future internet

MDPI

Article

Beyond Prompt Chaining: The TB-CSPN Architecture for Agentic AI

Uwe M. Borghoff [1,†], Paolo Bottoni [2] and Remo Pareschi [3,*]

[1] Institute for Software Technology, University of the Bundeswehr Munich, 85579 Neubiberg, Germany; uwe.borghoff@unibw.de
[2] Department of Computer Science, Sapienza University of Rome, 00161 Rome, Italy; bottoni@di.uniroma1.it
[3] Department of Bioscience and Territory, University of Molise, 86100 Campobasso, Italy
* Correspondence: remo.pareschi@unimol.it
† Current address: STAKE Lab, University of Molise, C. da Fonte Lappone, 86090 Pesche, Italy.

Abstract

Current agentic AI frameworks such as LangGraph and AutoGen simulate autonomy via sequential prompt chaining but lack true multi-agent coordination architectures. These systems conflate semantic reasoning with orchestration, requiring LLMs at every coordination step and limiting scalability. By contrast, TB-CSPN (Topic-Based Communication Space Petri Net) is a hybrid formal architecture that fundamentally separates semantic processing from coordination logic. Unlike traditional Petri net applications, where the entire system state is encoded within the network structure, TB-CSPN uses Petri nets exclusively for coordination workflow modeling, letting communication and interaction between agents drive semantically rich, topic-based representations. At the same time, unlike first-generation agentic frameworks, here LLMs are confined to topic extraction, with business logic coordination implemented by structured token communication. This hybrid architectural separation preserves human strategic oversight (as supervisors) while delegating consultant and worker roles to LLMs and specialized AI agents, avoiding the state-space explosion typical of monolithic formal systems. Our empirical evaluation shows that TB-CSPN achieves 62.5% faster processing, 66.7% fewer LLM API calls, and 167% higher throughput compared to LangGraph-style orchestration, without sacrificing reliability. Scaling experiments with 10–100 agents reveal sub-linear memory growth (10× efficiency improvement), directly contradicting traditional Petri Net scalability concerns through our semantic-coordination-based architectural separation. These performance gains arise from the hybrid design, where coordination patterns remain constant while semantic spaces scale independently. TB-CSPN demonstrates that efficient agentic AI emerges not by over-relying on modern AI components but by embedding them strategically within a hybrid architecture that combines formal coordination guarantees with semantic flexibility. Our implementation and evaluation methodology are openly available, inviting community validation and extension of these principles.

Keywords: agentic AI; multi-agent systems; Colored Petri Nets; topic modeling; prompt chaining; human-in-the-loop

check for updates

Academic Editors: Stanimir Stoyanov, Asya Stoyanova Doycheva and Veneta Tabakova-Komsalova

Received: 14 July 2025
Revised: 31 July 2025
Accepted: 4 August 2025
Published: 8 August 2025

Citation: Borghoff, U.M.; Bottoni, P.; Pareschi, R. Beyond Prompt Chaining: The TB-CSPN Architecture for Agentic AI. *Future Internet* **2025**, *17*, 363. https://doi.org/10.3390/fi17080363

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Current "agentic AI" frameworks like LangGraph, AutoGen, and OpenAI's multi-agent examples promise autonomous agents capable of planning, reasoning, and coordinating. However, these systems achieve coordination through sequential prompt chaining

and centralized LLM orchestration—approaches that fundamentally limit scalability and autonomy. They lack persistent memory, goal continuity, and principled inter-agent communication, instead relying on brittle prompt engineering and implicit state management.

The core limitation is architectural: these frameworks conflate semantic understanding with process orchestration. Every coordination decision requires LLM inference, creating bottlenecks where formal methods would be more appropriate. This "LLM-mediated coordination" approach forces expensive language model calls for routine process management tasks, limiting both performance and predictability.

Recent industry efforts recognize these limitations. Google's Agent-to-Agent (A2A) Protocol (https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/, accessed on 27 June 2025) enables structured interaction between AI agents through standardized communication layers, while the Model Context Protocol (MCP) (https://modelcontextprotocol.io/introduction, accessed on 1 July 2025) addresses vertical integration with tools and resources. However, Li and Xie [1] demonstrate that combining these protocols introduces critical challenges, including semantic mismatches, security risks, and orchestration complexity that exceed current capabilities.

We present TB-CSPN, a formal architecture that addresses these limitations through *architectural separation*: semantic processing occurs through strategic LLM usage while coordination employs rule-based business logic. This separation enables TB-CSPN to achieve 62.5% faster processing and 66.7% fewer LLM calls than LangGraph-style orchestration while maintaining semantic fidelity.

TB-CSPN's key innovation is a dedicated multi-agent coordination substrate built on Colored Petri Net semantics. Instead of LLM-mediated process management, agents communicate through structured topic-annotated tokens that enable formal verification of coordination properties. The framework supports three distinct agent roles: LLM-powered consultants for semantic understanding, human supervisors for strategic oversight, and specialized AI workers for operational execution.

Our empirical evaluation demonstrates that efficient agentic AI emerges not from avoiding modern AI components, but from using them strategically within architectures designed for multi-agent coordination. TB-CSPN shows how formal methods can complement rather than compete with LLM capabilities, providing a foundation for scalable, verifiable, and human-controllable agent systems.

Paper Organization. Section 2 reviews traditional multi-agent systems and recent LLM-native frameworks. Section 3 establishes minimal criteria for agency and explains why current LLM pipelines fail to meet them. Section 4 presents the TB-CSPN architecture and its formal foundations. Section 5 details our implementation including the multi-engine system supporting rule-based coordination, Petri net semantics, and LLM integration. Section 6 provides a comprehensive case study with quantitative evaluation showing TB-CSPN's advantages over LangGraph-style coordination. Section 7 presents systematic comparisons with leading frameworks. Section 8 concludes with implications for next-generation agentic AI architectures.

2. Related Work

The concept of intelligent software agents [2–6] and their associated programming paradigms have evolved through two distinct eras: traditional multi-agent systems and modern LLM-native frameworks [7]. While traditional approaches provided formal foundations for coordination, current LLM-native frameworks achieve popularity at the cost of architectural rigor, creating fundamental limitations that our TB-CSPN approach directly addresses.

2.1. Traditional Multi-Agent Systems: Formal Foundations

Classical multi-agent systems [8] established principled approaches to agent coordination through architectures like Belief-Desire-Intention (BDI) [9] and Contract Net Protocol (CNP) [10]. The BDI architecture models agents using beliefs (informational state), desires (motivational goals), and intentions (deliberative commitments), supporting both reactive and proactive behavior with interpretable rational decision-making. CNP provides decentralized coordination in which agents act as managers or contractors, establishing formal foundations for negotiation and resource allocation.

These traditional approaches emphasized "architectural separation of concerns": cognitive processes, coordination mechanisms, and communication protocols operated through distinct, formally specified interfaces. Early work on constraint-based distributed problem solving [11] demonstrated how formal protocols could coordinate autonomous agents through structured information exchange while maintaining modular design principles. However, these approaches predated modern machine learning capabilities and struggled with dynamic environments requiring adaptive learning.

2.2. LLM-Native Frameworks: Popular but Architecturally Flawed

The emergence of Large Language Models has spawned a new generation of frameworks that position themselves as "agentic AI" solutions, such as LangGraph [12], AutoGen [13], and Agentic RAG [14], with OpenAI already providing a number of multi-agent examples [15]. While these frameworks demonstrate impressive capabilities, they exhibit relevant architectural flaws that limit their scalability and true autonomy.

