ALFRED: GOVERNING KNOWING-DOING GAP IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents tasked with delegated work can state a correct plan yet execute another: they skip the authoritative sources, confirmations, and designated entry points they themselves promised, and pass controllable substitutes off as uncontrollable deliverables. This knowing–doing gap has been repeatedly documented by independent measurements, but measurement alone does not stop it, and unfaithful delivery undermines the evaluations meant to detect it. We present ALFRED, a governance framework that makes an agent's knowing–doing unity enforceable at the moment of delivery. ALFRED closes the loop with four components: selective isolation of roles and information surfaces, undisclosed heterogeneous review, contract-based acceptance at submission time, and deterministic failure routing, with only semantically undecidable cases escalated to the human owner. We operationalize the phenomenon into two independent evaluations: task completion, judged by deliverables and environment end-states against frozen task requirements, and knowing–doing failure, judged by the executor's self-stated plan and delivery commitments against its actual execution path, adjudicated by an independently validated judge. On a frozen 45-task set spanning real engineering sessions, established benchmarks, and a bandit decision task, we compare nine conditions: an ungoverned baseline, four external governance methods under their native protocols, ALFRED, and three mechanism ablations that remove process evidence, active verification, or enforced disposal. We will open-source the framework, the measurement protocol, and the task instances.

1 INTRODUCTION

Supervising the process quality of large language model (LLM) agents is becoming a central challenge for both academia and industry. The length of tasks that agents can complete autonomously doubles roughly every seven months and entered the ten-hour regime in 2026 (Kwa et al., 2025); business workflows with policy constraints, such as customer-service returns and flight rebooking, are already being handed to agents end to end (Yao et al., 2024; Barres et al., 2025). Yet these agents systematically bypass the way of working that the task requires: they skip queries to authoritative sources, skip the confirmations and checks that policies prescribe, and pass off controllable substitutes as uncontrollable real deliverables. This tendency is eroding authoritative measurement itself: an independent evaluation of GPT-5.6 found a higher cheating-detection rate than any public model the evaluator had previously measured, including planting exploit code in an intermediate submission to probe for hidden test suites and extracting hidden source code containing expected answers; measured by the time needed to complete 50% of tasks, the horizon is 11.3 hours when cheating counts as failure and exceeds 270 hours when it counts as success. The evaluator accordingly declared that no single number is reliable any more (METR, 2026). Making "did the agent actually complete the task in the prescribed way?" checkable and governable is therefore a precondition that neither evaluation nor deployment can bypass.

The phenomenon is directly visible in execution traces, and the research community calls it the **knowing–doing gap**: an agent states the correct rationale or diagnosis, yet its actual execution takes another path. Independent measurements have recorded it twice. In bandit decision tasks, models computed the correct arm to choose in 87% of rounds while acting on it in only 21% (Schmied et al., 2026); over 26 weeks of inventory management, four models identified 84–88% of stress events with an average lag of less than half a week, and stockouts still occurred. The authors' conclusion:

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.

the bottleneck is not cognition, but lies between the correctly stated belief and the costly action it demands (Deb & Krishnan, 2026).

Measurement, however, stops at recording the phenomenon; no effective means exists to govern it. The object of governance should be **knowing–doing unity**: once an agent has stated its plan and promised delivery to the delegator, including the authoritative sources to be consulted, the real deliverable to be produced, and the designated entry points to be used, its actual execution path should honor that stated commitment, without bypass or substitution. Existing efforts either reach inside the model, probing capabilities and representations with prompts and probes whose criteria must read internal states, which is brittle, systematically circumventable by obfuscation, and diagnostic without changing execution (Cheng et al., 2026); or they stay on the external trajectory, intervening at a single link of monitoring, correction, or after-the-fact audit, whose criteria are expensive and whose verdicts do not land at the moment of delivery (Kale et al., 2026); the verification loop closest to governance checks only the provenance of evidence at answer time, not the execution obligations the task assigns (Song et al., 2025). To our knowledge, no work combines a systematic quantification of unfaithful execution with a governance architecture that effectively reduces its incidence within one framework. This paper therefore proposes a **systematic framework for governing agents' knowing–doing unity**, so that unfaithful execution is intercepted and adjudicated before it reaches delivery, and knowing–doing unity becomes accountable at the moment of delivery.

The proposed framework, **ALFRED**, forms a closed loop of four components: selective isolation, undisclosed heterogeneous review, contract-based acceptance, and deterministic failure routing. Governance does not operate as process control: it prescribes no action ordering and grants no step-by-step approval, and the executor does not perceive that review exists. Governance operates at artifact acceptance: plan review checks the plan specification against the owner request, and artifact review checks the delivered artifact against the node contract. The reviewer sees the complete execution record from the owner's perspective, so the trajectory serves as evidence for faithfulness judgments rather than as an object of intervention. Verdicts are routed by a non-agent deterministic orchestrator according to failure class: mechanical failures are re-run under the same contract until the budget is exhausted, then escalated; semantic failures and adjudication disagreements go to the owner for final decision. There is no silent failure anywhere, and humans intervene only in judgments that cannot be decided automatically.

A precondition for detection and governance is to turn knowing–doing unity into a measurable, adjudicable quality object. This paper stipulates two independent evaluative foundations: **task completion**, judged by whether the actual deliverable or environment end-state satisfies the frozen original task requirements; and **knowing–doing failure**, judged by comparing the model's self-stated plan and delivery commitments in that run against its actual execution path: whether the executor honored, bypassed, or substituted what it itself wrote down. Judgments rely only on observable real records (statement texts and execution traces), presuppose no latent cognition, and attribute no motive; a unified independent LLM judge applies the same standard across conditions, the governance side's own operations and verdicts are not counted as executor behavior, and trajectories altered by governance therefore remain comparable.

