Evaluating Enterprise AI Operating Systems: A Research Protocol for Execution Correctness, Control Effectiveness, and Operational Outcomes

November 2026

Abstract

An enterprise AI operating system should be evaluated through the business states it creates, the authority it respects, and the obligations it leaves unresolved. A fluent response or successful demonstration provides limited evidence for those requirements. This paper proposes a research protocol combining representative workflows, controlled perturbations, independent outcome verification, and measurement of operating costs. It draws on language model and agent evaluation research while extending the unit of assessment to complete enterprise commitments. The protocol covers context validity, concurrent execution, permission enforcement, security boundaries, recovery, and human review. It also defines how deployment scope should relate to evidence, uncertainty, and the consequences of failure. MatruOS is an intended application context, with no experimental results reported. The contribution is an auditable evaluation design that supports bounded deployment decisions and identifies where further evidence is necessary before additional responsibility is delegated.

Keywords: enterprise AI evaluation, execution correctness, control effectiveness, operational outcomes, reproducibility, deployment evidence.

1. What an evaluation needs to establish

An agent can produce a correct explanation while leaving the underlying order unchanged. It can also complete the requested change while violating an approval condition. A single assessment of response quality does not distinguish those outcomes.

An enterprise AI operating system interacts with records, people, and external services. Its evaluation must therefore inspect the intended business state, the path by which that state was reached, and any remaining obligations. The relevant question is whether a defined responsibility can be delegated under stated operating conditions.

This paper asks how evidence should be collected and interpreted before expanding an AI system’s authority. The scope is an enterprise workflow rather than a foundation model in isolation. Model capability, application controls, integration behaviour, and human review all contribute to the observed outcome.

The protocol is intended for research and deployment assessment. It does not provide universal pass thresholds or certify a platform. Acceptance depends on the consequence of failure, the available recovery process, and the organisation’s capacity to supervise the selected scope.

2. Research foundations

τ-bench evaluates agents interacting with users and tools under domain policies, including verification of the resulting database state and consistency across repeated trials. This provides a foundation for evaluating observed effects rather than relying only on an agent’s narrative response. [1]

Holistic Evaluation of Language Models organises evaluation across scenarios and multiple metrics, with an emphasis on transparency and reproducible analysis. Its approach supports making coverage and omitted dimensions explicit rather than treating one aggregate result as a complete account of capability. [2]

Rabanser and colleagues examine agent reliability through consistency, robustness, predictability, and safety. Their work motivates testing variation and failure behaviour alongside ordinary task success. [3]

Kapoor and colleagues discuss cost, benchmark design, holdout data, and reproducibility in agent evaluation. These concerns are relevant when a system is tuned repeatedly against a small internal demonstration set and then presented as ready for broader use. [4]

AgentDojo supplies a framework for examining agent behaviour when tools return untrusted content containing adversarial instructions. It supports including security outcomes within evaluation while keeping them distinct from ordinary task completion. [5]

The contribution of this paper is an enterprise protocol that combines these perspectives around complete business commitments. It is a proposed method with no new benchmark results. The worked examples are hypothetical and serve to define measurements and decision rules.

3. Define the responsibility before the test

3.1 The evaluation contract

Each workflow should have an evaluation contract stating its trigger, eligible inputs, permitted actions, required approvals, and completion conditions. The contract should also define valid rejection and escalation. A system should not be penalised for declining a request that lies outside its authority.

For an order workflow, the contract might require correct customer identity, approved prices, valid stock reservation, and a durable order record. External confirmation may be a separate obligation. If the test ends before that communication is verified, the result should state the remaining scope.

The contract should identify what the system may assume. If customer identity is supplied as verified input, the test does not evaluate identity resolution. If the payment provider is simulated, the result does not establish reliability against the live provider. These boundaries are essential to interpreting the evidence.

3.2 Reference outcomes

A reference outcome should be determined from business rules and the initial state. It may include several acceptable paths where the business permits alternatives. The evaluator should not require one exact wording or tool sequence when several paths satisfy the same obligations.

Reference outcomes need review by people with the relevant domain knowledge. Disagreement should be recorded and resolved or retained as ambiguity. A model should not be judged against an undocumented preference that becomes apparent only after its response is inspected.

Some outcomes are inherently conditional. A request with missing authority may correctly end in a clarification or approval request. The reference should specify what information is needed and which effects must remain prohibited until it is supplied.

3.3 Consequence categories

Cases should be grouped by consequence as well as task type. A wrong draft, an unauthorised commitment, and a disclosure of protected information have different implications. Combining them into one error count can conceal a serious control failure inside many successful low consequence tasks.

The organisation should define which categories prevent deployment, which require manual review, and which can be tolerated within a correction process. These decisions should be made before results are known. Retrospective adjustment of thresholds can make a weak result appear acceptable.

4. Constructing a representative test set

The test set should include ordinary cases, difficult valid cases, prohibited requests, incomplete information, and operational interruptions. Sampling should reflect the intended deployment population where operational data is available. Synthetic cases can fill known gaps, but their origin should be explicit.

