CrewAI Organized Agents Into Teams. The Agents Still Have No Schema.

by Nick Clark | Published March 27, 2026 | PDF

CrewAI introduced role-based agent teams where agents with defined roles, goals, and backstories collaborate on sequential or parallel tasks. The metaphor is intuitive: assemble a crew, assign roles, define tasks, and let them work. But CrewAI agents are role descriptions attached to LLM instances, not structurally defined objects with typed fields for governance, memory, identity, and capabilities. The gap is between team orchestration and structural agent definition.

CrewAI made agent teams accessible by providing a clean API for role assignment and task delegation. The role-goal-backstory pattern is an effective way to configure LLM behavior. The gap described here is not about the team metaphor. It is about the structural foundation underneath it.

Roles are descriptions, not schemas

A CrewAI agent has a role (string), a goal (string), a backstory (string), and optionally tools and an LLM. The role defines what the agent does in natural language. The goal describes what it aims to achieve. The backstory provides context.

These are LLM prompt components, not typed schema fields. Two agents with different role strings but the same LLM and tools are structurally identical at the platform level. There is no typed field that distinguishes them structurally. The distinction exists only in the prompt.

Delegation without governance

CrewAI supports agent delegation: one agent can delegate a task to another. But delegation has no governance model. There is no trust slope validation between delegator and delegate. There is no capability envelope check to verify the delegate can handle the task. There is no governance policy that constrains what can be delegated.

In a canonical agent schema, delegation would be governed by the typed fields of both agents. The delegator's policy reference would specify delegation constraints. The delegate's capability envelope would be checked against the task requirements. Trust slope continuity between the agents would be validated. Delegation would be a governed operation, not an unchecked handoff.

What a canonical agent schema provides

A canonical agent schema gives every agent a structural definition with typed fields: identity, memory, governance, capabilities, execution state, and lineage. CrewAI's role-goal-backstory pattern would map to specific fields within this schema. Roles would become typed capabilities. Goals would become intent declarations validated against policy. Backstory would contribute to the agent's memory and identity.

With a canonical schema, CrewAI teams would gain structural governance. Each team member would be a validated agent object. Delegation would be governed. Task execution would be validated against capabilities. The team would not just collaborate. It would collaborate under structural constraints.

The remaining gap

CrewAI made agent teams intuitive. The remaining gap is in what those agents structurally are: typed objects with governance, memory, and capabilities as intrinsic fields, not natural language descriptions attached to LLM instances. That structural foundation is what makes teams governable at scale.

Agent Schema All 21 steps →

Define what an autonomous agent is — structurally.

Patent
US 19/452,651 · filed
Primary Technical Disclosure
○ Cognition-Compatible Semantic Agent Objects and Structural Validation
Secondary Technical
○ Partial Agent Structural Validity: Fewer Fields, Still Deterministic○ Minimum Two-Field Validation Threshold: The Floor of Semantic Structure○ Field Interaction Rules: Deterministic Constraints Between Canonical Fields○ Field-Based Role Typing: Agent Roles Derived From Structural Composition○ Semantic Templates: Predefined Field Arrangements as Agent Class Contracts○ Structural Scaffolding Logic: Resolving Missing Fields Through Inference or Defaulting○ Field-Aware Default Resolution: Deterministic Behavior When Fields Are Absent○ Traceable Semantic Lineage Graph: Mutation History Embedded in Agent Objects○ Serialization With Stateless Compatibility: Reconstruction Without External Session State○ Schema Governance Through Versioned Policies: Cross-Version Structural Interoperability
Applications (General)
○ Enterprise AI Agent Interoperability Through Canonical Schema○ Robotic System Standardization via Structural Field Composition○ Multi-Vendor AI Agent Interoperability○ Digital Twin Standardization Through Canonical Fields○ Healthcare AI Agent Portability○ Defense Coalition Interoperability○ Insurance Claims Processing Through Standard Agents○ Legacy System Integration via Schema Bridging
Applications (Specific)
○ LangChain Built the Agent Framework. It Did Not Define What an Agent Is.○ AutoGen Enabled Multi-Agent Conversations. The Agents Have No Structural Definition.● CrewAI Organized Agents Into Teams. The Agents Still Have No Schema.○ Semantic Kernel Integrated AI Into Enterprise Code. The Agents It Creates Have No Schema.○ OpenAI Assistants API Provides Agent Tooling. It Does Not Define Agent Structure.○ Google Vertex AI Agents Provide Managed Agent Infrastructure. The Agents Have No Canonical Schema.○ Amazon Bedrock Agents Orchestrate Foundation Models. The Agents Have No Structural Definition.○ Haystack Built Composable NLP Pipelines. The Pipeline Components Have No Agent Schema.○ LlamaIndex Built the Data Framework for LLM Applications. The Data Objects Have No Agent Schema.○ Dify Made LLM Application Development Visual. The Applications Have No Agent Schema.
Agent Schema overview →
AQ
deterministic
autonomy

Last updated: 2026-03-03

-
- Inventive Steps
- Licensing
- Patents
- Articles
- Legal
- Opportunities
- Sitemap

-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie