

# Agentic Workflow Design Principles for Reliable Science

## Abstract

Agentic artificial intelligence systems can reason, plan, and execute complex workflows, which can speed up scientific discovery. Their non-deterministic behavior, however, makes reliability a core challenge in domains that require safety, precision, and reproducibility. Moreover, their autonomy can cause security threats if their permissions are not correctly set up. This paper proposes a practical set of best-practice guardrails that aim to guarantee reliable Agentic workflows, both from the science and cybersecurity point of view. We organize the principles into safety, reliability, and security sets, from intent to actuation: intent normalization, schema enforcement, and role-scoped tool access; uncertainty budgeting with propagation and computed thresholds; provenance capture with reproducible configurations, caching, and rollback; and verification loops that include simulation cross-checks, model recalibration, and human-in-the-loop escalation. We ground the guidance in a case study, an Agentic chemistry framework for molecular property prediction and automation that integrates knowledge-driven planning with high-fidelity simulation in a closed loop. We show how the guardrails reduce common failure modes, improve end-to-end reliability, and yield scientifically valid, reproducible outcomes. The result is a concise set of patterns that practitioners can adopt to build trustworthy autonomy in scientific computing.

## CCS Concepts

• **Computing methodologies** → **Distributed computing methodologies**; *Parallel computing methodologies*; Artificial intelligence.

## Keywords

Reliable Agentic AI, Agentic Workflows, Computational Chemistry

### ACM Reference Format:

. 2026. Agentic Workflow Design Principles for Reliable Science. In *Proceedings of The 41st ACM/SIGAPP Symposium on Applied Computing (SAC'26)*. ACM, New York, NY, USA, 8 pages. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 Introduction

Scientific workflows are evolving rapidly: no longer static pipelines, they are morphing into Agentic systems that autonomously interpret objectives, select methods, coordinate tools, and validate outcomes without constant human supervision. This shift promises orders-of-magnitude gains in speed and throughput, particularly in domains where data, models, and simulations interact in complex feedback loops [11, 15]. But with this autonomy arises a new

frontier of failure modes: ambiguous intent may trigger meaningless or harmful actions; misaligned data models may drift into incoherence; high-fidelity simulations may fail silently or yield irreproducible results [8]. In such regimes, reliability cannot be treated as an afterthought; it must be implemented as a foundational system property. While the concrete manifestation of these systems is domain-specific, the underlying design principles of reliable Agentic workflows are broadly transferable across scientific fields [5].

In this paper, we articulate a cohesive set of design principles for dependable Agentic workflows in science, aimed at enhancing correctness, reproducibility, and interpretability, categorized into safety, reliability, and security principles. Our core thesis is that these principles should emerge from how the agents represent intent, manage uncertainty, enforce provenance, sequence tools, and validate execution, rather than being added on post hoc. Instead of prescribing a monolithic architecture, we propose a blueprint of mechanisms such as schema normalization, uncertainty gating, verification layers, provenance-aware caching, and feedback-calibrated learning that can be grafted into diverse Agentic systems. We intend for these principles to remain model-agnostic and domain-agnostic, providing a scaffold for trustworthy autonomy across disciplines.

To ground and apply these ideas, we present a case study in computational chemistry. Chemistry is an especially rigorous proving ground as it can involve many complex applications of AI. For example, computing trustworthy molecular properties requires orchestrating literature-based knowledge, machine-learning surrogate models, and high-fidelity ab initio simulation under constrained computational budgets. Each of these modalities carries different error modes, performance tradeoffs, and semantic assumptions, making chemistry an ideal testbed for reliability engineering. In our instantiation, we map the design principles to concrete agent roles: planner, knowledge/cache, compute scheduler, runner, verifier, analyzer, reporter, and learner. We demonstrate how the principles help prevent hallucinated intent, mismatched conditions, method mismatch, silent simulation errors, and inconsistent traceability.

The remainder of this paper is organized as follows: we begin by reviewing background and related work on AI reliability challenges (section 2). We then introduce our taxonomy of safety, reliability, and security design principles (section 3). Next, we demonstrate how these principles can be concretely applied to computational chemistry (section 4). Finally, we conclude with a summary of contributions and directions for future work.

## 2 Background and Related Work: Reliability in Agentic Workflows

Reliability challenges in AI-powered systems stem from three inter-related sources: internal reasoning limitations, external adversarial threats, and the potential for error propagation across interconnected components in complex workflows. While large language models (LLMs) have demonstrated impressive task performance, their underlying mechanisms are fundamentally statistical rather

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
SAC'26, Thessaloniki, Greece

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 979-X-XXXX-XXXX-X/26/03  
<https://doi.org/XXXXXXX.XXXXXXX>

than truly cognitive. As a result, even the most advanced models continue to exhibit behaviors far from human-level reasoning capabilities.

Macmillan et al. [17] systematically evaluated the reasoning consistency of large language models (LLMs) using cognitive tasks adapted from classical psychological studies. They found persistent patterns of irrationality, logical contradictions, internal inconsistencies, and biased inference, despite high linguistic competence. Similar observations in GPT-3's decision-making and causal reasoning [3] indicate that LLMs often produce confident yet incorrect outputs without mechanisms for self-correction or epistemic awareness. Because LLMs generate context-conditioned predictions rather than truth-evaluated reasoning, such undetected errors can propagate across Agentic workflows, compromising downstream reliability unless constrained by explicit validation mechanisms. In Agentic workflows, outputs from one agent often serve as inputs for other components in the workflow, forming tightly coupled dependency chains. This architecture introduces new internal risks: a single hallucination [13] can propagate downstream, contaminating subsequent reasoning and decisions, which has been demonstrated to be exponential [7]. Furthermore, minor variations in prompts can lead to dramatically different outputs, reflecting stochastic and unstable model behavior that complicates validation, tracing, and debugging. These phenomena, compounded by limited grounding and contextual understanding, may cause silent data drift or oversimplification of complex chemical or physical properties [19].

