Nick Clark · Published August 16, 2026 · PDF

Cordis Composability and Governed Agent Execution

Cordis, as publicly described, is a meta-framework of spatiotemporal composability: a reversible plugin system in which components mount, unmount, and hot-reload at runtime, and in which, as publicly described, a component's side effects can be completely reverted upon removal. United States Patent Application 19/230,933 describes an architecture in which each memory-bearing semantic agent carries a policy reference field, and in which the disclosure requires that field to be evaluated at runtime prior to any mutation, delegation, or propagation, with the action permitted or denied based on validation of the referenced policy. The comparison drawn here is between a composition model whose public description centers on reverting effects and a filed architecture that requires an authorization predicate to run before the act. Both concern change in a live system.

Two Different Questions About Change

A long-running system that loads and unloads parts of itself has to answer two questions about every change, and they are not the same question.

The first is a question about cleanup. If a component is installed and later removed, what state does it leave behind? Event listeners, timers, registered routes, cached handles, and entries in shared registries all outlive the component that created them unless something tracks and reverses them. Where nothing tracks them, the residue accumulates across reload cycles, and the eventual failure tends to surface far from the component that caused it.

The second is a question about authorization. Before a change is applied, is it one that this component, in this context, is allowed to make? An action that is later undone was still taken, and some actions cannot be undone by restoring memory state. An outbound message, a task delegated to another party, or a privilege a component granted itself and then used are events in the world before they are entries in a runtime log.

Neither question subsumes the other. Cleanup is about the aftermath of removal. Authorization is about the threshold of the act.

What Public Materials Describe About Cordis

Cordis is publicly described as a Meta-Framework of Spatiotemporal Composability, a framework for building frameworks. Public materials describe its core as a reversible plugin system, in which components can be mounted, unmounted, and hot-reloaded at runtime. It has served, as publicly described, as the foundation of the open-source Koishi chatbot framework for about four years. Public materials also state that Koishi runs Cordis v3 and that DeepSeek Harness runs v4.

A draft paper from DeepSeek and Peking University, titled "A Programming Paradigm for Spatiotemporal Composability" and dated August 13, 2026, gives the model a formal treatment. As publicly described, the paper defines temporal composability as "the ability to completely revert a component's side effects upon removal," and defines spatial composability in terms of declaring inter-component dependencies and managing them reactively. The formalization includes revertible effects, described in

public materials as every context transformation carrying an inverse that the runtime tracks, along with reactive coeffects, a unified context type, and a calculus of dynamic composition.

Those are the public descriptions, and this article takes them as stated and does not extend them. Two of them carry most of the weight for the comparison that follows. The first is the definition of temporal composability as complete reversion of a component's side effects upon removal, together with the statement in public materials that the runtime itself tracks the inverse of every context transformation. The second is the definition of spatial composability in terms of declaring inter-component dependencies and managing them reactively, which is a statement about how components find and respond to one another. Nothing beyond those public descriptions is assumed here.

Policy Reference Evaluated Before the Act

United States Patent Application 19/230,933 describes a cognition-native semantic execution platform in which the unit of execution is a memory-bearing semantic agent object. In the disclosed embodiments, a semantic agent 200 carries six structured fields: an intent field 201, a context block 202, a memory field 203, a policy reference field 204, a mutation descriptor field 205, and a lineage field 206. The disclosure describes these fields as carrying, inside the agent itself, the information needed to determine how the agent should behave, under what conditions it may mutate or delegate, and where it may be routed or rehydrated.

The policy reference field 204 is where the authorization question is answered. In the described embodiments it holds one or more cryptographically signed links to semantic policy contracts defining permissible behaviors, and the disclosure describes evaluating that field at runtime prior to any mutation, delegation, or propagation of the agent, with the action permitted or denied deterministically based on validation of the referenced policy, rather than by centralized authorization or post-execution filtering.

The middleware architecture 300 of FIG. 3 shows where that evaluation sits in the lifecycle. An agent arrives on the incoming agent bus 301 and enters the semantic router 302, which parses the context field to select the governance domain, or trust zone, in which the agent is eligible to execute. The structural validator 303 checks that required fields are present and valid, diverting incomplete agents to the delegation and fallback engine 304, which attempts reconstruction from contextual inference, lineage resolution, or local scaffolding, and which may quarantine or defer the agent if reconstruction fails. Only then does the agent reach the policy enforcement engine 305, which evaluates the embedded policy reference against the active trust zone governance, including cryptographic signature verification, scope parsing, and mutation eligibility assessment. Agents that clear that gate enter the mutation queue 306, where the mutation descriptor field is parsed against permitted mutation pathways, and validated mutations are recorded in the memory field as traceable events. The execution graph manager 307 maintains lineage across mutation events, delegation records, fallback resolutions, and zone transitions. The propagation interface 308 evaluates eligibility to leave the local substrate, using zone alignment and trust slope continuity between the agent hash and the substrate hash.

