BIZBENCH: UNVEILING HIDDEN CHALLENGES IN DEPLOYING LONG-HORIZON AGENTS FOR REAL-WORLD BUSINESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing evaluations of long-horizon agents often emphasize final task success while overlooking the validity of intermediate decisions. Real-world business applications, however, demand stricter assessment: an agent must behave correctly not only at the endpoint, but throughout the entire execution process. This requirement is particularly critical in high-stakes domains such as payments, where invalid intermediate actions can cause financial loss, regulatory violations, or irreversible downstream consequences even when the final goal is eventually achieved. Consequently, agents that perform well on existing benchmarks may still fail in deployment, as outcome-centric evaluation provides limited insight into the validity and downstream consequences of intermediate decisions.

In this work, we aim to narrow the gap between offline evaluation and real-world deployment of long-horizon agents. We introduce BizBench, derived from business procedures at a leading payments company. Its executable environment aligns policies, states, and tool behavior, with 789 knowledge documents and 601 discoverable business tools. Tasks combine resource, temporal, and commitment constraints, requiring agents to reconcile knowledge conflicts and identify outdated guidance. We replay reference executions to verify that each task can be completed while satisfying its constraints. The resulting 50 tasks evaluate final success and execution validity through deterministic checks of outcomes and histories, while permitting multiple legitimate strategies.

Across ten evaluated models, strict success peaks at 64.0%, and some runs exceed 200 tool calls. Even Claude Opus 5 and GPT-6 Sol achieve strict success rates of only 64.0% and 61.3%, respectively. Qwen3.8-27B achieves 42.0% goal attainment but only 32.7% strict success, showing that goal completion can mask invalid execution. BizBench provides a testbed for developing long-horizon agents that coordinate information gathering, planning, and action to complete business tasks while maintaining valid execution throughout.

1 INTRODUCTION

AI advances are reshaping real-world business operations, with customer service as a representative example. Early FAQ systems answered queries by matching them to curated question–answer pairs (Sakata et al., 2019; Li et al., 2017a). While effective for isolated queries, they struggled with contextual understanding across turns. Deep neural networks later enabled end-to-end task-oriented dialogue systems, extending customer service from isolated question answering to multi-turn task resolution (Raghu et al., 2021; Budzianowski et al., 2018; Li et al., 2017b). These systems tracked conversational context and accessed external knowledge, with capabilities largely centered on predefined dialogue acts, slots, and short-horizon state tracking. More recently, large language models (LLMs) have enabled agentic customer service, advancing toward autonomous execution of complex business tasks through policy interpretation, multi-step planning, and external tool orchestration (Yao et al., 2025; Shi et al., 2026; Tong et al., 2026). As agents may commit resources and constrain later choices, reliable real-world deployment requires both task completion and valid execution throughout (Yao et al., 2025; Ruan et al., 2024).

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

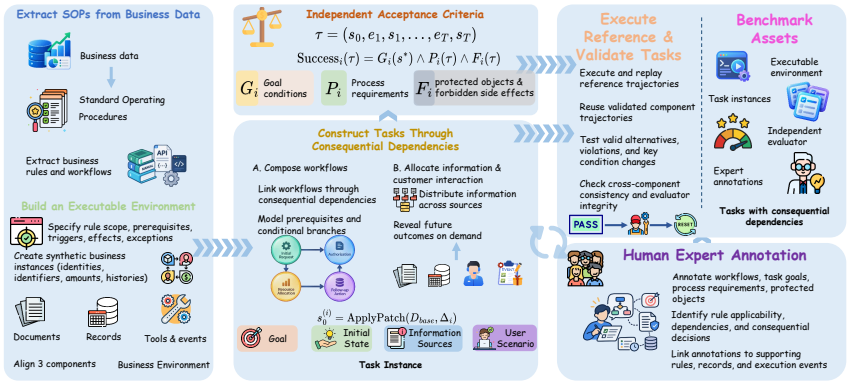

Figure 1: Overview of BizBench construction. Business procedures are normalized into an executable environment; tasks combine consequential dependencies with separate goal, process, and protection criteria, and are validated using reference executions and targeted counterexamples.

Reliable business deployment faces two challenges. First, agents must satisfy business requirements at critical intermediate steps. An unauthorized payment remains a violation even if the money is later returned (Yao et al., 2025; Ruan et al., 2024). Second, agents must determine which rules govern those steps. Business rules are scattered across policy manuals, FAQs, and operational notices (Tong et al., 2026; Shi et al., 2026). Different teams update these documents at different times. Staff turnover or changes in ownership can leave older guidance without active maintenance. Current and obsolete instructions may therefore coexist (Tong et al., 2026; Ouyang et al., 2025). Apparent conflicts also arise when rules apply to different regions, products, or effective periods. Finding a relevant document is therefore insufficient: agents must establish which instructions are authoritative and applicable to the current request (Shi et al., 2026; Nam et al., 2026). Existing long-horizon benchmarks cover policy-governed tool use and knowledge access (Yao et al., 2025; Lu et al., 2025; Shi et al., 2026). Yet final task success alone cannot establish whether agents followed the applicable rules at every critical step. This limits confidence in business deployment.

We therefore introduce BizBench to evaluate whether agents can identify applicable rules before taking consequential actions and complete business requests while maintaining valid execution throughout. The benchmark comprises 50 long-horizon tasks grounded in real-world business procedures from a leading global payments company. To test rule interpretation before action, we provide current and historical guidance with explicit scopes and effective periods across policy documents. Agents must determine which instructions govern the current request and use accompanying interface references to discover and invoke relevant tools. We instantiate the business procedures as executable workflows linked by shared resources, timing requirements, and prior commitments in a stateful environment. The environment retains action consequences, so later decisions must account for commitments already made.

To detect violations that final task success can conceal, a deterministic evaluator checks both outcomes and recorded execution histories against task goals and applicable rules. Success requires completing the requested goals while satisfying prior authorization requirements, persistent constraints, and restrictions on side effects. Completing a request therefore cannot compensate for a violation along the way, while different legitimate strategies remain acceptable. Reference replays verify that goals and constraints are jointly satisfiable; alternative valid executions and targeted violations test whether the evaluator accepts valid solutions and rejects invalid ones.

Our experiments show that even GPT-6 Sol and DeepSeek-V4.1-Flash achieve strict success rates of 61.3% and 43.3%, respectively, yet only 40.0% and 26.0% of tasks succeed in all three runs. By checking both outcomes and execution histories, BizBench detects invalid executions that goal-only evaluation would accept as successful. For example, it flags four DeepSeek-V4.1-Flash runs that achieve their goals and finish above the required cash-reserve floor despite breaching it earlier. Moreover, Qwen3.5-9B averages 187.2 tool calls per run, but over half return errors, mostly from queries to non-public resources; none of its 150 runs satisfies all task goals. These findings expose complementary weaknesses in execution consistency, process compliance, and effective tool use. BizBench provides a testbed for studying how agents coordinate information gathering, planning, and action to fulfill business requests while preserving execution validity.

2 Related Work

Long-horizon agent benchmarks. Agent benchmarks assess reasoning and tool use in interactive environments (Liu et al., 2024; Mialon et al., 2024). Web and desktop benchmarks cover shopping, browsing, and computer operation (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024b; Koh et al., 2024; Xie et al., 2024). SWE-bench evaluates repository-level issue resolution, while Terminal-Bench tests broader technical tasks through final-state checks (Jimenez et al., 2024; Merrill et al., 2026). SWE-Milestone checks feature completion and regressions at successive milestones (Deng et al., 2026); Long-Horizon-Terminal-Bench measures partial progress through subtask checks on the final state (Li et al., 2026). While these benchmarks emphasize technical progress and software integrity, BizBench models business environments with scattered tool documentation, noisy, outdated, or conflicting information, and changing conditions. It checks intermediate actions against applicable business requirements. An action's validity depends on the approvals, funds, and policy in force when it occurs.

Agents in business workflows. Business-oriented benchmarks introduce operational rules and organizational context. The τ family studies policy-governed interaction, user–agent coordination, and knowledge-grounded workflows (Yao et al., 2025; Barres et al., 2025; Shi et al., 2026). Enterprise suites cover browser-based knowledge work, compositional workflows, workplace collaboration, and CRM operations (Drouin et al., 2024; Boisvert et al., 2024; Xu et al., 2025; Huang et al., 2025; 2026). AutomationBench evaluates final application states without prescribing tool-call sequences, preserving flexibility in how agents solve tasks (Shepard & Salimans, 2026). BizBench likewise allows exploration and alternative strategies, while checking authorization and other business requirements at consequential steps, such as whether a transfer was approved before execution.

Long-horizon planning and adaptation. Tool-use learning and reasoning–action interleaving provide foundations for language agents (Schick et al., 2023; Yao et al., 2023b). Reflection, skill reuse, and search support multistep reasoning and planning (Shinn et al., 2023; Wang et al., 2023; Yao et al., 2023a; Zhou et al., 2024a), while clarification research examines when additional information is useful (Gulati et al., 2026). BizBench focuses on replanning in business systems where permitted actions can undermine later obligations and both acting and waiting can incur costs. Plans must accommodate commitments already made.

3 BizBench

BizBench evaluates agents that execute customer requests across real-world business workflows. Each task combines an objective, an initial business state, a customer scenario, and acceptance criteria. Agents use a shared environment of documents, records, and executable tools. Success requires fulfilling the request while respecting the requirements that govern its intermediate execution.

3.1 Business Context and Reconstructed Environment

BizBench draws on business procedures from one of the world's leading payments and financial technology companies. The company supports merchants and enterprises through cross-border payments, foreign exchange, and treasury services. Customer requests can span multiple currencies, legal entities, and financial operations. In practice, deployed agents must reconcile fragmented records and apply the right rules, even when payments or reviews are still pending. An action accepted by one system can still consume funds or capacity needed to fulfill a later obligation.

To preserve the confidentiality of business rules, we abstract sensitive details from the source workflows. We reconstruct their key business logic and execution mechanisms in an executable environment. Agents consult documents for business rules and tool instructions, and query records to establish the customer's current business state. They interact with a user simulator to obtain private customer information and required approvals. Business tools allow agents to execute operations and inspect their results. The frozen release contains 50 tasks and 789 knowledge documents. Agents have nine always-available tools and can discover 601 business capabilities; the user simulator has ten tools. Each run uses an independent copy of the database. Appendix B reports the environment inventory, task composition, and reference execution distributions.

3.2 Construction Insights from Our Business Workflows

Outdated and conflicting guidance. Business documents can remain accessible after their instructions have been superseded. Apparent conflicts can also arise because two rules govern different products, regions, or effective periods. Resolving a conflict requires establishing which source is authoritative and which rule applies to the particular business event. An expired insurance policy, for example, may still cover a past incident. BizBench includes current and historical guidance with explicit scopes and effective periods, making rule selection part of executing the request.

Authorization and execution validity. General consent does not substitute for approval of a specific operation. Approval must cover the actual operation and its terms, come from an agent with the required authority, and precede execution. Other requirements, such as a minimum reserve or continuous guarantee coverage, must hold throughout the workflow. An operation accepted by a tool can still breach these requirements. For example, an account holds \$60 and must retain at least \$20 throughout execution. Paying a \$60 retainer leaves the account at zero and immediately breaches this requirement, even though the payment is accepted. Later refunds and adjustments can restore the closing reserve without repairing the earlier breach (Appendix D.3). BizBench therefore evaluates the relevant approvals and state changes throughout execution alongside the requested outcomes.

Shared resources across workflows. Business obligations can draw on the same funds, authorization limits, inventory, or licenses. In our example, the agent must close the receivables file before releasing reserved funds to pay the review fee for a separate account application. The release route must provide sufficient funds early enough for the review to finish before its deadline (Figure 4). We execute linked workflows against the same resource records, with allocations in one workflow reducing availability for the others.

Delayed information and time-sensitive decisions. Reviews, refunds, and settlements may finish after another obligation requires action. In the delivery case (Appendix C.2), the priority booking window closes before warehouse confirmation arrives. Waiting for that confirmation makes the first delivery deadline unattainable. BizBench represents when information becomes observable and when actions become available, so that waiting can change whether the request remains feasible.

3.3 Evaluating Outcomes and Execution

Let $\tau = (s_0, e_1, s_1, \ldots, e_T, s_T)$ denote an execution history, where e_t is a tool execution or environment event. Task i succeeds when

$$\text{Success}_i(\tau) = G_i(\tau) \wedge P_i(\tau) \wedge F_i(\tau). \tag{1}$$

Here, G_i checks the requested outcomes and permitted terminal conditions, P_i checks requirements governing execution, and F_i checks protected objects and prohibited side effects. All three use the recorded evidence relevant to their requirements. For example, a payment goal may require an actual settlement receipt and its timestamp, while a reserve requirement needs the balance after each relevant state change.

