

Eval Research

Relevant Benchmark In Databank

Benchmark	Why it is relevant
τ-bench / τ^2-bench	Directly evaluates <u>multi-turn agent–user interaction</u> . A <u>simulated user</u> reveals information over the conversation. τ -bench was explicitly motivated by the fact that existing benchmarks often fail to test interaction with users . (arXiv1 ; arXiv2)
REAL	Evaluates agents on <u>practical multi-step web tasks</u> using deterministic replicas of real websites. It is useful for examining whether a benchmark can have a highly realistic environment and workflow while still giving the agent relatively structured task instructions. (arXiv)
SWE-bench / SWE-bench-Live	Uses <u>real GitHub issues and real software repositories</u> , making it a strong example of a benchmark grounded in real-world tasks. It is useful for asking whether a task can come from the real world but still be relatively well-specified , reducing the need for user clarification or intent elicitation. (arXiv1 ; arXiv2)
The Agent Company	Simulates a <u>real workplace environment</u> , where agents <u>browse websites, write code, run programs, and communicate with coworkers to complete professional tasks</u> . It broadens the analysis beyond one domain and lets us examine whether realistic workplace benchmarks capture the ambiguity and interaction patterns of actual user requests. (arXiv)

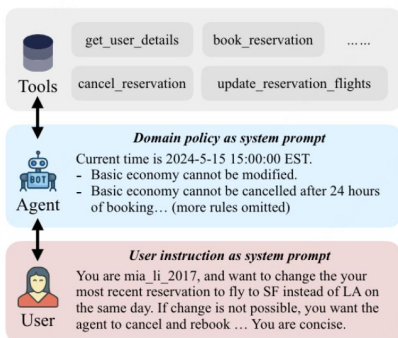
τ -bench / τ^2 -bench

τ -bench and τ^2 -bench are already explicitly trying to fix the exact weakness: traditional benchmarks give the model all the information upfront and **do not require realistic interaction with users**.

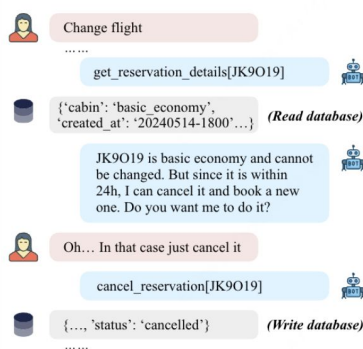
- τ -bench introduces multi-turn **Tool-Agent-User interaction**
- while τ^2 -bench goes further and **lets the user take actions in the environment**, not just provide information.