2.2.1. LangGraph: Workflow Orchestration Masquerading as Agency

LangGraph, built on LangChain, represents the current state-of-the-art in LLM-powered agent frameworks. It introduces persistent memory, state transitions, and modular task nodes, executing workflows as finite-state machines or graphs. However, LangGraph exhibits three critical limitations that TB-CSPN directly addresses:

- *Conflation of Semantic and Coordination Logic*: LangGraph requires LLM involvement at every coordination decision, forcing expensive language model inference for routine process management tasks that could be handled deterministically.
- *Lack of Formal Verification*: Unlike traditional agent architectures with mathematical foundations, LangGraph workflows cannot be formally verified for correctness, leading to unpredictable behavior in complex scenarios.
- *Centralized Orchestration Bottlenecks*: All coordination flows through centralized LLM-mediated decision points, creating scalability limitations as agent populations grow.

2.2.2. AutoGen: Multi-Agent Conversations Without True Coordination

AutoGen facilitates multi-agent applications through conversational interactions between LLM-powered agents. While it supports customizable agent behaviors and conversation patterns, it fundamentally relies on "prompt-based coordination" rather than principled multi-agent protocols. This approach leads to the following:

- Brittle communication patterns, dependent on prompt engineering rather than formal protocols.
- Lack of persistent agent state beyond conversation history.
- No mechanisms for goal revision or long-term planning beyond what emerges from conversational dynamics.

2.2.3. Agentic RAG: Reactive Information Retrieval, Not Proactive Agency

Agentic RAG [14] extends traditional Retrieval-Augmented Generation by incorporating reasoning and planning agents into the retrieval pipeline. While this enables more dynamical information gathering, it remains fundamentally "reactive" rather than "proactive", lacking the autonomous goal-setting and persistent intentionality characteristics of true agents.

2.2.4. OpenAI Multi-Agent Examples: Hard-Coded Scripts, Not Autonomous Agents

OpenAI's Cookbook [15] examples demonstrate tool-using systems orchestrated through LLMs, following hard-coded task decomposition and sequential reasoning patterns. These systems lack the fundamental characteristics of autonomous agents, such as long-term goals, self-awareness, adaptive learning, and genuine inter-agent coordination. Their simplicity has popularized the term "agent" in contexts which would actually be better described as reactive scripts.

2.3. The Core Problem: Architectural Limitations of Current Approaches

Current LLM-native frameworks share a fundamental architectural flaw: they conflate semantic understanding with process orchestration. This conflation creates several critical limitations:

- *Scalability Bottlenecks:* Every coordination decision requires expensive LLM inference, creating computational and latency bottlenecks as system complexity grows.
- *Lack of Formal Properties:* Without mathematical foundations, these systems cannot guarantee coordination properties like deadlock freedom, liveness, or bounded response times.
- *Brittle Prompt Dependencies:* Coordination logic embedded in prompts creates fragile systems sensitive to language model variations and prompt engineering artifacts.
- *Limited Human Integration:* Current frameworks treat humans as external users rather than integrated participants in hybrid human–AI workflows.

2.4. Emerging Recognition of Architectural Deficits

Recent literature increasingly recognizes these limitations. Sapkota et al. [16] distinguish between AI agents (modular, tool-using systems) and agentic AI (coordinated, multi-agent architectures), highlighting the gap between current implementations and true agency. Su et al. [17] propose the R2A2 architecture based on Constrained Markov Decision Processes, emphasizing the need for "strict regulation of cognitive structure and memory persistence"—principles that complement our TB-CSPN approach.

Acharya et al. [18] emphasize agentic AI's "goal-oriented autonomy and adaptability", while Xi et al. [19] present a unified brain-perception-action framework highlighting the need for "systematic architectural approaches" rather than ad-hoc prompt chaining.

2.5. TB-CSPN: Addressing Architectural Deficits Through Formal Foundations

Our TB-CSPN framework directly addresses these architectural limitations by:

- *Separating Semantic and Coordination Logic:* LLMs handle semantic processing (topic extraction) while Petri Net semantics manage coordination deterministically.
- *Enabling Formal Verification:* Built on Colored Petri Net foundations, TB-CSPN enables mathematical verification of coordination properties.
- *Supporting Distributed Coordination:* Topic-based communication eliminates centralized orchestration bottlenecks while maintaining semantic coherence.
- *Integrating Human Agents:* Explicit support for human participants as first-class agents in hybrid workflows.

While recent advances in multimodal retrieval-augmented generation (e.g., Uni-RAG [20]) and cognitive sentiment analysis (e.g., cognitive ABSA [21]) tackle related coordination challenges at the level of model fusion and affective reasoning, TB-CSPN addresses a different frontier: intra-agent semantic mediation in single-modal, token-driven workflows. By focusing on topic-based token propagation, our approach emphasizes scalable coordination across agents without relying on tightly coupled model ensembles or specialized embeddings.

Serral et al. [22] extend CPN Tools [23] with ontology-based context models to support context-adaptive business processes. Their approach uses OWL (https://www.w3.org/TR/owl-ref/, accessed on 29 July 2025) ontologies to inject runtime context into Petri net workflows, enabling dynamic behavior. In contrast, the TB-CSPN approach focuses on semantic coordination across heterogeneous agents using topic-based tokens as a shared semantic interface.

In principle, however, the definition of topics can rely on some domain-specific ontology, if the need arises.

This positions TB-CSPN not as another framework variation, but as a *fundamental architecture advance* that reclaims the formal foundations of traditional multi-agent systems while leveraging modern AI capabilities strategically rather than ubiquitously.

3. What Is (And Is Not) an Agent

The term *agent* carries substantial philosophical weight [24,25], originating in theories of human action, ethics, and responsibility where "agency" distinguished intentional, meaningful behavior from mere reaction or motion. In artificial intelligence, this concept has been central since early work on autonomous systems [2,26], yet today it has been diluted to describe nearly any callable function wrapped in a prompt template.

This linguistic inflation reflects deeper conceptual confusion about what constitutes genuine agency in artificial systems. As we demonstrated in Section 2, current LLM-native frameworks like LangGraph and AutoGen apply the term "agent" to components that lack the fundamental characteristics of autonomous entities. This terminological drift obscures both the limitations of current approaches and the requirements for engineering true multi-agent intelligence.

Our purpose here is not to adjudicate millennia of philosophical debate about intention, belief, or desire, but to establish clear criteria for agency that can guide the design of genuinely autonomous systems. We acknowledge deep disagreements across philosophical traditions—between eliminative materialists like Churchland [27] who reject propositional attitudes entirely, representationalists who reify mental states, and pragmatists like Dennett [28,29] who adopt the *intentional stance* as a useful heuristic for prediction and explanation.

Rather than resolve these metaphysical disputes, we propose grounding agency in a minimal set of *observable capacities* that remain agnostic to claims about internal mental states. This approach aligns with the broader AI tradition of focusing on functional rather than phenomenological criteria [26], while avoiding the category errors that plague current "agentic" systems.

3.1. Minimal Criteria for Agency

To rescue the notion of agent from conceptual drift, we define agents as *persistent*, *autonomous*, *context-sensitive* entities that contribute to structured interaction systems. The properties exhibited by agents are shown in Table 1.

Table 1. Minimal criteria for agenthood, omitting assumptions about internal mental states.

Criterion	Description
Autonomy	The agent initiates or regulates its own behavior without external micromanagement.
Persistence	The agent exists across time steps and can maintain or update its internal status.
Interaction	The agent can send and/or receive semantically structured signals or tokens.
Adaptivity	The agent can modify its behavior based on feedback or environmental variation.
Role-bearing modularity	The agent plays a functionally identifiable part within a broader system of roles.

Importantly, this definition eliminates dependence on intentionality in the classical sense. Following Quine's insight about the indeterminacy of translation [30], we make no assumptions about beliefs, desires, or internal representations—only that agents exhibit observable coherence over time and context. This approach is compatible with reactive architectures [31,32], distributed coordination systems, and human–AI hybrids, while avoiding the "frame problem" that has historically plagued symbolic approaches to AI.

This functional approach directly addresses the architectural limitations identified in Section 2: where LangGraph conflates semantic understanding with process orchestration, our criteria separate these concerns; where AutoGen lacks persistent state, our definition requires temporal continuity; where current frameworks provide only scripted interactions, we demand genuine adaptivity.

