CORECRAFT: TRAINING GENERALIZABLE AGENTS ON HIGH-FIDELITY RL ENVIRONMENTS

Sushant Mehta[*], **Logan Ritchie, Suhaas Garre, Ian Niebres, Nick Heiner, Edwin Chen**
Surge AI Research

ABSTRACT

We show that training AI agents on high-fidelity reinforcement learning environments produces capabilities that generalize beyond the training distribution. We introduce CORECRAFT, the first environment in EnterpriseBench, Surge AI's suite of agentic RL environments. CORECRAFT is a fully operational enterprise simulation of a customer support organization, comprising over 2,500 entities across 14 entity types with 23 unique tools, designed to measure whether AI agents can perform the multi-step, domain-specific work that real jobs demand. Frontier models such as Claude Opus 4.6, GPT-5.2, and Gemini 3.1 Pro solve fewer than 35% of tasks when all expert-authored rubric criteria must be satisfied. Using this environment, we train GLM 4.6 with Group Relative Policy Optimization (GRPO) and adaptive clipping. After a single epoch of training, the model improves from 25.37% to 36.76% task pass rate on held-out evaluation tasks. More importantly, these gains transfer to out-of-distribution benchmarks: +4.5% on BFCL Parallel, +7.4% on τ^2-Bench Retail, and +6.8% on Tool Decathlon (Pass@1). We believe three environment properties are consistent with the observed transfer: task-centric world building that optimizes for diverse, challenging tasks; expert-authored rubrics enabling reliable reward computation; and enterprise workflows that reflect realistic professional patterns. Our results suggest that environment quality, diversity, and realism are key factors enabling generalizable agent capabilities.

1 INTRODUCTION

The deployment of AI agents in production settings remains fairly limited despite rapid capability improvements on research benchmarks. A recent survey of 306 practitioners found that 68% of deployed agents execute ten or fewer steps before human intervention, with reliability cited as the primary development challenge (Pan et al., 2025). This gap between benchmark performance and deployment readiness suggests that current training approaches may not fully develop the robust, transferable skills required for real-world operation.

We hypothesize that this gap stems partly from the characteristics of training environments. Many existing agent benchmarks use simplified simulations, synthetic data, or contrived task structures that fail to capture the complexity of real-world workflows (Zhou et al., 2024; Xie et al., 2024). When agents are trained on such environments, they may learn environment-specific heuristics rather than generalizable problem-solving strategies.

This paper presents evidence supporting an alternative approach: training on high-fidelity environments designed around realistic enterprise workflows. We introduce results from CORECRAFT, a reinforcement learning environment simulating a customer support agent at a PC components company. CORECRAFT was developed for training and evaluating frontier models, where it revealed systematic capability gaps and an empirically-derived hierarchy of agentic capabilities (Ritchie et al., 2026). In this work, we demonstrate that realistic environments also serve as effective training substrates. CORECRAFT optimizes for task quality through three core design principles:

1. **Task-centric design**: Every entity, tool, and data source exists to support diverse, challenging tasks rather than to maximize world complexity.

[*]Correspondence to: sushantmehta@surgehq.ai

2. **Expert-authored evaluation**: Domain experts design both tasks and detailed rubrics, enabling reliable automated reward computation.
3. **Realistic workflows**: Tasks mirror genuine professional patterns including multi-step reasoning, constraint handling, and structured communication.

Using this environment, we train GLM 4.6 with GRPO and demonstrate that a single epoch of training produces substantial improvements that transfer to external benchmarks. Our contributions are:

- We demonstrate +11.39 percentage point improvement on held-out CORECRAFT tasks, exceeding the capability gap between Claude Sonnet 4.5 and Claude Opus 4.5 (+7.05 pp).
- We show transfer to out-of-distribution benchmarks: +4.5% on BFCL Parallel function calling, +7.4% on τ^2-Bench Retail customer service, and +6.8% on Tool Decathlon long-horizon tool use.
- We provide qualitative analysis of learned behaviors, identifying three categories of improvement: multi-step workflow execution, constraint handling, and response quality.
- We present evidence that environment quality, diversity, and realism are important factors enabling generalization, and discuss implications for agent training methodology.