τ -bench

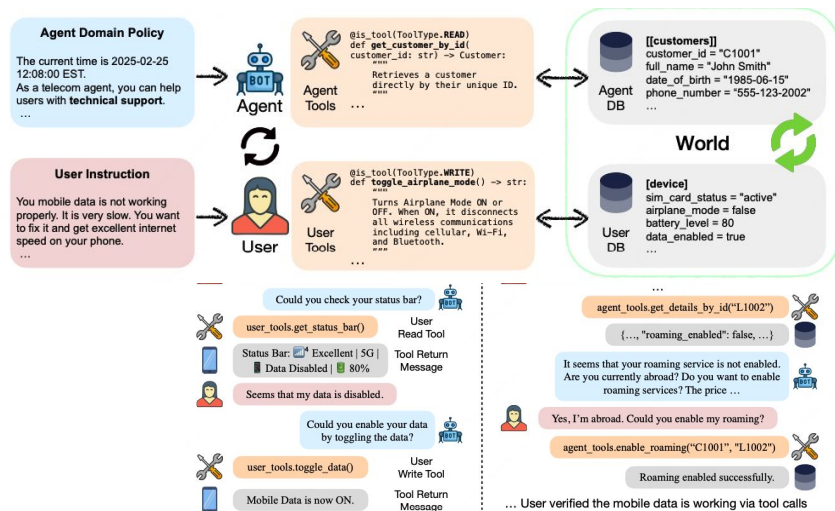
(a) τ -bench setup



(b) Example trajectory in τ -airline



τ^2 -bench



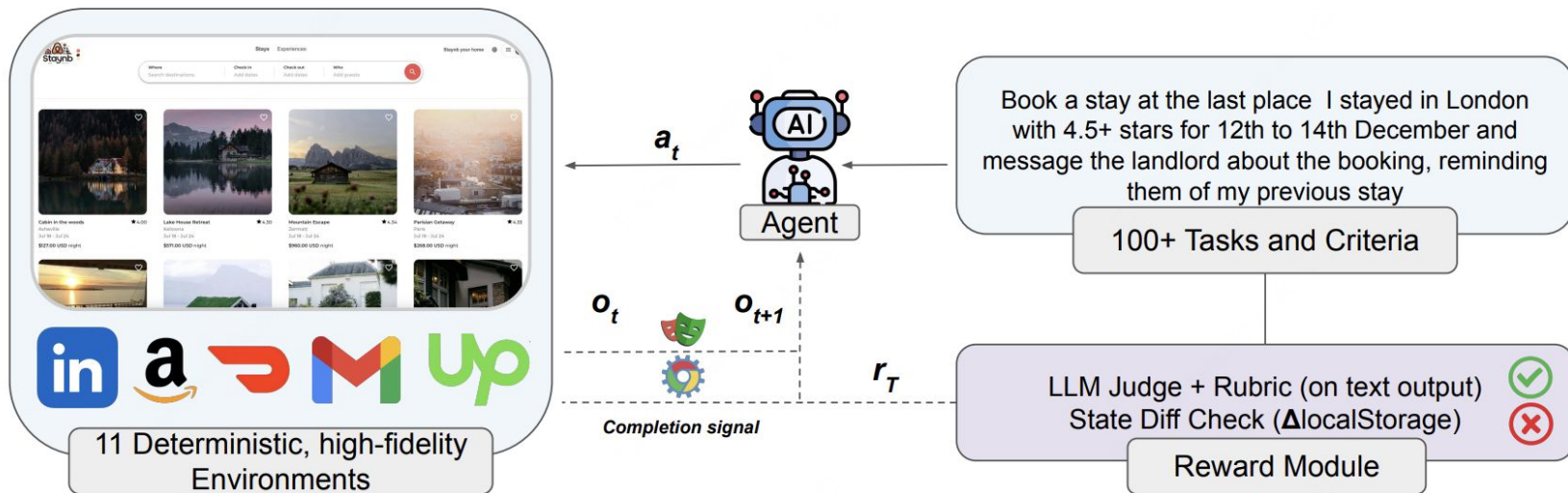
Dimension	τ -bench	τ^2 -bench
Overall structure	Simulated User $\leftrightarrow$ Agent $\leftrightarrow$ Tools/Database (just agent has tools) . The agent talks with the user, gathers missing information, follows domain policy, and calls tools.	Simulated User $\leftrightarrow$ Agent , and both sides have tools that act on a shared environment. Main idea: dual-control interaction.
Domains	Retail and airline.	Retail, airline, plus a new telecom troubleshooting domain.
Task source / construction	Synthetic benchmark environment; tasks are manually constructed and iteratively refined so that each task leads to a unique valid outcome .	Telecom tasks are programmatically composed from atomic subtasks, with manual refinement; designed for verifiability and controllable complexity .
Task specification	The agent sees only the conversation, but the simulated user is given a detailed hidden instruction containing identity, intent, preferences, and fallback behavior .	Same general setup, but task instructions are more structured, including reason for call, known info, unknown info, and task instructions .
Single-turn or multi-turn?	Multi-turn . The agent must progressively gather information from the user .	Multi-turn , with stronger emphasis on coordination and step-by-step troubleshooting.
User type	LLM-simulated user.	LLM-simulated user, more tightly coupled to the environment and tool state.
Can the user act on the environment?	Mostly no . The user provides information and preferences; the agent controls tools.	Yes . Both agent and user have tools and can modify different parts of the shared world. This is the key “dual-control” contribution.
Clarification / information gathering	Yes . A core motivation is that the agent must gather information rather than receive everything upfront.	Yes , plus the agent must guide the user through actions and diagnosis .