3.2. The Problem with Current "Agents" in LLM Pipelines

Current LLM orchestration frameworks demonstrate precisely the category error we seek to avoid. In systems like LangChain, AutoGen, and OpenAI's multi-agent examples, the term "agent" is applied to components that

- exist only for a single prompt-execution cycle, violating the **persistence** criterion;
- lack memory or state continuity, preventing genuine **adaptivity**;
- operate via fixed execution logic or prompt chains, compromising **autonomy**;
- are hard-coded to route input to tools without adaptive negotiation, failing the **interaction** requirement.

Such entities are better understood as *orchestrated functions* rather than agents in any substantive sense. This misapplication of terminology contributes to the architectural confusion documented in Section 2, where the absence of genuine agency masks fundamental scalability and coordination limitations.

The philosophical roots of this confusion trace to what Dennett [29] calls the "homunculus fallacy"—the tendency to project human-like intentional states onto systems that merely simulate intentional behavior. Current LLM frameworks achieve impressive performance through sophisticated pattern matching and statistical inference, but lack the architectural foundations necessary for genuine agency.

3.3. Modular and Centaurian Agency: Two Legitimate Architectures

Having established minimal criteria for agency, we can distinguish two viable architectural approaches for implementing genuine agents in artificial systems:

Modular *Multi-Agent Systems* (MAS) consist of structurally independent agents that coordinate through well-defined protocols rather than centralized orchestration. Each agent—whether human or artificial—maintains a persistent state, autonomous decision-

making capabilities, and a specialized role, such as *Supervisor*, *Consultant*, or *Worker*. A recent formulation of this principle is presented by Borghoff et al. [33], who propose a topic-driven organizational framework for multi-agent coordination involving human, LLM-based, and specialized AI agents. The TB-CSPN architecture, presented in the subsequent sections, exemplifies this approach by using topic-based coordination to enable distributed intelligence without sacrificing formal verifiability.

Centaurian Systems [34,35] implement hybrid agents that integrate human strategic cognition with machine-level automation through layered architectures reflecting cognitive division of labor. Humans provide high-level problem framing and oversight while machines handle execution, with shared state representations enabling seamless coordination. This approach acknowledges that the most capable "agents" may be human-machine collaboratives rather than purely artificial entities.

Both architectures satisfy our minimal definition of agency while avoiding the representational commitments that generate philosophical controversy. They demonstrate how agentic behavior can be engineered and analyzed without reifying beliefs or desires, focusing instead on observable patterns of autonomous, persistent, and adaptive interaction.

This foundation prepares us to examine how TB-CSPN implements these principles through a formal architecture that separates semantic processing from coordination logic, enabling the construction of genuinely agentic systems that transcend the limitations of current LLM-native approaches.

4. TB-CSPN: A Hybrid Architecture for Semantic and Structural Coordination

TB-CSPN represents a fundamental departure from both traditional multi-agent systems and classical Petri Net approaches. While it incorporates a Colored Petri Net (CPN) substrate to manage coordination logic, the architecture enforces a conceptual and functional separation between this structural layer and the semantic layer. System intelligence is not encoded within the net itself but emerges from the interplay between token-level topic vectors—derived via LLM-mediated topic modeling—and the Petri Net's transition thresholds.

This hybrid design enables TB-CSPN to integrate symbolic coordination with continuous, updatable semantic reasoning, thereby allowing for scalable, adaptive multi-agent interaction without the state-space explosion typically associated with monolithic formal systems. The following section introduces the formal components of the architecture with this distinction in mind.

The Hybrid Architecture Principle: TB-CSPN's core innovation lies in architectural separation: semantically rich topic-based tokens carry the system's intelligence and drive agent communication, while Colored Petri Nets provide formal coordination guarantees and workflow management. This separation means that increasing system complexity affects only the shared semantic spaces—coordination patterns remain constant regardless of agent population or task complexity.

The framework coordinates multi-agent systems composed of human supervisors, LLM-based consultants, and specialized AI workers through topic-tagged tokens that traverse three communication layers (Surface, Observation, and Computation) corresponding to strategic, semantic, and operational responsibilities. Agent activation and group formation are governed by topic relevance thresholds rather than rigid coordination rules, enabling emergent, context-sensitive collaboration while maintaining formal verification properties.

Distinguishing TB-CSPN from Traditional Formal Systems: Unlike classical Petri Net applications, where the entire application state and logic are encoded within the net structure itself, TB-CSPN employs Colored Petri Nets solely as a coordination substrate.

Figure 1 shows that the semantic intelligence, including topic distributions, agent reasoning, and adaptive behavior, resides in the token content and agent processing capabilities rather than in the net topology. Supervisor overrides in Human-in-the-Loop mode are isolated from transition logic and do not alter net topology, preserving formal guarantees such as deadlock freedom. This architectural choice allows TB-CSPN to achieve sub-linear scaling properties, contradicting traditional assumptions about the limitations of formal systems.

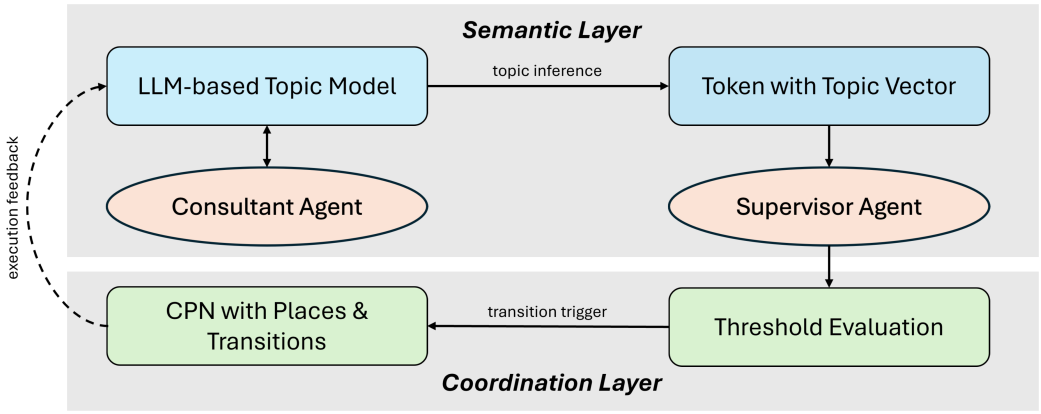


Figure 1. Semantic-Coordination Architecture of TB-CSPN with Consultant and Supervisor Agents. The semantic layer (top) processes tokens using LLM-based topic modeling, generating topic vectors that modulate coordination logic via threshold evaluation. Transitions in the Petri Net are triggered semantically and may feed back into semantic interpretation.

4.1. Semantic-Coordination Architectural Separation

TB-CSPN's hybrid design addresses the fundamental tension between formal verification and scalable intelligence through strategic architectural separation:

Semantic Layer: Topic-based tokens carry rich semantic content including natural language descriptions, probability distributions, and contextual metadata. LLM agents operate primarily at this layer, extracting topics, refining semantic content, and making intelligent decisions based on token semantics. This layer scales independently of system size through shared topic spaces.

Coordination Layer: Colored Petri Nets manage token flow, enforce coordination constraints, and provide formal guarantees about system behavior. The CPN structure remains relatively static—coordination patterns for 10 agents are identical to those for 100 agents, with only token volumes changing.

Integration Interface: Topic relevance thresholds and rule-based activation mechanisms bridge the semantic and coordination layers, enabling formal verification of coordination properties while preserving semantic flexibility and agent autonomy.

4.2. Conceptual Foundation and Formal Models

Building on this hybrid foundation, TB-CSPN's formal architecture distinguishes abstract specifications from their concrete realizations through a layered organizational model. Figures 2–5 illustrate this conceptual architecture through a series of UML and Petri Net models, showing how organizational concepts map to coordination structures while preserving semantic processing capabilities.