Trust zones 150 are described as scoped governance domains layered over memory-resident environments called nests. The disclosure draws the distinction directly: a nest determines what memory an agent can reach, while a zone determines what the agent is permitted to do within that memory. FIG. 8 shows a mutation governance sequence 800 in which Agent_X 801 submits a mutation request inside Trust Zone A 802 along with its semantic state, memory trace, and policy reference. Decentralized policy validators 810, shown as Validator_1 through Validator_n at 811 through 814, each evaluate the proposal against the memory field, the mutation descriptor 205, and the zone's active policy contract, and each issues a recorded vote. On quorum approval the agent transforms into a new instance 820 with extended memory and updated lineage.

Without quorum, rollback or quarantine 825 pauses execution and freezes the memory field. Contested cases may escalate to a meta-policy layer 830, which either authorizes an override at 831 or finalizes the quarantine at 832.

FIG. 10 illustrates the self-modification case, which is the sharpest version of the authorization question. Agent_X 1001 proposes altering its own mutation descriptor field to permit downstream delegation without quorum validation. Because the proposed change concerns the agent's own structural privileges, the meta-policy contract 1006 is invoked, and the Adaptive Consensus Protocol Layer 1005 extracts conditions under which such a change may be permitted, described in this example as requiring prior scoped validator consensus or lineage-based authorization. Where those preconditions are unmet, the described substrate denies the mutation, isolates the agent, and initiates semantic quarantine 1009, with rollback to the last verified state available in certain configurations. The denial itself is recorded in the agent's trace, so a refused action leaves an audit record.

Convergence in Category, Divergence in Layer

Both architectures treat runtime change as a first-class concern rather than an exception, and both locate their answer inside the runtime rather than in a build step or an external orchestrator. That is the convergence, and it is real.

The divergence is which moment in the change each one instruments. As publicly described, the reversible plugin model pairs each context transformation with an inverse the runtime tracks, so that a component's side effects can be reverted when the component is removed. The filed architecture instruments the predicate that runs before the transformation, and requires that predicate to travel inside the moving object. The signed policy reference sits in field 204 and moves with the agent, so an agent that migrates from one nest to another and into a new zone is, in the described embodiments, evaluated against the governance active where it lands rather than against the governance it left.

The problems addressed differ accordingly. Untracked side effects that accumulate across reload cycles are one problem. An action that should not have been taken, taken by a component that had the capability to take it, is another. The disclosed embodiments also treat the record of refusal as an output: validator votes are cryptographically recorded and, where required, appended to the memory field 203 for later auditability or trust slope analysis, and a denied self-modification is recorded in the agent's trace.

Coexistence and Boundaries

A plausible deployment runs the two at different levels of the same stack. A composition framework of the kind publicly described handles the mechanics of loading, unloading, and dependency reactivity for the modules that make up a service, with a removed module's side effects reverted on removal as those materials describe. An agent layer of the kind described in the filing governs what running components are permitted to do to each other and to the outside world: whether a given mutation of an agent's own descriptor is allowed, whether a delegation may be handed onward, whether an agent may propagate out of its current substrate. The first concerns the state left behind when a component departs. The second, in the described embodiments, produces a decision before the action and a record of what was denied.

The disclosed architecture leaves a good deal to other layers. Its fallback path handles structurally incomplete agents through inference and scaffolding, which is a different problem from unwinding the side effects of a departing module. Rollback in the described embodiments restores an agent to its last verified state within the semantic memory field, a scope defined by the agent's own record rather than a general inverse for every effect a component may produce. A team that wants clean unloading semantics for its modules and a policy predicate before consequential agent actions is describing two pieces of infrastructure, not one. Composability and governance are separable design axes, and the disclosed subject matter is aimed at the second.

Disclosure Scope

This article is a technical description of subject matter disclosed in United States Patent Application 19/230,933, a pending application. It describes embodiments and reference numerals as they appear in that filing and does not characterize the scope of any claim. Nothing here should be read as a statement of what any claim covers or requires.