2 RELATED WORK

2.1 AGENT BENCHMARKS AND EVALUATION

The landscape of agent evaluation has evolved from simplified web interfaces (Liu et al., 2018; Shi et al., 2017) toward realistic, execution-based environments. WebArena (Zhou et al., 2024) introduced self-hostable web environments with 812 long-horizon tasks. OSWorld (Xie et al., 2024) extended evaluation to desktop environments across operating systems. AppWorld (Trivedi et al., 2024) provides a controllable environment for benchmarking interactive coding agents across 9 apps and 457 APIs.

For software engineering, SWE-bench (Jimenez et al., 2024) evaluates agents on 2,294 real GitHub issues, while Terminal-Bench (Merrill et al., 2026) focuses on command-line tasks spanning scientific workflows, network configuration, and data analysis. TheAgentCompany (Xu et al., 2024) simulates a software company with 175 diverse professional tasks requiring web browsing, coding, and communication with simulated coworkers, finding that best agents complete only 24% of tasks autonomously. This consistent finding across benchmarks underscores the challenge of building reliable agents for real-world deployment.

Customer service evaluation has been addressed by τ-bench (Yao et al., 2024) and its successor τ^2-bench (Barres et al., 2025), which introduced the pass^k metric measuring reliability across multiple attempts. AgentBoard (Ma et al., 2024) provides multi-turn evaluation with subgoal progress tracking across diverse agentic tasks.

Recent surveys have organized this growing landscape. Yehudai et al. (2025) provide a comprehensive taxonomy across four dimensions: fundamental capabilities, application-specific benchmarks, generalist agents, and evaluation frameworks. Our work contributes to this literature by demonstrating that high-fidelity environments designed for evaluation can also serve as effective training substrates.

2.2 TOOL USE AND FUNCTION CALLING

Tool use represents a critical capability for agents interacting with external systems. The Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025) established systematic evaluation across 2,200+ test cases covering serial and parallel function calls. Gorilla (Patil et al., 2024) demonstrated that specialized training on API documentation can surpass GPT-4 on function calling tasks.

ToolLLM (Qin et al., 2024) introduced ToolBench with 16,464 real APIs and proposed depth-first search reasoning for complex tool sequences. T-Eval (Chen et al., 2024) decomposed tool utilization into six sub-processes for fine-grained capability diagnosis. These works highlight that tool use involves not just API knowledge but also planning, error recovery, and output formatting.

2.3 Reinforcement Learning for Language Model Agents

Modern RL approaches for language models trace from proximal policy optimization (PPO) (Schulman et al., 2017) through InstructGPT's three-stage training (Ouyang et al., 2022) to more recent methods. Direct Preference Optimization (DPO) (Rafailov et al., 2023) simplified training by eliminating explicit reward models, though subsequent work found PPO often outperforms DPO when properly tuned (Ivison et al., 2024).

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates the critic network by estimating baselines from group scores, significantly reducing memory requirements. DAPO (Yu et al., 2025) introduced adaptive clipping and other techniques preventing entropy collapse during training. DeepSeek-R1 (DeepSeek-AI, 2025) demonstrated that reasoning capabilities can emerge through pure RL with verifiable rewards.

Recent work has begun extending RL training to agentic settings. Agent-R1 (Cheng et al., 2025) proposed a modular RL framework for training LLM agents with end-to-end reinforcement learning on multi-step tasks. In instruction following, rubric-based reward signals have shown promise: He et al. (2025) introduced RIFL, a pipeline that uses expert-curated rubrics as reward signals for RL training, achieving substantial improvements on instruction-following benchmarks. Our work applies a similar rubric-based reward approach to agentic tasks in realistic enterprise environments, demonstrating that this paradigm extends beyond instruction following to multi-step tool-use workflows.

