; Jo, Y. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. *arXiv preprint arXiv:2503.00564* **2025**.
84. Gou, B.; Huang, Z.; Ning, Y.; Gu, Y.; Lin, M.; Qi, W.; Kopanev, A.; Yu, B.; Gutiérrez, B.J.; Shu, Y.; et al. Mind2web 2: Evaluating agentic search with agent-as-a-judge. *arXiv preprint arXiv:2506.21506* **2025**.
85. Patil, S.G.; Zhang, T.; Wang, X.; Gonzalez, J.E. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems* **2024**, *37*, 126544–126565.
86. Farn, N.; Shin, R. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775* **2023**.
87. Wei, J.; Sun, Z.; Papay, S.; McKinney, S.; Han, J.; Fulford, I.; Chung, H.W.; Passos, A.T.; Fedus, W.; Glaese, A. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516* **2025**.
88. Mialon, G.; Fourrier, C.; Wolf, T.; LeCun, Y.; Scialom, T. Gaia: a benchmark for general ai assistants. In Proceedings of the The Twelfth International Conference on Learning Representations, 2023.
89. Patil, S.G.; Mao, H.; Yan, F.; Ji, C.C.J.; Suresh, V.; Stoica, I.; Gonzalez, J.E. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
90. Lu, J.; Holleis, T.; Zhang, Y.; Aumayer, B.; Nan, F.; Bai, H.; Ma, S.; Ma, S.; Li, M.; Yin, G.; et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025, 2025, pp. 1160–1183.
91. Xu, Y.; Chen, Q.; Ma, Z.; Liu, D.; Wang, W.; Wang, X.; Xiong, L.; Wang, W. Toward Personalized LLM-Powered Agents: Foundations, Evaluation, and Future Directions. *arXiv preprint arXiv:2602.22680* **2026**.
92. Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* **2023**.
93. Xu, F.F.; Song, Y.; Li, B.; Tang, Y.; Jain, K.; Bao, M.; Wang, Z.Z.; Zhou, X.; Guo, Z.; Cao, M.; et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161* **2024**.
94. Yehudai, A.; Eden, L.; Li, A.; Uziel, G.; Zhao, Y.; Bar-Haim, R.; Cohan, A.; Shmueli-Scheuer, M. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416* **2025**.
95. Mohammadi, M.; Li, Y.; Lo, J.; Yip, W. Evaluation and benchmarking of llm agents: A survey. In Proceedings of the Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2, 2025, pp. 6129–6139.
96. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. OpenClaw as Language Infrastructure: A Case-Centered Survey of a Public Agent Ecosystem in the Wild. *Preprints.org* **2026**.
97. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. The AutoResearch Moment: From Experimenter to Research Director. *Preprints.org* **2026**.