Dimension	τ -bench	τ^2 -bench
Simulator behavior	User reveals information through natural-language interaction, but follows a predefined latent goal.	More constrained : the simulator must follow scenario instructions, disclose information progressively, and only call tools when requested.
Personas	No explicit systematic persona framework.	Adds None / Easy / Hard personas , mainly varying technical familiarity and behavior.
Success metric	Final database state + required output information ; also uses pass^k to measure consistency across repeated runs.	Supports DB checks, state assertions, communication checks, natural-language assertions, and action matching; telecom mainly uses state assertions.
Main finding	Agents struggle with consistency, rule following, compound requests, and reasoning over partial information.	Performance drops substantially when moving from no-user mode to dual-control, showing that communication and coordination are separate bottlenecks from pure reasoning .
What is still missing vs. real users?	Real users may have ambiguous, evolving, inconsistent, or not-yet-formed goals, while τ -bench deliberately removes ambiguity to preserve a unique ground truth.	User behavior is still highly controlled for reliability ; users are instructed not to invent information and to follow a predefined scenario . It also does not fully model the expert–novice mental-model gap .
Main evaluation trade-off	More interaction realism, but ambiguity is intentionally reduced to keep evaluation objective and verifiable.	More interaction and coordination realism, but stronger constraints on the simulator improve reliability at the possible cost of natural behavioral realism .
Relevance to our research	Shows that modern benchmarks already move beyond fully specified single-turn tasks, but still rely on a fully specified latent user goal .	Strong evidence that interaction changes measured performance, while also raising the question of whether a highly constrained simulator behaves like a real user .

REAL

REAL is explicitly trying to fix: existing web-agent benchmarks often trade off realism for reproducibility, while **live websites are realistic but unstable and hard to evaluate** consistently.

- REAL introduces **high-fidelity, deterministic replicas of real websites**
- and evaluates agents on **practical multi-step web tasks** while keeping the environment reproducible and safely measurable.



REAL

Dimension	REAL
Overall structure	Natural-language task → Agent → deterministic high-fidelity website replica → evaluation. The agent repeatedly observes the browser and takes actions until completion.
Domains	11 web environments covering e-commerce, travel, communication, scheduling, professional networking, food delivery, property search, freelance marketplaces , etc.
Task source / construction	112 practical tasks designed to mirror common real-world web workflows. The environments are synthetic replicas of real websites , with fixed data, timestamps, and UX states for reproducibility; some synthetic data is AI-generated.
Task specification	The agent receives a natural-language user goal upfront , potentially with multiple constraints and an environment configuration.
Single-turn or multi-turn?	Multi-step environment interaction , but not multi-turn user conversation .
User type	No live or simulated user during execution; the user is represented by a fixed natural-language instruction .
Can the user act on the environment?	No . After the task is given, only the agent interacts with the website.

REAL

Dimension	REAL
Clarification / information gathering	The agent may need to retrieve information from the website, but cannot ask the user for clarification or progressively elicit preferences .
Simulator behavior	N/A . No user simulator.
Personas	None . No explicit user persona framework.
Success metric	Action tasks : programmatic check of final localStorage state . Retrieval tasks : rubric-based LLM judge . Combined tasks : both must pass.
Main finding	Frontier agents still struggle substantially; best reported end-to-end success is about 41% . Common failures include poor state verification and inability to recover from navigation dead ends.
What is still missing vs. real users?	No evolving conversation, clarification, hidden preferences, changing goals, user mistakes, or user feedback during execution.
Main evaluation trade-off	High environment/workflow realism + high reproducibility , but relatively low user-interaction realism .
Relevance to our research	Strong example that realistic environment ≠ realistic user interaction: REAL makes the website and workflow realistic, but largely assumes the user goal has already been specified before execution.

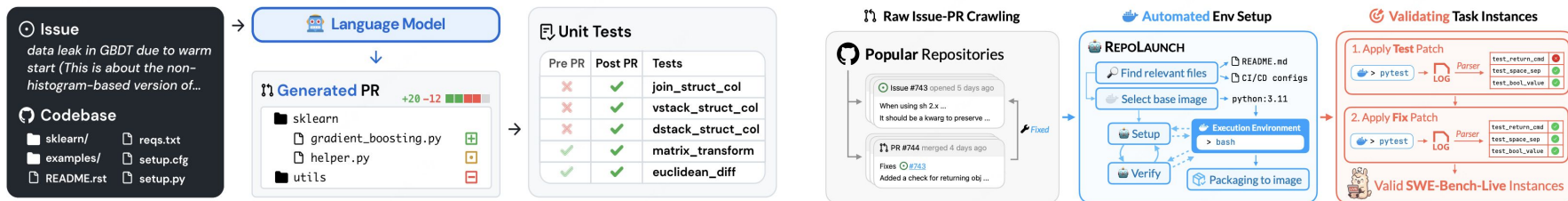
SWE-bench / SWE-bench-Live

SWE-bench is trying to fix: traditional coding benchmarks use small, self-contained programming problems that do not reflect the complexity of real software engineering.

