ERP-AgentBench: A Live Benchmark for Autonomous Workflow Agents in Enterprise Resource Planning Systems

Alex Liu*
Korso, Inc.
United States
alex@korsoai.com

Daichi Hiraoka*
Korso, Inc.
United States
daichi@korsoai.com

Martin Pan*
Korso, Inc.
United States
martin@korsoai.com

Abstract

Autonomous language-model agents are increasingly evaluated in realistic web, desktop, and software-engineering environments, yet comparatively little evaluation infrastructure exists for enterprise resource planning (ERP) workflows, where agents must operate over mutable systems of record, respect business constraints, and avoid unsafe side effects. We present ERP-AgentBench, a live benchmark for evaluating autonomous workflow agents inside a seeded Odoo manufacturing environment. The benchmark uses a trigger–wait–check protocol: each task injects a realistic operational event into a live ERP instance, the agent under test acts through its normal integration path, and the harness evaluates the resulting ERP state against weighted rubrics. The current release contains 16 scenarios across five diagnostic capability levels: tool proficiency, execution planning, business judgment, recovery from unexpected state, and multi-issue orchestration. It includes a 718+ record seed dataset, resettable scenario manifests, pre/post state snapshots, surgical cleanup utilities, and rubric-based scoring that combines required state changes, absence of harmful side effects, and behavioral proxies for information gathering. By evaluating agents through actual ERP state transitions rather than transcript-only answers or toy API calls, ERP-AgentBench exposes failures central to enterprise deployment: wrong object grounding, unsafe writes, incomplete multi-step workflows, poor prioritization, and brittle exception handling. The contribution of this paper is the resource and its evaluation protocol; no baseline runs have yet been executed against the release, so the paper reports no agent scores. We will release the benchmark harness, seed dataset, scenario definitions, and documentation to support reproducible research on agents for high-stakes business operations; the artifact is not yet public, and the repository URL and archival DOI will be added in a future version of this preprint.

Keywords

autonomous agents, enterprise resource planning, agent evaluation, workflow agents, benchmark, resource paper, Odoo, manufacturing operations, side-effect safety

1 Introduction

Autonomous agents built on large language models are increasingly expected to act inside operational software. For enterprise systems, the problem extends beyond interaction. An ERP agent that confirms a purchase order, validates a receipt, posts an invoice, starts a manufacturing order, or creates a backup supplier order changes the system of record. These actions can affect inventory, financial commitments, supplier relationships, customer promises, and compliance records. Evaluation therefore has to measure whether an agent changes business state safely and correctly, not merely whether it produces a plausible answer.

Recent benchmarks have moved agent evaluation beyond static question answering. WebArena evaluates agents in realistic websites [6], OSWorld evaluates open-ended computer-use tasks [5], SWE-bench evaluates code changes against real software issues [3], and WorkArena evaluates agents on enterprise knowledge-work tasks in ServiceNow [1]. However, ERP workflows remain underrepresented. ERP tasks require grounding natural language in provider-specific business objects, chaining actions across related transactional records, reasoning about quantities and deadlines, handling exceptions, and avoiding spurious writes. These properties are central to enterprise deployment and are not captured by transcript-only or toy API benchmarks.

We introduce ERP-AgentBench, a resource for evaluating autonomous workflow agents in a live Odoo manufacturing environment. The benchmark follows a trigger–wait–check protocol (Figure 1). The harness creates a realistic operational event, the agent under test acts through its normal integration path, and the checker evaluates the resulting ERP state using scenario-specific weighted rubrics. The benchmark treats the agent as a black box; any model, planner, prompting strategy, memory system, or tool runtime can be tested as long as it can observe and act on the ERP instance.

The design follows three principles. *Stateful execution*: agents are evaluated by their effects on live ERP records, not by natural-language answers alone. *Diagnostic decomposition*: scenarios are organized into capability levels so that a failure indicates which layer of agent behavior broke, whether tool use, planning, judgment, recovery, or orchestration. *Side-effect awareness*: rubrics reward correct actions and penalize spurious or unsafe record mutations.

This paper makes four contributions.

