AutoGen Enabled Multi-Agent Conversations. The Agents Have No Structural Definition.

by Nick Clark | Published March 27, 2026 | PDF

Microsoft's AutoGen made multi-agent conversation patterns practical, enabling agents to collaborate through structured message passing with human-in-the-loop capabilities. The conversation patterns are flexible and powerful. But AutoGen agents are defined by their conversational role and system message, not by a canonical schema with typed fields for governance, memory, identity, and capabilities. The structural gap is between multi-agent conversation orchestration and structural agent definition.

AutoGen's contribution to multi-agent research and practice is valuable. Its conversation patterns, code execution support, and group chat abstractions opened new possibilities for agent collaboration. The gap described here is not about conversation orchestration. It is about the structural identity of the agents that converse.

Agents are roles, not structures

An AutoGen agent is primarily defined by its system message, LLM configuration, and conversation behavior. An AssistantAgent has a system prompt that describes its role. A UserProxyAgent represents human input. A GroupChat coordinates multiple agents.

But none of these agents have a typed schema. There is no field for governance policy. There is no field for capability envelope. There is no lineage tracking mutations. The agent's identity is its name string and system message. Two agents with the same system message are functionally identical. There is no structural distinction.

Multi-agent collaboration without governance

When AutoGen agents collaborate in a group chat, the orchestration determines speaking order and termination conditions. But there is no governance validation at each turn. An agent that has lost trust, exceeded its capability envelope, or violated its policy reference continues to participate because the orchestration has no model of governance.

In a canonical agent schema, every agent contribution would be validated against the agent's typed fields. An agent whose confidence has dropped below threshold would be structurally prevented from contributing. An agent whose integrity has deviated would have its contributions weighted accordingly. This requires the platform to understand agent schema, not just conversation sequence.

What a canonical agent schema provides

A canonical agent schema defines six typed fields: identity, memory, governance, capabilities, execution state, and lineage. Every agent, whether it participates in multi-agent conversations or operates independently, carries these fields as structural components.

With a canonical schema, AutoGen's conversation patterns would gain structural governance. Each agent's contribution would be validated against its typed fields. Multi-agent collaboration would inherit the governance properties of the participating agents. The conversation would be not just orchestrated but governed.

The remaining gap

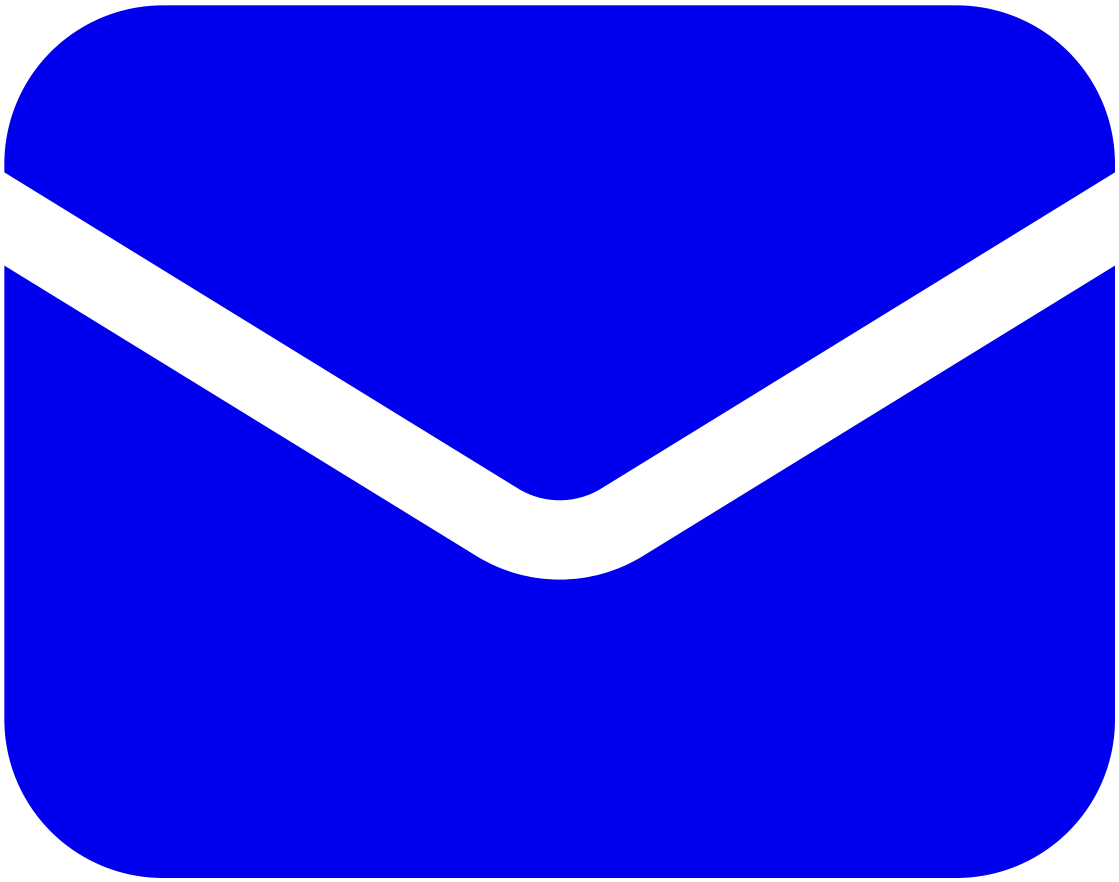
AutoGen made multi-agent conversations practical. The remaining gap is in what agents structurally are: typed objects with governance, memory, identity, and capabilities as intrinsic fields rather than conversational roles defined by system messages. That structural definition is what makes agents governable.

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



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

Invented by Nick Clark | Founding Investors: Devin Wilkie