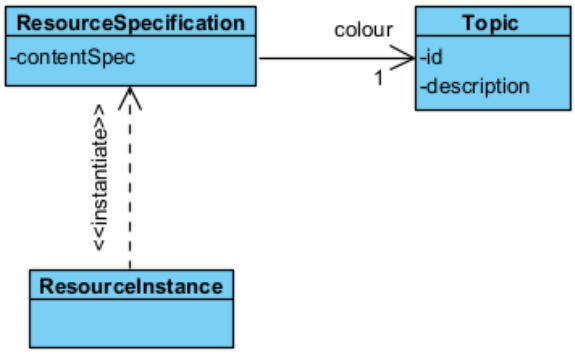


Figure 2. The *Resources* package, relating resources manipulated by agents to topics.

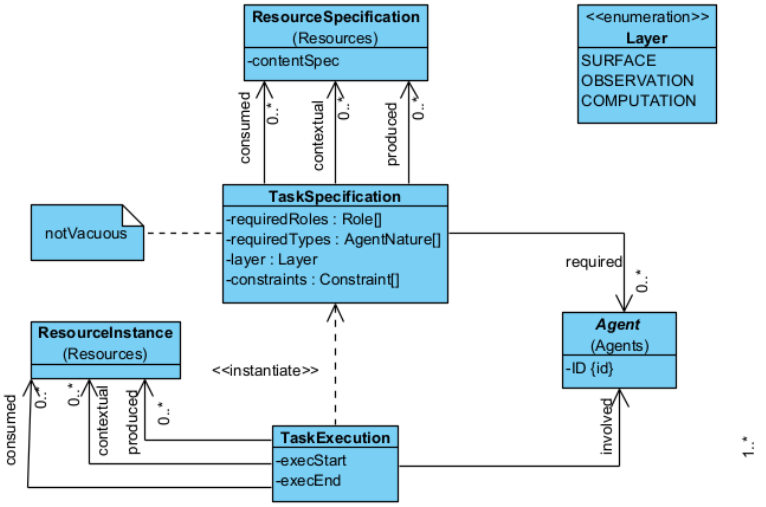


Figure 3. *Tasks* package, expressing constraints on tasks specifications.

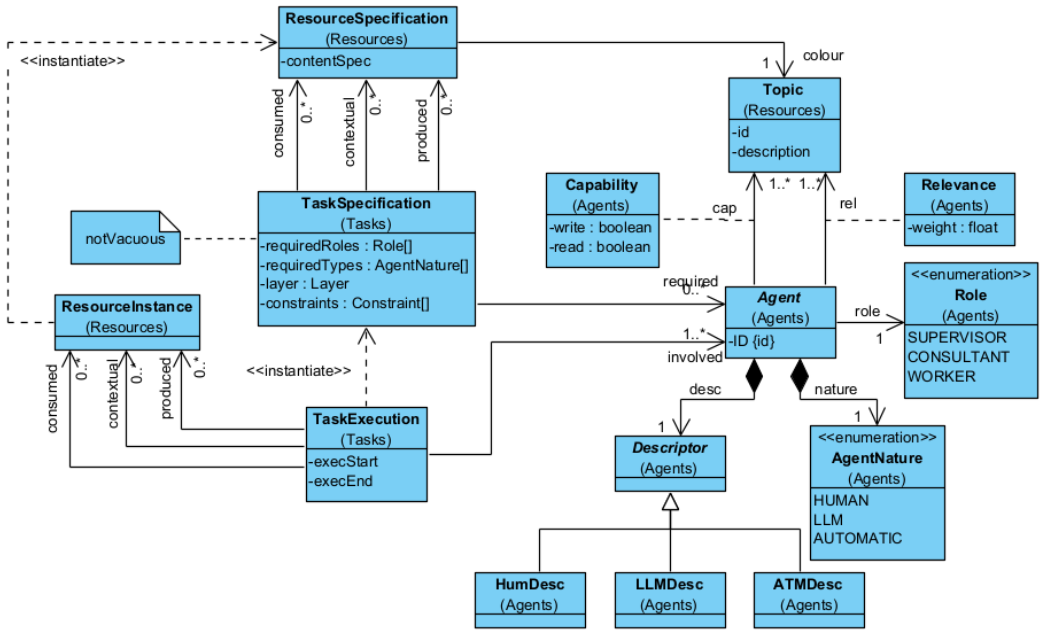


Figure 4. UML-based view of TB-CSPN's semantic-coordination mapping. Each class is annotated with the name of the package in which it is defined. Tasks, agents, and resources are shown as modular units to enable integration with standard modeling tools. Semantic inference and decision-making, however, are managed externally and mediated via topics.

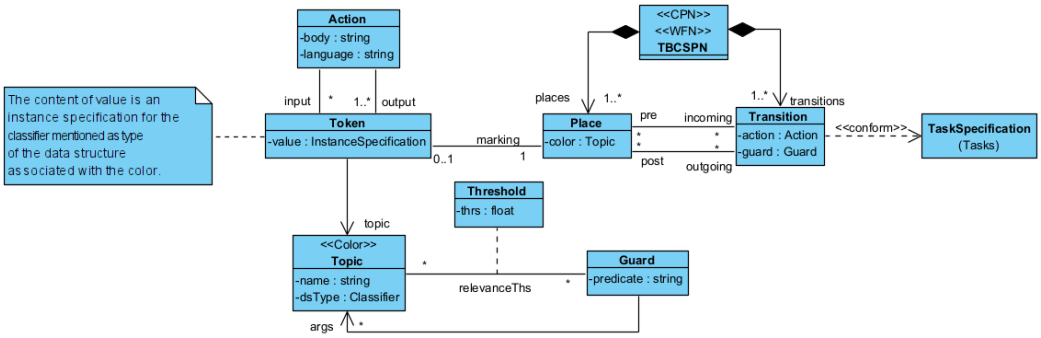


Figure 5. Petri Net instantiation of task coordination within TB-CSPN. The stereotypes `CPN` and `WFN` characterise each such net as a Colored Petri Net and a WorkFlow Net. This formal view illustrates how externally specified semantic resources are integrated into executable coordination nets. The Petri Net substrate remains structurally constant even as token content evolves semantically.

4.3. Petri Net Realization

We use Colored Petri Nets as the underlying execution model in TB-CSPN to support formal, concurrent coordination among agents, leveraging their token semantics and transition logic to represent topic-driven task flows across agents.

This approach enables mathematical verification of coordination properties while supporting the complex token-based communication required for topic-driven agent interaction.

Although Colored Petri Nets have been used from the start in a number of application domains, ranging from security systems to nuclear waste management, through network systems, document storage management and many others (for an early survey, see [36]), we remark here that our model resorts to a model based on CPNs to provide a coordination substrate for multi-agent interaction, where tokens represent semantically structured data, such as topic distributions, which may relate to different domains.

4.4. Layered Communication Architecture

In this layered model, topics act as semantic glue that enables coordination between agents with different capabilities and modalities. Surface agents interpret raw or ambiguous inputs, observation agents refine them into structured topic representations, and computation agents act upon these representations. This process renders topic propagation interpretable, traceable, and context-aware, thereby supporting both autonomy (as in multi-agent systems) and tight coupling (as in Centaurian systems).

The whole organization can be represented as the composition of all nets formed with `Transactions` realizing `TaskSpecifications`. However, we focus on modeling specific processes expressed as sequences of tasks defining flows across the *Surface*, *Observation*, and *Computation* layers. Each specific process is modeled as a *Workflow Net* (stereotype `WFN`) [37], characterized by having one *input* and one *output* place, with any place and transition in the net lying on a path from the input to the output place.