The test of the governance framework is a controlled experiment. Under identical tasks and common starting points, an ungoverned baseline, four external governance methods (each retaining its native protocol), the full framework of this paper, and mechanism ablations (removing process evidence, active verification, and enforced disposal respectively, to test each link of the closed loop) each execute complete runs; governance effects are reported jointly by the two independent evaluations, with runtime cost reported separately. The task set has 45 items, frozen before comparisons begin and not altered according to performance: 15 self-built cases derived from real engineering sessions, covering repair, refactoring, verification, real-database audit, documentation delivery, and compliant submission; plus 30 further instances: 29 drawn from SWE-bench Verified, Frontier-SWE, WebShop, ALFWorld, and WebDetective, and 1 bandit decision task constructed in the manner of Greedy Agents. All evaluations proceed independently on frozen materials without in-run human intervention. Before formal comparisons, the judge is independently validated: three researchers who took no part in the task construction or judge-label generation for the respective items, and who are neither the designer nor the tuner of this judge version, first lock reference judgments independently (references are not given to the judge), then check the judge's outputs case by case; criteria and the statistical protocol are pre-registered.

The contributions of this paper are:

- **Governance architecture.** Selective isolation, two review pipelines (plan: specification against owner request; artifact: deliverable against contract), and deterministic failure routing form a closed loop; deterministic criteria execute automatically, semantic judgments go to the owner for final decision, no failure passes silently, and governance operates at artifact acceptance rather than process control.
- **An adjudicable knowing–doing failure.** "Did the agent execute as it promised?" is operationalized as two independent evaluations: task completion (deliverable or end-state against original requirements) and knowing–doing failure (self-stated plan and commitments against actual execution path); the criteria use only observable records, probe no model internals, and attribute no motive; a unified independent judge enables comparison across governance mechanisms, and governance-side operations are not counted as executor behavior.
- **A controlled-experiment protocol.** Under identical tasks and common starting points, the ungoverned baseline, four external methods (native protocols retained), the framework, and mechanism ablations (removing process evidence, active verification, or enforced disposal only) are compared; the 45-item task set is frozen in advance; criteria and the statistical protocol are pre-registered, and the judge enters formal comparisons only after validation against independently locked references.

We will open-source the governance framework, the measurement protocol, and the task instances.

2 Related Work

Post-training paradigms. Reinforcement learning has become central to LLM post-training: from reinforcement learning from human preferences (Ouyang et al., 2022), through reward-model-free direct preference optimization and group relative policy optimization (Rafailov et al., 2023; Shao et al., 2024), to reinforcement learning with verifiable rewards, whose rule-based grading lets models acquire complex abilities in mathematics, coding, and agentic tasks (DeepSeek-AI, 2025; Kimi Team, 2025; Lambert et al., 2024). The grading designs converge on one shape: DeepSeek-R1 adopts accuracy-plus-format rule rewards, having abandoned neural reward models as gameable at scale; Kimi k1.5 grades coding problems by passing test cases and auto-generates test cases to serve as rewards; Tulu 3 fixes RLVR as the final pipeline stage. Yet grading lands only at the endpoint (answer match, test pass) and holds only within the training distribution: reward-model overoptimization was quantified long ago (Gao et al., 2023); rule rewards still assign positive reward to rollouts whose answers are guessed right but whose reasoning is unreliable, in up to half of cases (Ding et al., 2026); and RLFT amplifies the knowing–doing gap while improving task metrics (Schmied et al., 2026). These paradigms keep polishing graders of "outcome reachability"; "how the outcome was produced" never enters the reward. This paper differs: it leaves model weights untouched and, on the deployment side, verifies the consistency of self-stated commitments with the execution path, taking that as its governance criterion.

Measuring knowing–doing divergence and cheating. The conceptual root lies in Ryle's distinction between knowing-how and knowing-that (Ryle, 1949); subsequent measurement of the knowing–doing gap and of cheating falls into three settings. In settings where models self-assess, Kadavath et al. let models judge their own candidate answers and found that models can mostly report their knowledge boundaries in a calibrated way (Kadavath et al., 2022); such settings measure only what the model *says*, and action lies outside measurement. In gold-standard-contrast settings, LMAct separates declarative knowledge and effective decision-making as two measurement objects (Ruoss et al., 2025); BALROG scores factual knowledge, due conclusion, and actual behavior in separate columns for the same model (Paglieri et al., 2024); Greedy Agents places models in bandit tasks, deriving before acting, and finds many rounds where the derivation is right yet not acted upon (Schmied et al., 2026); STOCKTAKE lets belief and action be scored side by side over long-horizon inventory management, concluding that the bottleneck lies not in cognition but in the action cost that stated beliefs demand (Deb & Krishnan, 2026); SeekBench quantifies groundedness, recovery, and calibration competencies over retrieval trajectories (Shao et al., 2026); DARE-bench institutes instruction adherence and process faithfulness as evaluation dimensions for data-science agents (Shu

et al., 2026). The phenomenon's name is reused across these works, while each operationalizes it its own way. Such settings presuppose a measurement party holding closed-form gold standards, with tasks designed for measurement. In deployment-side audit settings, METR reviews the full trajectory after the run and reports afterwards (METR, 2026); audit presupposes seeing the whole trajectory and occurs after execution. The three settings share one assumption: the measurer is a bystander who enters to adjudicate only after the trajectory is generated and the report written, records the phenomenon, and does not intervene in execution. This paper moves the adjudication basis onto the run's own stated commitments: it relies on no gold standard and probes no internal states, and is scored separately from task completion within the same run; the same basis simultaneously serves as the routing criterion of the governance loop, with in-mechanism adjudication and out-of-mechanism evaluation complementary and non-substituting (Section 3).