(1) **A live ERP agent benchmark.** We contribute a benchmark for stateful, side-effect-aware evaluation of agents acting in a seeded Odoo manufacturing environment.

(2) **Diagnostic task taxonomy.** We organize 16 scenarios into five capability levels: tool proficiency, execution planning, business judgment, recovery, and orchestration.

(3) **Trigger–wait–check harness.** The harness comprises scripts for scenario triggering, pre/post snapshots, rubric scoring, manifest-based reset, and repeatable level-by-level evaluation.

(4) **Side-effect-aware scoring.** Rubrics reward correct state transitions while penalizing missing evidence gathering, incorrect records, and spurious mutations.

*All authors contributed equally as first authors.

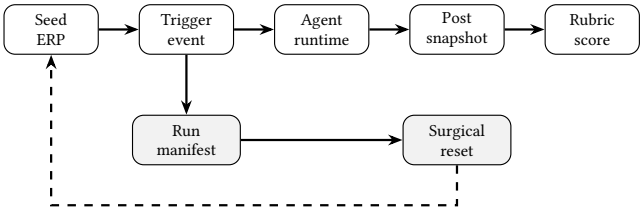

Figure 1: Trigger–wait–check protocol. The harness injects a realistic event into a seeded ERP instance, allows the black-box agent to act, evaluates post-state, and uses manifests for repeatable cleanup.

We report no agent scores in this paper. No baseline runs have been executed against the current release, so the contribution is the environment, the scenario suite, the rubrics, and the reporting protocol of Section 6, whose result table is deliberately left unfilled.

2 Related work

Interactive agent benchmarks. WebArena created realistic web environments for language-guided agents and showed that strong GPT-4-based agents remained far below human end-to-end task success [6]. OSWorld broadened interactive evaluation to real desktop environments and multi-application tasks [5]. These benchmarks motivate environment-based evaluation, but their tasks are not centered on ERP systems of record.

Outcome-based evaluation. SWE-bench evaluates whether agents can modify codebases to resolve real GitHub issues [3]. Its key methodological lesson is that success should be grounded in external outcome checks rather than self-reported completion. ERP-AgentBench applies this principle to business operations: success is measured by ERP state after the agent acts.

Enterprise agent evaluation. WorkArena evaluates web agents on ServiceNow knowledge-work tasks [1]. It is the closest existing benchmark family in spirit, but ERP workflows differ in their object model, transactional side effects, and operational constraints. An ERP benchmark must test inventory, purchasing, manufacturing, sales, invoicing, receipt handling, quality exceptions, and multi-issue triage.

ERP-AgentBench is therefore complementary rather than competing. Like WebArena and OSWorld, it evaluates agents through environment interaction; like SWE-bench, it grounds success in external state rather than self-reported completion; like WorkArena, it targets real enterprise software. Its distinguishing focus is the mutable ERP system of record, where a single action commits transactional business state and where business-object grounding and safe side-effect management, rather than navigation or patch generation, are the dominant sources of failure.

3 Resource design

3.1 Environment and black-box interface

ERP-AgentBench runs against a live Odoo instance configured as a manufacturing business. The seed dataset contains 718+ records across 18 seed phases, including products, raw materials, vendors, vendor prices, customers, sales orders, purchase orders, manufacturing orders, work orders, stock quantities, quality alerts, and operational messages. The benchmark does not prescribe an agent architecture. An agent may use webhooks, polling, scheduled sweeps, retrieval, planning, tool calling, or human-in-the-loop review. The only interface assumption is that the agent can observe relevant ERP events and mutate the ERP through its normal integration path. This allows the benchmark to compare end-to-end systems rather than isolated model calls.

Table 1: Benchmark structure. Scenarios are grouped by the capability layer they primarily exercise.