Checks operate on business states, ledgers, receipts, and event histories. They distinguish an executed action from a proposal, an approval, or a rejected call without side effects. Different valid action sequences can satisfy the same criteria; the evaluator does not match executions to a reference trajectory. The primary score is determined by executable checks and does not use an LLM judge or score the quality of the final prose response.

4 Data Construction

We organize data construction into five stages. First, we organize business rules into structured procedures and adapt them for simulation (Section 4.1). Second, we build executable tasks by defining states, operations, and acceptance checks, and linking workflows through resource and timing constraints (Section 4.2). Third, we distribute task information across sources and define the customer scenario and simulator policy (Section 4.3). Fourth, we validate tasks through reference replay, valid alternatives, and targeted violations (Section 4.4). Finally, experts review task annotations and select the final benchmark (Section 4.5).

4.1 Grounding Tasks in Business Procedures

We use standard operating procedures (SOPs) as an intermediate representation for adapting business rules under confidentiality constraints. The source rules contain proprietary business information and therefore cannot be publicly released in full. Rules in the source documents are scattered across workflow descriptions, policy clauses, and exceptions, and changes to one rule can affect related requirements. Replacing individual names or values is therefore insufficient to adapt the business logic consistently. We first consolidate the source material into structured SOPs, then rewrite sensitive content at the procedure level. The SOPs used for benchmark construction are simulated adaptations of the source workflows.

Algorithm 1 summarizes the Candidate–Plan–SOP process. *Candidates* link workflow content to source evidence, and *Plans* group related Candidates into hierarchical business branches. The resulting *SOPs* specify steps, decisions, and transitions, including returns for repeated steps.

Algorithm 1 Constructing simulated SOPs from business documents

Input: Business documents $\mathcal{D}$
Output: Simulated SOPs $\widetilde{\mathcal{S}}$
$\mathcal{C} \leftarrow$ extract Candidates from $\mathcal{D}$, retaining source evidence links
$\mathcal{P} \leftarrow$ group related Candidates into hierarchical Plans with evidence links
$\mathcal{S} \leftarrow$ generate SOPs from $\mathcal{P}$ and their source evidence
$\widetilde{\mathcal{S}} \leftarrow \emptyset$
for each SOP $s \in \mathcal{S}$ **do**
    Specify steps, decisions, next-step links, and return links in s
    Review s via its Plan and Candidates; attach only applicable, non-conflicting rules
    $\tilde{s} \leftarrow$ rewrite sensitive business content in s for simulation
    $\widetilde{\mathcal{S}} \leftarrow \widetilde{\mathcal{S}} \cup \{\tilde{s}\}$
end for
return $\widetilde{\mathcal{S}}$

Before adapting an SOP for simulation, we review its rules against source evidence, link them to relevant steps where possible, and exclude content with unresolved conflicts. The resulting simulated SOPs provide task requirements organized by scope, affected objects, prerequisites, actions, and exceptions. Our authoring scheme distinguishes requirements grounded in the source material, interpretations needed to make them executable, and synthetic extensions introduced for task construction. Task instances use synthetic identities, identifiers, amounts, and transaction histories while retaining business distinctions such as separate legal entities, product-specific authorization, and the continuing obligations of an account being closed. Additional resource limits, event schedules, and linking conditions are made explicit in the task's documents and records.

4.2 Building Executable Tasks

We map each task requirement to the records and operations that determine whether it is satisfied. Documents state the requirement, records provide the relevant facts, and action handlers or scheduled events implement its effects. Acceptance checks identify the corresponding evidence. For a continuous reserve, this mapping connects the account and required minimum to cash changes and the resulting balance history.

Each instance specifies customer goals, initial resources, existing obligations, prior transactions, and scheduled events in a shared database. Composite tasks connect workflows through resource use, authorization, state prerequisites, or explicit timing requirements. Linking conditions state which resource or event connects the obligations. A shared resource changes later availability; an explicit ordering requirement constrains the permitted sequence. We use dependency graphs to organize these relationships and their conditional branches during authoring. The agent receives the public rules and interfaces, while authoring graphs and reference strategies remain unavailable.

4.3 Assigning Information and Customer Roles

We assign required information to documents, queryable records, customer materials, or observable event results. Rule applicability is expressed through conditions such as entity, product, purpose,

version, and effective period. The authoring review checks that apparent conflicts can be resolved using the available evidence and that information needed for a decision is available at the appropriate stage. Unrevealed external outcomes remain hidden until their events occur.

The customer scenario specifies private facts, preferences, available materials, and permitted actions. The simulator policy requires these facts and fixed business constraints to remain consistent across the interaction. Review checks whether responses stay within this information boundary and preserve fixed contract constraints. Validation checks whether the supplied information supports a valid strategy under the task's declared event conditions.

4.4 Executing References and Validating Tasks

Acceptance criteria are specified from the customer objective and applicable rules independently of a particular reference sequence. We execute reference strategies through the task's tools and customer interactions, then replay recorded actions from the initial state. For supported composite tasks, validated component trajectories can be scheduled under the linking conditions. A passing reference provides an executable witness that the encoded requirements can be satisfied together. It does not establish a shortest or optimal solution.

The validation protocol covers valid alternatives, targeted violations, and changes to conditions that determine feasibility. It also examines whether harmless variations, such as an idempotent retry, leave the judgment unchanged. These tests check both acceptance of valid behavior and rejection of actual violations. Document, interface, simulator, and evaluator discrepancies are recorded for correction and retesting. The protocol records the tasks and requirements exercised by each test category. All 50 tasks have a passing reference execution and a validated alternative. Each evaluation uses an independent initial state and retains the resulting execution history.

4.5 Human Expert Annotation

Expert annotation assesses whether the encoded requirements reflect the intended business procedures. Five researchers and a team of business-domain experts contributed to task annotation and review. The rubric records each task's workflow, dependencies, consequential decisions, and protected objects. For each requirement, annotators identify its scope, governing rule, relevant action or event, and observable evidence for its acceptance check.