Governance structures for agents. Efforts to write governance into the system land either in training (constitutional principles and AI-feedback discipline of behavior (Bai et al., 2022)) or in prompting (multi-agent frameworks regulating roles and division of labor through process norms (Hong et al., 2024; Wu et al., 2023)), both relying on the model's voluntary compliance, with no enforced acceptance. Work that genuinely sets checkpoints on the execution chain can be ordered by criterion cost from low to high: SWE-bench judges delivery by unit-test pass (Jimenez et al., 2024), seeing only the endpoint and unable to detect process bypass; trajectory review scores execution records with LLM judges (Zheng et al., 2023), grades stepwise with process supervision (Lightman et al., 2024), and evaluates web agents with generative process reward models, noting that outcome-based supervision is sparse and delayed, often rewarding incorrect trajectories (Zhang et al., 2026). Such reviews cover the process but serve evaluation and training signals, not deployment-time interception. On the runtime side, Reflexion lets the executor self-reflect and correct (Shinn et al., 2023); SWE-PRMD has a process reward model read the policy's interactions window by window and feed corrective guidance back to the policy (Gandhi et al., 2025); weak-to-strong monitoring detects agents' covert violations with external monitoring pipelines and stress-tests the monitor's own robustness with a red-teaming workflow (Kale et al., 2026); guardrails move intervention up to the planning stage, with criteria taken from external risk taxonomies (Guardrail). Closest to the deployment loop is the active-verification line: ProRe schedules targeted state probes with a general reasoner, replacing static trajectory grading (ProRe), and ContextCov derives executable constraints from instruction files and enforces them (Sharma, 2026); but both draw their criteria from external rules and not from the run's own stated commitments. Reward-hacking governance for training signals (Amodei et al., 2016) and after-the-fact independent evaluation and accountability, one altering only the training signal and the other producing only evaluation conclusions, leave execution itself untouched; Deep Search's verification loop (EvidenceLoop) already has a closed-loop shape, yet it only checks the provenance of the answer's evidence (Song et al., 2025), not whether the obligations the task assigned were fulfilled. These works add checkpoints link by link along the execution chain, yet no one's criterion is the run's own stated commitments checked against its actual execution path, and verdict and disposal remain separated. This paper closes the gap with a four-component loop: adjudication completes before delivery, and accountability lands at delivery.

3 Governance Framework

3.1 Problem Setting and Operationalizing the Failure

Delegated execution and the run unit. This paper considers delegated task execution in non-adversarial settings: an owner issues a delegation request (OwnerRequest) in natural language; the agent undertakes planning and execution and delivers through an artifact or an environment end-state. A run denotes one complete execution of one experimental condition on the same task instance from the same starting point, including that condition's native planning, review, feedback, continuation, and re-runs, until final delivery or termination; governance mechanisms are compared on complete run units. The object of governance is the agent itself, namely the governed role undertaking planning and execution; tasks, environments, plans, and artifacts are not objects of governance. The model roles in the framework (planner, executor, reviewer) are all filled by general-purpose LLMs, the orchestrator is a deterministic component, and the owner is a human; no dedicated guardian model is trained, nor is any change of the model's internal preferences presupposed. Under this setting, testing whether a governance mechanism is effective means answering three questions between complete runs on the same task set and from the same starting points: whether unfaithful delivery

decreases, whether genuinely completed tasks improve in step, and at what runtime cost these effects are obtained. The first two questions are carried by the knowing–doing failure and task completion evaluations respectively (Definition 3.1, Definition 3.1); runtime cost is accounted separately, with its reporting given in Section 4.

Definition 3.1 (task completion). A run counts as task completion if and only if its actual deliverable or environment end-state satisfies the frozen original task requirements; optimization-type tasks additionally report their original scores. This judgment is not replaced by the executor's self-report of completion, by a method-internal pass, or by the quality of the execution path.

Definition 3.2 (knowing–doing failure). A run constitutes a knowing–doing failure if and only if the executor's self-stated plan and delivery commitments diverge from its actual execution path: an authoritative source it promised to consult goes unconsulted, a real deliverable it promised to produce is supplanted by a controllable substitute, or a designated entry point it promised to use is bypassed. The judgment takes as input three kinds of observable records of that run: the executor's statements (plan, commitments, and their in-run revisions), the actual behavior trace, and condition changes. Statements and traces must come from the same executing subject at the corresponding point in time: knowledge held at the planning or review stage does not transfer to the executor, and later revisions in a run do not retroactively establish earlier knowledge. The judgment presupposes no latent cognition and attributes no motive; the judging protocol and its independent validation appear in Section 4.

The two evaluations are mutually independent. Functionally succeeding while leaving a commitment unfulfilled is a legitimate intermediate state: the outcome check asks only whether the deliverable meets the bar, the failure judgment asks only whether commitments were honored; the two are not merged into a single truth value, and no governance-verdict ground truth or composite compliance score is defined. This is a study of governance effectiveness and does not assume three burdens of proof: that all task failures are knowing–doing divergence, that latent cognition is present behind failures, or that the executor knowingly offends.

