From Pipelines to Agents: Reconsidering the Architecture of Information Fusion Systems in CopForge

CINN - TI/IA

December 16, 2025

Abstract

This document examines the concept of agency in the context of Large Language Model (LLM)-based systems, arguing that agency should be understood as a spectrum rather than a binary classification. We present a critical analysis of the initial pipeline-based architecture adopted in the TIFDA and early CopForge projects for information ingestion and fusion, concluding that such architectures function as "sophisticated translators" rather than true agents. Subsequently, we describe the redesigned CopForge Ingest Agent, which implements a genuine agentic architecture through an LLM reasoning loop. This architectural shift enables contextual decision-making, proactive duplicate detection, and intelligent multi-sensor correlation—capabilities that were not possible with the previous fixed-flow approach.

1 Introduction: The Spectrum of Agency

The term "agent" has become ubiquitous in the field of artificial intelligence, yet its definition varies substantially across different frameworks and research communities. Rather than constituting a binary property, agency can be more accurately conceptualized as a *spectrum* that ranges from purely deterministic functions to fully autonomous systems capable of self-directed goal pursuit. Understanding where a given system falls on this spectrum has significant implications for architectural decisions, capability expectations, and the selection of appropriate communication protocols.

Anthropic, the organization behind Claude, employs a relatively strict definition of agency. According to this perspective, a true agent must exhibit: (i) significant autonomy in decision-making, (ii) the capacity for planning and goal-directed behavior, (iii) the ability to select and use tools based on self-determined necessity, and (iv) persistence of objectives across multiple interactions. Under this definition, many systems commonly labeled as "agents" would more accurately be classified as tool-augmented language models or orchestrated pipelines. However, alternative perspectives exist in the literature and industry practice. LangChain defines an agent as any system that uses an LLM to decide which actions to take. CrewAI conceptualizes agents as specialized roles equipped with objectives and tool access. The classical AI definition from Russell and Norvig [1] characterizes an agent as any entity that perceives its environment and acts upon it. In the reinforcement learning paradigm, an agent is defined as an entity that maximizes cumulative reward through sequential action selection.

These divergent definitions reflect genuine differences in the design philosophy and intended use cases of various systems. The key insight is that agency admits of degrees, and the appropriate level of agency for a given application depends on the specific requirements, constraints, and trade-offs involved.

2 The Agency Spectrum: A Taxonomy

To operationalize the concept of agency as a spectrum, we propose a taxonomy based on the degree of autonomous decision-making exhibited by a system. This agency vary along at least four orthogonal dimensions:

1. Control of Execution Flow
    - Externally controlled: execution order is fixed (pipelines, DAGs).
    - Conditionally routed: limited branching determined by predefined rules.
    - Model-controlled: the LLM decides which step to execute next.
2. Knowledge Source
    - Parametric only: the model relies solely on its internal weights.
    - Augmented: retrieval, databases, or external state supplement the model.
3. Tool Autonomy
    - No tools: pure text generation.
    - Callable tools: tools invoked only when explicitly instructed.
    - Selectable tools: the model decides which tool to use and when.
4. Temporal Coherence
    - Stateless: each invocation is independent.
    - Session-scoped: short-term memory within a request.
    - Persistent: objectives and context span multiple interactions.

A system becomes "more agentic" not when it uses an LLM, but when control flow, tool choice, and temporal structure migrate from code into model reasoning.

Table 1 presents a practical example of this taxonomy with representative elements.

Table 1: The spectrum of agency in LLM-based systems

Level	**Characteristics**	**Example**	**Decision Capability**
Pure Function	Deterministic, stateless	Traditional parser	None
MCP Tool	Invocable, no agenda	`find_duplicates`	None (executes on demand)
LangGraph Node	State + conditional routing	Pipeline with if/else	Predefined branching
CrewAI Agent	Role + objective + tools	Specialized worker	Within assigned scope
Autonomous Agent	Dynamic planning, tool selection	AutoGPT, reasoning loop	Full autonomy

At the lowest end of the spectrum, pure functions and MCP tools exhibit no agency whatsoever—they execute precisely what they are instructed to do, without deviation or initiative. LangGraph nodes with conditional routing represent a modest increase in apparent agency: the system can take different paths based on runtime conditions, but the set of possible paths is statically defined

at design time. The LLM within such a node performs a specific task (e.g., entity extraction) but does not decide *which* task to perform or *whether* to perform it.

CrewAI agents occupy an intermediate position. Each agent has a defined role, an objective, and access to tools, and can exercise judgment within its assigned scope. However, the overall workflow is typically orchestrated externally, and the agent's autonomy is bounded by its role definition. At the highest end of the spectrum, fully autonomous agents determine dynamically which actions to take, which tools to invoke, and when to terminate. They operate in a continuous reasoning loop, observing the results of their actions and adapting their strategy accordingly.