3 The Corecraft Environment

3.1 Design Philosophy

CORECRAFT simulates a customer support agent at Corecraft Computers, Inc., a fictional online PC parts retailer. The environment provides a stateful world in which agents interact with databases, tools, and simulated customers to complete support tasks.

Our central design principle is that high-quality, diverse tasks **provide a useful training signal**; everything else exists to support task diversity and difficulty. This contrasts with approaches that solely maximize entity or tool counts without sufficient complexity or diversity.

We prefer worlds with realistic entities that support more diverse, challenging tasks to worlds with many entities supporting redundant or trivial tasks.

3.2 World Structure

The environment is packaged as a self-contained Docker bundle, enabling stateful interactions where the container maintains world state (orders, tickets, inventory) for the duration of each episode. The bundle includes:

- **Entities**: Customer records, order histories, product catalogs, support tickets, shipping information, and company policies stored in structured JSON format.
- **Tools**: Database queries, order management APIs, communication interfaces, and diagnostic utilities exposed via Model Context Protocol (MCP), enabling standardized tool invocation across different agent frameworks.
- **Tasks**: Customer support scenarios with associated rubrics, each containing prompts, system context, and verifiable evaluation criteria.

Tasks range from simple lookups to complex multi-step workflows requiring coordination across multiple data sources. The MCP server handles all tool calls, routing agent requests to the appropriate backend operations while maintaining transactional consistency.

3.3 Task Categories

Tasks span several categories of different difficulty levels:

Information Retrieval: Basic tasks requiring agents to locate and present information from databases, such as order status queries or product specifications.

Communication: Tasks requiring professional customer-facing responses that synthesize the retrieved information into clear, actionable messages.

Reasoning: Tasks requiring the application of business rules, temporal constraints, or policy conditions. For example, determining refund eligibility based on purchase date, return window, and product condition.

Multi-Step Workflows: Complex tasks requiring sequential operations in the correct order: validate input, identify issues, apply fixes, compute totals, and format output.

3.4 Rubric Design

Each task includes expert-authored rubrics that enable automated evaluation. Rubrics decompose success into verifiable criteria, such as:

- **Completeness**: Did the agent address all required aspects?
- **Correctness**: Are factual claims accurate given the world state?
- **Constraint Satisfaction**: Are business rules and policies correctly applied?
- **Format Compliance**: Does the output follow required structure?

This decomposition enables reward computation for RL training while providing diagnostic information about failure modes. Task-level pass rates require *all* rubric criteria to be satisfied, providing a strict measure of end-to-end task completion.

4 Training Methodology

We train a frontier-scale language model using reinforcement learning on CORECRAFT, using the environment's expert-authored rubrics as verifiable reward signals. The goal is twofold. First, we aim to improve performance on the held-out CORECRAFT evaluation set, demonstrating that rubric-based RL training produces measurable gains on challenging agentic tasks. Second, and more importantly, we evaluate whether these gains *transfer* to external benchmarks that the model was never trained on, testing our hypothesis that high-fidelity environments teach generalizable agentic skills rather than environment-specific heuristics.

4.1 Base Model and Data

We use GLM 4.6 as our base model, a 357B parameter mixture-of-experts architecture with 32B active parameters. We used a random train-test split from the CORECRAFT environment with 1000 tasks used for training and 150 tasks in the held-out evaluation test.

4.2 Training Infrastructure

Figure 1 illustrates the overall architecture. The training pipeline consists of three stages operating in a continuous loop:

Rollout Generation: For each training prompt, we generate 16 rollouts. Each rollout interacts with its own stateful CORECRAFT Docker container. The rollout engine generates model responses, with tool calls routed to the Model Context Protocol (MCP) server running inside the container. The container maintains world state (orders, tickets, inventory) for the duration of the rollout, enabling multi-turn trajectories that proceed until the agent produces a final response.