Coverage should include customer types, product variants, transaction values, languages, and channels relevant to the workflow. A multilingual deployment needs tests of the structured business interpretation, not only language fluency. Similar wording can represent different quantities or obligations in different contexts.

Temporal variation is also important. Policies can change after retrieval, approvals can expire, and external updates can arrive late. Tests should specify the timing of those events and the expected revalidation. A static set of records cannot reveal every failure created by changing state.

Concurrent cases should compete for shared resources. Two individually valid orders may jointly exceed stock or credit. The evaluator should inspect the combined outcome, not grade each conversation independently against the same initial snapshot.

The test set should include work that remains unresolved. An uncertain payment or failed cancellation must have an expected containment and handoff outcome. Treating such cases as outside the benchmark can exclude the operational work that most affects deployment suitability.

5. Independent observation of outcomes

The evaluator should inspect authoritative records and external effects through a controlled observation channel. The agent’s statement that an order was created is not sufficient. The test should verify the actual record, relevant constraints, and any associated messages or payments.

Observation should include intermediate states when they matter. A workflow may end in an acceptable state after briefly exposing protected data or issuing a duplicate request that was later corrected. A final snapshot alone can miss those events. Tool traces and controlled external endpoints can provide additional evidence.

The observation process should avoid altering the behaviour being measured. If the evaluator repairs records or supplies hints during a run, that intervention must be recorded. Human assistance should be part of the declared condition rather than an unreported benefit available only during testing.

For some qualitative outputs, a reviewer may assess appropriateness or clarity. Reviewers should use a documented rubric and, where feasible, be unaware of the configuration that produced the output. Disagreement should be examined rather than hidden through a single unreviewed score.

6. Measures that should remain separate

6.1 Execution correctness

Execution correctness concerns whether the resulting business state satisfies the contract. Report the number of eligible cases, verified completions, valid rejections, valid escalations, incorrect effects, and unresolved cases. The denominator should remain visible for every reported rate.

A task can be partly correct. The record should identify which obligation failed rather than assigning only a broad success label. This supports investigation and prevents a correct document from concealing an incorrect stock reservation or missing approval.

6.2 Control effectiveness

Control measures include unauthorised actions attempted, attempts blocked, and protected effects that occurred. A blocked attempt is evidence of containment but may still indicate a planning or interpretation defect. These outcomes should be reported separately.

Approval validity should be tested against the exact action executed. Tenant separation should be checked across retrieval, storage, exports, and background jobs. Revocation should be tested while work is queued or in progress. A permission check in one interface does not establish control across the full application.

6.3 Reliability and recovery

Repeat cases under controlled variation to assess consistency. Change wording, document order, timing, and service availability where those variations are relevant. The goal is to identify whether the system behaves predictably within the intended operating conditions.

Recovery measures should include duplicate effects, time to verified resolution, age of unresolved work, and manual effort. A test should continue long enough to observe delayed responses and retries. Closing measurement when the agent stops speaking can omit consequential effects.

6.4 Operational cost and human effort

Record model usage, infrastructure, external charges, review time, correction, and maintenance required for the test. Cost per verified completion should include resources consumed by failed and unresolved attempts. Separate implementation investment from recurring operation.

Human review should be measured for both workload and effectiveness. A system may appear reliable because a skilled reviewer corrects every proposal. That is a valid assisted configuration if reported accurately, but it does not establish autonomous execution capability.

7. Experimental design and comparison

The protocol should define the configuration before execution: model version, prompts, tools, policies, retrieval settings, and environment. Changes during testing should create a new condition or be documented as an intervention. Otherwise, the reported result may combine incompatible systems.

A controlled comparison should isolate the design question. To test approval binding, keep the model and workflow constant while changing the binding mechanism in a safe environment. To test retrieval validity controls, use the same cases and model while changing the applicability checks. Changing several components at once limits causal interpretation.

Development cases should be separated from held out evaluation cases. Repeatedly tuning against every test can produce a system adapted to those examples. The final evaluation set should include unseen combinations and realistic variations within the intended scope.

Ablation can help identify the contribution of a control, but removing safeguards should occur only in an isolated test environment with synthetic effects. The purpose is to understand the mechanism, not to expose real customers or business records to an intentionally weakened system.

The report should include failed runs, excluded cases, and reasons for exclusion. If a test harness fails independently of the agent, that should be recorded and rerun under a documented rule. Selective reruns of disappointing agent outcomes would distort the result.

8. Interpreting small samples and uncertainty

A finite test cannot establish that a rare failure will never occur. Under an illustrative model of independent trials with a constant failure probability, observing zero failures in 100 trials gives an approximate one-sided 95 percent upper bound of 2.95 percent for that probability. This follows from solving the probability of observing no failures as 0.05. The calculation is not a claim about any platform.

The assumptions matter. Repeated runs of nearly identical cases may be correlated and may not represent the deployment population. A shared integration defect can affect many cases together. A confidence calculation based on independence would then understate uncertainty.