Level	Capability	N	Example scenario	Failure exposed
L1	Tool proficiency	4	Reorder raw material; confirm draft PO	Wrong model, field, or action
L2	Execution planning	4	Validate delivery, create and post invoice	Incomplete action chain
L3	Business judgment	3	Choose vendor under cost/urgency tradeoff	Acts without evidence
L4	Recovery	3	Handle overdue supplier or quality failure	Brittle exception handling
L5	Orchestration	2	Triage rush order, capacity, and overdue PO	Missed issues or spurious writes

3.2 Trigger–wait–check protocol

Each scenario has three phases. **Trigger** creates or modifies ERP records to simulate an operational event, such as a stock drop, overdue supplier message, quality failure, or customer status request. **Wait** leaves the agent runtime unconstrained; the evaluator invokes the check phase once the agent has finished acting. **Check** snapshots relevant ERP models, computes a diff against the pre-trigger state, and evaluates a weighted rubric. This protocol deliberately rejects transcript-only completion. If an agent claims that it created a purchase order but no corresponding purchase order exists, the scenario fails. If it completes the target action but creates unrelated records, side-effect checks penalize the run.

3.3 Reset and reproducibility

Every trigger writes a manifest of created and modified records. The reset utility deletes scenario-created records in dependency-safe order and restores modified records. This enables single-scenario iteration, level-by-level evaluation, and full-suite runs. For dependent scenarios, the harness can reset prerequisite artifacts automatically. Both trigger and reset support dry-run modes so researchers can inspect intended mutations before changing the ERP.

4 Benchmark suite

The first release contains 16 scenarios across five capability levels, summarized in Table 1 and enumerated scenario by scenario in Appendix A. The levels identify failure classes rather than ranking scenarios by difficulty.

Level 1: tool proficiency. Level 1 tests whether an agent can use ERP tools correctly for single-model or single-action tasks. Scenarios include reordering raw material when stock falls below a

Table 2: Representative rubric for a Level 1 raw-material reorder scenario.

Check	Weight
Purchase order exists for the target raw material	0.40
Purchase order uses the correct vendor	0.30
Ordered quantity is within allowed bounds	0.20
No unrelated purchase, sales, or manufacturing records created	0.10

threshold, checking stock availability without making writes, looking up vendor pricing, and confirming a draft purchase order.

Level 2: execution planning. Level 2 tests clear multi-step workflows. Scenarios include fulfilling a ready-to-ship sales order by validating delivery and creating/posting an invoice, processing a receipt and updating stock, checking material availability after a sales order triggers manufacturing, and completing remaining work orders.

Level 3: business judgment. Level 3 tests situations with multiple defensible outcomes. For example, an agent may need to choose between a cheaper slower vendor and a more expensive faster vendor under urgent manufacturing demand. Rubrics check that the agent gathered relevant evidence before acting.

Level 4: recovery. Level 4 tests unexpected or degraded state, including overdue suppliers, quality failures, and partial receipt mismatches. The agent must diagnose the situation, inspect alternatives or downstream impact, and either take a safe corrective action or escalate.

Level 5: orchestration. Level 5 tests simultaneous operational issues. Scenarios combine rush orders, manufacturing conflicts, overdue suppliers, invoice handling, payment follow-up, receipts, and manufacturing checks. These scenarios measure prioritization and coverage across multiple concerns.

5 Scoring

Each scenario defines a weighted rubric whose checks sum to 1.0, as in the representative Level 1 rubric of Table 2. A scenario passes when its score is at least 0.70. The harness reports each check, its weight, and its pass/fail status. The current implementation uses four check families:

(1) **State checks** verify that target ERP records reach expected values, such as a purchase order state or delivery state.
(2) **Existence checks** verify that required records were created with correct entities, quantities, and monetary values.
(3) **Absence checks** verify that the agent did not create unrelated purchase orders, sales orders, manufacturing orders, invoices, or other harmful artifacts.
(4) **Behavioral proxies** verify information gathering indirectly through ERP-visible traces, messages, activities, or state changes.

The scoring design is intentionally operational. An agent receives no credit for merely explaining what it would do. It also cannot maximize its score by making broad speculative writes, because absence checks penalize unrelated mutations. For judgment and recovery tasks, behavioral proxies make evidence gathering visible without requiring access to private chain-of-thought or proprietary agent traces. Per-check reporting also makes failures actionable: a failed fulfillment scenario may show that delivery validation succeeded and invoice creation succeeded, but that invoice posting failed or used the wrong customer.

Table 3: Reporting template, not results. Recommended reporting format for baseline evaluation. No baseline runs have been performed: this table reports no results. Every result cell is intentionally unfilled and is rendered as ·, meaning *not reported*; a cell may be populated only from a completed benchmark run, and no value here is estimated, imputed, or otherwise inferred.

Reporting template: no baseline runs have been performed						
Agent	**L1**	**L2**	**L3**	**L4**	**L5**	**Overall**
Reference agent	·	·	·	·	·	·
Alternative agent	·	·	·	·	·	·
Human/scripted oracle	·	·	·	·	·	·
Spurious-write rate	· for each agent above					
Every cell above is intentionally unfilled						

· **denotes *not reported***: the run that would produce the cell has not been performed. This table is a specification of the reporting format, not a record of any measurement.

6 Baseline reporting protocol

No baseline runs have been executed against the current release, so this section specifies how results should be reported rather than reporting any. A complete benchmark report should include end-to-end score by scenario and by level, number of ERP reads and writes, escalation count, trigger-to-first-action latency, trigger-to-final-state latency, spurious-write rate, model configuration, and cost where available. Table 3 is that reporting template: it fixes the rows, columns, and units of a benchmark report so that independent evaluations remain comparable, and it carries no results.

The level decomposition is useful even before comparing many models. An agent that performs well on L1 and L2 but poorly on L3 likely has adequate ERP tool use but weak evidence gathering or prioritization. An agent that performs well on L3 but poorly on L4 may instead require better exception handling. An agent that completes target actions but has a high spurious-write rate is unsafe for deployment even if its nominal pass rate is high.

7 Availability and extension

The release artifact, which is not yet public, contains seed scripts, scenario definitions, CLI commands for `list`, `run`, `check`, and `reset` (Appendix B), rubric implementations, scoring output, setup documentation, and versioned benchmark metadata.

Status of the release. At the time of writing, the artifact is not yet public. We will release the benchmark harness, the seed dataset, the scenario definitions, and the setup documentation through a public repository, and we will archive that release in a deposit that mints a citable DOI. This version of the paper prints no repository URL and no archival identifier, because neither exists yet; publishing an address that does not resolve would be worse than publishing none. The repository URL and the archival DOI will be added in

a future version of this preprint, once the repository is public and the deposit is minted. The license for the harness code and the license for the seed dataset have not been settled, so this paper names no license; both will be specified at release and recorded in the repository. Readers should treat the resource as described here but not yet obtainable.

Reproducibility also requires pinning the environment. Exact version pinning will accompany the release: the released artifact will record the Odoo version, the Python version, the dependency versions, and the seed commit that a reported run was produced against, and a future version of this preprint will quote those pins. No version numbers are stated here, because the release that would fix them has not been made.

The seed dataset is synthetic manufacturing data; it contains no customer credentials, no customer communications, and no proprietary or production ERP data, and none will be added to the public artifact. The deposit will be documented in the style of a datasheet for datasets [2], recording provenance, intended use, composition of the seed dataset, and known limitations. It will be organized to follow the FAIR principles [4]: the archival deposit will be citable by DOI, the scenario manifests and rubrics will be machine-readable, and the harness will be versioned alongside a changelog so that any score can be tied to an exact benchmark revision.

Researchers can extend the benchmark in three ways. First, they can add new Odoo scenarios using the same trigger–wait–check abstraction. Second, they can add additional workflow domains such as accounting close, service operations, or distribution. Third, they can port the design to other ERPs, including SAP Business One, SAP S/4HANA, Microsoft Dynamics 365 Business Central, and NetSuite.

8 Limitations and ethics

ERP-AgentBench should not be interpreted as proof that an agent is safe for production ERP deployment. It measures behavior on a finite set of synthetic tasks in an isolated Odoo environment. The current release focuses on manufacturing workflows, so results may not transfer directly to other industries or ERP providers. Some behavioral proxies are indirect because black-box agents may not expose internal reasoning traces. The 16-scenario suite is diagnostic but not exhaustive; future versions should expand coverage for accounting controls, permissions, concurrent users, long-running workflows, and adversarial or ambiguous instructions.