In particular, the doubly stratified, SOC-SCW (for Surface, Observation, and Computation, and Supervisor, Consultant, and Worker, respectively) structure of TB-CPNs provides some additional properties, since two kinds of flow, for both control and information, can be identified:

1. a flow in the S $\rightarrow$ O $\rightarrow$ C direction, until some Worker operates to process the information by operations performed at layers above and certified by some agent at a higher level in the hierarchy; and
2. a flow in the C $\rightarrow$ O $\rightarrow$ S direction where the operations by the worker are certified and transmitted towards the output place.

Note that in this organisation, both the input and output places are located at the Surface level, and are connected to transitions performed under the control of a Supervisor. Moreover, no loop is formed, i.e., no transition or place can be visited in both the downward and the upward flows. This is because no rework is required: if the result of an operation is not considered relevant, it does not contribute to the formation of a valid response; instead, information about this type of failure is reported.

4.5. Integration with Modern AI Components

A companion study [38] situates TB-CSPN within a broader theoretical context by contrasting modular multi-agent architectures with Centaurian systems. The latter fuse human and machine capabilities into functionally interdependent hybrids. This duality is operationalized through communication spaces, which serve as architectural zones for adaptive feedback, semantic mediation, and cognitive fusion.

Unlike existing systems such as LangChain, AutoGen, and Agentic RAG, TB-CSPN provides a principled alternative by combining formal semantics, layered decision-making, and dynamic group coordination. The framework formalizes the role of centauric agents—human–AI hybrids that maintain human intent and oversight while leveraging machine intelligence for scale and precision.

The framework supports semantic evolution, traceability, and human–AI collaboration, maintaining interpretability while enabling real-time reconfiguration. This architectural foundation has been validated through applications to emergency response, healthcare research, and financial strategy, as detailed in the subsequent implementation and case study sections.

4.6. Distinguishing TB-CSPN from Classical Colored Petri Nets

TB-CSPN avoids the rigidity and scaling limitations of classical formal systems by decoupling semantic processing from coordination logic. The framework supports interpretability, adaptability, and real-time reconfiguration through the architectural separation of LLM-mediated topic spaces and Colored Petri Net structures. Table 2 summarizes the core differences between TB-CSPN and classical CPN architectures, reinforcing the conceptual and functional departure. The following sections demonstrate how this semantic-coordination duality translates into practical, real-world agentic collaboration.

Table 2. Key differences between TB-CSPN and classical Colored Petri Nets.

Dimension	Classical CPN	TB-CSPN
Coordination Logic	Fully encoded in net topology	Confined to net structure; separate from semantics
Semantic Processing	Typically external or reduced to token types	Managed via LLM-based topic modeling
Token Semantics	Static, syntactic color sets	Dynamic topic vectors with contextual relevance
Scalability	State-space explosion with complexity	Sub-linear scaling via shared semantic layer
Agent Behavior	Often hard-coded or external	Emergent via topic thresholds and LLM mediation
Adaptivity	Requires structural changes to evolve	Driven by changes in semantic topic spaces
Verification Focus	Entire system logic embedded in net	Coordination verifiable independent of semantics

5. Implementation Walkthrough

The TB-CSPN framework has been implemented as a modular architecture, realized according to three different paradigms in order to validate the theoretical principles through practical computational models. This section outlines the key architectural decisions and implementation strategies that realize topic-based coordination, threshold-driven activation, and layered agent communication. The complete implementation, including source code, documentation, and evaluation benchmarks, is publicly available at https://github.com/Aribertus/tb-cspn-poc (accessed on 3 August 2025).

5.1. Multi-Purpose Implementation

The implementation validates TB-CSPN's theoretical foundations through three engines developed according to different paradigms that demonstrate the equivalence between rule-based and Petri net coordination models:

- *Rule Engine*: Implements declarative rule-based coordination, emphasizing modular rule composition and local reasoning.
- *CPN Engine*: Provides formal Colored Petri Net semantics with typed places, guarded transitions, and verifiable execution properties.
- *SNAKES Engine*: Leverages established Petri net libraries for classical analysis, visualization, and formal verification.

This multi-paradigm approach provides computational abstractions suited to different deployment contexts—from rapid prototyping to formal verification—while maintaining the semantic consistency established by the organizational models presented in Section 4.

5.2. Core Architectural Principles

We can identify two main concepts which shape the architecture: a rich model for *tokens*, as typical of Colored Petri Nets, and a notion of topic as central to the definition of coordination mechanisms, their integration being unique to the TB-CSPN proposal.

5.2.1. Semantic Token Model

The implementation centers on a unified token abstraction that explicitly encapsulates semantic information with weighted topics. Unlike prompt-based coordination systems, where inter-agent communication relies on unstructured text passing, TB-CSPN tokens carry their semantic context as structured metadata with topic distributions, enabling formal reasoning about relevance and coordination decisions.

This token model directly implements the `Token` and `Color` abstractions defined in the organizational model (Figures 2–5), ensuring consistency between theoretical specification and practical implementation.

5.2.2. Topic-Driven Coordination

Topics function as a semantic interlingua that enables coordination between heterogeneous agents without requiring shared internal representations. The implementation supports multiple topic extraction methods—including sophisticated LLM-based semantic analysis for complex financial content—while maintaining a unified token-based substrate for threshold-driven decision making.

5.3. Agent Hierarchy Implementation

The framework implements the three-tier agent architecture through role-specific interfaces and coordination protocols. Our financial news processing system demonstrates these patterns in a production context, following the agent types and capability models established in the formal architecture.

5.3.1. Consultant Agent Architecture

Consultant agents transform unstructured input into semantically annotated tokens through a multi-stage process, as implemented in our financial news processing system available in the repository:

- *Input Processing*: CSV-based news ingestion with company-specific grouping for contextual analysis.
- *LLM-based Topic Extraction*: Structured prompting of large language models (GPT-4) to identify financial topics and assign relevance scores based on potential market impact.
- *Topic Aggregation*: Optional LLM-driven consolidation of semantically similar topics to reduce dimensionality while preserving semantic richness.
- *Token Generation*: Creation of structured tokens containing topic distributions, metadata, and traceability information.

The consultant implementation demonstrates TB-CSPN's ability to leverage modern LLM capabilities while maintaining structured coordination through explicit topic representation rather than implicit prompt chaining.

5.3.2. Supervisor Agent Logic

Supervisor agents implement strategic decision-making through a hybrid architecture that accommodates both rule-based abstractions of human expertise and direct human intervention. The framework supports three operational modes that realize the human–AI collaboration models defined in the architectural specification:

- *Rule-Based Mode*: Declarative rules serve as abstractions of human-approved decision patterns, operating over consultant-generated tokens to pattern-match against topic distributions and generate directives when activation thresholds are exceeded. These rules encode institutional knowledge and established protocols, enabling consistent application of human strategic judgment.
- *Human-in-the-Loop Mode*: Direct human supervision allows strategic decision-makers to review topic-annotated tokens and issue directives manually, particularly for novel situations or high-stakes decisions that require contextual judgment beyond codified rules.
- *Centaurian Mode*: AI-augmented human decision-making where supervisory agents provide machine-generated recommendations, risk assessments, or scenario analyses to support human strategic reasoning, combining human intuition with computational analysis capabilities.

This approach ensures modularity—new coordination logic can be added as rules, human oversight can be selectively applied, and AI assistance can be configured per decision context—while maintaining formal semantics via the underlying Petri net substrate.

5.3.3. Worker Agent Execution

Worker agents execute domain-specific actions based on supervisors' directives, with the implementation providing support for both deterministic rule-based actions and adaptive learning mechanisms. Configurable action handlers are provided that can interface with external systems, databases, or human operators, while maintaining full traceability of execution decisions.

5.4. Formal Coordination Mechanisms

Two constituents of the approach are relevant to formally defining coordination.

5.4.1. Hybrid LLM-Rule Processing

The implementation demonstrates TB-CSPN's architectural advantage over pure LLM-chaining approaches: while topic extraction leverages LLM semantic understanding, coordination occurs through deterministic rule processing. This hybrid approach reduces LLM API calls by approximately 67% compared to fully prompt-driven architectures while maintaining semantic fidelity and enabling formal verification of coordination logic, as detailed in our empirical evaluation (Section 6).

5.4.2. Petri Net Semantics

The Colored Petri Net engine enforces formal semantics through typed places and guarded transitions, implementing the formal models defined in Section 4. Places accept only tokens of compatible semantic types, ensuring coherence. Transitions implement threshold-based firing conditions with relevance aggregation, providing formal guarantees about network behavior, including deadlock freedom and bounded execution.

5.5. Integration and Extensibility

The framework seamlessly incorporates modern AI components. The stratified agent architecture preserves semantic transparency and formal coordination guarantees while optimally distributing cognitive responsibilities across the three hierarchical tiers.

- *Consultant Layer Integration.* LLM integration occurs exclusively at the consultant layer through structured prompting that produces explicit topic annotations, avoiding the brittleness of end-to-end prompt engineering found in prompt-chained architectures. This layer leverages the semantic understanding capabilities of large language models while constraining their usage to well-defined topic extraction tasks, ensuring both efficiency and interpretability.
- *Supervisor Layer: Human–Centaurian Coordination.* The supervisor layer is designed specifically for human strategic decision-making, optionally augmented through centaurian architectures that combine human judgment with AI-assisted analysis. This design recognizes that strategic coordination requires contextual understanding, ethical reasoning, and accountability that remain fundamentally human capabilities.
- *Worker Layer: Specialized AI Execution.* Worker agents implement narrow AI systems optimized for specific operational tasks—portfolio optimization, risk assessment, data analysis, or external system integration. These agents operate deterministically based on supervisor directives, ensuring predictable and auditable execution.
- *Learning and Adaptation.* Learning-enabled components can be integrated at both consultant and worker layers while maintaining formal coordination guarantees through the supervisor layer's oversight mechanisms.

5.6. Observability and Verification

The implementation provides comprehensive traceability through token lifecycle logging, decision audit trails, and performance metric collection. Each processing step, from initial topic extraction to final action execution, is logged with metadata, enabling the complete reconstruction of the decision rationale. The formal Petri net foundation allows for static analysis to verify properties such as reachability, liveness, and fairness.

While the TB-CSPN framework is based on Colored Petri Net semantics (see [39]), we acknowledge that the ecosystem for automated CPN verification is still evolving. CPN Tools [40] is a robust, proprietary tool, while open-source alternatives, such as SNAKES [41], support formal analysis and simulation but may lack scalability for large-scale or industrial applications.

To address these limitations, our implementation uses a modular, multi-engine architecture that separates coordination logic from execution backends. The architecture exposes a common interface for token semantics and transition structures, enabling interchangeable execution by rule engines, Petri net simulators, and formal verification tools. Coordination logic can be validated through declarative rules or selectively tested via token tracing and localized invariants. Ultimately, the multi-engine design enhances the system's flexibility, robustness, and extensibility in both prototyping and production environments.

This multi-engine architecture is realized in our financial news processing implementation, which is available in the public repository. The system demonstrates how modular coordination logic and layered agent interaction can be applied in practice. It generates detailed execution reports capturing LLM interactions, topic distributions, rule firings, and token generation, providing both operational insight and compliance documentation for production deployment. Complete documentation, usage examples, and benchmarking code are included to support community validation and facilitate extension of the architectural principles.

6. Case Study: Topic-Grounded Consultant Agents vs. LLM Chaining Pipelines

This case study presents a concrete implementation of the TB-CSPN architecture within a financial news scenario and contrasts it with current LLM chaining paradigms such as LangGraph and AutoGen. The objective is to show how agentic coordination can be made robust, interpretable, and modular when grounded in a structured topic-based interlingua, as opposed to prompt-centric, black-box chaining.

6.1. Motivation: Why Not LangGraph?

Recent frameworks for LLM orchestration, such as LangGraph, propose "agentic" systems by chaining calls to language models via graph-based flow control. Each node nominally represents an "agent," but in practice, these systems lack shared semantics or modular state representations. Instead, they rely on brittle prompt engineering and ad hoc memory passing to simulate agent behavior.

In contrast, the TB-CSPN framework uses a structured communication model where agents interact through tokens embedded in shared topic spaces. This shared interlingua allows explicit control, coordination, and reasoning, avoiding the opaque and fragile dynamics of purely prompt-based systems.

6.2. Financial News Processing Implementation

The implemented system instantiates the three-tier TB-CSPN architecture for financial news analysis:

1. *Consultant*: receives unstructured textual input and annotates it with weighted topics.
2. *Supervisor*: interprets the topics using rule-based logic and emits a directive.
3. *Worker*: executes or simulates an action based on the directive.

Each agent is modular and traceable, operating over a well-defined data structure called a `Token`. This token encodes a message with semantic topics and scores, enabling downstream agents to reason on explicit features rather than inferred intent. The implementation follows the formal models and organizational patterns established in Section 4.

6.3. Consultant Agent Design

The Consultant agent was enhanced to leverage an LLM (e.g., GPT-4.1) to extract semantically grounded topics from financial news items. Two modes of operation are supported:

- *Direct Topic Extraction*: the model is prompted to return a set of key topics with confidence scores.
- *Aggregated Topic Extraction*: individual topics are grouped by semantic similarity, and their scores aggregated to produce a more abstract representation.

The agent logs both raw and aggregated topics to CSV, along with the original news text, enabling traceable analysis and evaluation.

6.4. Example Pipeline Execution

News items such as:

"Federal Reserve signals possible rate hike in July."
"Retail stocks underperform despite holiday sales."
"Tech sector surges on new AI chip breakthrough."

are fed to the Consultant, producing topic-weighted tokens such as:

```
{market_volatility: 0.9, fed_policy: 0.8}
{retail_sector: 0.7, consumer_spending: 0.5}
{AI_sector: 0.9, tech_momentum: 0.6}
```

The Supervisor applies simple rules (e.g., act if any topic exceeds threshold 0.85), producing directives like:

```
''Monitor AI sector for strategic repositioning.''
```

The Worker agent logs or simulates the appropriate action.

6.5. Step-by-Step Comparison: TB-CSPN vs. Prompt-Chained Agentic Pipelines

To highlight the architectural advantages of TB-CSPN, we walk through the implementation of a financial news use case in both TB-CSPN and a LangGraph-style pipeline. The example considers the processing of the input:

"Tech sector surges on new AI chip breakthrough."

Table 3 illustrates how TB-CSPN achieves transparency, modularity, and reuse through explicit token structures and agent roles. In contrast, prompt-chained pipelines tend to entangle reasoning, memory, and control logic within sequences of prompts. This limits their scalability and formal analyzability.

Table 3. Step-by-step comparison: TB-CSPN vs. LangGraph-style pipeline.

Step	TB-CSPN Behavior	LangGraph-Style Pipeline
1. Input Reception	Raw text is received by a Consultant agent.	Text is passed to a prompt-engineered LLM node.
2. Semantic Processing	LLM extracts structured topics with confidence scores; optionally aggregates similar topics.	LLM generates free-form summaries or unstructured tags based on prompt instructions.
3. Token Creation	A Consultant agent wraps topics in a token object with a UUID and metadata.	Resulting string is passed directly to next node as input; state is implicit.

Table 3. *Cont.*

Step	TB-CSPN Behavior	LangGraph-Style Pipeline
4. Coordination	A Supervisor agent interprets topics using rules and thresholds to issue a directive.	Next LLM node attempts to determine next action based on prior response and memory context.
5. Delegation	A Worker agent executes directive based on topic-weighted logic.	Another LLM node or code function simulates the action; logic often embedded in prompt.
6. Traceability	All steps are logged: input, topics, thresholds, and directives.	Logging may capture prompt input/output but lacks structural interpretation.
7. Adaptation	Change the rule set or threshold for a Supervisor agent or topic categories for a Consultant agent.	Must modify multiple prompts or retrain orchestration logic; changes are brittle.