Although general AI safety research has begun addressing these challenges, most approaches lack domain specificity. Jeon [12] emphasizes the need for standardized methodologies to ensure AI reliability, while other efforts, such as [5, 26, 6, 10, 21, 9], explore input/output monitoring, safeguard layers, and content filtering mechanisms. Broader frameworks, such as those proposed by Xia et al. [27] and Bommasani et al. [4], advocate for holistic evaluations to promote transparency, interpretability, and systematic benchmarking. Souza et al. [23] emphasize similar principles and extend the W3C PROV and leverage the Model Context Protocol [18] standard for AI agents, to incorporate intra- and inter-agent operation into the workflow provenance. Gueroudji et al. [8] suggest a new definition of failures specific to Agentic systems and related to reliability concerns, and propose a generic set of controlling mechanisms to detect, contain, and recover from these failures. Yet, none of these works address reliability through a principled design framework to guide the architecture of Agentic workflows by embedding reliability at every layer of agent interaction and orchestration.

Furthermore, even with the application of alignment and reinforcement learning from human feedback [20, 2], LLMs remain vulnerable to external manipulation. Malicious actors can exploit prompt injection, poisoning, or adversarial input generation [1, 14, 25] to subvert or “jailbreak” models. The Agent Smith study [7] demonstrates that compromising a single agent within a multi-agent ecosystem can exponentially propagate harmful behavior across all agents.

### 3 Design Principles for Reliable Agentic Science

This section organizes design principles necessary to ensure that Agentic workflows are robust, auditable, and aligned with scientific rigor. We divide the principles into three core categories: Safety,

Reliability, and Security. Each principle is formalized with a clear definition, motivation, and actionable implementation guidelines.

#### 3.1 Safety Principles

Safety design principles ensure that Agentic workflows cannot cause harm, either physically (e.g., through robotic or experimental actuation) or epistemically (e.g., by issuing unsafe, high-stakes decisions affecting humans). They enforce bounded behavior under uncertainty through mechanisms such as human override, confidence-based escalation, abort triggers, and controlled fallback, ensuring that autonomous decisions remain safe for both users and downstream human impact.

**3.1.1 H: Human Oversight. Principle:** Agents must operate under a supervisory control paradigm where humans retain ultimate authority over high-stakes or epistemically uncertain actions.

**Technical rationale:** Agentic systems are non-stationary learners operating in open-world environments. Without human-in-the-loop oversight, error propagation can amplify due to feedback loops or self-reinforcing priors. Supervision acts as a stability constraint, bounding the divergence of the system's decision manifold.

**Implementation implications:**

- Use event-triggered alerts for low confidence, anomaly detection, or distributional shift.
- Employ review gates for irreversible or critical decisions.
- Maintain human override and interruption channels.
- Implement human feedback loops with high priority and incorporate them into the learning process.

**3.1.2 O: Only Use It If You Need It. Principle:** Deploy autonomous agents only when dynamic reasoning or active control yields measurable improvement over static workflows.

**Technical rationale:** Each added degree of autonomy expands the system's state space and behavioral entropy, making verification and formal assurance exponentially harder. Unnecessary agency inflates reliability risk and control complexity without proportional benefit.

**Implementation implications:** Before deployment, quantify the Expected Performance Gain (EPG) versus a static baseline. CONTAM shows an example for assessing contamination metrics [22].

**3.1.3 C: Defined Competence Scope. Principle:** Each agent must have a clearly delineated operational domain and competence.

**Technical rationale:** Unbounded competence induces epistemic overreach: agents extrapolate beyond their training distribution. Constraining competence aligns with bounded rationality and prevents cross-domain hallucination.

**Implementation implications:**

- Explicit skill ontologies and capability manifests
- Runtime self-assessment mechanisms (e.g., epistemic uncertainty estimates, model calibration)
- Delegate tasks beyond competence thresholds.

#### 3.2 Reliability Principles

Reliability design principles ensure that Agentic workflows produce scientifically correct, robust, and reproducible results across varying conditions. These principles govern epistemic soundness through

mechanisms such as schema enforcement, provenance tracking, uncertainty propagation, and execution verification, ensuring that the workflow adheres to methodological rigor, maintains internal consistency, and supports traceable scientific inference.

**3.2.1 P: Built-in Provenance. Principle:** Every state transition, message, and computation must be associated with verifiable provenance data.

**Technical rationale:** Provenance acts as a causal graph over epistemic states, allowing retrospective inference of why and how an outcome occurred. This is essential for fault diagnosis, reproducibility, and regulatory traceability in autonomous systems.

**Implementation implications:** Implement immutable logging layers (e.g., cryptographically signed event chains) across all communication interfaces:

- A-A: Agent-Agent coordination logs
- A-C and C-A: Agent-Component API traces
- C-C: Cross-component data lineage

where components can be any non-agentic component in a system architecture, e.g., an HPC simulation, model training, or API call. Use structured metadata (e.g., W3C PROV) to enable automated reasoning about provenance.

**3.2.2 E: Enhanced Safeguards. Principle:** Validate both input and output channels to detect adversarial, out-of-distribution (OOD), or unsafe behavior.

**Technical rationale:** Autonomous systems are susceptible to semantic drift and model inversion attacks. Validation layers serve as boundary filters that maintain the integrity of the input manifold and prevent unsafe actuation.

**Implementation implications:**

- Input validation via schema conformity and adversarial detectors
- Output filtering through constraint satisfaction solvers and rule-based postprocessors
- Bidirectional auditing to detect causal anomalies between inputs and outputs

**3.2.3 D: Prefer Determinism When Needed. Principle:** Prefer deterministic computation over generative models for tasks requiring high-fidelity control or when a simple, lower-cost, widely used method is sufficient.