The most significant limitation is that the release has not been characterized empirically. Because no baseline runs have been executed, this paper offers no evidence about how difficult the 16 scenarios are in practice, whether the 0.70 pass threshold separates agents usefully, whether the five capability levels are ordered by observed difficulty, or whether the behavioral proxies correlate with the evidence gathering they are intended to detect. Those are properties of the resource, not of any agent, and they can be established only by running agents against the suite. Until that is done, the levels and weights presented here are design commitments rather than validated instruments.

The benchmark is designed to reduce evaluation risk. It uses synthetic data, resettable environments, and side-effect checks. Researchers should run agents only against isolated ERP instances. Any extension using real business data should follow institutional review, privacy, and data-governance requirements. Because ERP operations can affect financial and operational records, human review remains necessary before deploying agents in real systems of record.

9 Conclusion

ERP-AgentBench provides a live, stateful, side-effect-aware benchmark for evaluating autonomous workflow agents in ERP systems. By combining a seeded manufacturing environment, resettable triggers, pre/post snapshots, weighted rubrics, and diagnostic capability levels, it supports reproducible evaluation of agents that must act inside mutable enterprise systems of record. The broader claim is that enterprise agent evaluation should measure operational state changes, evidence gathering, exception handling, and harmful side effects rather than natural-language task completion alone. The resource fills a gap between general web-agent benchmarks and the requirements of real operational automation, and it gives the community a way to measure progress toward safe enterprise agents once baseline runs exist.

A Scenario inventory

Table 4 lists every scenario in the current release with its identifier and a one-line description.

Table 4: All 16 scenarios in the current release.

ID	Scenario
L1-1	Reorder raw material after stock drops below minimum
L1-2	Check stock levels for a draft sales order without writes
L1-3	Look up vendor pricing for a draft purchase order
L1-4	Confirm a draft purchase order
L2-1	Fulfill ready-to-ship sales order and post invoice
L2-2	Process receipt and update stock
L2-3	Check material availability after sales-order-triggered manufacturing
L2-4	Complete remaining work orders and close manufacturing order
L3-1	Select vendor under cost-versus-urgency tradeoff
L3-2	Decide partial ship versus wait after customer status request
L3-3	Prioritize manufacturing order under competing demand
L4-1	Handle overdue supplier
L4-2	Recover from quality failure
L4-3	Process partial receipt mismatch and downstream impact
L5-1	Triage rush order, urgent manufacturing, and overdue supplier
L5-2	End-of-period close across invoices, receipts, payment, and manufacturing

B CLI workflow

The harness exposes four primary commands. `list` shows available scenarios and status. `run` triggers a scenario, level, or full suite and stores a pre-snapshot plus run manifest. `check` evaluates the current ERP state against the relevant rubrics. `reset` performs surgical cleanup for a scenario, level, or full suite. The recommended workflow is to reset, run one level, wait for the agent, check outcomes, and then continue to the next level.

GenAI Usage Disclosure

Large language models were used for limited drafting, editing, and format-conversion assistance. LLMs may also be components of agents evaluated by the benchmark, but the benchmark resource, scenario definitions, scoring code, and final claims were reviewed and validated by the authors. No confidential customer data was provided to the AI system.

References

[1] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks?. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, 11642–11662. arXiv:2403.07718 [cs.LG] https://proceedings.mlr.press/v235/drouin24a.html

[2] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for Datasets. *Commun. ACM* 64, 12 (2021), 86–92. doi:10.1145/3458723

[3] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *arXiv preprint arXiv:2310.06770* (2023). arXiv:2310.06770 [cs.CL] https://arxiv.org/abs/2310.06770 Also appeared at ICLR 2024.

[4] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, et al. 2016. The FAIR Guiding Principles for Scientific Data Management and Stewardship. *Scientific Data* 3 (2016), 160018. doi:10.1038/sdata.2016.18

[5] 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. 2024. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972* (2024). arXiv:2404.07972 [cs.AI] https://arxiv.org/abs/2404.07972 Also appeared at NeurIPS 2024 Datasets and Benchmarks Track.

[6] 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. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854* (2023). arXiv:2307.13854 [cs.AI] https://arxiv.org/abs/2307.13854 Also appeared at ICLR 2024.