- SWE-bench introduces **real GitHub issues** paired with real repository snapshots and merged pull-request solutions
- and evaluates models by asking them to **generate repository-level patches** that must pass real tests, making the benchmark both realistic and execution-verifiable.

SWE-bench-Live keeps this real-world issue-resolution setup, but fixes another weakness: static benchmarks become stale, narrow, and increasingly vulnerable to contamination or overfitting.

- SWE-bench-Live introduces **continuously updatable, recent GitHub issues** across many more repositories
- and automates task/environment construction so evaluation stays **fresh, scalable, executable, and more contamination-resistant**. Its initial release contains 1,319 tasks from 93 repositories, based on issues created since 2024



SWE-bench / SWE-bench-Live

Dimension	SWE-bench	SWE-bench-Live
Overall structure	Real GitHub issue + repo snapshot → model → patch → tests	Same core structure: recent GitHub issue + executable repo environment → agent/model → patch → tests
Domains	12 popular open-source Python repositories.	Much broader repository coverage: initial release spans 93 repositories. (
Task source / construction	2,294 tasks mined from real issues and merged PRs; heavily filtered and execution-validated.	Real, recent GitHub issues; automated curation pipeline handles instance creation and environment setup to support continuous updates.
Task specification	Fixed issue description + codebase/context. Issue text is naturally occurring and may be noisy or incomplete.	Same basic idea: the issue is still treated as a fixed task specification, but issues are newer and more diverse.
Single-turn or multi-turn?	Essentially single-shot issue-to-patch in the original benchmark.	Still primarily an issue-resolution benchmark, not a user conversation benchmark.
User type	No live/simulated user; user intent is represented by the real GitHub issue.	Same. No interactive issue author/user.
Can the user act on the environment?	No.	No.

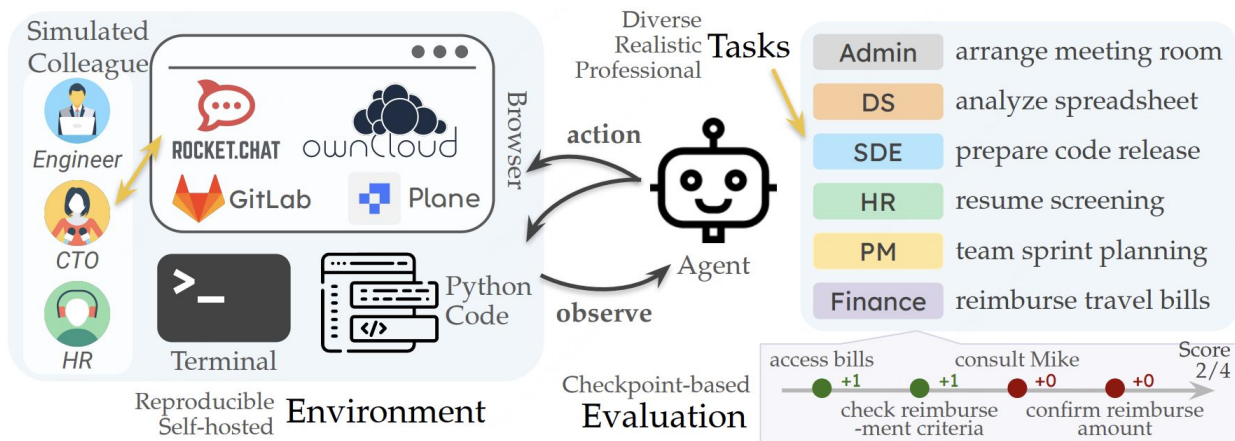
SWE-bench / SWE-bench-Live

Dimension	SWE-bench	SWE-bench-Live
Clarification / information gathering	Model can inspect code/context, but cannot ask the issue author follow-up questions.	Same core limitation. Freshness does not add user clarification.
Personas	None	None
Success metric	Patch must apply; all FAIL_TO_PASS tests pass and PASS_TO_PASS tests remain passing.	Same execution-based philosophy, with reproducible task-specific environments/Docker images.
Main finding	Repository-scale issue resolution is much harder than traditional code generation; models struggle with localization, long context, and multi-file reasoning.	Agents perform substantially worse on the fresh Live benchmark than on static SWE-bench-style benchmarks, suggesting benchmark age/contamination matters.
What is still missing vs. real users?	No clarification dialogue, evolving requirements, feedback, or iterative communication with the issue author.	Same limitation. Updating the data improves freshness, not user-interaction realism.
Main evaluation trade-off	High task-source realism + objective verification, but low interaction realism.	Adds freshness, diversity, scalability, and lower contamination risk , while retaining the same low user-interaction realism.
Relevance to our research	Shows that real-world task source $\neq$ realistic user interaction .	Even a fresher, more contamination-resistant benchmark still freezes a real user/developer request into a static task.

