Grounded Agency v2: A Standard for Composable, Auditable, World-Modeling Agents

Daniel Bentes (Synapti.ai)
January 2026

Abstract

We propose **Grounded Agency**, an open standard for building agents that can reason and act in real and digital environments with explicit uncertainty, strong provenance, and safe mutation semantics. The standard separates *atomic capabilities* (what an agent can do), *workflows* (how capabilities compose), *policies* (rules and guardrails), and *schemas* (data contracts). Grounded Agency includes (1) a capability ontology of **99** composable primitives with formal input/output contracts and **60** dependency edges; (2) a canonical world-state and event model supporting versioned snapshots, append-only observations, reversible transitions, and typed uncertainty; (3) an identity and authority layer that resolves aliases, merges/splits entities under explicit thresholds, and ranks conflicting claims via authority weights and temporal decay; and (4) a validator that resolves external schema references, infers binding types from schemas, and flags unsatisfied preconditions and type incompatibilities at design time. The result is a publishable baseline for “agents as systems”: reliable composition, bounded risk, and inspectable reasoning traces.

1. Motivation

Modern LLM-based agents increasingly operate beyond text: they read and modify repositories, call tools, and trigger actions in external systems. But agent engineering is currently split across two disconnected vocabularies: **Capability taxonomies** describe what AI can do (e.g., detect/identify/forecast/act), mostly from the perspective of product ideation and model functionality. **Operational primitives** describe what makes agents reliable (checkpointing, provenance, type-safe dataflow, rollback, policy gates). Grounded Agency bridges the two by treating capabilities as *atomic, typed, composable interfaces*, and by making state, evidence, and reversibility first-class. This work is directly motivated by the “capability-level” decomposition of AI examples into actions and inferences (Levels 1–4) from Yildirim et al. (DIS ’23), and by the practical extension mechanisms in Claude Code (skills, subagents, hooks, MCP).

2. Design principles

Grounded Agency is built from three invariants: **I1 — Every claim is grounded.** Any derived fact must reference evidence anchors (raw events, files, URLs, tool outputs), plus a provenance record describing transformations and assumptions. **I2 — Uncertainty is explicit and typed.** Confidence is represented as a probability in [0,1] and categorized as *epistemic*, *aleatoric*, or *mixed*, enabling downstream workflows to gate actions and widen bounds rather than “pretend certainty”. **I3 — Every mutation is reversible and auditable.** Any action that can produce side effects requires a checkpoint and an inverse strategy. Audit logs link every executed action back to the plan, verification

criteria, approvals, and evidence.

3. Standard artifacts

The standard is distributed as a repository with four normative artifact families: **(A) Capability Ontology** (ontology/capability_ontology.yaml): 99 primitives organized by layer, each with input/output JSON-Schema fragments, preconditions (requires/soft_requires), risk classification, default tool permissions, timeouts and retry defaults, and a directed dependency graph (60 edges). **(B) Workflow Catalog + DSL** (workflows/workflow_catalog.yaml, docs/specs/workflow_dsl_spec_v2.md): declarative workflows composed of steps with typed input_bindings, conditions, gates, parallel_groups, recovery_loops, and cross-workflow invocation. **(C) World Modeling Schemas** (docs/schemas/world_state_schema.yaml, docs/schemas/event_schema.yaml): canonical *WorldState* snapshots and append-only *ObservationEvent* logs with versioning, retention, provenance, uncertainty, transition rules, and indexes. **(D) Policy Layer** (docs/policies/*.yaml): identity resolution (namespaces + alias scoring + merge/split rules) and authority trust model (source weights + temporal decay + field-specific authority + override audit requirements).

4. Formal model

Grounded Agency defines four core object types. **Capability** C is a function signature with a contract: C: Input $\rightarrow$ Output, plus metadata (risk, preconditions, tool permissions). Capabilities must be composable: outputs are valid inputs for other capabilities via bindings. **Workflow** W is a directed program of steps over capabilities with explicit dataflow. Steps may execute sequentially, conditionally, in parallel groups, or inside bounded recovery loops. **Skill** S is an implementation packaging of one or more capabilities or workflows as invocable markdown, optionally running inside isolated subagent contexts. Skills are the human-facing boundary. **World State** Ω is a versioned snapshot of entities, relationships, state variables, and derived assertions. Ω is updated only via validated transitions driven by observation events and policies. The model deliberately separates *what happened* (ObservationEvent, append-only) from *what we believe* (WorldState, inferred, versioned), so that beliefs can be revised without rewriting history.