References to Cordis are to public materials and are used for comparison only; no relationship, endorsement, or infringement is asserted. No assertion is made that Cordis practices, overlaps with, or was influenced by the subject matter of the cited filing, and no comparison here implies any order of development between the two.

Outcomes described in the filing are conditioned on the bounds the filing declares: on the presence and validity of a signed policy reference, on the governance active in the trust zone where an agent executes, and on the validator and meta-policy conditions defined for that zone.

Execution Platform (/execution-platform)

All 49 steps → (/inventive-steps)

The complete runtime for governed, persistent agents.

U.S. 19/230,933 (/patents/19-230933)

PRIMARY TECHNICAL DISCLOSURE

- A Cognition-Native Execution Platform for Distributed, Stateful, and Governable Agents (/articles/a-cognition-native-execution-platform-for-distributed-stateful-and-governable-agents)

SECONDARY TECHNICAL

- Six-Field Canonical Agent Schema: Structural Definition of Autonomous Semantic Agents (/articles/execution-platform/canonical-schema)
- Semantic Nest Instantiation: Dynamic Execution Environments From Agent Density and Entropy (/articles/execution-platform/nest-instantiation)

- Trust Zone Overlay Governance: Logical Policy Domains Independent of Network Topology (/articles/execution-platform/trust-zone-overlay)
- Scoped Quorum Mutation Validation: Independent Validators With Meta-Policy Escalation (/articles/execution-platform/quorum-validation)
- Meta-Policy Override Resolution: Higher-Level Governance for Local Quorum Decisions (/articles/execution-platform/meta-policy-override)
- Semantic Router: Schema-Aware Propagation Replacing Address-Based Forwarding (/articles/execution-platform/semantic-router)
- Dynamic Agent Hash Derivation: Deterministic Identity From Memory and Mutation History (/articles/execution-platform/dah-derivation)
- Dynamic Device Hash Derivation: Substrate Identity From Device-Local Entropy (/articles/execution-platform/ddh-derivation)
- Content Anchor Hash Derivation: Perceptual Identity for Non-Executing Digital Content (/articles/execution-platform/cah-derivation)
- DAH-DDH Slope Entanglement: Binding Agent Identity to Host Device Lineage (/articles/execution-platform/dah-ddh-entanglement)
- Trust Slope Validation Across Zone Migration: Continuity Verification With Quarantine (/articles/execution-platform/zone-migration)
- Pseudonymous Propagation: Recognition by Slope Rather Than Global Identifier (/articles/execution-platform/pseudonymous-propagation)
- Alias Slope-Band Indexing: Symbolic Resolution Through Slope-Indexed Anchor Pathfinding (/articles/execution-platform/slope-band-indexing)
- Fallback Rehydration: Recovering Partial Agents Through Contextual Policy Inference (/articles/execution-platform/fallback-rehydration)
- Structural Validator With Fallback Routing: Schema Verification Before Execution (/articles/execution-platform/structural-validator)
- Execution Graph Manager: Structured Lineage of Agent Reasoning and Transformation (/articles/execution-platform/execution-graph)
- Full and Partial Agent Interoperability: Cross-Boundary Semantic Exchange Under Policy (/articles/execution-platform/agent-interoperability)
- Cross-Topology Substrate Deployment: Identical Agent Structure Across All Substrates (/articles/execution-platform/cross-topology)

APPLICATIONS · GENERAL

- Governable AI Agents: Auditable Reasoning, Policy-Constrained Orchestration, and Training-Artifact Traceability (/articles/execution-platform/ai-agent-governance)

- Multi-Cloud Agent Orchestration Without a Centralized Scheduler (/articles/execution-platform/multi-cloud-orchestration)
- Autonomous Fleet Coordination Through Self-Governing Agents (/articles/execution-platform/fleet-coordination)
- Enterprise Workflow Automation Without Orchestration Servers (/articles/execution-platform/enterprise-workflow-automation)
- Smart Contract Alternative Without Blockchain Latency: Governed Contract Execution (/articles/execution-platform/smart-contract-alternative)
- Reproducible Scientific Computing With Provenance-Bearing Governed Agents (/articles/execution-platform/scientific-computing)
- Supply Chain Autonomous Agents (/articles/execution-platform/supply-chain-agents)
- Distributed Energy Grid Management With Governed Autonomous Agents (/articles/execution-platform/energy-grid-management)
- Disaster Response Coordination Without Central Command (/articles/execution-platform/disaster-response-coordination)
- Sovereign Agent Runtimes: Running AI Agents Air-Gapped and On-Premises for Defense and Regulated Industries (/articles/execution-platform/sovereign-agent-runtimes)
- When an Agent Outlives the Server That Started It (/articles/execution-platform/execution-platform-stakes)