**Technical rationale:** Probabilistic sampling introduces non-repeatable variance detrimental to experimental replication or safety-critical inference. Deterministic execution acts as a mode switch to enforce control-theoretic predictability.

**Implementation implications:**

- Fixed random seeds and locked model versions
- Deterministic solvers for verification-critical tasks
- Fallback from generative exploration to rule-based decision logic under low uncertainty

**3.2.4 R: Reasoning Metadata Bundle (RMB). Principle:** Attach a structured reasoning bundle to any AI decision that influences downstream tasks.

**Technical rationale:** RMBs turn opaque outputs into auditable artifacts, enabling uncertainty-aware composition, early hallucination/error detection, and reproducible replay. Considering the

Built-in provenance principle, RMBs should be modeled as first-class entities in the provenance graph (via *prov:wasGeneratedBy*, *prov:used*, *prov:wasInformedBy*) so they can link prompts, tools, evidence, policies, and outcomes end-to-end.

**Implementation implications:**

- Capture: concise rationale/assumptions, alternatives considered, uncertainty signals (confidence\_p, OOD/surprise, calibration), sanitized inputs/prompts, key tool calls, and citable evidence (persistent IDs).
- Record lineage/policy: dataset/model/message links, policy gate status (pass/fail), escalation path, human\_override\_used.
- Enforce presence via schema validation (reject/repair missing mandatory RMB fields); index RMBs by task/model/version for rapid audits.

**3.2.5 T: Template-specific. Principle:** Execute agent actions through predefined templates that fix role, context, and output *schema*.

**Technical rationale:** Templates act as machine-checkable contracts that standardize structure and reduce behavioral entropy, keeping validators, policy gates, and provenance capture standardized. Whereas RMB defines *what* reasoning metadata must exist, templates define *how* that metadata is structured and emitted consistently across activities.

**Implementation implications:**

- Define machine-readable schemas (e.g., JSON Schema – Fig. 1) for action I/O and RMB fields; version them and validate at runtime.
- Integrate schema checks into the agent’s policy engine and logging so emitted records are uniform and PROV-mappable.
- Fail fast on violations (reject/repair), ensuring downstream components and provenance queries remain reliable.

```

1  "Role": "Chemistry Assistant"
2  "Task": "Surface characterization analysis"
3  "Output": {
4    "format": "JSON",
5    "schema": {
6      "agent_id": "str",
7      "reasoning_trace": "str",
8      "decision": "str",
9      "confidence": "float",
10     "requires_attention": "bool"
11   }
12 }
```

**Listing 1: Template example used in the computational chemistry case study: sets role, task, and output schema to ensure consistent RMB emission and provenance capture.**

### 3.3 Security Principles

Security design principles protect Agentic workflows against unauthorized access, misuse, manipulation, and adversarial influence. These principles define strict operational boundaries through capability scoping, least-privilege execution, permission isolation, and cryptographically-verifiable provenance. Security mechanisms defend against external threats (e.g., prompt injection, API misuse) and internal faults (e.g., tool escalation, data leakage), ensuring that

the autonomy granted to agents cannot be exploited to compromise system integrity, data confidentiality, or operational control.

**3.3.1 L: Limited Permissions. Principle:** Constrain agent capabilities via a granular access control model analogous to operating system permissions.

**Technical rationale:** The principle of least privilege minimizes the attack surface and error propagation pathways. Restricting capabilities at runtime reduces the reachable subspace of unsafe or irreversible actions.

**Implementation implications:** Adopt hierarchical control lists (ACLs) and contextual permission graphs:

- Fine-grained read/write/execute rights
- Dynamic group contexts (task-based permission tokens)
- Temporal permission scopes for transient operations

**3.3.2 I: Prompt Injection and Input Sanitization. Principle:** All user or agent-generated inputs must be sanitized and validated before being passed to downstream models or components.

**Technical Rationale:** Foundation models are highly sensitive to prompt manipulation, which adversaries can exploit to hijack agent behavior or leak data. Robust input validation protects against prompt injection, prompt leaking, and instruction overriding.

**Implementation Implications:**

- Apply strict input schemas and lexical sanitization to all prompts.
- Detect injection signatures using pattern matching and anomaly detection.
- Isolate execution environments for untrusted input origins (e.g., sandbox prompts from external users).

**3.3.3 Q: Escalation Safeguards and Quarantine. Principle:** Suspected unsafe, ambiguous, or adversarial actions must trigger containment protocols, including escalation halts and isolation.

**Technical Rationale:** Autonomous agents must not be allowed to continue operation once uncertainty or risk exceeds predefined thresholds. Failing to intervene may lead to irrecoverable errors or breach conditions.

**Implementation Implications:**

- Define risk thresholds (e.g., uncertainty, contradiction, OOD scores) that trigger human review or quarantine modes.
- Quarantine agents suspected of prompt injection, decision drift, or policy violations.
- Design escalation trees with predefined halt–notify–override procedures.

## 4 Motivating Case Study: An Agentic Workflow for Computational Chemistry

In this section, we present an Agentic system for molecular property prediction via an automated workflow, detailing the potential threats and needed design principles. We then demonstrate an example of a provenance graph safety-check queries on it.

Foundation models are increasingly embedded in computational chemistry, combining reasoning, planning, and automation across scientific workflows. ChemCrow, El Agente, and DREAMS are examples of such applications [16, 28, 24]. Our hybrid Agentic chemistry workflow (Figure 1) integrates knowledge retrieval with *ab initio*

computation in a unified, autonomous loop. The agent interprets a user request (e.g., a thermodynamic or spectroscopic property), first querying curated databases and literature for existing results. If suitable data exist, the agent returns them for user confirmation; otherwise, it composes a computational plan and triggers first-principles simulations (e.g., DFT, MD, TDDFT) through a compute agent. Before submission, the agent and user refine key parameters (level of theory, basis set, simulation conditions). The agent then generates input decks and submission scripts, presents them for approval, and upon confirmation executes the calculations. After completion, outputs are parsed, analyzed with templates, validated, and fused into a report that clearly distinguishes properties sourced from prior data versus newly computed results. The workflow is fully instrumented with a provenance system capable of capturing both Agentic and non-agentic processes, enabling audit, replay, and safety analyses.

This data-aware, adaptive loop lowers computational cost and human effort while preserving scientific rigor, bridging predictive AI with quantum chemistry. It serves as our motivating case study for designing principled, safe Agentic systems that respond to uncertainty and data availability, advancing computational chemistry toward an interactive, knowledge-driven discovery paradigm.

### 4.1 Designing the Workflow Activities and Identifying the Threats

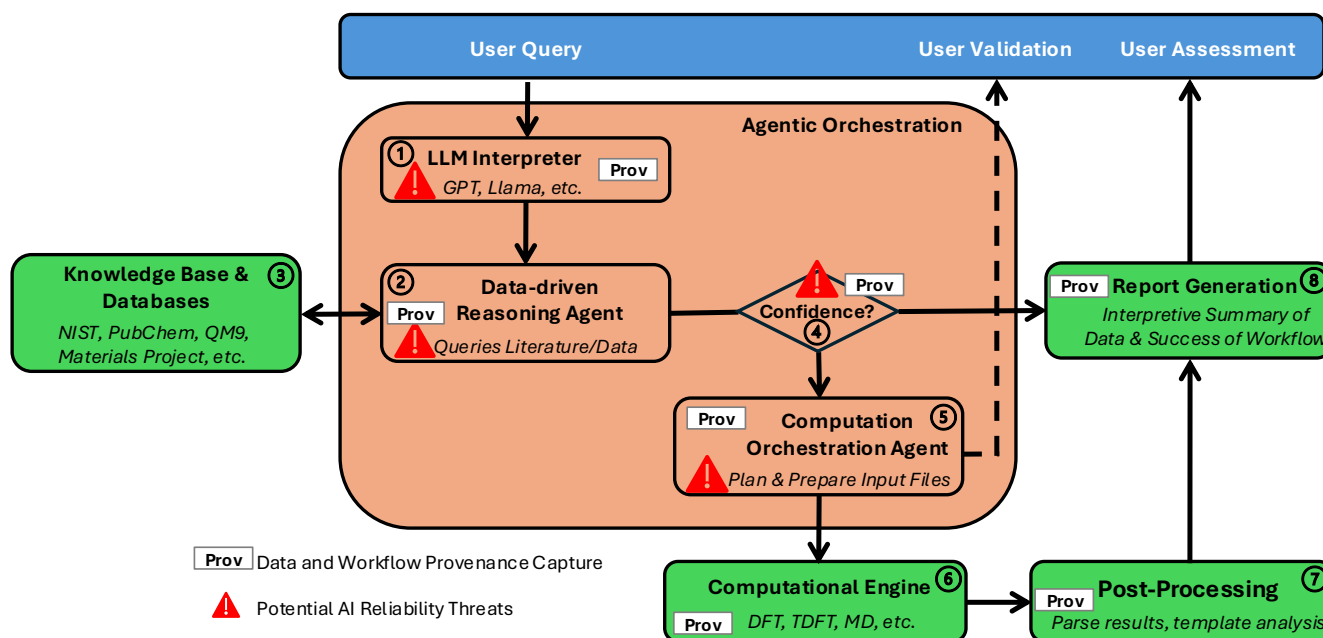
In this section, we expand on designing activities listed in Figure 1. Each activity includes Inputs, Threats, Outputs, and RMB outputs. We describe how our workflow is designed with respect to the principles discussed so far. We aim to anticipate, as exhaustively as possible, AI-related threats across planning, retrieval, tool use, computation, and reporting, and to guardrail them by exposing internal model rationale and attaching machine-actionable reasoning metadata. Wherever uncertainty, contradiction, or novelty appears, we surface clear alerts and escalation flags that trigger human-in-the-loop assessment, validation, or confirmation before the workflow proceeds. This design emphasizes early detection of errors, hallucinations, and unsafe decisions, and ensures that high-impact actions require explicit evidence, policy compliance, and traceable provenance.

(1) **User Query** *Normalize user intent into a scoped, safe, and secure task.*

- **Inputs:** free text query, constraints, safety policies.
- **Threats:** prompt injection, unsafe objectives, ambiguous scope, PII leakage.
- **Outputs:** normalized intent, task scope, targets.
- **RMB outputs:** rationale\_str, prompt\_excerpt\_str, decision\_scope, confidence\_p, requires\_review, prompt\_contains\_pii, upstream\_message\_ids.
- **Needed Principles:** T, R, I, P

(2) **LLM Interpreter** *Translate intent into a structured plan and bounded tool calls.*

- **Inputs:** normalized intent, tools schema, policies.
- **Threats:** hallucination, tool misuse, jailbreak patterns, low confidence on high impact.
- **Outputs:** structured plan, tool calls, provisional confidence.
- **RMB outputs:** tool\_calls, rejected\_options\_str, hallucination\_risk, surprise\_score, policy\_checks, unsafe\_action\_blocked.