Reviewers compare these annotations with the customer request, public documents, records, and reference, alternative, and violation traces. They examine whether the agent has the information needed for a valid strategy, whether the evaluator's judgments match the business rules, and whether tool or simulator behavior changes the stated terms. Revised tasks are rerun through the validation protocol. The five researchers unanimously approved the 50 tasks selected for the final benchmark.

5 Experiments

Our experiments examine whether agents complete customer requests while satisfying the requirements that govern execution. We report goal, process, protection, and strict success across models, then use information ablations and execution traces to study customer interaction, document retrieval, and long-horizon tool use. We also assess the stability of the user simulator.

5.1 Experimental Setup

We evaluate BizBench on the same 50 tasks with proprietary and open-source models (OpenAI, 2026; DeepSeek-AI, 2026a;b; Yang et al., 2025) (Table 1). Each run starts from an independent copy of the initial database and uses the same public documents, tools, and acceptance criteria. In the default condition, agents retrieve documents, discover business capabilities, and interact with a DeepSeek-V4.1-Flash user simulator. Reference trajectories, evaluator predicates, and unrevealed event outcomes remain unavailable to the agent.

The runner imposes no cumulative limit on generated tokens, tool calls, or scheduler steps, and sets per-response output-token limits high enough to avoid routine truncation. Episodes retain a wall-clock timeout and a tool-error threshold. Each request otherwise includes the accumulated dialogue

and tool history. Following Shi et al. (2026), if a provider reports context-window overflow, the oldest quarter of eligible knowledge-search and document-read receipts is replaced in the model-facing context, while the complete execution record is preserved.

For models evaluated three times per task, we average single-run success across the three trials; other models have one trial per task. Information ablations use the same tasks, bank-side tools, database, and acceptance criteria. We track infrastructure and simulator protocol failures separately from business outcomes, and resume transient transport interruptions from the verified prior state.

5.2 Overall Performance and Process Validity

Table 1: Goal (G), process (P), protection (F), intermediate failure, and strict success (%). Intermediate failure means that a process or protection requirement failed. Claude Opus 5, GLM-5.3-Flash, and Muse Spark 1.3 have one run per task; the other models have three.

Model	**Goal** ($\mathbf{G}$)	**Process** ($\mathbf{P}$)	**Protection** ($\mathbf{F}$)	**IF** ($\neg(\mathbf{P} \wedge \mathbf{F})$)	**Strict** ($\mathbf{G} \wedge \mathbf{P} \wedge \mathbf{F}$)
Claude Opus 5	64.0	96.0	98.0	4.0	**64.0**
GPT-6 Sol	64.7	94.7	100.0	5.3	**61.3**
GLM-5.3-Flash	50.0	80.0	90.0	26.0	**46.0**
DeepSeek-V4.1-Flash	46.7	92.0	96.7	9.3	**43.3**
DeepSeek-V4-Pro	41.3	85.3	88.7	24.7	**39.3**
Muse Spark 1.3	38.0	94.0	98.0	6.0	**38.0**
Qwen3.8-27B	42.0	74.0	93.3	28.0	**32.7**
DeepSeek-V4-Flash	7.3	53.3	95.3	47.3	**5.3**
Qwen3.5-397B-A17B	1.3	46.7	93.3	56.7	**0.7**
Qwen3.5-9B	0.0	18.7	89.3	83.3	**0.0**

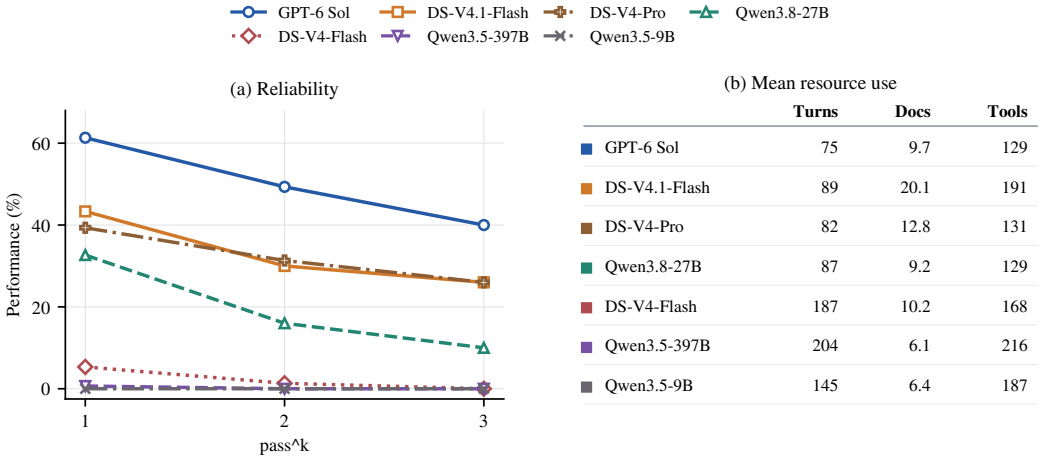

Figure 2: Reliability and resource use for models with three runs per task. Left: pass^k is the fraction of tasks successful in every run of a k-run subset, averaged across subsets. Right: mean assistant turns, distinct documents read, and assistant tool calls per run.

The main comparison separates goal attainment from valid execution using the criteria in Equation 1. Table 1 reports the pass rates for goals and permitted terminal conditions (G), process requirements (P), protection requirements (F), full success ($G \wedge P \wedge F$), and intermediate failure (IF, $\neg(P \wedge F)$). For this analysis, we map acceptance checks to these categories and decompose checks combining different requirements without changing the full-success criterion.

Frontier models perform well but leave substantial room for improvement: Claude Opus 5 achieves the highest strict success rate at 64.0%, followed by GPT-6 Sol at 61.3%. The next highest strict success rates are 46.0% (GLM-5.3-Flash), 43.3% (DeepSeek-V4.1-Flash), and 39.3% (DeepSeek-V4-Pro), while Qwen3.5-9B completes none of its 150 runs. Every model incurs intermediate failures, from 2 of 50 runs (4.0%) for Claude Opus 5 to 125 of 150 (83.3%) for Qwen3.5-9B; such failures are generally more frequent among lower-scoring models. Process pass rates are below protection

pass rates for every model, spanning 18.7–96.0% for P versus 88.7–100.0% for F. Figure 2(a) shows that reliability declines as k increases: GPT-6 Sol falls from 61.3% at $k = 1$ to 40.0% at $k = 3$, while DeepSeek-V4.1-Flash falls from 43.3% to 26.0%. At $k = 3$, Qwen3.8-27B reaches 10.0%, and DeepSeek-V4-Flash and both Qwen3.5 models reach 0%. These declines indicate that the models remain unreliable across repeated runs.