Reward Computation: Once a trajectory completes, the final response is evaluated against task-specific rubrics using an LLM judge. Each rubric criterion is a verifiable assertion (e.g., "The response should identify that the PSU wattage is insufficient for the GPU's requirements"). Rewards are computed as the proportion of satisfied criteria.

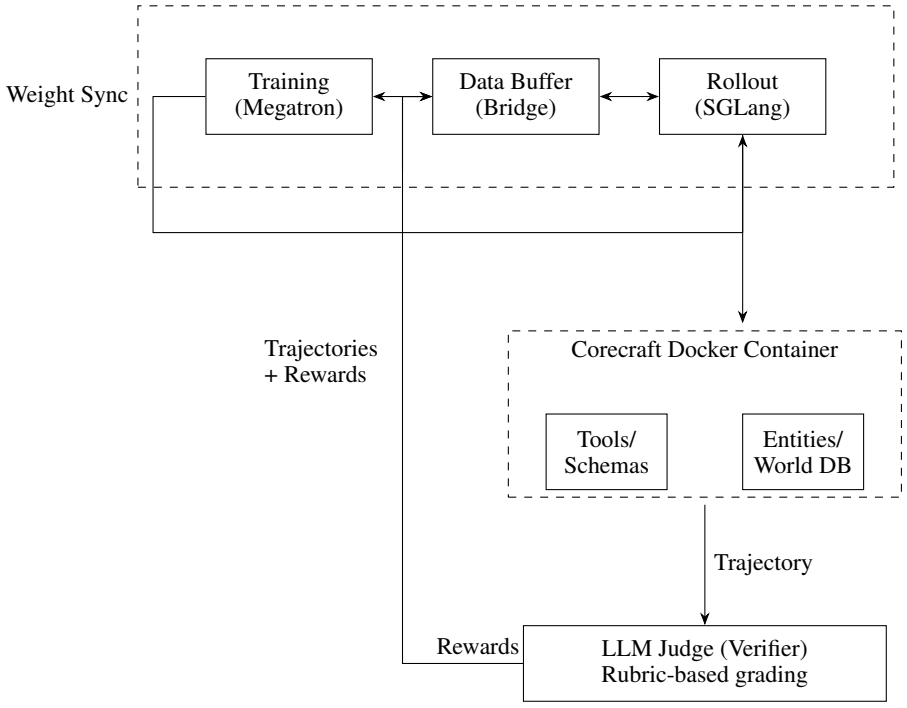

Figure 1: Architecture of the RL training loop with CORECRAFT. The rollout engine generates agent responses, routing tool calls to stateful Docker containers running the MCP server. Completed trajectories are evaluated by an LLM judge against task rubrics, with rewards flowing back to the training loop.

Training Update: Rollout results (trajectories and rewards) are written to a data buffer. The Megatron training loop reads batches from the buffer, computes policy gradients using GRPO, and updates model weights. Updated weights are synchronized back to the SGLang rollout engine for the next iteration.

4.3 GRPO WITH ADAPTIVE CLIPPING

We train using Group Relative Policy Optimization (Shao et al., 2024) with modifications inspired by DAPO (Yu et al., 2025). GRPO eliminates the critic network by computing advantages relative to other completions in a group.

4.4 REWARD COMPUTATION

Rewards are computed from rubric evaluations using an LLM judge. Each task completion receives scores on individual criteria, aggregated into a continuous score for training:

$$r = \frac{1}{|C|} \sum_{c \in C} \mathbf{1}[\text{criterion } c \text{ satisfied}] \tag{1}$$

where C is the set of rubric criteria. This provides a dense reward signal while maintaining interpretable task-level metrics.

Judge Reliability. We use an LLM as the grading mechanism, the rubric criteria are designed to be granular and objective, substantially reducing the ambiguity that can undermine LLM judge reliability. Each task is decomposed into several concrete, factual criteria that test a specific, verifiable assertion grounded in the environment state.