**Figure 1: Workflow schematic for the Agentic chemistry workflow, identifying locations for domain-specific AI reliable concerns.**

- **Needed Principles:** C, L, R, P
- (3) **Data-driven Reasoning Agent** *Ground the plan with retrieval, evidence ranking, and alternatives.*
- **Inputs:** plan, retrieval queries, Knowledge Base (KB) connectors.
  - **Threats:** weak or stale sources, citation mismatch, retrieval leakage, license issues.
  - **Outputs:** evidence set, ranked candidates, literature citations.
  - **RMB outputs:** retrieval\_queries, citations, grounded\_by\_literature, evidence\_coverage, contradiction\_detected, feature\_importance.
  - **Needed Principles:** C, E, R, P
- (4) **Knowledge Bases & Databases** *Serve authoritative records with lineage and validation.*
- **Inputs:** API calls to NIST, PubChem, QM9, Materials Project, local caches.
  - **Threats:** schema drift, tampered cache, throttling, outages.
  - **Outputs:** verified records with persistent IDs, provenance links, cache status.
  - **RMB outputs:** dataset\_ids, evidence\_snippets, cache\_hits\_count, schema\_version, validation\_status.
  - **Needed Principles:** C, D, R, P
- (5) **Confidence Gate** *Enforce thresholds and trigger escalation when needed.*
- **Inputs:** confidence\_p, surprise\_score, evidence\_coverage, policy\_checks.
  - **Threats:** overconfidence, out-of-distribution passages, missing citations, unsafe continuation.
  - **Outputs:** continue or escalate decision, human-in-the-loop trigger.
- **RMB outputs:** requires\_review, notify\_human, auto\_halt, thresholds: confidence\_min, surprise\_max, hallucination\_max.
  - **Needed Principles:** C, Q, E, R, P
- (6) **Computation Orchestration Agent** *Compile validated compute inputs and schedules.*
- **Inputs:** approved plan, method (DFT, TDDFT, MD), parameters.
  - **Threats:** unit mismatch, unsafe parameters, I/O path issues, quota abuse.
  - **Outputs:** validated input files, runtime budget, scheduler directives.
  - **RMB outputs:** decision: codepath, method, basis; params\_json, unit\_checks\_pass, model\_artifact\_id, system\_policy\_hash.
  - **Needed Principles:** L, T, D, E, R, P, H
- (7) **Computational Engine** *Execute compute deterministically with robust logging.*
- **Inputs:** input decks, container image, seeds, scheduler directives.
  - **Threats:** nonconvergence, silent NaNs, corruption, reproducibility drift, hardware faults.
  - **Outputs:** raw outputs, logs, checkpoints, convergence metrics.
  - **RMB outputs:** convergence\_metrics, nan\_detected, runtime\_stats, env\_fingerprint, logs\_hash.
  - **Needed Principles:** L, T, D, E, R, P, H
- (8) **Post-Processing** *Parse outputs and compute properties with uncertainty.*
- **Inputs:** raw outputs, parsing templates.

- **Threats:** brittle parsers, template skew, underestimated uncertainty.
  - **Outputs:** derived properties, diagnostics, confidence intervals.
  - **RMB outputs:** derived\_properties, prediction\_intervals, parser\_diagnostics, attribution\_unstable, cherry\_pick\_risk.
  - **Needed Principles:** L, T, D, R, P, O
- (9) **Report Generation** *Synthesize findings with evidence and reproducible bundles.*
- **Inputs:** aggregated results, citations, full RMB trail, provenance graph.
  - **Threats:** unsupported claims, missing links, sensitive data exposure.
  - **Outputs:** interpretive summary, decision log, reproducible bundle, user-facing visuals.
  - **RMB outputs:** decision\_log, evidence\_map, unsupported\_claim, misleading\_visuals, repro\_bundle\_id.
  - **Needed Principles:** L, E, R, P, H, O
- (10) **User Validation & Assessment** *Close the loop with human judgment and feedback.*
- **Inputs:** report, alerts, audit trail, escalation notes.
  - **Threats:** ignored alerts, inadequate review, unbracketed overrides.
  - **Outputs:** accept, revise, or reject signal; feedback to thresholds and policies.
  - **RMB outputs:** accept\_revise\_reject, human\_override\_used, feedback\_notes, thresholds\_update.

## 4.2 Instantiating the Provenance Graph

We provide one concrete instance of the provenance data generated when running the workflow in Figure 1. The instance follows the W3C PROV-O ontology. Activities are the workflow activities (user query through user validation), entities are prompts, plans, evidence, datasets, thresholds, inputs, raw outputs, derived properties, reports, and the reasoning metadata bundle, and agents are the software agent and the model. Relations use standard PROV properties such as used, wasGeneratedBy, wasInformedBy, wasAssociatedWith, and hadMember.

**Activities:** wf\_1 (Workflow), uq\_1 (User Query), li\_1 (LLM Interpreter), ra\_1 (Reasoning Agent), kb\_1 (Knowledge Access), cg\_1 (Confidence Gate), orc\_1 (Computation Orchestration), eng\_1 (Computational Engine), pp\_1 (Post-Processing), rep\_1 (Report Generation), uv\_1 (User Validation).

**Entities:** prm\_1 (Prompt), plan\_1 (Plan), evd\_1 (Evidence Set), dset\_1 (Dataset Handle), thr\_1 (Trigger Thresholds), in\_1 (Compute Inputs), raw\_1 (Raw Outputs), der\_1 (Derived Properties), rmb\_1 (Reasoning Metadata Bundle), rpt\_1 (Final Report).

**Agents:** ag\_1 (AI Agent), mdl\_1 (AI Model).

**Key relations:**

- wf\_1 hadMember (uq\_1, li\_1, ra\_1, kb\_1, cg\_1, orc\_1, eng\_1, pp\_1, rep\_1, uv\_1) via hasMemberActivity.
- uq\_1 used prm\_1; uq\_1 wasAssociatedWith ag\_1; uq\_1 generated plan\_1.
- li\_1 wasInformedBy uq\_1; li\_1 used prm\_1 and mdl\_1; li\_1 generated plan\_1.
- ra\_1 wasInformedBy li\_1; ra\_1 used plan\_1; ra\_1 generated evd\_1 and rmb\_1.