APPLICATIONS · SPECIFIC

- Kubernetes Orchestrates Containers. It Does Not Know What They Are Doing. (/articles/execution-platform/kubernetes)
- Temporal Alternative for Governed Agent Execution: Durable Workflows Have No Semantic Identity (/articles/execution-platform/temporal-io)
- Apache Airflow vs. Governed Agent Execution: DAG Scheduling or Agent-Level Governance? (/articles/execution-platform/apache-airflow)
- Prefect Alternative for Governed Agent Execution: Beyond Python Task Scheduling (/articles/execution-platform/prefect)
- AWS Step Functions Alternative for Governed Agent Execution (/articles/execution-platform/aws-step-functions)
- Azure Durable Functions vs a Governed Execution Platform: Where Does Step Authority Live? (/articles/execution-platform/azure-durable-functions)
- HashiCorp Nomad vs. a Governance-Bearing Execution Platform: Where Does Workload Authority Live? (/articles/execution-platform/nomad)
- Docker Swarm Alternative for Governed Agent Execution: Beyond Opaque Containers (/articles/execution-platform/docker-swarm)

- Apache Mesos Managed Datacenter Resources. The Resources Had No Semantic Governance. (/articles/execution-platform/mesos)
- Argo Workflows Alternative for Governed Pipelines: Kubernetes-Native DAGs Without a Governance Substrate (/articles/execution-platform/argo-workflows)
- Dagster Alternative for Governed Pipelines: Software-Defined Assets Without a Governance Substrate (/articles/execution-platform/dagster)
- Luigi Alternative for Governed Agent Execution: Beyond Task-Dependency Pipelines (/articles/execution-platform/luigi)
- Camunda vs Governed Agent Execution: BPMN Orchestration Beyond the Process Engine (/articles/execution-platform/camunda)
- Zeebe vs Governed Agent Execution: Does Governance Scale With Throughput? (/articles/execution-platform/zeebe)
- AWS RoboMaker vs Governed Agent Execution at the Fleet Edge (/articles/execution-platform/aws-robomaker)
- NVIDIA Cosmos vs Governed Agent Execution: World Models Need a Runtime (/articles/execution-platform/nvidia-cosmos)
- NVIDIA DRIVE vs Governed Agent Execution: A Cross-Vehicle Substrate Alternative (/articles/execution-platform/nvidia-drive)
- NVIDIA Isaac vs a Governed Agent Execution Substrate (/articles/execution-platform/nvidia-isaac)
- NVIDIA Metropolis vs Governed Agent Execution: A Metropolis Alternative for Edge Cognition (/articles/execution-platform/nvidia-metropolis)
- LangGraph (LangChain) alternative: where does agent state, policy, and lineage actually live? (/articles/execution-platform/langgraph)
- Microsoft AutoGen vs a substrate-embedded execution platform: where does agent orchestration state live? (/articles/execution-platform/microsoft-autogen)
- CrewAI alternative: where does delegation and policy state live at runtime? (/articles/execution-platform/crewai)
- Ray (Anyscale) alternative: where does governance and identity live when agents move between nodes? (/articles/execution-platform/ray-anyscale)
- **Cordis Composability and Governed Agent Execution (/articles/execution-platform/cordis-composability)**

HOW-TO GUIDES

- How to Enforce AI Agent Permissions Before Execution Instead of After (/articles/guides/enforce-agent-permissions-before-execution)
- How to Move a Running AI Agent's State Between Servers Without Losing It (/articles/guides/move-agent-state-between-servers)

- How to Run Untrusted Third-Party AI Agents Safely on Your Platform (/articles/guides/run-untrusted-third-party-agents-safely)
- How to Scope What a Delegated AI Subagent Is Allowed to Do (/articles/guides/cryptographically-scoped-agent-delegation)

TERMINOLOGY

- cognition-native semantic execution platform (/glossary#cognition-native-semantic-execution-platform)
- dynamic agent hash (/glossary#dynamic-agent-hash)
- memory-bearing semantic agent object (/glossary#memory-bearing-semantic-agent-object)
- memory-native substrate (/glossary#memory-native-substrate)
- slope entanglement (/glossary#slope-entanglement)
- trust slope (/glossary#trust-slope)

Execution Platform overview → (/execution-platform)