3 The Multi-Agent Ambiguity: CrewAI as a Case Study

The CrewAI framework illustrates an interesting ambiguity in the definition of agency. In a CrewAI application, multiple "agents" are defined in a configuration file, each with its own role, goal, backstory, and tool access. These agents collaborate within a "Crew" to accomplish a complex task.

This raises a definitional question: is the agent each individual CrewAI agent, or is the agent the Crew as a whole? From a reductionist perspective, each CrewAI agent qualifies as an agent because it has its own LLM, objective, and decision-making capacity. From a holistic perspective, however, the individual "agents" are more akin to specialized organs within a single organism—they only make sense in the context of the Crew, and their collective behavior constitutes the true agentic system. This ambiguity suggests that the level of analysis (individual vs. system) is itself a design choice with practical implications. For inter-system communication using protocols such as A2A (Agent-to-Agent)[6], it matters whether the protocol endpoint corresponds to a single role or to an entire collaborative system.

4 From TIFDA to CopForge: Recognizing the "Sophisticated Translator"

The TIFDA project[7] (Tactical Information Fusion and Dissemination Agent) and the early CopForge[8] architecture adopted a pipeline-based approach to information ingestion, normalization, and fusion with the Common Operational Picture (COP). In this architecture, sensor messages traversed a fixed sequence of processing stages:

1. **Firewall**: Security validation (injection detection, coordinate validation)
2. **Parser**: Format-specific parsing (ASTERIX, drone telemetry, radio intercepts)
3. **Multimodal**: Processing of associated files (audio transcription, image analysis)
4. **LLM Extract**: Entity extraction and enrichment using a language model
5. **Validate**: Output validation against the EntityCOP schema
6. **COP Update**: Insertion into the Common Operational Picture

This architecture, illustrated in Figure 1, was implemented using LangGraph as a directed acyclic graph (DAG) with conditional edges. While the presence of an LLM in the extraction stage might suggest agency, a closer analysis reveals that this system functions as a *sophisticated translator* rather than a true agent.

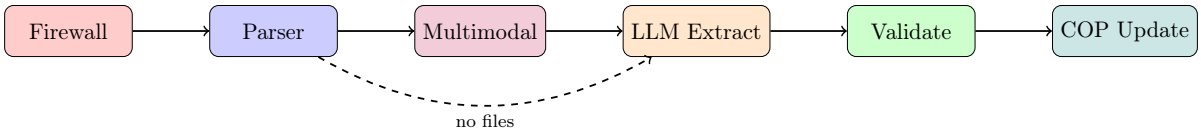

Figure 1: The pipeline architecture: a fixed DAG with conditional routing

The critical observation is that the LLM in the `llm_extract` node does not *decide* what to do—it simply performs entity extraction on whatever input it receives. The flow of control is entirely determined by the DAG structure and the conditional routing logic. The LLM never:

- Queries the COP to obtain context about existing entities
- Decides whether to check for duplicates
- Chooses which tools to invoke
- Adapts its strategy based on intermediate results

Consequently, despite the sophistication of its individual components, this architecture occupies a relatively low position on the agency spectrum—closer to a LangGraph node with routing than to a true autonomous agent.

5 The CopForge Ingest Agent: True Agentic Architecture

Recognizing the limitations of the pipeline approach, we redesigned the CopForge Ingest Agent to implement a genuine agentic architecture based on an LLM reasoning loop. In this architecture, the language model operates in a continuous loop, deciding at each iteration which tool to invoke based on its current understanding of the task and the results of previous tool calls.

5.1 Architecture Overview

The reasoning loop architecture is depicted in Figure 2. Unlike the fixed pipeline, the agent has access to all available tools at every step and autonomously decides the sequence of operations.

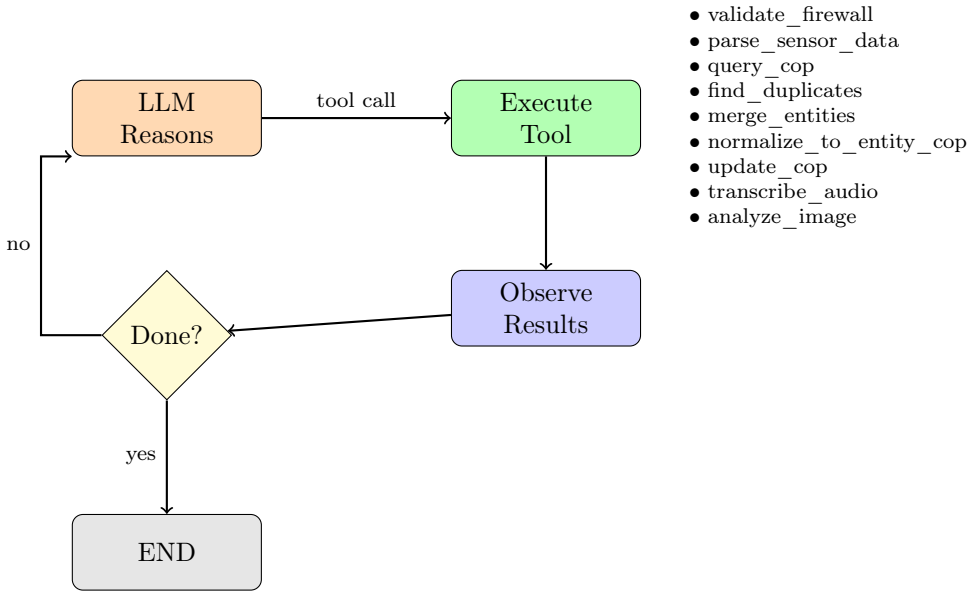

Figure 2: The agent architecture: LLM reasoning loop with tool selection

5.2 Key Capabilities

The reasoning loop architecture enables several capabilities that were not possible with the fixed pipeline:

Contextual Decision-Making. The agent can query the COP to obtain context before deciding how to process a new entity. For example, upon receiving a radar track, the agent might first query existing entities in the same geographic area to determine whether the track corresponds to a known contact.

Proactive Duplicate Detection. Rather than blindly inserting entities and relying on post-hoc deduplication, the agent proactively checks for potential duplicates when contextually appropriate. The decision of *when* to check for duplicates is made by the LLM based on the specifics of the current situation.

Intelligent Multi-Sensor Correlation. When processing data from multiple sensors, the agent can reason about correlations. For instance, if an image analysis reveals a tank at coordinates close to an existing "unknown ground vehicle" radar contact, the agent can decide to merge these observations and update the classification.

Adaptive Processing. The agent can adapt its processing strategy based on intermediate results. If an initial parse yields ambiguous results, the agent might decide to invoke additional tools or request more context.

5.3 Comparison of Architectures

Table 2 summarizes the key differences between the pipeline and agent architectures.

Table 2: Comparison of pipeline and agent architectures

Characteristic	**Pipeline**	**Agent**
Control flow	Fixed DAG	Dynamic (LLM decides)
Tool selection	Predetermined	Autonomous
COP context queries	Not used	Proactive
Duplicate detection	Post-hoc only	Proactive
LLM calls per message	1	3–10
Latency	2–3 seconds	10–30 seconds
Cost per message	Low	Higher
Predictability	High	Medium
Auditability	Easy (fixed flow)	Requires logging
Multi-sensor correlation	Limited	Intelligent

5.4 Trade-offs and Applicability

The agent architecture is not universally superior to the pipeline approach. The increased autonomy comes with costs in terms of latency, computational expense, and reduced predictability. For high-volume, low-complexity data streams (e.g., thousands of radar tracks per second), the pipeline architecture remains more appropriate due to its efficiency and determinism.

The agent architecture is most advantageous when:

- The sequence of reasoning steps cannot be fully specified at design time, and must be dynamically constructed based on intermediate results.

- The system must autonomously decide which actions to take next, including whether to query additional context, invoke tools, or terminate.
- Low-volume, high-value data where the computational cost and latency of multiple LLM calls are acceptable relative to the operational value of the outcome.
- Decision latency is acceptable (on the order of seconds to minutes), such that bounded real-time guarantees are not required.
- The task involves planning, explanation, or synthesis rather than direct transformation or classification.
- Relationships between information sources are open-ended or hypothesis-driven, rather than fixed by known fusion rules.

A hybrid approach, where simple high-volume data flows through the pipeline while complex data is routed to the agent, may offer the best of both architectures.

6 Conclusion

The concept of agency in LLM-based systems is best understood as a spectrum rather than a binary classification. By analyzing the initial TIFDA/CopForge pipeline architecture through this lens, we recognized that despite its sophistication, it functioned as a "sophisticated translator" with minimal true agency. The redesigned CopForge Ingest Agent implements a genuine agentic architecture through an LLM reasoning loop, enabling contextual decision-making, proactive duplicate detection, and intelligent multi-sensor correlation.

This architectural evolution reflects a broader insight: the mere presence of an LLM in a system does not confer agency. True agency requires that the LLM participate in the control flow, making decisions about *what* to do rather than merely *how* to do a predetermined task. As LLM-based systems become more prevalent in critical applications such as information fusion for situational awareness, careful attention to the appropriate level of agency will be essential for balancing capability, efficiency, and reliability.

References

[1] Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

[2] LangChain. (2024). *Introduction to Agents*. https://python.langchain.com/docs/concepts/agents/

[3] Anthropic. (2024). *Building effective agents*. https://www.anthropic.com/research/building-effective-agents

[4] CrewAI. (2024). *Documentation*. https://docs.crewai.com/

[5] Anthropic. (2024). *Model Context Protocol Specification*. https://modelcontextprotocol.io/

[6] Google. (2025). *Agent-to-Agent Protocol*. https://github.com/google/A2A

[7] Martínez-Agulló, P. (2025). *TIFDA: GenAI-Enabled Tactical Information Fusion and Dissemination Agent*. https://github.com/MartinezAgullo/genai-tifda

[8] Martínez-Agulló, P. (2025). *COP Forge: Information Fusion System for Common Operational Pictures*. https://github.com/MartinezAgullo/copforge