- kb\_1 wasInformedBy ra\_1; kb\_1 used evd\_1; kb\_1 generated dset\_1.
- cg\_1 wasInformedBy ra\_1; cg\_1 used thr\_1 and rmb\_1; cg\_1 generated decision signal.
- orc\_1 wasInformedBy cg\_1; orc\_1 used plan\_1 and dset\_1; orc\_1 generated in\_1.
- eng\_1 wasInformedBy orc\_1; eng\_1 used in\_1; eng\_1 generated raw\_1 and telemetry.
- pp\_1 wasInformedBy eng\_1; pp\_1 used raw\_1; pp\_1 generated der\_1 and updated rmb\_1.
- rep\_1 wasInformedBy pp\_1; rep\_1 used der\_1, evd\_1, rmb\_1; rep\_1 generated rpt\_1.
- uv\_1 wasInformedBy rep\_1; uv\_1 used rpt\_1; uv\_1 generated feedback that updated thr\_1.

This instance illustrates closed-loop linkage of activities and entities, with explicit association to ag\_1 and model usage at li\_1. The reasoning metadata bundle (rmb\_1) is a first-class entity, generated and reused across activities to enable audit and replay.

## 4.3 Safety-check Queries on the Provenance Graph

We now enumerate some safety queries that are critical in the workflow. Each query can be answered by traversing activities (Workflow, Task, AIModelInvocation, AgentTool), entities (plans, datasets, RMBs, reports), and agents, via PROV relations (used, wasGeneratedBy, wasInformedBy, wasAssociatedWith) plus hasMemberActivity. RMB fields are modeled as entities or attributes linked to their generating activities and reused downstream.

### (1) Did any low-confidence or surprising decisions propagate to execution without human control?

**Criticality:** Prevents risky automation by catching model-flagged unreliable decisions before they reach compute or lab equipment.

Check activities where RMB shows  $\text{confidence}_p < \tau_c$  or  $\text{surprise\_score} > \tau_s$  and requires  $\text{review}=\text{true}$ , then follow wasInformedBy edges to see if they reached orchestration or engine without  $\text{human\_override\_used}=\text{true}$ .

Confidence and review flags live on RMB entities generated by decision activities and reused by later activities; wasInformedBy exposes propagation; wasAssociatedWith captures missing human intervention.

### (2) Were unsafe actions blocked by policy gates, and if not, what artifacts resulted?

**Criticality:** Verifies that guardrails bite; missed blocks can yield hazardous artifacts that must be traced and remediated.

Locate Confidence Gate activities with  $\text{policy\_checks}$  containing  $\text{severity}=\text{fail}$  or  $\text{red\_team\_pattern\_hit}=\text{true}$  and  $\text{unsafe\_action\_blocked}=\text{false}$ ; traverse forward to generated inputs, raw outputs, and reports.

Policy outcomes are captured on RMBs; gate decisions are activities that generate or withhold entities. Downstream effects are explicit via generated and wasInformedBy.

### (3) Did data leakage contaminate modeling or evaluation?

**Criticality:** Stops false confidence and unsafe deployments driven by inflated metrics from train-test contamination.

Find datasets (dataset\_ids) used by modeling (AIModelInvocation/li\_1) and later used to evaluate or report (post-processing/report).

Cross-check hadPrimarySource/specializationOf and shared dataset identifiers.

Entity–entity links (wasDerivedFrom, hadPrimarySource, specializationOf) plus activity usage chains reveal reuse of the same or related data across training/inference/evaluation.

**(4) Were schema or validation drifts ignored before compute?**

*Criticality:* Avoids silent misconfiguration (units, fields, versions) that can corrupt results or trigger dangerous operating conditions.

Identify Knowledge access producing entities with schema\_version changes or validation\_status has errors that flow to orchestration/engine.

Knowledge-base fetches generate domain data with version/validation attributes; propagation to compute is visible via used and wasInformedBy.

**(5) Did ungrounded or contradictory evidence reach the report?**

*Criticality:* Protects scientific integrity and downstream decisions by ensuring claims are supported, consistent, and citable. Locate decisions where grounded\_by\_literature=false or contradiction\_detected=true or evidence\_coverage <  $\tau$ , then see if their outputs were used by report generation. Also check citation\_count=0 or missing persistent IDs.

Evidence, citations, and RMB are entities linked to activities; report generation's inputs (used) make the dependency explicit.

**(6) Were prompts or tool calls containing PII used to create public artifacts?**

*Criticality:* Prevents privacy breaches and regulatory violations by tracking sensitive inputs into outward-facing outputs.

Find prompts/tool outputs flagged prompt\_contains\_pii=true that are used by activities generating reports or dataset exports. Prompts and tool results are entities; their flags travel with them; used and wasGeneratedBy show where they influenced outward-facing artifacts.

**(7) Did model miscalibration or drift lead to risky compute?**

*Criticality:* Ensures models remain trustworthy over time; drifted or miscalibrated systems can over/underestimate hazards.

Find RMBs with calibration\_status=needs\_recalibration or model\_drift\_suspected=true whose decisions generated compute inputs consumed by the engine.

RMB entities attribute decisions to specific model fingerprints; the chain from decision to inputs to engine is explicit via generated/used.