5.3 Information Acquisition

Table 2: DeepSeek-V4.1-Flash strict success under three information conditions on 50 tasks.

Information condition	Strict (%)
Vanilla	43.3
+ Gold-doc	50.0
+ Full customer disclosure	70.0

Table 3: DeepSeek-V4.1-Flash strict success with tool-call caps (three-run mean); R_i is the Golden call count.

Tool-call cap	Strict (%)
None	43.3
$1.2 \times R_i$	0.7
$1.5 \times R_i$	6.7
$2.0 \times R_i$	26.7

Strict success rises from 43.3% to 70.0% when all customer-known task information is supplied in the opening request without a user simulator, compared with 50.0% when the revised Gold-doc package replaces retrieval (Table 2). Required customer approvals remain in force. The larger gain from removing multi-turn customer dialogue implies that eliciting customer-specific constraints and retaining their terms through execution are difficult. Gold-doc removes the search for relevant rules, but the agent must still establish current facts from records and apply those rules across linked actions; the modest gain suggests that finding the right documents is only part of the knowledge challenge. Even with all customer facts available at the outset, 30.0% of tasks remain unsolved; document use, planning, and valid execution still matter.

5.4 Long-Horizon Behavior and Budget Sensitivity

The Golden reference traces average 111.8 assistant tool calls. DeepSeek-V4.1-Flash averages 191.3 calls (1.71 times the reference). The excess is concentrated in information access: record queries average 111.1 versus 59.2 in the references, and knowledge searches or document reads average 43.3 versus 11.5, while business capability calls and request executions are similar (26.9 versus 28.7).

Figure 2(b) shows that longer traces need not reflect greater progress across models. GPT-6 Sol averages 75.3 assistant turns and 129.1 tool calls, whereas Qwen3.5-397B-A17B averages 203.6 turns and 216.4 calls but succeeds in only one of 150 runs. Appendix D reports the full call distributions.

Table 3 retrospectively limits each task to $\lceil \alpha R_i \rceil$ assistant tool calls, where R_i is its Golden reference count. Across three runs per task, strict success falls from 65/150 (43.3%) without a cap to 1/150 (0.7%) at $1.2\times$, 10/150 (6.7%) at $1.5\times$, and 40/150 (26.7%) at $2.0\times$. Even twice the reference call count excludes many otherwise successful runs, quantifying the cost of the agents' information search and repeated checking.

5.5 Trajectory Analysis

Table 4: Trajectory statistics across stronger and weaker models. Tool calls, identical repeated database requests and responses, and tool-error replies are means per completed run. The last column counts runs reaching the goal but failing process or protection requirements.

Model	Tool calls	Repeat DB calls	Tool errors	$\mathbf{G} \wedge \neg(\mathbf{P} \wedge \mathbf{F})$
DeepSeek-V4.1-Flash	191.3	16.4	2.0	5/150
Qwen3.8-27B	129.2	7.8	4.2	14/150
DeepSeek-V4-Flash	168.0	23.7	8.2	3/150
Qwen3.5-9B	187.2	16.5	101.6	0/150

Progress and repetition. Table 4 distinguishes additional tool use from literal repetition: a repeated database call has the same tool, arguments, and returned content as an earlier call in that run. DeepSeek-V4.1-Flash makes 16.4 such calls per run; DeepSeek-V4-Flash makes 23.7 despite using

fewer tools overall. Qwen3.5-9B receives 101.6 tool-error replies per run, 71% of them for querying non-public resources, and reads only 6.4 distinct documents, compared with 20.1 for DeepSeek-V4.1-Flash. Thus, long traces mix information gathering with redundant reads and invalid navigation; tool count alone does not measure progress.

Intermediate violations. Among DeepSeek-V4.1-Flash runs, five reach the requested goal but fail the full criteria: four cross a required cash-reserve floor before later restoring the balance, and one attempts execution before approval and receives a rejection. Qwen3.8-27B has 14 such runs, including eight rejected pre-approval execution attempts; DeepSeek-V4-Flash has three. Qwen3.5-9B fails process requirements in 122/150 runs, including 88 pre-approval execution attempts, but reaches no goals. These histories explain why final goal checks alone overstate valid execution.

Business-document scope. Golden references open 5.4 distinct documents per task, whereas DeepSeek-V4.1-Flash opens 20.1. Additional reads include articles on related payment operations and products with similar terminology but different applicability. This shows that the complexity of the document collection makes identifying applicable guidance challenging for the model. Appendix D.4 gives a tax-payment example.

5.6 Simulator Reliability and Evaluation Robustness

We assess whether the user simulator follows the interaction protocol under two prompt variants. The revised prompt explicitly specifies the order of customer actions and when to return control to the agent. Using DeepSeek-V4.1-Flash for both roles, we observe protocol violations on 4 of 50 tasks with the original prompt and none with the revised prompt. These violations involve batched approvals or further tool use before returning control. Appendix D.2 details the protocol and paired analysis.

We also examine sensitivity to the choice of user-simulator model. Qwen3.8-27B executes the tasks in all four conditions. Figure 3 reports the mean strict success rate over three runs for each simulator. The mean rates range from 30.7% to 36.0%. The largest deviation from the default DeepSeek-V4.1-Flash setting (32.7%) is 3.3 percentage points. This small spread suggests that agent performance is relatively robust to the choice of user simulator in this comparison.

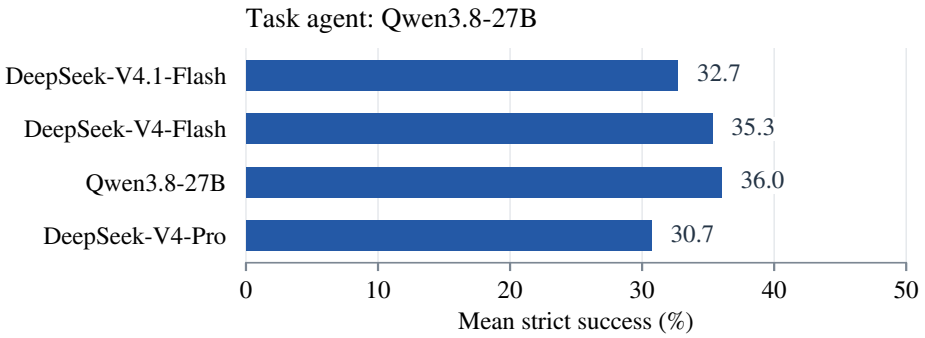

Figure 3: User-simulator model comparison using Qwen3.8-27B as the agent. Bars show the mean strict success rate over three runs; DeepSeek-V4.1-Flash is the default simulator.