4.5 ENVIRONMENT ARCHITECTURE

The CORECRAFT training bundle is designed to be framework-agnostic, providing all components needed for RL integration:

Component	Location	Purpose
Tasks	`tasks/*.json`	Prompts and rubrics for training/eval
MCP Server	`server/`	Handles tool calls from the agent
World Data	`entities/`	Products, customers, orders, tickets
System Prompts	`system-prompts/`	Agent context and policies
Tool Definitions	`mcp-tools.json`	Available tools and schemas

Table 1: Components of the CORECRAFT Docker bundle for RL integration.

CORECRAFT can integrate with other RL frameworks supporting MCP tool-use, multi-turn trajectory collection, and custom reward functions.

5 RESULTS

5.1 IN-DISTRIBUTION PERFORMANCE

Table 2 presents task pass rates on the held-out CORECRAFT evaluation set. A task is counted as passed only when *all* rubric criteria are satisfied, making this a strict measure of end-to-end task completion. After one epoch of training, GLM 4.6 improves from 25.37% to 36.76%, a gain of 11.39 percentage points.

For comparison, this improvement of 11.39 percentage points exceeds the gap between Claude Sonnet 4.5 (26.44%) and Claude Opus 4.5 (33.49%), which is 7.05 percentage points, representing a substantial capability gain.

5.2 OUT-OF-DISTRIBUTION GENERALIZATION

Table 3 presents results on benchmarks outside the CORECRAFT training distribution. The trained model shows consistent improvements across long-horizon tool-use benchmarks.

BFCL Results. The Berkeley Function Calling Leaderboard (Patil et al., 2025) evaluates parallel and simple function calling scenarios. The trained model improves by 4.5% on parallel calls, which require invoking multiple APIs correctly in a single turn, suggesting better tool coordination learned from CORECRAFT's multi-tool workflows.

τ^2-Bench Results. The τ^2-Bench Retail domain (Barres et al., 2025) evaluates customer service conversations requiring database lookups, policy application, and response generation. The 7.4% improvement demonstrates that customer support skills transfer across different retail contexts.

Toolathlon Results. Toolathlon (**?**) evaluates language agents on 108 diverse, long-horizon tasks spanning 32 software applications and 604 tools, requiring an average of approximately 20 interaction turns per task. The trained model improves from 18.8% to 25.6% (+6.8%), placing it between Claude Haiku 4.5 and Grok-4 in our evaluations. This is a particularly notable transfer result because Toolathlon tasks (e.g., managing Kubernetes deployments, grading assignments on Canvas, syncing product inventories across databases) are substantially different from CORECRAFT's customer support domain, suggesting that the trained model acquired broadly applicable agentic skills rather than domain-specific shortcuts.

5.3 QUALITATIVE ANALYSIS OF LEARNED BEHAVIORS

To understand what the model learned, we analyzed paired trajectories comparing baseline and trained model behavior on identical tasks. Three categories of improvement emerged:

Model	Pass Rate (%)
GPT-5.1 High	36.86
GLM 4.6 (1 Epoch)	**36.76**
Claude Opus 4.5	33.49
GPT-5	30.45
Claude Sonnet 4.5	26.44
GLM 4.6 (Baseline)	**25.37**
Gemini 3 Pro Preview	24.84
GPT-5.2	24.77
Claude Haiku 4.5	7.21

Table 2: Task pass rates on the CORECRAFT held-out evaluation set. Pass rate requires all rubric criteria to be satisfied. The trained model surpasses Claude Opus 4.5 and approaches GPT-5.1 High performance.

Benchmark	Baseline	Trained	Δ
BFCL Parallel	91.0%	95.5%	+4.5%
BFCL Simple	91.5%	93.5%	+2.0%
τ^2-Bench Retail	68.7%	76.1%	+7.4%
Toolathlon	18.8%	25.6%	+6.8%

Table 3: Out-of-distribution benchmark results showing transfer from CORECRAFT training. Improvements are consistent across function calling, customer service, and long-horizon tool-use benchmarks.