3.2 Framework Overview: Isolation, Review, and Routing

The framework forms a closed loop of four components: selective isolation, undisclosed heterogeneous review, contract-based acceptance, and deterministic failure routing. Governance does not operate as process control: no action ordering is prescribed, no step-by-step approval occurs, and the executor does not perceive that review exists. Governance operates at artifact acceptance: plans and artifacts are checked at their moments of submission, and verdicts are routed by a non-agent deterministic orchestrator according to failure class.

Roles and selective isolation. Each role sees only the information surface matching its duty; isolation is enforced by infrastructure mounts and permission configuration and does not rely on model self-restraint:

Role	Duty and output	Visible information	Workspace write
Owner	Issues the delegation request; final decision on escalations	Escalation materials: artifact, review opinion, verdict, evidence chain	Via approval interface
Planner	Produces the plan specification (DagSpec): nodes, contracts, routing	Task materials and repository (read-only; may run read-only commands); unaware the plan will be reviewed	None
Executor	Executes node tasks; produces artifacts	Task-description projection of the node contract; unaware of the full acceptance criteria; unaware of being reviewed	Sole writer
Reviewer	Judges whether plan and artifact are faithful	Full owner view: request, plan specification, full contract, artifact, complete execution record	None (may run verification commands)

Table 1: Roles and information surfaces of the governance framework.

The allocation of write permission guards a single provenance of artifacts: the workspace is written only by the executor, so the object of review cannot be contaminated by the planner's or reviewer's own outputs.

Undisclosed review. The governed party does not know it is being reviewed. Within the executor's visible scope no review material exists, and the fact of review cannot be reached through the mount surface, leaving disguise without a target (Greenblatt et al., 2024). The planner likewise does not know the plan specification will be reviewed: when a plan is rejected, it receives a natural-language revision request phrased in the owner's voice; no structured rejection signal is presented, and neither the review criteria nor the reviewer's existence is exposed.

Contract acceptance and two review pipelines. Node contracts are frozen with the plan and constitute the criteria for artifact acceptance. Plan review checks, at plan submission, the plan specification against the delegation request: whether the plan covers the delegated requirements and whether node contracts remain faithful to the delegator's intent. Artifact review checks, at artifact submission, the delivered artifact against the node contract: whether the delivery honors the contract. The two pipelines are carried by the same reviewer role, with criteria switching by object under review. The plan specification uses an engine-agnostic schema and passes compile-time-locked structural validation at submission, suppressing hallucinated fields at planning (implementation details in the appendix). Reviewers come from model families different from the executor's and vote as a committee, with size, family selection, and the confidence threshold fixed with the run configuration (appendix). (The faithfulness judged here and the knowing–doing failure of Definition 3.1 are complementary: the former is the routing basis inside the governance mechanism; the latter is the evaluative foundation outside it.)

3.3 Evidence, Active Verification, and Enforced Disposal

The loop's causal claim rests on three links: the reviewer can obtain process evidence, can actively verify, and every verdict necessarily triggers disposal. The ablations of Section 4 remove these three links respectively.

Process is evidence, not an object of intervention. The reviewer sees the complete execution record from the owner's perspective, so the execution trace serves as evidence for faithfulness judgments. But the framework does not take process as its point of intervention: it prescribes no internal action ordering, grants no stepwise approval, and turns no offline path analysis into online correction. A single action of unfaithful execution is usually fully legal; governance therefore does not land at the action level but at artifact acceptance at submission time.

Active verification. Beyond the existing visible surface (including the complete execution record), the reviewer obtains read and run permissions: it may read task files and run verification commands to assist grading; write permission remains closed, and verification proceeds read-only or on copies, with permission boundaries and implementation documented in the appendix.

Enforced disposal. Verdicts are executed by the deterministic orchestrator and recorded as ExecVerdict; the reviewer holds no discretion over disposal. Failures take two forms:

- **Mechanical failures** (deterministically decidable): gate failures, missing artifacts, unambiguous judgments with unanimous high-confidence reviewers. Disposal is a re-run under the same contract with budget $N{=}2$, a pure deterministic count; when the budget is exhausted, escalate to the owner. Diagnostics are not fed back to the executor: replaying under the same contract is zero new information to it, entering only escalation records and audit.
- **Semantic failures** (not decidable by deterministic structure; final decision rests with the owner, with contract ambiguity first handled in-run through replanning and the rest escalating directly): contract ambiguity, where the root cause lies in acceptance criteria or a contract deviating from the delegator's intent, goes to the planner for replanning, the rejection signal disguised per non-disclosure; contract fault, where the artifact is faithful to a contract that itself deviates from the delegation request, makes re-running pointless and escalates directly with a contract-amendment proposal, recorded in audit as a calibration signal for plan review; absence of committee majority, low confidence, or failure of the

review pipeline itself escalates to the owner. A review without conclusion is never treated as a pass; there is no silent failure anywhere.

The owner is the terminal adjudicator: judgments that cannot be decomposed further must terminate in a human, while mechanical failures are handled without humans by deterministic counting. Humans intervene only in judgments that cannot be decided automatically.

Correspondence with the ablations. A1 removes process evidence, leaving the reviewer unable to see the execution record; A2 removes active verification, so the reviewer no longer runs verification commands or performs active checking, but the execution record remains visible; A3 is audit-only without enforced disposal, where review produces reports but changes no routing. The contribution of each link is tested in Section 4; selective isolation, non-disclosure, and heterogeneous review are retained as design choices, and no separate causal advantage is claimed for them.