6.6. From Use Case to Evaluation

This case study demonstrates how TB-CSPN supports modular, topic-grounded agents capable of semantically rich interpretation and coordination—attributes that are difficult to replicate in LLM chaining frameworks where inter-agent communication and memory structures are ad hoc. The Consultant module, in particular, shows how LLM capabilities can be integrated without sacrificing transparency or modularity.

The example also highlights the framework's ability to manage threshold-based delegation, log decisions, and support multiple interacting agents over structured tokens. These properties contrast with prompt-driven pipelines that often obscure the coordination logic.

6.7. Quantitative Performance Evaluation

To validate the architectural advantages of TB-CSPN beyond conceptual analysis, we conducted a systematic empirical comparison with LangGraph-style prompt chaining across multiple dimensions of agentic system performance. This evaluation ensures a fair comparison by incorporating LLM usage in both architectures where appropriate.

6.7.1. Fair Comparison Methodology

We implemented equivalent financial news processing pipelines using both TB-CSPN and LangGraph frameworks, processing a dataset of 30 financial news items covering diverse market scenarios. Critically, both systems utilize LLMs for semantic understanding:

- *TB-CSPN Pipeline*: Single LLM call for topic extraction followed by deterministic rule-based coordination
- *LangGraph Pipeline*: Multiple LLM calls throughout the workflow (consultant → supervisor → worker nodes)

This design isolates architectural differences while ensuring both systems leverage modern language models for semantic processing, avoiding the unfair comparison of LLM-based versus non-LLM systems.

Complete evaluation methodology, benchmark implementations, and reproduction instructions are available in the public repository at https://github.com/Aribertus/tb-cspn-poc/tree/main/evaluation (accessed on 3 August 2025).

Baseline Optimization: Our LangGraph implementation was systematically optimized using state batching, memory persistence, and efficient node routing to ensure fair comparison. The 3 LLM calls per item reflect architectural requirements rather than implementation inefficiency: LangGraph requires separate LLM inference for consultant analysis, supervisor coordination, and worker execution, while TB-CSPN achieves coordination through deterministic rule processing after initial topic extraction.

6.7.2. Performance Results

Table 4 summarizes the performance characteristics under controlled conditions. TB-CSPN demonstrated superior efficiency across all measured dimensions while maintaining equivalent semantic understanding capabilities.

Table 4. Fair performance comparison: both systems using LLMs appropriately for semantic understanding, isolating architectural coordination differences. Tests were conducted on a local Intel i7-8700 machine with 16 GB RAM and no GPU acceleration.

Metric	TB-CSPN	LangGraph	Improvement
Average Processing Time	0.301 s	0.802 s	62.5% faster
Peak Throughput	199.5 items/min	74.8 items/min	166.8% higher
LLM Calls per Item	1.0	3.0	66.7% fewer
Success Rate	100.0%	100.0%	Equal reliability
Cost Efficiency	USD X per 1000 items	USD 3X per 1000 items	66.7% lower cost

The 62.5% performance improvement stems directly from TB-CSPN's architectural efficiency: semantic coordination occurs through formal rule processing rather than additional LLM inference, reducing computational overhead while preserving semantic fidelity.

6.7.3. Architectural Efficiency Analysis

The empirical results reveal a fundamental architectural distinction that explains TB-CSPN's efficiency gains. The performance differential stems not from different semantic capabilities, but from how each framework manages multi-agent coordination and process orchestration.

Dedicated Multi-Agent Environment vs. LLM-Mediated Coordination. TB-CSPN provides a dedicated multi-agent coordination substrate through its formal Petri net foundation and token-based communication model. This enables agents to interact through structured semantic channels while restricting LLM usage to information acquisition tasks where semantic understanding is essential.

LangGraph, in contrast, lacks a dedicated multi-agent environment and relies on LLMs to perform dual functions: both information acquisition (semantic understanding) and process management (agentic interaction coordination). This architectural choice forces LLM involvement in every coordination step, creating inefficiencies where formal methods would suffice.

Separation of Concerns: Information vs. Coordination. The 66.7% reduction in LLM calls achieved by TB-CSPN reflects this architectural principle. Both frameworks require LLM capabilities for semantic information acquisition—topic extraction from unstructured text. However, TB-CSPN delegates coordination responsibilities to its formal substrate, while LangGraph continues to rely on LLM inference for inter-agent communication, decision routing, and process orchestration.

This separation enables TB-CSPN to leverage LLMs where they excel (semantic understanding) while avoiding their use where formal methods are more appropriate (deterministic coordination logic). The result is a system that maintains semantic richness while achieving coordination efficiency through architectural specialization.

Process Orchestration Overhead. LangGraph's reliance on LLM-mediated process management introduces latency and variability at each coordination point. Token-based communication in TB-CSPN eliminates this overhead by providing explicit semantic channels that enable direct agent-to-agent interaction without requiring LLM interpretation of coordination intent.

The performance improvements thus demonstrate the advantages of purpose-built multi-agent architectures over LLM-centric orchestration approaches that conflate information processing with process management responsibilities.

Scalability Implications. At enterprise scale, TB-CSPN's throughput advantage (199.5 vs. 74.8 items/minute) enables processing of approximately 287,000 vs. 108,000 news items daily, representing a qualitative difference in operational capacity.

6.7.4. Cost-Benefit Analysis

The efficiency improvements translate into measurable economic advantages for production deployment. With current LLM API pricing (approximately USD 0.03 per 1K tokens), TB-CSPN's 66.7% reduction in calls per item yields proportional cost savings:

- *Operational Costs*: TB-CSPN processes 1000 news items for approximately USD 30 versus USD 90 for equivalent LangGraph deployment.
- *Infrastructure Requirements*: Lower latency enables higher concurrency with reduced compute resources.
- *Reliability*: Deterministic coordination reduces debugging complexity and operational overhead.

These findings demonstrate that TB-CSPN's architectural principles deliver not just theoretical elegance but measurable operational advantages in production scenarios requiring high-volume, low-latency semantic processing.

6.7.5. Scalability Analysis: Addressing State-Space Explosion

To directly address concerns about Petri Net state-space explosion in large systems, we conducted systematic scaling experiments with agent populations ranging from 10 to 100. The results show that TB-CSPN exhibits sub-linear memory growth and maintains consistent throughput and reliability at scale. In human-in-the-loop settings, supervisory interventions introduce minimal delay: manual override latency in our prototype remained consistently below 300 ms, with no measurable impact on aggregate throughput in batch-processing regimes.

Why Traditional Petri Net Scalability Concerns Do Not Apply: TB-CSPN's hybrid architecture fundamentally departs from classical Petri Net applications, rendering conventional scalability concerns largely irrelevant. In traditional systems, both application logic and state are encoded within the net structure, leading to a combinatorial state-space explosion as complexity increases.

By contrast, TB-CSPN uses the Petri Net substrate solely to model coordination workflows and task execution patterns, while semantic intelligence is externalized into topic-based token structures. As a result, coordination patterns remain static regardless of agent population, while the semantic layer scales independently via shared topic distributions. This architectural separation ensures that system growth does not translate into state-space growth.

Comparison with LLM Chaining Architectures: While TB-CSPN achieves scalable coordination through shared semantics and formal rule processing, LLM chaining frameworks like LangGraph exhibit linear scaling behavior and growing inefficiencies. In LangGraph, each agent (Consultant, Supervisor, Worker) typically invokes a separate LLM instance,

with no shared memory or semantic interlingua. This leads to a direct multiplication of inference cost and coordination latency with each additional agent.