Multi-Step Workflow Execution. The most significant improvement was in end-to-end execution with correct task decomposition. The trained model follows intended operation sequences: validate constraints, identify incompatibilities, apply fixes, compute totals, present structured output.

Example: A PC build compatibility task requires verifying component compatibility, identifying issues, recommending fixes, and computing updated totals. The baseline performs partial compatibility checks with incorrectly sequenced recommendations. The trained model executes the correct workflow: (1) retrieve order and catalog data, (2) identify case-motherboard incompatibility first, (3) recommend case replacement in correct sequence, (4) compute accurate price totals.

Constraint Handling and Reasoning. The trained model significantly improved at applying constraints correctly: filtering data by time windows, handling "before/after" relationships, and correctly joining records based on ordering.

Example: A task requires finding orders placed after a support ticket creation within a 30-day window with specific status constraints. The baseline retrieves orders but applies inconsistent time comparisons. The trained model uses explicit datetime comparisons, filters to post-ticket orders only, enforces the 30-day window, and produces clean structured output with timestamps.

Response Quality and Structure. The trained model produces more complete, actionable outputs aligned with realistic customer communication patterns.

Example: A shipment status email task requires generating professional communication with carrier details and next steps. The baseline produces inconsistent phrasing without clear guidance. The trained model produces structured email with: (1) order reference, (2) current shipment status with carrier name, (3) expected delivery window, (4) clear customer next steps, (5) support contact information.

These three competencies are not specific to PC components or Corecraft's particular tools, and they help explain the observed transfer to external benchmarks. Task decomposition transfers directly to any multi-step agent workflow. Improved constraint handling aligns with the +4.5% gain on

BFCL Parallel, which requires correct multi-constraint function calls. Better response structuring aligns with the +7.4% improvement on τ^2-Bench Retail, which evaluates customer communication quality. Because CORECRAFT tasks mirror realistic enterprise workflows, the model learns general professional patterns rather than environment-specific shortcuts.

6 DISCUSSION: ENVIRONMENT DESIGN AND ITS IMPLICATIONS FOR AGENT TRAINING

The production agent deployment landscape shows that reliability remains the primary barrier to real-world adoption (Pan et al., 2025). Our results suggest that the design properties of training environments play an important role in addressing this challenge.

Many existing agent benchmarks rely on synthetic data generation to achieve scale (Qin et al., 2024). While synthetic approaches enable large task counts, they may introduce distributional artifacts that limit transfer. The CORECRAFT environment takes a different approach, grounding tasks in realistic professional workflows through three properties that we believe contribute to the observed generalization. First, CORECRAFT features diverse task types spanning information retrieval, communication, reasoning, and multi-step workflows, requiring agents to develop a broad repertoire of skills rather than narrow heuristics. Second, expert-authored rubrics decompose success into verifiable sub-criteria, providing a richer reward signal than binary pass/fail. This rubric-based approach shares a common insight with recent work on rubric-based training for instruction following (He et al., 2025): decomposing evaluation into fine-grained criteria enables more effective RL training. While that work demonstrated the approach for single-turn instruction following, our results extend the paradigm to multi-step agentic tasks with tool use. Third, tasks are designed to mirror authentic workplace challenges, encouraging models to learn professional patterns (e.g., structured communication, constraint-aware data retrieval) that apply across domains.

The transfer to τ^2-Bench Retail is particularly encouraging given that customer service represents one of the most common deployment domains for production agents. Pan et al. (2025) report that finance, technology, and corporate services represent the highest-concentration deployment domains, with customer support among the most frequent application areas. Improvements in this domain could have immediate practical impact.

While we do not claim that scale is unimportant, our results suggest that environment quality, diversity, and realism play a meaningful role in producing transferable capabilities. A promising future direction would be to combine expert-designed core scenarios with synthetic variations to optimize for both quality and coverage.