The Agent Company

TheAgentCompany is trying to fix: existing agent benchmarks often **evaluate narrow task types or isolated interfaces**, while real workplace tasks span **multiple roles, tools, long-horizon workflows, and communication with coworkers**.

- TheAgentCompany introduces a **self-contained simulated software company** with realistic workplace tools, data, and **LLM-simulated colleagues**
- and evaluates agents on **175 professional tasks across SWE, PM, data science, HR, admin, and finance**, using checkpoint-based evaluation that gives partial credit for long-horizon progress.



Original user = gives you the task.
Coworker = helps you complete the task.

The Agent Company

Dimension	TheAgentCompany
Overall structure	Task intent → Agent → workplace environment + tools + simulated colleagues → checkpoint-based evaluation. The agent browses internal websites, writes/runs code, uses terminal/Python, and may communicate with coworkers.
Domains	Professional work in a simulated software company: Software Engineering, Project Management, Data Science, Administrative, Human Resources, Finance, and Other.
Task source / construction	175 hand-curated tasks , not scraped directly from real user logs. Task categories are grounded partly in the U.S. Department of Labor O*NET database, plus domain-expert experience and LLM brainstorming. All tasks were manually designed, implemented, tested, and reviewed by the authors/domain experts.
Task specification	Each task starts with an English task intent designed to be clear enough that a human worker should not need to ask the original user for more instructions, although they may need to ask coworkers for missing information.
Single-turn or multi-turn?	Multi-step and often long-horizon. Some tasks also include genuine multi-turn interaction with simulated colleagues.
User type	No persistent “user” simulator like τ -bench. Instead, the agent gets a fixed task instruction and may interact with LLM-simulated coworkers/NPCs during execution.
Can the user act on the environment?	The original task-giver does not act. Simulated colleagues can interact conversationally , provide information, negotiate, and influence task progress, but they do not function as a τ^2 -style co-controller of a shared environment.

The Agent Company

Dimension	TheAgentCompany
Clarification / information gathering	Yes, but mainly from coworkers rather than from the original user. Some tasks require contacting colleagues to obtain information not present in the initial task description.
Simulator behavior	Simulated coworkers are LLM-backed characters with predefined profiles containing name, role, responsibilities, and project affiliations . They communicate through RocketChat and are fixed to Claude-3.5-Sonnet in the experiments.
Personas	No Easy/Hard persona framework. Instead, colleagues have role-based profiles such as CTO, PM, engineer, finance director, HR manager, etc.
Success metric	Tasks are decomposed into checkpoints with weighted points . The benchmark reports both full completion and partial completion . Most evaluators are deterministic; some complex outputs use rubric-based LLM evaluation.
Main finding	Even the best tested agent/model completed only about 30% of tasks fully . Agents struggled especially with social interaction, complex web UIs, long-horizon coordination, and administrative/finance workflows .
What is still missing vs. real users?	The initial task intent is intentionally well-scoped and clear, so ambiguity in the original user request is limited . Tasks are also manually curated rather than sampled directly from real enterprise interactions. The paper itself notes that more vague intents would better simulate real-world situations where the goal is not immediately clear.
Main evaluation trade-off	High workplace/workflow realism + multi-tool interaction + coworker communication , but task intents are deliberately made clear and benchmark tasks are curated for evaluability, limiting real-world ambiguity.