 doi:10.20944/preprints202604.0201.v1

98. Singla, A.; Sukharevsky, A.; Hall, B.; Yee, L.; Chui, M. The State of AI in 2025: Agents, Innovation, and Transformation. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai, 2025. Accessed 2026-03-30.
99. Stein, M. How are AI agents used? Evidence from 177,000 MCP tools. *arXiv preprint arXiv:2603.23802* **2026**.
100. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. The Synthetic Media Exchange: When Lineage Becomes Currency **2026**.
101. He, C.; Zhou, X.; Wang, D.; Xu, H.; Liu, W.; Miao, C. From Prompts to Portfolios: AI Agents as Agentic Multimedia Firms **2026**.
102. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* **2020**, *33*, 9459–9474.
103. Chen, L.; Zaharia, M.; Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176* **2023**.
104. Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.L.; Wu, T.; Gonzalez, J.E.; Kadous, M.W.; Stoica, I. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665* **2024**.
105. Varangot-Reille, C.; Bouvard, C.; Gourru, A.; Ciancone, M.; Schaeffer, M.; Jacquenet, F. Doing More with Less: A Survey on Routing Strategies for Resource Optimisation in Large Language Model-Based Systems. *arXiv preprint arXiv:2502.00409* **2025**.
106. Zheng, H.; Xu, H.; Lin, Y.; Fan, S.; Chen, L.; Yu, K. DiSRouter: Distributed Self-Routing for LLM Selections. *arXiv preprint arXiv:2510.19208* **2025**.
107. Mozannar, H.; Sontag, D. Consistent estimators for learning to defer to an expert. In Proceedings of the International conference on machine learning. PMLR, 2020, pp. 7076–7087.
108. Leitão, D.; Saleiro, P.; Figueiredo, M.A.; Bizarro, P. Human-AI collaboration in decision-making: beyond learning to defer. *arXiv preprint arXiv:2206.13202* **2022**.
109. Spitzer, P.; Holstein, J.; Hemmer, P.; Vössing, M.; Kühl, N.; Martin, D.; Satzger, G. Human delegation behavior in human-AI collaboration: The effect of contextual information. *Proceedings of the ACM on Human-Computer Interaction* **2025**, *9*, 1–28.
110. Gu, W.; Li, M.L.; Zhu, S. Who Should Do What? Adaptive Delegation in Human-AI Collaboration. In Proceedings of the NeurIPS 2025 Workshop MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making, 2025.
111. Biyik, E.; Yao, F.; Chow, Y.; Haig, A.; Hsu, C.w.; Ghavamzadeh, M.; Boutilier, C. Preference elicitation with soft attributes in interactive recommendation. *arXiv preprint arXiv:2311.02085* **2023**.
112. Muldrew, W.; Hayes, P.; Zhang, M.; Barber, D. Active preference learning for large language models. *arXiv preprint arXiv:2402.08114* **2024**.
113. Amershi, S.; Weld, D.; Vorvoreanu, M.; Fourney, A.; Nushi, B.; Collisson, P.; Suh, J.; Iqbal, S.; Bennett, P.N.; Inkpen, K.; et al. Guidelines for human-AI interaction. In Proceedings of the Proceedings of the 2019 chi conference on human factors in computing systems, 2019, pp. 1–13.
114. Kraus, M.; Wagner, N.; Callejas, Z.; Minker, W. The role of trust in proactive conversational assistants. *IEEE Access* **2021**, *9*, 112821–112836.
115. Lyu, Y.; Zhang, X.; Yan, L.; de Rijke, M.; Ren, Z.; Chen, X. Deepshop: A benchmark for deep research shopping agents. *arXiv preprint arXiv:2506.02839* **2025**.
116. Peeters, R.; Steiner, A.; Schwarz, L.; Yuya Caspary, J.; Bizer, C. WebMall–A Multi-Shop Benchmark for Evaluating Web Agents. *arXiv e-prints* **2025**, pp. arXiv–2508.
117. Kim, S.; Heo, R.; Seo, Y.; Yeo, J.; Lee, D. AgenticShop: Benchmarking Agentic Product Curation for Personalized Web Shopping. *arXiv preprint arXiv:2602.12315* **2026**.
118. Zhou, S.; Xu, F.F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* **2023**.
119. Yoran, O.; Amouyal, S.J.; Malaviya, C.; Bogin, B.; Press, O.; Berant, J. Assistantbench: Can web agents solve realistic and time-consuming tasks? In Proceedings of the Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024, pp. 8938–8968.
120. Jimenez, C.E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770* **2023**.
121. Yang, J.; Jimenez, C.E.; Wettig, A.; Lieret, K.; Yao, S.; Narasimhan, K.; Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems* **2024**, *37*, 50528–50652.

 doi:10.20944/preprints202604.0201.v1

122. Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T.J.; Cheng, Z.; Shin, D.; Lei, F.; et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems* **2024**, *37*, 52040–52094.
123. Cemri, M.; Pan, M.Z.; Yang, S.; Agrawal, L.A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657* **2025**.
124. Zhou, H.; Wan, X.; Sun, R.; Palangi, H.; Iqbal, S.; Vulić, I.; Korhonen, A.; Arık, S.Ö. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533* **2025**.
125. Shen, L.; Shen, X. Auto-SLURP: A Benchmark Dataset for Evaluating Multi-Agent Frameworks in Smart Personal Assistant. *arXiv preprint arXiv:2504.18373* **2025**.
126. Zhang, H.; Feng, T.; You, J. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In Proceedings of the The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.
127. Allouah, A.; Besbes, O.; Figueroa, J.D.; Kanoria, Y.; Kumar, A. What Is Your AI Agent Buying? Evaluation, Biases, Model Dependence, & Emerging Implications for Agentic E-Commerce. *arXiv preprint arXiv:2508.02630* **2025**.
128. Xiu, Z.; Sun, D.Q.; Cheng, K.; Patel, M.; Zhang, Y.; Lu, J.; Attia, O.; Vemulapalli, R.; Tuzel, O.; Cao, M.; et al. ASTRA-bench: Evaluating Tool-Use Agent Reasoning and Action Planning with Personal User Context. *arXiv preprint arXiv:2603.01357* **2026**.
129. He, C.; Zhou, X.; Xu, H.; Liu, W.; Miao, C.; Wang, D. Human-AI productivity claims should be reported as time-to-acceptance under explicit acceptance tests, 2026.