6 Conclusion

With BizBench, business requirements become testable conditions on both the outcomes and execution of long-horizon tasks. Our experiments show that task completion, process compliance, and consistency across runs reveal different weaknesses in current agents. Information ablations suggest that obtaining the right customer facts can substantially improve performance, while trajectory analysis shows that extensive tool use does not necessarily indicate meaningful progress. These findings motivate closer coordination between information gathering and action selection: agents need to establish when they have sufficient evidence to proceed and which obligations must remain satisfied as execution unfolds.

REFERENCES

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

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. WorkArena++: Towards compositional planning and reasoning-based common knowledge work tasks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5996–6051. Curran Associates, Inc., 2024. doi: 10.52202/079017-0195. URL https://doi.org/10.52202/079017-0195.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 5016–5026, 2018.

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026a. URL https://arxiv.org/abs/2609.19969.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026b. URL https://arxiv.org/abs/2606.19348.

Gangda Deng, Zhaoling Chen, Zhongming Yu, Haoyang Fan, Yuhong Liu, Yuxin Yang, Dhruv Parikh, Rajgopal Kannan, Le Cong, Mengdi Wang, Qian Zhang, Viktor Prasanna, Xiangru Tang, and Xingyao Wang. SWE-Milestone: Evaluating AI Agents on Continuous Software Evolution, 2026. URL https://arxiv.org/abs/2603.13428. ICML 2026; arXiv:2603.13428.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. doi: 10.52202/075280-1220. URL https://doi.org/10.52202/075280-1220.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Anmol Gulati, Hariom Gupta, Elias Lumer, Sahil Sen, and Vamse Kumar Subbiah. Ask early, ask late, ask right: When does clarification timing matter for long-horizon agents?, 2026. URL https://arxiv.org/abs/2605.07937. arXiv preprint arXiv:2605.07937.

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3830–3850, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.194. URL https://aclanthology.org/2025.naacl-long.194/.

Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. CRMArena-Pro: Holistic assessment of LLM agents across diverse business scenarios and interactions. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2505.18878.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, pp. 54107–54157, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visually grounded web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, Guwei Jin, and Wei Chu. AliMe Assist: An intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2495–2498. ACM, 2017a. doi: 10.1145/3132847.3133169. URL https://doi.org/10.1145/3132847.3133169.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 733–743, 2017b.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-Horizon-Terminal-Bench: Testing the Limits of Agents on Long-Horizon Terminal Tasks with Dense Reward-Based Grading, 2026. URL https://arxiv.org/abs/2607.08964. arXiv preprint arXiv:2607.08964.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pp. 52989–53046, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces, 2026. URL https://arxiv.org/abs/2601.11868. arXiv preprint arXiv:2601.11868.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *International Conference on Learning Representations*, pp. 9025–9049, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Youngeun Nam, Joeun Kim, Hwanjun Song, Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. TimelyRAG: Semantic-temporal hybrid retrieval for time-critical question answering in overlapping-evolving documents, 2026. URL https://arxiv.org/abs/2609.11572. arXiv preprint arXiv:2609.11572.

OpenAI. GPT-6 Astra: A new generation of intelligence. OpenAI, September 2026. URL https://openai.com/index/gpt-6-astra/.

Jie Ouyang, Tingyue Pan, Mingyue Cheng, Ruiran Yan, Yucong Luo, Jiaying Lin, and Qi Liu. HoH: A dynamic benchmark for evaluating the impact of outdated information on retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6036–6063, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.301. URL https://aclanthology.org/2025.acl-long.301/.

Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, and Mausam. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4348–4366, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.357. URL https://aclanthology.org/2021.emnlp-main.357/.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.15817.

Wataru Sakata, Tomohide Shibata, Ribeka Tanaka, and Sadao Kurohashi. FAQ retrieval using query-question similarity and BERT-based query-answer relevance. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1113–1116. ACM, 2019. doi: 10.1145/3331184.3331326. URL https://doi.org/10.1145/3331184.3331326.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2997. URL https://doi.org/10.52202/075280-2997.

Daniel Shepard and Robin Salimans. AutomationBench, 2026. URL https://arxiv.org/abs/2604.18934. arXiv preprint arXiv:2604.18934.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-Knowledge: Evaluating conversational agents over unstructured knowledge, 2026. URL https://arxiv.org/abs/2603.04370. arXiv preprint arXiv:2603.04370.

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

Yongqi Tong, Xiaoyun Feng, Lyuxin Xue, Jianshe Li, Xin Zhang, and Jiang-Ming Yang. How to train a real-world silicon concierge? internalizing complex business workflow to only OneModel. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1303–1316, San Diego, California, USA, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-industry.91. URL https://aclanthology.org/2026.acl-industry.91/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291. arXiv preprint arXiv:2305.16291.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://doi.org/10.52202/079017-1650.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-0315. URL https://doi.org/10.52202/085713-0315.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://doi.org/10.52202/068431-1508.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. doi: 10.52202/075280-0517. URL https://doi.org/10.52202/075280-0517.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, pp. 9965–10017, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024a. URL https://proceedings.mlr.press/v235/zhou24r.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, pp. 15585–15606, 2024b. URL https://openreview.net/forum?id=oKn9c6ytLx.