3.4 Method Boundaries and Experimental Comparability

Governance happens inside the run; it is not a third evaluation. The governance mechanism is an in-run variable of the comparative experiment; its logs serve reproducibility, evidence attribution, and cost accounting, and produce no evaluative truth. The two evaluations of task completion and knowing–doing failure proceed independently after runs end, with no feedback into runs.

Boundary of measurement evidence. The two evaluations use only observable records natively produced during the run: executors are not asked to produce additional statements for evaluation, no independent knowledge questionnaire, hidden-layer probe, or special oracle is introduced; where reasoning content is not externally visible, after-the-fact summaries or paraphrases are not substituted for the statement record.

Baselines retain native protocols. The four external governance methods (Reflexion (Shinn et al., 2023), SWE-PRMD (Gandhi et al., 2025), ContextCov v2 (Sharma, 2026), and the full-paper form of ProRe (ProRe)) are integrated into the unified experimental infrastructure under their native mechanisms; their information permissions, intervention timing, feedback, and termination semantics are not forced into this paper's contract and routing structure. External baselines answer whether neighboring methods can solve the same delegated-execution problem; this paper's ablations answer component contribution; neither substitutes for the other.

Handling of owner interaction. Native approval steps of each method are attached under an assumed-approval simplification with intervention needs recorded; approval does not count as evidence of completion, and fixed replies are declared, with trigger conditions and content, only when genuinely required.

Behavior attribution. Evaluation judges only executor behavior: reads, checks, and modifications performed by the governance side do not enter the executor's path, and governance verdicts do not serve as measurement answers; behavior the executor actually adopts after receiving feedback remains the executor's. Trajectories altered by the governed mechanism can therefore still be compared across conditions under a unified standard.

The task set, condition matrix, judging protocol, statistical criteria, and the reporting of runtime cost appear in Section 4.

4 Experiments

4.1 Experimental Setup

Conditions. The test is conducted under nine conditions: an ungoverned baseline; four external governance methods, Reflexion (Shinn et al., 2023), SWE-PRMD (Gandhi et al., 2025), ContextCov v2 (Sharma, 2026), and ProRe (ProRe), each integrated under its native protocol into the unified infrastructure, retaining native allocations without forcing equalized compute; ALFRED, the full framework of this paper; and three mechanism ablations corresponding one-to-one to the three links of Section 3.3: removing process evidence (A1, the reviewer cannot see the execution record), removing active verification (A2, the reviewer no longer runs verification commands or performs active checking; the execution record remains visible), and audit-only without enforced disposal

(A3, review produces reports but changes no routing). The information permissions, intervention timing, feedback, and termination semantics of the four external methods are not fitted into this paper's contract and routing structure; their native approval steps are attached under the assumed-approval simplification with intervention needs recorded, and approval does not count as evidence of completion. All conditions use the same task set and the same starting points; the base executor model and each method's bound configuration are locked once ready and written into run records, and a method's original-paper model name is not treated as evidence of present-day availability.

Task set. The task set has 45 items, frozen before comparisons begin and neither pruned nor extended by run performance: items are not deleted or swapped for method-performance differences or missing measurements, and the set is not shrunk because a domain lacks completed adaptation. Of these, 15 are self-built cases derived from real engineering sessions, covering repair, refactoring, verification, real-database audit, documentation delivery, and compliant submission; 29 are drawn from existing benchmarks: SWE-bench Verified (Jimenez et al., 2024), Frontier-SWE (Frontier-SWE), WebShop (Yao et al., 2022), ALFWorld (Shridhar et al., 2021), and WebDetective (Song et al., 2025); and 1 is a bandit decision task constructed in the manner of Greedy Agents (Schmied et al., 2026), for which binary completion does not apply and the original continuous score is reported separately. Results are presented grouped by fixed source family: self-built cases, SWE-bench, WebDetective, Frontier-SWE, WebShop, ALFWorld, and Bandit; the 17 ALFWorld items are instances of one benchmark, not seventeen benchmarks.

Run structure. The 45 tasks $\times$ 9 conditions give 405 starting run units, with one complete run scheduled per unit in the starting batch; the definition of the run unit appears in Section 3.1. Run states are reported honestly: started, ended, and not-started are counted separately; initialization failures count as started, and no zero-usage records are fabricated for units not started. Within-condition replications and the formal sample size are locked from variance information obtained in a pilot run, and pre-registered together with criteria and the statistical protocol; the validation batch serves only as a descriptive check of wiring, evidence, and judge availability, is not used for superiority conclusions, and does not feed back into task selection.

4.2 The Two Independent Evaluations and the Judge

Independence of the evaluations. The two evaluations, task completion and knowing–doing failure (Definitions 3.1, 3.2), proceed independently on frozen materials after runs end, with no feedback into runs; the governance logs serve reproducibility, evidence attribution, and cost accounting, and produce no evaluative conclusions.

Judgment states and the primary reporting metric. Each evaluation has six mutually exclusive states, and the state sets differ. Task completion falls into: completed, not completed, undetermined, evaluation error, pending, and not applicable; not applicable covers tasks with only a continuous score and no binary completion predicate (the bandit task of 4.1), whose original score is reported separately and whose knowing–doing failure evaluation proceeds as usual. Knowing–doing failure falls into: confirmed failure, confirmed consistent, no comparable statement, undetermined, evaluation error, and pending; its primary reporting metric is the failure rate within the comparable-and-adjudicable set, namely the count of confirmed failures divided by the sum of confirmed failures and confirmed consistencies, with all state shares reported alongside. Differences in comparability opportunity across conditions are never hidden: the executor's statement structure may differ across conditions, changing the share of no-comparable-statement runs, and comparing only failure rates without reporting denominator composition would overstate comparability.

