Bruno: A Constrained Coordination Agent for Scientific Research Teams

Andreas Haupt [*1] Prashaant Ranganathan [*2]

Abstract

We propose Bruno, a semi-autonomous coordination agent for scientific research teams. Rather than generating hypotheses or running experiments, Bruno covers the coordination layer: it ingests state from the tools a lab already uses (Slack, GitHub, Overleaf, Weights & Biases, calendars, transcripts), maintains a project-scoped model of tasks, decisions, and artifacts, and surfaces that state through Slack and dashboards. Its action space is restricted by design to messages, dashboards, and human-confirmed task mutations; it has no write access to code, manuscripts, datasets, or instruments. We argue this constraint is the load-bearing design choice for deploying agents in high-stakes scientific workflows, and describe an evaluation roadmap built around a pilot in a project-based graduate computational biology course.

1. Motivation

Frontier autonomous-science systems target the generative stages of research: ideation, simulation, experiment design, manuscript drafting. Real laboratories continue to lose the bulk of their working knowledge in a less glamorous failure mode — the coordination layer. The graduation of a senior student from the lab leads to a loss of practical knowledge. Two coauthors writes up the same result. Protocol adjustments, failed sweeps, handoff context: none reach the manuscript, lab notebook, or tracker.

Existing PM tools (Linear, Asana, Monday) target software teams whose work product is code in a repository. Research teams' work product is experiments, data, and manuscripts, distributed across heterogeneous software tools. Electronic Lab Notebooks and Laboratory Information Management Systems solve record-keeping but not live cross-tool coordination. We see this gap as the natural niche for a constrained, semi-autonomous agent.

[*]Equal contribution [1]Stanford University [2]Independent. Correspondence to: Andreas Haupt <h4upt@stanford.edu>.

Accepted by ICML 2026 AI for Science Workshop.

2. Scope and Autonomy

Bruno is scoped to planning and the meta-layer across all stages of the workflow, not to the stages themselves.

Sprint planning/review. On a configurable cadence (e.g., fortnightly), Bruno solicits constraints on Slack and proposes an updated plan against deadlines.

Task decomposition Bruno proposes decompositions of stated goals ("ship the Enformer baseline by Friday") into todos managed on a web dashboard; humans confirm.

Cross-tool linking. A finished W&B sweep auto-comments on the relevant todo with the top run; a closing PR is logged; an Overleaf edit on an open task is flagged.

Decision logging & divergence detection. From meeting transcripts, Bruno drafts decision logs and flags when stated plans diverge from observed execution.

By construction, no writes to code, manuscripts, datasets, or instruments. The action space is restricted to: (i) Slack messages, threads, and canvases, (ii) web dashboards on a hosted endpoint, and (iii) mutations to a Bruno-internal task store.

3. Architecture and Tooling

Figure 1 gives the component-level view. Four layers:

Ingest. Per-project, read-only scopes to Slack (primary I/O), GitHub, Overleaf, Weights & Biases,

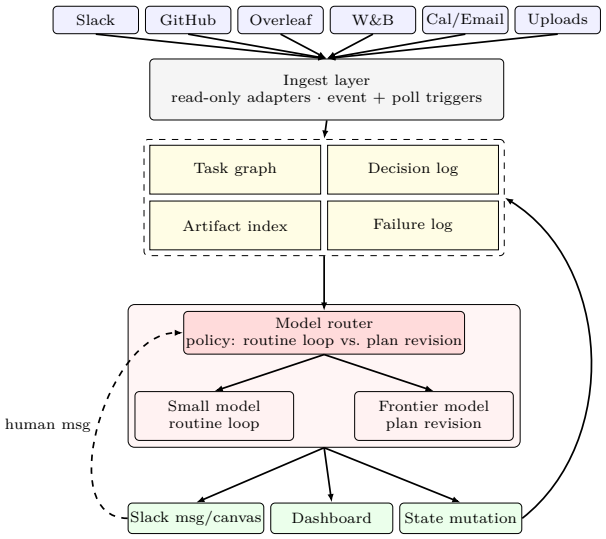


Figure 1. Bruno component view. Solid arrows: data flow. The reasoning agent is split into an explicit model router that dispatches between a small-model routine loop and frontier-model plan revision. The agent's only write paths are Slack messages, the dashboard, and confirmation-gated mutations to the project state store; ingest sources and external artifacts are read-only. The dashed arrow is the human-in-the-loop path: the human reads the emitted Slack message and replies.

Google Calendar, email, and uploaded files. Every connection is opt-in.

State. A typed, project-scoped store: (i) task graph (owners, blockers, deadlines, provenance edges to source events); (ii) decision log (open questions, go/no-go points, contradictions); (iii) artifact index (manuscripts, repos, runs, datasets); (iv) "what failed and why" log capturing negative results and protocol changes that would otherwise vanish.

Reasoning. A tool-using LLM invoked on timed (sprint cadence) and event triggers (new W&B run, GitHub PR, Slack mention). The coordination task is narrow and verifiable, so we run the production loop on small models (∼7–13B) for routine state-keeping and reserve a frontier model for plan revision and ambiguous parsing.

Further Integration. Integration of programmable lab automation (Opentrons) and broader instruments via SiLA2 or data layers (TetraScience, Ganymede) is a next step to increase observability. This does not change the commitment to a limited action space for the agent.

4. Evaluation Roadmap

A quarter-long pilot in a project-based graduate computational biology course, using Overleaf, GitHub, optionally W&B, with Slack as the discussion channel. The design is within-team and longitudinal rather than comparative: we make no attempt to construct a non-Bruno control arm, and instead track the trajectory of validated coordination measures across the quarter.

Repeated administration of validated team-coordination instruments over the quarter: a transactive-memory-systems scale (specialization, credibility, coordination), a team shared-mental-model measure, and a perceived-coordination-effectiveness scale.

5. Governance and Safeguards

Action-space restriction as commitment device. Bruno's inability to write to code, manuscripts, experiments, or instruments allows for deployments where the cost of a silent edit is high.

Privacy. All outputs are visible to the team owning the project; no instructor view, by design, to avoid creating educational records. Source connections (GitHub OAuth, Overleaf/W&B tokens) are user-initiated. The only LLM provider in the stack is contractually prohibited from training on customer content; remaining processors act as data processors under standard commercial terms.

Attribution. Bruno does not take actions, but can reconstruct contributions in tools. If LLMs are connected to Bruno, then also AI contributions can be tracked in Bruno.

Failure modes and limitations. The load-bearing assumption is that an LLM can map heterogeneous tool events onto a coherent project state. This mapping is the system's most brittle component: event parsing can hallucinate tasks, mislabel ownership, or raise spurious plan/execution divergence flags, and the failure log—precisely the high-value, low-redundancy content—is the hardest to reconstruct correctly from noisy signals. The typed, provenance-edged state store bounds rather than eliminates this risk: every state assertion links back to its source events, so errors are auditable. Further, context is limited. Bruno observes only what is instrumented: coordination that happens in calls, direct messages, or hallway conversation is invisible, so the state model is structurally partial and most degraded exactly when a team is under pressure and communicating off-tool.