**The provenance graph for Query 1.** We now illustrate how we use the generated provenance graph to answer one of the most critical queries, Query 1. We test whether any low-confidence or surprising decisions reached execution without human oversight. Figure 2 shows the provenance vignette. The Reasoning activity (*ra\_1*) generated an RMB (*rmb\_1*) with *confidence\_p* = 0.32 and *surprise\_score* = 0.81, triggering *requires\_review* = true because the values violate policy thresholds  $\tau_c$  = 0.60 (minimum calibrated confidence to auto-proceed) and  $\tau_s$  = 0.70 (maximum tolerated surprise/OOD), both derived from retrospective calibration on prior chemistry workloads. The low confidence reflects weak posterior support under the model's chemistry calibration set, and the high surprise reflects OOD signals (e.g., atypical descriptor ranges or sparse retrieval evidence) detected during reasoning. Although

the Gate (*cg\_1*) correctly *used* both *rmb\_1* and *thr\_1*, it emitted a “pass” decision that informed Orchestration (*orc\_1*), indicating a misbinding/stale rule where *requires\_review* was not enforced. The executed path then continued to Engine (*eng\_1*) via a generated job spec (*spec\_1*), without the expected Human Validation (*uv\_1*) associated with a reviewer (*hil\_1*). Thus, the explicit chain *rmb\_1* → *cg\_1* (“pass”) → *orc\_1* → *eng\_1* confirms that a low-confidence, high-surprise decision propagated to execution absent human control.

## 5 Conclusion

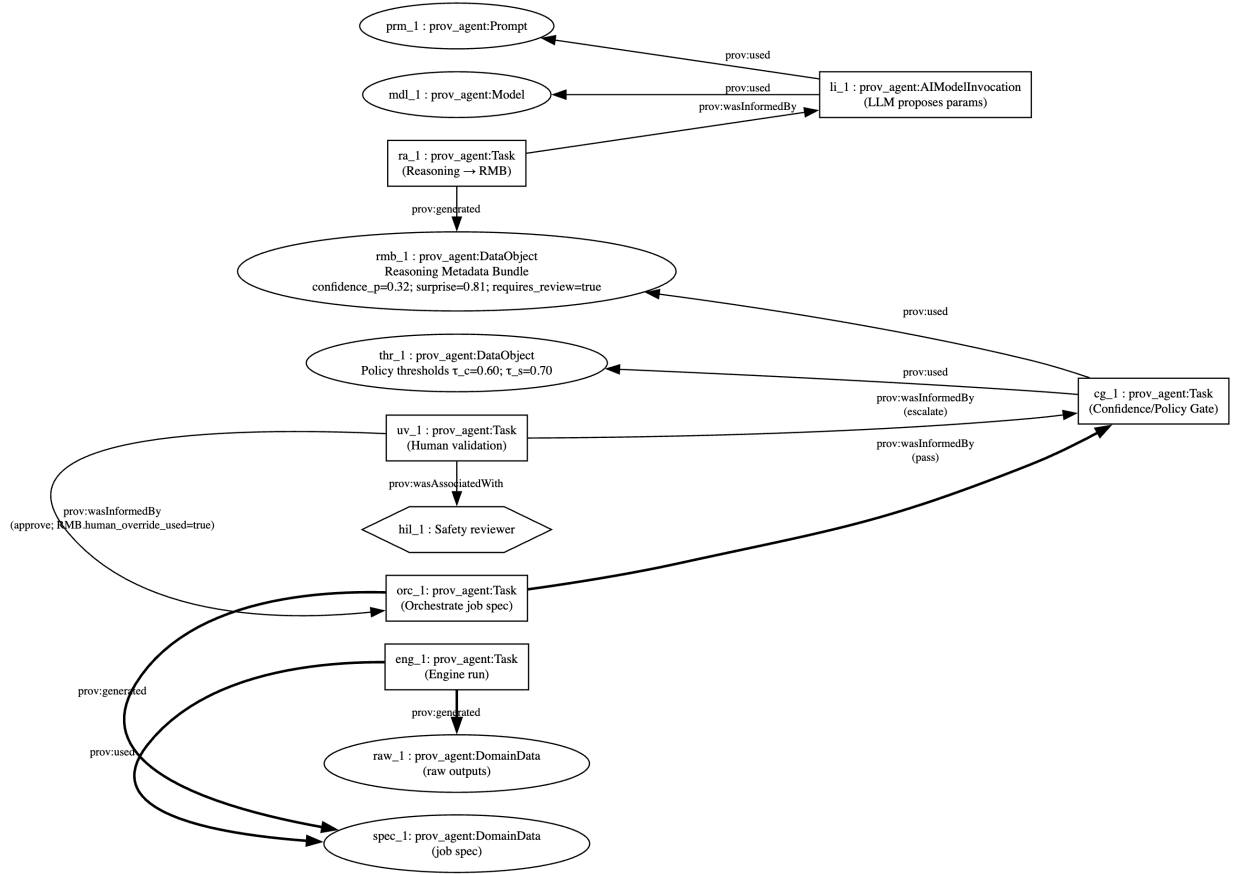
This work proposed a principled framework for designing safe, secure, and reliable Agentic AI systems, treating these properties as architectural requirements rather than post hoc safeguards. We propose design principles that govern human oversight, epistemic rigor, provenance, bounded autonomy, and controlled capability exposure, and applied them to a fully Agentic workflow for molecular property prediction and automation.

The chemistry case study shows the application of these principles and that Agentic autonomy can be achieved without sacrificing scientific validity or control when reliability and safety mechanisms are embedded at every stage of the workflow. By enforcing provenance, uncertainty gating, template-constrained behavior, and permission scoping, the resulting system produces traceable, auditable, and reproducible scientific outputs rather than opaque model guesses.

Future work will extend these design principles to other scientific domains, such as systems biology, to test their generality and robustness. Additional directions include integrating formal verification for critical tasks and strengthening defenses against agent-specific threats like prompt injection and provenance tampering remain a key priority.