The evaluation should report uncertainty appropriate to its design and avoid presenting one decimal place as evidence of precision. Where sample sizes are small, the report should examine individual cases and failure causes, with explicit limits on any ranking. Consequence severity should also influence how much evidence is required.

Zero observed security incidents in benign tests provides no direct evidence about adversarial robustness. Similarly, no observed recovery failure during uninterrupted operation says little about outage handling. Coverage and statistical uncertainty are different limitations, and both should be stated.

9. A worked deployment decision

Assume a synthetic evaluation contains 120 order cases: 70 ordinary requests, 20 cases requiring approval, 15 incomplete or conflicting requests, and 15 interrupted operations. These counts are an example of test composition, not an empirical result or a recommended universal sample size.

Suppose the system handles ordinary orders consistently but executes one changed order under an approval tied to an earlier version. That outcome is a control defect even if the overall completion rate appears acceptable. The deployment decision should address the approval mechanism before allowing that responsibility.

Suppose another configuration blocks the invalid approval but refers nearly every ordinary order to a manager. The control result is useful, but review capacity may limit operational suitability. The decision could permit draft preparation while withholding unattended execution, pending further work on the valid case handling.

Suppose interrupted operations produce truthful pending status and complete recovery after service restoration. The report should inspect how long resolution takes and whether duplicate effects occur. A correct initial message is insufficient to establish recovery.

This example shows how evidence can support a bounded decision rather than a binary declaration that the entire AI operating system is ready. Each permitted responsibility should be associated with the conditions and evidence supporting it.

10. Deployment stages and continuing evaluation

A practical sequence begins with isolated tests, proceeds to observation against representative work, and then permits assisted or bounded execution where evidence supports it. Movement between stages should be recorded by the responsible business and technical owners.

Production monitoring should use the same outcome definitions where feasible. It should track unresolved work, overrides, control denials, and corrections. Monitoring only usage or model latency cannot establish that business obligations remain correct.

Changes to models, prompts, tools, policies, or integrations should trigger targeted regression tests. The scope of retesting should reflect the changed component and its dependencies. A new model may affect interpretation, while a connector change may affect identity, ordering, or external status semantics.

The organisation should also define conditions for reducing delegated scope. A rising unresolved queue, repeated invalid approvals, or loss of a necessary data source may justify returning to assisted operation. Continuing evaluation is useful only if it can change the deployment decision.

11. Application to MatruOS and research outputs

For MatruOS, the protocol provides a structure for evaluating specific workflows across its business modules. The first research artefact should be a versioned evaluation contract and test set, followed by execution traces, outcome labels, and an analysis of failures. Public reporting can use synthetic or appropriately de-identified examples while protecting customer information.

The evaluation should state the author’s relationship to the platform and distinguish internal testing from independent assessment. External reviewers can inspect the protocol and results where access and confidentiality permit. Independence should be described accurately rather than implied by the existence of a formal report.

A reproducible research package should include configuration identifiers, case definitions, expected outcomes, metric calculations, and known exclusions. Where proprietary components prevent full reproduction, the report should identify that limitation and provide sufficient detail for meaningful scrutiny.

This paper does not report that such a package has been executed. It defines the evidence needed for future empirical work and provides a common method for testing the architectural propositions developed across the research series.

12. Limitations and conclusion

The protocol depends on clear business rules, representative cases, and access to authoritative outcomes. Some consequences emerge only after long periods or through human relationships that are difficult to simulate. These limitations should influence deployment scope and follow up research.

A benchmark can support a decision within its coverage. It cannot replace continuing operational review or establish reliability under untested conditions. The appropriate interpretation is a statement about a defined system, task population, environment, and observation period.

Enterprise AI evaluation should connect execution correctness, control effectiveness, recovery, cost, and human effort. Keeping those dimensions visible prevents a single success metric from concealing consequential weaknesses.

The proposed protocol supports evidence-based delegation by identifying what has been tested, what remains uncertain, and which responsibilities the evidence can reasonably support. For an AI operating system, that is the foundation for responsible expansion from useful assistance to dependable business execution.

Author affiliation and disclosure

The author is affiliated with Code Colonies Private Limited, Ahmedabad, and leads the MatruOS team. This paper proposes an evaluation protocol. It reports no completed benchmark or independent product assessment.

References

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

https://arxiv.org/abs/2406.12045

[2] Liang, P., et al. (2023). Holistic Evaluation of Language Models. arXiv:2211.09110, version 2.

https://arxiv.org/abs/2211.09110v2

[3] Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S., and Narayanan, A. (2026). Towards a Science of AI Agent Reliability. Proceedings of the 43rd International Conference on Machine Learning.

https://proceedings.mlr.press/v306/rabanser26a.html

[4] Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., and Narayanan, A. (2024). AI Agents That Matter. arXiv:2407.01502.

https://arxiv.org/abs/2407.01502

[5] Debenedetti, E., et al. (2024). AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. arXiv:2406.13352, version 3.

https://arxiv.org/abs/2406.13352v3