Cross-benchmark comparison

Dimension	τ -bench	τ^2 -bench	REAL	SWE-bench	TheAgentCompany
Primary realism target	User interaction	User interaction + shared control	Web environment / workflow	Real-world task source	Workplace / cross-tool / coworker interaction
Task source	Synthetic, manually designed customer-service scenarios	Synthetic / programmatically composed scenarios, especially telecom	Researcher-designed practical web tasks	Real GitHub issues + merged PRs	Hand-curated professional tasks grounded partly in O*NET + domain-expert experience
Environment realism	Medium	High	High	High for real codebases, but not a full workplace environment	High
User-request realism 	Medium: agent sees incomplete info, but latent goal is <u>predefined</u>	Medium: the user goal is still <u>predefined</u> , but interaction is richer because the simulated user can take actions and participate in troubleshooting.	Low-Medium: fixed natural-language goal given upfront	Medium: real issue text, but frozen as a static task	Medium: task intent is realistic, but deliberately made clear/well-scoped
Multi-turn user interaction	Yes	Yes	No	No	Not with original user; yes with coworkers
Can agent ask clarification?	Yes, to simulated user	Yes, to simulated user	No user available	No issue author available	Yes, mainly to simulated coworkers

Cross-benchmark comparison

Dimension	τ -bench	τ^2 -bench	REAL	SWE-bench	TheAgentCompany
User / human agency	User mainly provides information	User can also act via tools	None after task is issued	None after issue is issued	Coworkers can respond, negotiate, provide information
Hidden information / preferences	Yes	Yes	Mostly no user-side hidden state	Possible in natural issue text, but not interactively recoverable	Yes, sometimes held by coworkers
Ambiguity allowed? 	Limited; tasks refined toward a unique valid outcome	Limited / controlled	Limited; task goal is given upfront	Naturally occurring issue text may be noisy or incomplete	Limited in initial task intent; paper explicitly says tasks are made clear enough for human workers
Can goals evolve during interaction? 	Essentially no	Essentially no	No	No	Very limited; coworker feedback can affect execution, but task intent stays fixed
Personas / user variability	No systematic persona framework	None / Easy / Hard personas	None	None	Role-based coworker profiles rather than user personas
Long-horizon execution	Moderate	Moderate–High	High	High in code reasoning, but one-shot patch generation in original setup	High

Cross-benchmark comparison

Dimension	τ -bench	τ^2 -bench	REAL	SWE-bench	TheAgentCompany
Cross-tool / cross-app work	Limited to domain tools	More extensive shared environment	Website-focused	Code repository / execution	Strong: browser, terminal, Python, GitLab, Plane, RocketChat, ownCloud
Evaluation style	Final DB state + required info, pass^k	State / action / communication assertions	Programmatic state checks + LLM judge	Execution-based tests	Checkpoint-based , mostly deterministic + some LLM evaluation
Partial credit 	<u>Limited</u>	More flexible assertions, but still task success oriented	Mostly binary outcome	Mostly solved / not solved	Yes, explicit weighted checkpoint partial credit
Reproducibility / control	High	High	Very high	High	Very high
Main realism strength	Captures information gathering from a user	Captures coordination with an active user	Captures realistic website operation	Captures real developer issues and repository-scale problems	Captures professional workflows, multiple tools, and coworker communication
Main realism limitation	User's latent goal is still fully predefined	User simulator is still tightly constrained	No user interaction; task starts after intent is specified	Real request, but turned into a static issue-to-patch task	Initial task intent is intentionally clear; coworker interaction $\neq$ unclear user intent

Conclusion

1. User-request realism

- **Gap:** Most benchmarks still assume that the user already knows what they want.



Proposed Method: Real users may begin with **partially formed or ambiguous goals** and further revise them during interaction. This can include **goal refinement from vague to specific requests, constraint emergence, preference revision after seeing trade-offs, goal switching when the original plan becomes infeasible, and preference inconsistency across turns.** Evaluate whether agents can not only recover missing information, but also help clarify and adapt to user intent as it is formed and changes over time.

2. Interaction-aware evaluation

- **Gap:** Evaluation is still largely **outcome-centric**; benchmarks rarely give explicit credit for **good intermediate interaction behavior**.
 - a. **τ -bench:** mainly evaluates final state and required information.
 - b. **τ^2 -bench:** adds state/action/communication assertions, but remains task-success oriented.
 - c. **REAL:** primarily evaluates final environment state or retrieval correctness.
 - d. **SWE-bench:** essentially resolved vs. not resolved through tests.
 - e. **TheAgentCompany:** supports weighted checkpoint partial credit, but checkpoints mainly capture execution progress rather than whether the agent asked the right follow-up question, detected changing intent, or clarified ambiguity appropriately.