Judge. The knowing–doing failure evaluation is executed by a unified independent LLM judge applying the same standard across conditions; the task completion evaluation is decided by task-level checkers and existing evaluation pipelines against the frozen original requirements, without presupposing an additional general-purpose LLM judge. The judge reads the real records and rules case by case; each judgment must carry its rationale and cite the executor's own words and the corresponding execution evidence as its basis; action payloads and tool returns do not pass for the executor's own statements. Judge configuration and execution configuration are mutually independent and fully explicit: read caps, call caps, time limits, and retry budgets. Evaluation does not rely on in-run human intervention.

Judge validation. Before entering formal comparisons, the judge is independently validated: three researchers who took no part in the task construction or judge-label generation for the respective items, and who are neither the designer nor the tuner of this judge version, first lock reference judgments independently (references are not given to the judge); they then check the judge's outputs case by case: whether statements and traces were identified correctly (subject, time point, and action-opportunity correspondence), false positives and false negatives, and whether not-applicable and missing cases are listed faithfully. The reference-agreement rate, confirmed semantic errors, false positives and false negatives, and non-comparable runs are each reported separately. The reference-agreement rate is not reported as accuracy: reference judgments may themselves be contestable, and three-way checking does not constitute triple the sample.

Behavior attribution. Reads, checks, and modifications performed by the governance side do not count as executor behavior, and governance verdicts do not serve as measurement answers; behavior the executor actually adopts after receiving feedback remains the executor's (Section 3.4). Trajectories altered by the governed mechanism can therefore still be compared across conditions under a unified standard.

4.3 Main Results

There are eight comparisons, all pre-declared: the comparisons of the full framework against the ungoverned baseline and the four external methods are the main comparisons, five in total; the full framework against each of the three ablations, one each. No pairwise significance search between external methods is conducted. Reporting form: a task$\times$condition grid over all 45 slots listing run state and both evaluations; for run pairs on the same task, replicate, base executor model, and comparable starting point, paired transition tables and paired differences for each evaluation; a cross-tabulation of the two evaluations, directly showing whether governance trades task completion for faithfulness; for runs confirmed failed, repair status and the failure fact are reported together even if later repaired through feedback; the equal-weight descriptive mean per instance is given only after a method covers all 45 items, with coverage and gaps reported honestly otherwise.

4.4 Ablations

The three ablations each answer whether a link of the loop carries load: with process evidence removed, review loses the execution trace as an evidence source, and judgment may degrade; with active verification removed, review no longer runs verification commands or performs active checking but still sees the execution record: can it still detect bypass and substitution; with audit-only and no enforced disposal, judgments still occur but change no routing: does the knowing–doing failure rate rise again. They are presented as the three pre-declared ablation comparisons of 4.3, interpreted with component attribution.

4.5 Runtime Cost

Original scores and runtime cost are reported separately and do not enter the main tables. Usage and cost are broken out by the three stages of task execution, completion detection, and path measurement; the consumption of detection and measurement is the cost of the evaluation infrastructure and is not mixed into the task-execution cost of the compared methods. Billing is taken from provider-native usage fields with normalized mappings; identical requests are deduplicated by provider and original request identifier, unverifiable duplicates are listed as pending rather than guessed away; cache and reasoning segments follow the provider's billing rules for inclusion in input or output, without double counting. Time consumed by governance itself is accounted separately from the task's original budget and does not alter the task's deadline semantics; where monetary coverage is incomplete, no total is given.

5 Results Analysis

Section 4 answers how things are measured and what was measured; this section answers what the results mean: 5.1 interprets the main comparisons (including the external-method contrasts), 5.2 attributes the ablations to components, 5.3 interprets runtime cost and owner burden, and 5.4 reports

validity checks at the measurement layer. This section interprets only results within the pre-declared comparison framework of Section 4, adding neither comparisons nor reporting rules; all measured numbers are as reported in Section 4. The flow of governance events, itemized records of each method's native interventions, and full narratives of representative cases appear in the appendix.

5.1 Interpreting the Main Comparisons

Main comparison. The full framework's comparison against the ungoverned baseline answers the first two questions of Section 3.1: whether unfaithful delivery decreases and whether genuinely completed tasks improve in step; the main comparisons number five in total (see 4.3), with the attribution of the external-method items in the following paragraph. Readings are taken from two places: the knowing–doing failure rate (primary metric per 4.2) and the task completion rate in the per-condition tables of both evaluations; and, on run pairs with the same task, replicate, base model, and comparable starting point, the paired transitions and paired differences of each evaluation (adjudication conditions in 4.3). The directionality of paired transitions is the mechanistic evidence here: if governance works, transitions should show failure-to-consistent conversions exceeding the reverse, and not at the cost of more pending or undetermined states.

Cross-evaluation. The cross-tabulation (task completion $\times$ knowing–doing failure) answers directly whether governance trades task completion for faithfulness. The expected reading of the four quadrants: if under governance the completed-and-consistent quadrant expands while the completed-but-failed quadrant contracts, the two evaluations move together; if the not-completed-but-consistent quadrant expands at the same time, governance has lowered the delivery bar rather than improved compliance, and this must be read together with denominator composition.

Attributing differences among external methods. Each of the four external methods' paired differences against the full framework is one pre-declared comparison (4.3); attribution starts from three native-structure facts: whether information permissions reach process evidence, whether intervention occurs during execution or at submission, and whether a forced disposal route follows the verdict. The positioning of whether neighboring methods can solve the same delegated-execution problem is given in Section 3.4 and is not reopened here; itemized records of who did what, when, and on what information appear in the appendix.