5. Capability levels 1–4 as an extraction lens

Yildirim et al. decompose AI examples into a grammar of *Action* + *Inference* + *Data/Entity/Metric* and organize them into four abstraction levels. At the highest abstraction, they identify eight verbs: **Detect, Identify, Estimate, Forecast, Compare, Discover, Generate, Act**. Grounded Agency treats these as a *capability extraction lens*, but extends them in two ways: 1) **Operationalization:** each capability becomes a typed interface with preconditions and audit requirements, enabling safe composition. 2) **Missing atoms:** real-world agents require primitives that do not appear in ideation-centric taxonomies: retrieve-by-ID, schema modeling, relationship graphing, constraint enforcement, checkpoint/rollback, trust resolution, and workflow invocation.

6. Validator and type inference

The validator performs three classes of checks: **V1 — Referential integrity:** all capabilities referenced by workflows exist; all referenced schema paths ($ref) are resolved into concrete schema trees. **V2 — Preconditions and ordering:** hard requires edges are satisfied prior to invocation, including checkpoint requirements for mutating steps. **V3 — Binding correctness:** for each binding expression (e.g., ${inspect_out.identity.id}) the validator infers its type from the producing step's output schema (including external refs), then checks compatibility with the consumer's expected input schema. Type annotations in bindings are optional. The validator infers types automatically; annotations are only required when schema inference yields a union/anyOf ambiguity (e.g., string|number) or when a binding targets an untyped free-form field. When a mismatch is detected, the validator emits diagnostics (path, expected, actual, confidence) and *coercion hints* via a registry of safe transforms (e.g., string → array<string> by wrapping; number → string via formatting).

7. Canonical digital-twin sync example

The repository includes a full "digital twin sync loop" workflow: ingest raw logs, transform into canonical observation events, resolve entity identities, integrate with authority-aware conflict resolution, compute a new world-state snapshot (versioned), diff it against the previous snapshot, and publish deltas. This workflow demonstrates: deterministic ETL (transform mapping), identity resolution (namespaces + alias scoring), authority trust weighting with temporal decay, schema conformance assertions, and safe mutation gates (checkpoint before side-effectful publication).

8. Related work

Grounded Agency is adjacent to, but distinct from, several research and systems lines: • **LLM agent prompting patterns:** ReAct combines reasoning traces with tool actions; Reflexion adds self-evaluation and memory to improve performance over iterations. Grounded Agency complements these by defining typed capability interfaces, auditability, and reversible state transitions. • **Cognitive architectures:** Soar and ACT-R decompose cognition into modules and productions. Grounded Agency borrows the "atomic primitive" idea, but targets pragmatic engineering constraints: schemas, evidence, and operational safety. • **Agent product ideation taxonomies:** Yildirim et al.'s four-level capability resource provides an empirical lens for describing AI features. Grounded Agency extends it toward execution-time systems: identity, authority, conflict resolution, and workflow validation. • **Claude Code extension system:** Claude Code's skills/subagents/hooks/MCP demonstrate a practical layering of agent extension points. Grounded Agency provides a standards-grade data and capability substrate that can be implemented as skills and composed into workflows.

9. Limitations and scope

Grounded Agency standardizes the interfaces, contracts, and safety semantics of agent workflows; it does not prescribe a specific planner, model, or tool runtime. It also does not solve open problems such as long-horizon autonomous governance, value alignment, or real-time actuation safety in physical robotics. Instead, it makes these concerns *legible*: every decision becomes inspectable, every mutation is gated, and every claim has a traceable lineage.

10. Conclusion

Grounded Agency turns “agents” from ad-hoc prompt scripts into systems with contracts. By combining a capability ontology, workflow DSL, world modeling schemas, identity + authority policy, and a schema-aware validator with automatic type inference, the standard provides a concrete baseline for composable and auditable agent engineering. The v10 repository acts as both *specification* and *reference implementation*, and is designed to be extended: new capabilities can be added without breaking existing workflows, and new domains can be modeled by extending the entity taxonomy and transform mappings.

References

[R1] Nur Yildirim et al. “Creating Design Resources to Scaffold the Ideation of AI Concepts.” DIS ’23. https://doi.org/10.1145/3563657.3596058

[R2] Claude Code Docs: “Extend Claude Code.”

[R3] Claude Code Docs: “Skills.”

[R4] Anthropic Engineering: “Equipping agents for the real world with Agent Skills.”

[R5] Shunyu Yao et al. “ReAct: Synergizing Reasoning and Acting in Language Models.” OpenReview (ICLR 2023).

[R6] Noah Shinn et al. “Reflexion: Language Agents with Verbal Reinforcement Learning.” OpenReview (NeurIPS 2023).

[R7] John E. Laird. “The Soar Cognitive Architecture.” MIT Press, 2012.

[R8] Frank E. Ritter et al. “ACT-R: A cognitive architecture for modeling cognition.” WIREs Cognitive Science, 2019.