7 CONCLUSION

We presented evidence that high-fidelity RL environments enable generalizable agent training. Using CORECRAFT, a realistic customer support environment, we showed that single-epoch GRPO training produces 11.39 percentage point improvement on held-out tasks and transfers to out-of-distribution benchmarks (+4.5% BFCL Parallel, +7.4% τ^2-Bench Retail, +6.8% Toolathlon).

Our results highlight the importance of environment quality, diversity, and realism in agent training. Effective training environments should feature diverse and challenging tasks designed by domain experts, rubric-based evaluation enabling reliable reward computation, and realistic workflow patterns that mirror genuine professional settings. These design principles may help bridge the gap between benchmark performance and production deployment reliability.

The trained model learned transferable competencies in task decomposition, constraint handling, and response structuring rather than environment-specific heuristics. This suggests that careful environment design can shape what agents learn, steering them toward robust capabilities applicable across domains.

REFERENCES

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*, 2025.

Chen, Z., Du, W., Zhang, W., et al. T-Eval: Evaluating the Tool Utilization Capability of Large Language Models Step by Step. In *Proceedings of ACL*, 2024.

Cheng, M., Li, Q., Liu, Z., et al. Agent-R1: Training Powerful LLM Agents with End-to-End Reinforcement Learning. *arXiv preprint arXiv:2511.14460*, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645:633–638, 2025.

He, Y., Chen, H., Liu, B., et al. Rubric-Based Benchmarking and Reinforcement Learning for Advancing LLM Instruction Following. *arXiv preprint arXiv:2511.10507*, 2025.

Ivison, H., Wang, Y., Liu, J., et al. Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback. In *Advances in Neural Information Processing Systems*, 2024.

Jimenez, C.E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K.R. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *International Conference on Learning Representations*, 2024.

Liu, E.Z., Guu, K., Pasupat, P., Shi, T., and Liang, P. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *International Conference on Learning Representations*, 2018.

Ma, C., Hu, X., Shen, L., et al. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems*, 2024.

Merrill, M.A., Shaw, A.G., Carlini, N., et al. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Ouyang, L., Wu, J., Jiang, X., et al. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, 2022.

Pan, M.Z., Arabzadeh, N., Cogo, R., et al. Measuring Agents in Production. *arXiv preprint arXiv:2512.04123*, 2025.

Patil, S.G., Zhang, T., Wang, X., and Gonzalez, J.E. Gorilla: Large Language Model Connected with Massive APIs. In *Advances in Neural Information Processing Systems*, 2024.

Patil, S.G., Yan, F., Mao, H., et al. The Berkeley Function Calling Leaderboard: From Tool Use to Agentic Evaluation. In *International Conference on Machine Learning*, 2025.

Qin, Y., Liang, S., Ye, Y., et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations*, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., and Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*, 2023.

Ritchie, L., Mehta, S., Heiner, N., Yu, M., and Chen, E. The Hierarchy of Agentic Capabilities: Evaluating Frontier Models on Realistic Enterprise Environments. *arXiv preprint arXiv:2601.09032*, 2026.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, Z., Wang, P., Zhu, Q., et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

Shi, T., Karpathy, A., Fan, L., Hernandez, J., and Liang, P. World of Bits: An Open-Domain Platform for Web-Based Agents. In *International Conference on Machine Learning*, 2017.

Trivedi, H., Khot, T., Hartmann, M., Manber, R., and Sabharwal, A. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In *Proceedings of ACL*, 2024.

Xie, T., Zhang, D., Chen, J., et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, 2024.

Xu, F.F., Song, Y., Li, B., et al. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. *arXiv preprint arXiv:2412.14161*, 2024.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yehudai, A., Schwartz, R., and Goldberg, Y. Survey on Evaluation of LLM-based Agents. *arXiv preprint arXiv:2503.16416*, 2025.

Yu, Q., Liu, G., Liu, L., et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhou, S., Xu, F.F., Zhu, H., et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, 2024.