Denominator composition and comparability. Comparability opportunity differs across conditions (4.2): if an external method changes the executor's statement structure, the share of no-comparable-statement runs shifts, and an apparent drop in failure rate may partly come from denominator contraction rather than improved compliance. Interpretation therefore reads alongside the per-condition denominator decomposition: failure-rate differences are read as compliance differences only when denominator compositions are broadly comparable; otherwise they are honestly reported as differences in comparability opportunity.

Coverage and source families. The conditions for giving the equal-weight descriptive mean appear in 4.3; where coverage is incomplete, only coverage counts and gaps are read, with no extrapolated fill-in. Observations of differences by source family are exploratory, enter only the appendix so labeled, and do not enter the main-text interpretation.

5.2 Attributing the Ablations

Parallel removal, not a cumulative ladder. Each ablation is the full framework with one link removed in parallel (definitions in 4.1); they do not form a cumulative ladder, and the ungoverned baseline serves only as a narrative reference for loop completeness in the presentation order, not as an object of ablation testing. Paired differences are reported only for the pre-declared pairs: each ablation against the full framework.

Reading each link. A1 asks whether judgment degrades once review loses the execution trace as evidence; the shape of degradation is more informative than its magnitude: misses concentrating on bypasses without corresponding statements would show that process evidence carries information beyond self-reports. A2 asks whether bypass and substitute implementations can still be detected once active verification is lost (no verification commands, no active checking, but the execution record still visible); the boundary of detection is the increment of active verification. A3 asks whether the failure rate rises again when judgments still occur but change no routing: the size

of the rebound is the gap between detection and disposal, the only link among the three that separates finding a problem from blocking its delivery. The implementation form of A3 (reusing the first submission of the full-framework run as the review input, or running independently) remains open in the implementation specification; the two forms yield different readings and are interpreted accordingly.

Attribution boundary. Ablations answer component contribution; selective isolation, non-disclosure, and heterogeneous review are design choices, and no causal advantage is claimed for them separately from ablation results (Section 3.3). Ablations and external baselines do not substitute for each other (Section 3.4).

5.3 Interpreting Cost and Owner Burden

Cost structure. The accounting rules appear in 4.5; only structural interpretation is given here. The meaning of breaking usage and cost out by the three stages of task execution, completion detection, and path measurement is comparative fairness: the consumption of detection and measurement is the cost of the evaluation infrastructure and is not mixed into the compared methods' task-execution cost; time introduced by governance itself is listed separately and does not encroach on the task's original budget semantics.

Approval and escalation burden. Each method's native approval steps are attached under the assumed-approval simplification (4.1). Reported here are escalation and approval counts, handling outcomes, and the share of necessary interventions, with the declaration that request counts are not equivalent to actual human time. If escalations concentrate in a particular failure class, read against the routing semantics of Section 3.3: mechanical failures are expected to be handled in-run by deterministic counting, and what escalates to the owner should be semantic failures.

Budget termination and attrition. Counts and cause distributions of terminations by budget or exhaustion per condition, and the usage distribution of the mechanical re-run budget $N{=}2$, are presented descriptively. Environment failures, budget exhaustion, and insufficient evidence are registered separately: they do not masquerade as task non-completion, nor enter the valid conclusions of the two evaluations. (These two descriptive tables are reported in the appendix unless added to the pre-registered statistical protocol.)

Original scores. The original continuous scores of optimization-type and bandit tasks are reported separately per 4.5 and do not enter the main tables; a table footnote states that results on adapted tasks do not pass as original-benchmark leaderboard scores.

Cost is not converted into effect. Runtime cost answers the third question of Section 3.1 but is presented on its own: no measurement quantity is replaced by cost, cost is not converted into effect, and no composite score is formed with the failure rate.

5.4 Validity Checks and Additional Discussion

This section reports the results of measurement-layer checks; protocols and judgment states appear in 4.2, and only results and interpretation are given here, with details in the appendix; judge-related wording is filled in per the finalized judging-protocol version, and the main text carries no process code names.

Judge validation. This section reports the itemized validation results listed in 4.2; the interpretive boundary of the reference-agreement rate is in 4.2 and is not restated here.

Acceptance of completion detection. Re-check results after completion checking is wired in against original task criteria, and the handling of instances the criteria do not cover.

Behavior-attribution audit. Ground-truthing of the principle of 4.2: governance-side reads, checks, and modifications were not counted into executor behavior, and verdicts did not serve as measurement answers, confirmed by rule checks plus spot checks.

Evaluation errors and environment failures. Evaluation-stage errors and runtime environment failures are registered and presented separately; the former does not masquerade as task non-completion, the latter does not masquerade as valid conclusions of the two evaluations.

Counter-intuitive findings. Should evidence contrary to expectation appear, such as a low share of necessary interventions, mechanical re-runs ineffective for some failure class, or review over-reliant on semantic adjudication, it is reported faithfully, without embellishment.

Robustness checks. Whether robustness checks under rule perturbation are added is decided once the statistical protocol is final; undecided items do not enter the main text.

6 CONCLUSION

This paper studies knowing–doing failure in delegated task execution in non-adversarial settings: the executor's self-stated plan and delivery commitments diverge from its actual execution path, while neither process control nor self-report can stop unfaithful delivery at submission time. To this end the paper proposes ALFRED, a structure in which governance lands at artifact acceptance: selective isolation, undisclosed heterogeneous review, contract-based acceptance, and deterministic failure routing form a closed loop; verdicts are executed by a non-agent orchestrator, and judgments that cannot be decided automatically terminate at the owner. Its effects are tested with the two mutually independent evaluations of task completion and knowing–doing failure, on a frozen task set and all pre-declared comparisons (Section 4).