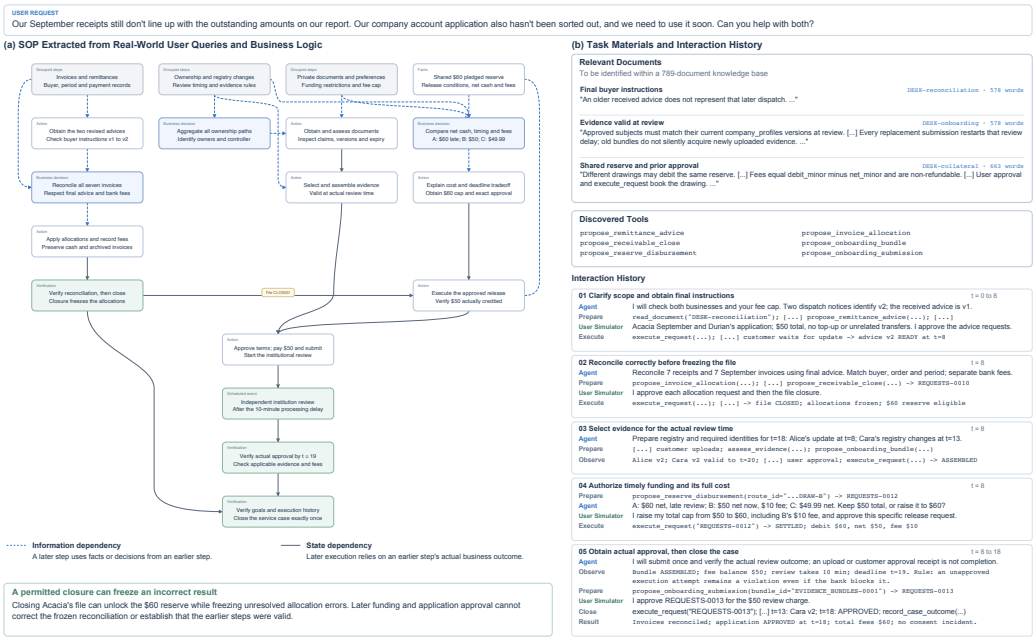

Figure 4: **A complete task example in BizBench.** In Q03, the customer asks to reconcile Acacia's September receipts and complete Durian Trading's account application. **(a)** The simulated SOP connects two rule-applicability decisions—resolving superseded buyer instructions and selecting evidence valid at review time—to reconciliation, funding and institutional approval. Closing the file unlocks a $60 reserve but freezes its allocations. **(b)** The agent must discover relevant documents and tools, obtain customer facts and request-specific approvals from the user simulator, and execute the linked workflows. Critical intermediate steps require correct reconciliation before closure and explicit consent to the revised $60 total fee cap and the actual request terms before execution; final application approval alone cannot establish these requirements. Document labels report body word counts; amounts are USD and t denotes logical minutes.

A Funding Arrangements for the Example Task

The three routes in Table 5 debit the same $60 reserve after Acacia's file is closed. In the reference execution, the customer explicitly raises the total fee cap from $50 to $60 and approves route B, which funds a single $50 review submission at $t = 8$; the application is approved at $t = 18$, within the deadline of $t = 19$. From the same release time, route A credits funds at $t = 18$, leaving the ten-minute review beyond the deadline, while route C falls one cent short of the submission fee. These comparisons are derived from the release terms, rather than observed model failures. The customer permits neither a top-up nor transfers from unrelated accounts.

Each task instance comprises *a customer objective and opening request*; *a user-simulator scenario* governing private facts and cooperation; *an initial state* recording resources, histories, obligations, and scheduled events; *access to shared documents and executable tools*; and *deterministic acceptance criteria* for final outcomes and execution histories. A replayable reference execution and validation evidence establish feasibility and check the evaluator's treatment of valid alternatives and violations. The displayed dialogue is constructed from a reference execution; the SOP graph, reference trace, and evaluator predicates are not supplied to the evaluated agent.

Table 5: Three releases of the same $60 reserve. The $50 institution review charge is additional to the release fee; route B therefore requires explicit authorization of $60 in total fees.

Release route	Net credit	Release fee	Credit delay
A: Standard settlement	$60	$0	10 minutes
B: Same-cycle processing	$50	$10	Immediate
C: Reduced-net processing	$49.99	$10.01	Immediate

B DATASET STATISTICS

All statistics refer to the 50 released tasks. Reference-call statistics use one designated executable Golden trace per task. Table 6 summarizes the shared environment and the available construction and validation records. Table 7 reports document lengths and reference execution distributions.

Task composition and requirements. The construction records explicitly pair 2 source workflows in 30 tasks (60%), using 43 distinct source components. This count describes recorded composition; the other tasks may also involve multiple business procedures. Every task includes execution-authorization requirements and checks protecting unrelated records or other customers. The construction validation records contain a passing reference and a passing alternative for every task. Requirement coverage indicates which tasks contain a requirement, separately from validation coverage or verifier accuracy.

Table 6: Scale and recorded coverage of the released dataset. Shared inventories include background material; task counts use the complete released set. Validation coverage reflects the construction checks.

Dataset and environment	**Count**
Tasks	50
Tasks explicitly composed from two source workflows	30 (60%)
Distinct source components used by these tasks	43
Always-available agent tools	9
Discoverable business capabilities	601
User tools across product portals	10
Knowledge documents	789
Total document-body tokens	598,827
Queryable collections in the shared base state	398
Records in these shared base-state collections	2,994
Tasks with execution-authorization requirements	50/50
Tasks with unrelated-record or customer-isolation checks	50/50
Tasks with a validated reference	50/50
Tasks with a validated alternative	50/50

Documents, tools, and records. Each document is counted once by its identifier. Token counts use document bodies and the `cl100k_base` tokenizer, excluding titles and metadata. Tool counts distinguish permanent agent interfaces, discoverable capabilities, and the union of customer tools across the product portals. Operations within a capability do not add interfaces. The shared base-state inventory includes empty collections and background cases before task-specific patches; 227 of its 398 public collections are nonempty. Private customer files and hidden event payloads are excluded. Collections are counted by query resource name.

Reference execution. Each task contributes one designated valid reference. Agent and customer calls are counted separately, and batched calls count individually. The Golden references average 111.8 agent calls and 28.6 customer calls. Record queries account for 42.1% of these combined calls.

Table 7: Document lengths and reference execution distributions across the full release. Reference calls and document openings describe a valid strategy; neither establishes a minimum requirement or measures model dialogue turns.