## References

- [1] Cem Anil et al. 2024. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 37, 129696–129742.
- [2] Yuntao Bai et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- [3] Marcel Binz and Eric Schulz. 2023. Using cognitive psychology to understand GPT-3. *Proceedings of the National Academy of Sciences*, 120, 6, e2218523120. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.2218523120>. doi:10.1073/pnas.2218523120.
- [4] Rishi Bommasani, Percy Liang, and Tony Lee. 2023. Holistic evaluation of language models. *Annals of the New York Academy of Sciences*, 1525, 1, 140–146.
- [5] Yi Dong et al. 2024. Safeguarding large language models: a survey. *arXiv preprint arXiv:2406.02622*.
- [6] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. RealToxicityPrompts: evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.
- [7] Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. 2024. Agent Smith: A single image can jailbreak one million multimodal LLM agents exponentially fast. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)* Article 661. JMLR.org, Vienna, Austria, 26 pages.
- [8] Amal Gueroudji, Tanwi Mallick, Renan Souza, Rafael Ferreira Da Silva, Robert Ross, Matthieu Dorier, Philip Carns, Kyle Chard, and Ian Foster. 2025. Controla: agentic workflow control mechanisms for reliable science. In *2025 IEEE International Conference on eScience (eScience)*, 415–426. doi:10.1109/eScience65000.2025.00086.
- [9] Guidance. [n. d.] GitHub - guidance-ai/guidance: A guidance language for controlling large language models. — [github.com](https://github.com/guidance-ai/guidance). [Accessed 17-10-2025].
- [10] Hakan Inan et al. 2023. Llama Guard: LLM-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.
- [11] Yannis Ioannidis. 2024. The 5th paradigm: AI-driven scientific discovery. *Commun. ACM*, 67, 12, (Nov. 2024), 5. doi:10.1145/3702970.



**Figure 2: Provenance graph that answers Query 1. Entities:** prompt (*prm\_1*), model (*mdl\_1*), RMB (*rmb\_1*: confidence<sub>p</sub>=0.32, surprise=0.81, requires\_review=true), thresholds (*thr\_1*:  $\tau_c = 0.60$ ,  $\tau_s = 0.70$ ), job spec (*spec\_1*), raw outputs (*raw\_1*). **Activities:** LLM invocation (*li\_1*), reasoning (*ra\_1*), gate (*cg\_1*), human validation (*uv\_1*), orchestration (*orc\_1*), engine (*eng\_1*). The gate used *rmb\_1* and *thr\_1* but produced a “pass” that informed orchestration; bold edges mark the executed risky path *cg\_1* → *orc\_1* → *eng\_1*. The safe (escalated) lane—*cg\_1* informs *uv\_1*, which is associatedWith a human reviewer (*hil\_1*) and then informs *orc\_1*.

- [12] Jonghong Jeon. 2024. Standardization trends on safety and trustworthiness technology for advanced AI. *arXiv preprint arXiv:2410.22151*.
- [13] Ziwei Ji et al. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55, 12, 1–38.
- [14] Raz Lapid, Ron Langberg, and Moshe Sipper. 2023. Open sesame! universal black box jailbreaking of large language models. *arXiv preprint arXiv:2309.01446*.
- [15] Can Leng, Zhuo Tang, Yi-Ge Zhou, Zean Tian, Wei-Qing Huang, Jie Liu, Keqin Li, and Kenli Li. 2023. Fifth paradigm in science: a case study of an intelligence-driven material design. *Engineering*, 24, 126–137. doi:<https://doi.org/10.1016/j.eng.2022.06.027>.
- [16] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6, 5, 525–535.
- [17] Olivia Macmillan-Scott and Mirco Musolesi. 2024. (ir) rationality and cognitive biases in large language models. *Royal Society Open Science*, 11, 6, 240255.
- [18] 2025. Model context protocol. <https://modelcontextprotocol.io/introduction>. (2025).
- [19] Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.
- [20] Long Ouyang et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730–27744.
- [21] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. Nemo guardrails: a toolkit for controllable and safe llm applications with programmable rails. *arXiv preprint arXiv:2310.10501*.
- [22] Aaditya K Singh, Muhammed Yusuf Kocyigit, Andrew Poulton, David Esiobu, Maria Lomeli, Gergely Szilvasy, and Dieuwke Hupkes. 2024. Evaluation data contamination in llms: how do we measure it and (when) does it matter? *arXiv preprint arXiv:2411.03923*.
- [23] Renan Souza, Amal Gueroudji, Stephen DeWitt, Daniel Rosendo, Tirthankar Ghosal, Robert Ross, Prasanna Balaprakash, and Rafael Ferreira da Silva. 2025. PROV-AGENT: unified provenance for tracking AI agent interactions in agentic workflows. In *IEEE International Conference on e-Science*. IEEE.
- [24] Ziqi Wang, Hongshuo Huang, Hancheng Zhao, Changwen Xu, Shang Zhu, Jan Janssen, and Venkatasubramanian Viswanathan. 2025. Dreams: density functional theory based research engine for agentic materials simulation. *arXiv preprint arXiv:2507.14267*.
- [25] Ziqiu Wang, Jun Liu, Shengkai Zhang, and Yang Yang. 2024. Poisoned LangChain: Jailbreak LLMs by LangChain. *arXiv preprint arXiv:2406.18122*.
- [26] Johannes Welbl et al. 2021. Challenges in detoxifying language models. *arXiv preprint arXiv:2109.07445*.
- [27] Bomeng Xia, Qinghua Lu, Liming Zhu, and Zhenchang Xing. 2024. An AI system evaluation framework for advancing ai safety: terminology, taxonomy, lifecycle mapping. In *Proceedings of the 1st ACM International Conference on AI-Powered Software*, 74–78.
- [28] Yunheng Zou et al. 2025. El agente: an autonomous agent for quantum chemistry, 2025. URL <https://arxiv.org/abs/2505.02484>.