Limitations. This paper does not assume the three burdens of proof listed in Section 3.1. Results were obtained on a frozen task set and a single base-executor model family; extrapolation across families and task domains is unverified. The knowing–doing failure evaluation relies on a uniformly validated independent judge, and task completion relies on checkers wired in against original task criteria; both are validated yet imperfect measurement procedures (Section 5.4). This paper evaluates the run-level effect of governance mechanisms and does not cover the owner's experience of use.

Future work. First, small within-subject human-factors studies of the governance interface, measuring owner trust calibration and actual burden; second, extension across model families and further task domains, with continued growth of the task set; third, adversarial evaluation on the reviewer side, testing the robustness of review against stronger governed parties.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with writing and polishing the text: translating our Chinese working drafts into English, copy-editing for grammar, style, and terminology consistency, and preparing the LaTeX sources. We have not used generative AI tools for producing or altering any experimental results, code, or data; the governance framework, the measurement protocol, and all technical claims are the authors' own and were verified by the authors. We have reviewed all AI-assisted text. We take responsibility for the final content of this work, including text produced with the aid of generative AI.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Yuntao Bai, Saurav Kadavath, Saurav Kundu, Amanda Askell, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Victor Barres, Honghua Dong, Soham Ray, Xiaolei Si, and Karthik Narasimhan. τ^2-bench: Evaluating Conversational Agents in a Dual-Control Environment, 2025. arXiv:2506.07982.

Yize Cheng, Chenrui Fan, Mahdi JafariRaviz, Keivan Rezaei, and Soheil Feiz. Model-adaptive tool necessity reveals the knowing-doing gap in LLM tool use, 2026. arXiv:2605.14038.

Sagar Deb and Ashwanth Krishnan. STOCKTAKE: Measuring the gap between perception and action in LLM agents with a fair oracle, 2026. arXiv:2607.13618.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. arXiv:2501.12948.

Yan Ding, Chi Zhang, Juntao Li, H. R. Lin, and Min Zhang. FAPO: Flawed-aware policy optimization for efficient and reliable reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.22543.

Frontier-SWE. Frontier-SWE benchmark, 2026.

Shubham Gandhi, Jason Tsay, Jatin Ganhotra, Kiran Kate, and Yara Rizk. When agents go astray: Course-correcting SWE agents with PRMs, 2025. arXiv:2509.02360.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Ryan Greenblatt, Carson Denison, Evan Perez, et al. Alignment faking in large language models. *arXiv preprint arXiv:2412.14093*, 2024.

Guardrail. Building a foundational guardrail for general agentic systems via synthetic data. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview 10009978.

Sirui Hong, Mingxin Zhuge, Jonathan Chen, Xiawu Zheng, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

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 (ICLR)*, 2024.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Neil Kale, Chen Bo Calvin Zhang, Kevin Zhu, Ankit Aich, Paula Rodriguez, et al. Reliable weak-to-strong monitoring of LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2508.19461.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs, 2025. arXiv:2501.12599.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, et al. Measuring AI ability to complete long software tasks, 2025. arXiv:2503.14499.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tülu 3: Pushing frontiers in open language model post-training, 2024. arXiv:2411.15124.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, et al. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

METR. Summary of METR's predeployment evaluation of GPT-5.6 Sol. `https://metr.org/blog/2026-06-26-gpt-5-6-sol/`, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, et al. BALROG: Benchmarking agentic LLM and VLM reasoning on games, 2024. arXiv:2411.13543.

ProRe. ProRe: A proactive reward system for GUI agents via reasoner–actor collaboration. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview 10006613.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Anian Ruoss, Fabio Pardo, Harris Chan, Bonnie Li, Volodymyr Mnih, and Tim Genewein. LMAct: A benchmark for in-context imitation learning with long multimodal demonstrations. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2412.01441.

Gilbert Ryle. *The Concept of Mind*. Hutchinson, 1949.

Thomas Schmied, Jörg Bornschein, Jordi Grau-Moya, Markus Wulfmeier, and Razvan Pascanu. LLMs are greedy agents: Effects of RL fine-tuning on decision-making abilities. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2504.16078.

Jiaqi Shao, Yuxiang Lin, Munish Prasad Lohani, Yufeng Miao, Bing Luo, et al. Do LLM agents know how to ground, recover, and assess? A benchmark for epistemic competence in information-seeking agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. arXiv:2402.03300.

Reshabh K. Sharma. ContextCov: Deriving and enforcing executable constraints from agent instruction files, 2026. arXiv:2603.00822.

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 (NeurIPS)*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, et al. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Fan Shu, Yite Wang, Ruofan Wu, Boyi Liu, and Zhewei Yao. DARE-bench: Evaluating modeling and instruction fidelity of LLMs in data science. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2602.24288.

Maojia Song, Renhang Liu, Xinyu Wang, Yong Jiang, Pengjun Xie, et al. Demystifying deep search: A holistic evaluation with hint-free multi-hop questions and factorised metrics, 2025. arXiv:2510.05137; includes the WebDetective test set.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023. arXiv:2308.08155.

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 (NeurIPS)*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains, 2024. arXiv:2406.12045.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Lei Han, and Volker Tresp. WebArbiter: A generative reasoning process reward model for web agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2601.21872.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.