Characteristic	**Mean**	**Median**	**Range**
Document collection: per document			
Document length in tokens	759.0	452	44–10,755
Reference execution: per task			
Agent tool calls	111.8	100.5	51–282
Customer tool calls	28.6	21.5	10–74
Distinct agent tools or capabilities	11.6	11	9–17
Distinct documents opened in reference	5.4	5	2–15
Distinct business records read	29.9	23.5	4–96
Business instructions executed	14.2	11	6–36
Customer approvals	14.2	11	6–36
Event-update batches returned by waits	3.4	3	0–14

Dispatch calls count toward their target capability without counting the wrapper again. Distinct-tool counts exclude the unlocking interface, which remains included in total call counts. Distinct records count successful full-record reads, deduplicated by resource name and record identifier. They measure observed access rather than the minimum or exhaustive set of relevant records. Executed instructions count successfully executed request identifiers; customer approvals count successful approval actions. Event-update batches count successful customer waits that return new environment events. One batch may contain several events, and events produced internally by other operations are outside this count. These batches therefore measure observed updates rather than semantic decision stages.

C Illustrative Business Cases

The following examples focus on payment and delivery decisions in Q24 and Q34. All monetary amounts are in US dollars; times are logical minutes.

C.1 A Limited Net-Delivery Guarantee (Q24)

Two suppliers must receive their exact remaining invoice balances under current banking instructions. The customer permits at most $75 in total bank charges, including the earlier transfer's return charge. After that transfer resolves, the remaining balances are $300 and $400. A single-use guarantee offers exact net delivery for a fixed fee; the standard route exposes the recipient to intermediary deductions, followed by a separately charged top-up.

Consider assigning the guarantee to the $300 balance and sending the $400 balance through the standard route. Both suppliers ultimately receive the correct amounts. However, the $20 return charge, $20 guarantee fee, $35 intermediary deduction, and $5 top-up fee total $80. The fee budget is exceeded despite correct final payments. The case requires allocating a limited service across obligations and accounting for charges incurred at different stages of settlement.

C.2 Payment Reconciliation and Delivery Windows (Q34)

A merchant must reconcile an already-paid order without charging again and deliver three compatible units: one rated at least 15 by minute 2, and two rated at least 10 by minute 5. All require the same socket. The priority delivery service takes one minute but must be booked by minute 0; ordinary delivery takes three minutes.

Consider repairing the merchant mapping and propagating two captures totaling $10 and a $1 refund, yielding the order's $9 net payment, before waiting for the warehouse result at minute 1. The priority booking is then rejected because its window has closed. Compatible inventory is identified and the two later units arrive on time, but the early unit cannot meet its deadline through the remaining service. This sequence shows why an agent must coordinate information gathering with expiring

service rights: correct reconciliation and suitable stock alone do not complete the commercial objective.

D Execution-History Analysis

D.1 Model execution length and tool use

We analyze DeepSeek-V4.1-Flash Vanilla trajectories. Table 8 counts batched calls individually and separates assistant and user tools. Record queries use `search_records` or `get_record`. An exact repeat has the same name, arguments, and response as an earlier read in the same run, excluding queries that observe changed state.

Table 8: DeepSeek-V4.1-Flash Vanilla execution distribution.

Per completed trajectory	Mean	Median	90th percentile
Assistant tool calls	191.3	182	287
User portal tool calls	28.2	26	41
Record queries	111.1	101	180
Distinct documents read	20.1	22	34
Exact repeated reads	16.4	14	32

Successful runs average 13.7 exact repeated reads, versus 18.5 in failed runs. Rechecking unchanged records is measurable redundancy, although some rechecks may be deliberate validation.

D.2 Customer-simulator turn-taking patch

The comparison uses the same 50 task assets and DeepSeek-V4.1-Flash for both roles. The user model has `parallel_tool_calls=false` and a 393,216-token per-response limit in both runs. The unpatched prompt asks the customer to handle one write at a time and report each receipt; the patch adds an explicit priority order for mixed questions and actions. The patched run is an existing complete evaluation. A protocol violation means that one customer response executes multiple write or wait tools, or that the customer calls another tool after a successful write or wait receipt before reporting it in text and returning control to the agent.

The four unpatched protocol violations occur on four tasks: two batch customer approvals and two continue using the portal after a receipt. The paired exact McNemar test gives $p = 0.125$. Neither run has a user output-limit failure. Mean customer text turns are 28.4 without the patch and 30.7 with it. Other customer-tool errors occur on five tasks in each run; these include bank-state refusals induced by agent requests and therefore are not all simulator errors. Task data and role configurations match across runs; later verifier and one business-runtime revision do not enter the message-level protocol detector.

D.3 Two goal-passing, process-failing executions

The single-run Vanilla audit contains two executions with $G = 1$ and $P = 0$.

Rejected premature execution (Q13). The agent coordinates insurance settlement and a stored-value order payment. It invokes `execute_request` on the payment before portal approval; the tool returns `CUSTOMER_APPROVAL_REQUIRED`, with no payment or balance change. The customer then approves and the agent completes the goals. The premature attempt violates approval discipline: $G = 1$, $P = 0$, $F = 1$. No unauthorized transfer occurs.

Actual reserve breach (Q18). The customer needs interest corrections and accountant payments: a \$60 retainer at minute 0 and a \$40 balance by minute 2, while retaining \$20 in the operating account throughout. The agent pays the retainer from the initial \$60, leaving zero. A \$18.66 refund settles and the agent posts a \$50 adjustment at minute 1; the final account balance after the remaining payment is \$28.66. The goals and closing reserve pass, but the earlier zero balance violates the continuous requirement: $G = 1$, $P = 0$, $F = 1$. Only the execution history exposes this breach.

D.4 BUSINESS-DOCUMENT SCOPE IN RETRIEVAL

In Q01, the customer asks the agent to pay suppliers in euros and pounds and match earlier tax payments to the correct periods. The applicable Operations Desk rules are described in `DESK-tax` and `DESK-fx`. Searching for `tax payment matching periods reconciliation` returns `PX-B-77`, an overpayment-credit article for a different product, in first position and `PX-B-76`, a tax-agent article for that product, in fifth position. The agent opens both articles, whose terminology matches the query although neither governs this Operations Desk task.

E LIMITATIONS

The current release focuses on payment-related workflows; extending the task collection to other business domains would broaden its coverage. The primary score evaluates task outcomes and compliance with business requirements, without separately assessing the clarity or style of the final response. Complementary measures of communication quality could support a more comprehensive assessment of the customer-service experience.

AI DISCLOSURE

We used generative AI tools to assist in synthesizing the benchmark's simulated business documents, database states, tool interfaces, tasks, and reference executions. Section 4 describes the data-construction and validation procedures. We also used generative AI tools to assist with polishing the manuscript. All authors take responsibility for the final data, code, results, and claims in this paper.