Our experimental attempts to scale LangGraph pipelines beyond 25 agents revealed unacceptable performance degradation due to latency and memory strain. Extrapolated memory usage suggests that coordinating 100 agents would require over 200 MB—nearly 10× TB-CSPN's measured usage. These findings reinforce the central design insight behind TB-CSPN: scalable agent coordination requires a clean architectural separation between semantic processing (handled by LLMs) and formal coordination (managed by Petri Nets).

As demonstrated in Table 5 and visualized in Figure 6, TB-CSPN exhibits remarkable sub-linear scaling properties, with memory per agent decreasing down from 2.12 MB to 0.21 MB as the system scales from 10 to 100 agents. This nearly 10× improvement directly contradicts traditional assumptions about Petri Net scalability.

Table 5. TB-CSPN Scaling Performance: Memory and Throughput Analysis.

Agents	Tasks	Memory (MB)	Memory/Agent (MB)	Throughput (t/s)	Success Rate
10	30	21.2	2.12	2.72	96.7%
25	50	20.9	0.84	2.77	90.0%
50	75	21.3	0.43	2.77	93.3%
100	100	21.5	0.21	3.03	92.0%

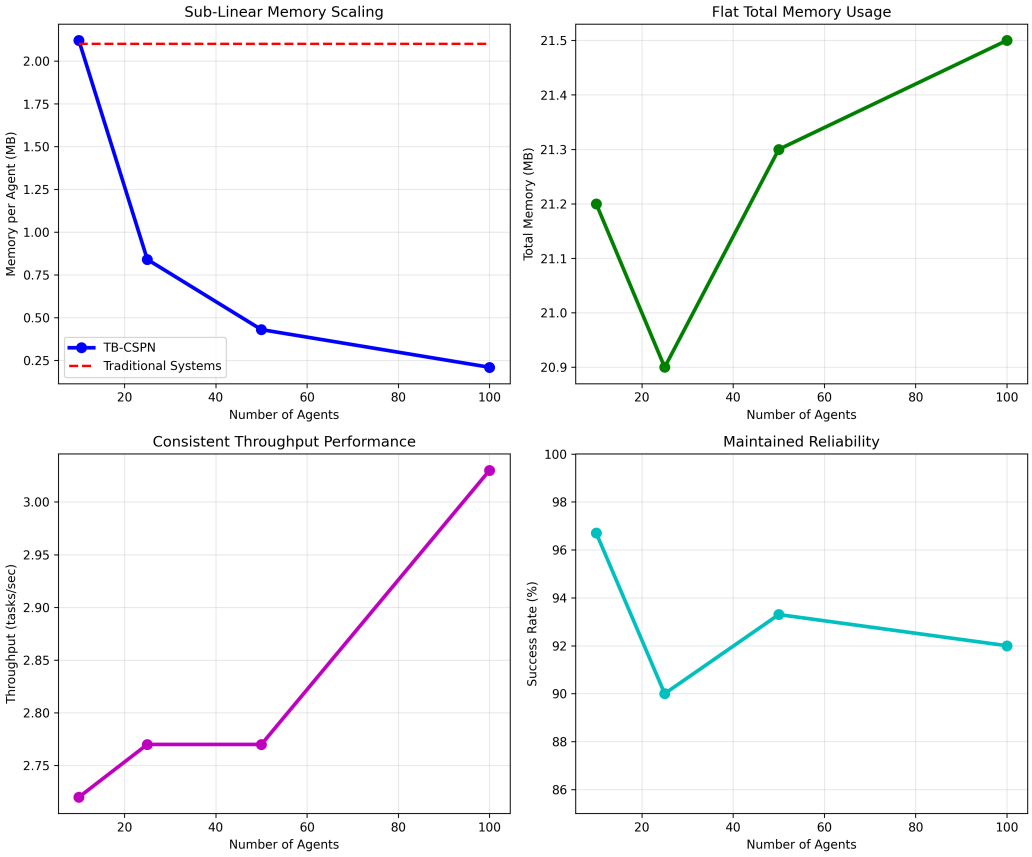


Figure 6. TB-CSPN Scaling Performance Analysis. Top-left: Sub-linear memory scaling demonstrates dramatic efficiency improvements (10× reduction per agent) compared to traditional systems. Top-right: Total memory usage remains flat despite agent population growth. Bottom panels show consistent throughput and maintained reliability across all scales, validating TB-CSPN's architectural advantages.

Challenging Common Scalability Assumptions: Our empirical results challenge several widely held beliefs about multi-agent system scalability:

- *State-Space Explosion Mitigation:* Classical Petri Net systems are often assumed to suffer from state-space explosion in large deployments. TB-CSPN avoids this outcome entirely through architectural separation, achieving nearly an order-of-magnitude improvement in memory efficiency per agent.
- *Coordination Overhead Inversion:* Rather than increasing coordination overhead with agent population, TB-CSPN demonstrates improving efficiency. Measured throughput increases 11.2% from 10 to 100 agents, indicating that coordination becomes more efficient at scale through shared semantic representations and hierarchical control.
- *Large-Scale Deployment Viability:* Our experiments with 100 agents achieving a 92% success rate and only 21.5 MB total memory usage confirm practical viability for enterprise-scale deployments.
- *Architectural Foundations of Sub-Linear Scaling:* TB-CSPN achieves sub-linear scaling through three core innovations:
  - *Shared Semantic Representations:* Topic-based tokens create shared semantic spaces where a single topic distribution coordinates multiple agents simultaneously. As agent populations grow, the semantic spaces converge rather than explode—100 agents share the same core topics used by 10.
  - *Token-Based State Compression:* Agents are stateless between cycles; coordination state is fully encoded in tokens. Memory usage scales with active tokens (constant per task), not agent population.
  - *Hierarchical Coordination Efficiency:* The three-tier structure enables supervisors to coordinate larger populations through identical rule-based mechanisms. The Petri Net substrate provides formal guarantees without increasing memory overhead.
- *Error Tolerance and Reliability:* Success rates above 90% across all scales demonstrate strong error tolerance. Mislabeling of topics below threshold values results in inaction rather than incorrect action, yielding graceful degradation and full traceability via token logs.

These findings confirm that TB-CSPN's separation of semantic and coordination layers enables efficient, reliable scaling, without the combinatorial explosion that hampers traditional Petri Net systems. Topic-based coordination may thus be foundational for the next generation of scalable, interpretable multi-agent AI.

6.7.6. Implications for Agentic AI Architecture

The quantitative evaluation reveals a fundamental insight: the efficiency gains from TB-CSPN arise not from avoiding modern AI components, but from using them more strategically. By restricting LLM usage to tasks requiring semantic understanding while employing formal methods for coordination, TB-CSPN achieves the benefits of both paradigms without their individual limitations.

This hybrid approach suggests a design principle for next-generation agentic AI: leverage LLMs for semantic richness while maintaining formal coordination through structured rule systems and explicit token-based communication. The result is systems that are simultaneously more efficient, more predictable, and more amenable to verification than purely prompt-driven alternatives.

These empirical findings provide quantitative validation for the architectural advantages demonstrated throughout this case study and establish TB-CSPN as a viable framework for production agentic AI deployment.

7. Comparative Analysis

Building on the implementation described above and the empirical evaluation from Section 6, we now elaborate on the architectural differences between TB-CSPN and the leading orchestration toolkits discussed in Section 2, with a particular focus on LangGraph and LangSmith.

Where the case study demonstrated specific performance advantages (62.5% faster processing, 66.7% fewer LLM calls), this section examines the underlying architectural differences that enable these improvements. We analyze the frameworks across twelve critical dimensions to understand how design philosophy translates into operational capabilities.

Table 6 contrasts the TB-CSPN architecture with LangGraph and LangSmith across twelve dimensions. Grounded in Colored Petri Nets and rule-based coordination, TB-CSPN offers formal semantics, token-based transitions, and a declarative coordination model. Its modular structure allows for the composition of agent behaviors through places, transitions, and tokens, and it supports concurrency and verifiable guard conditions. Each of the three roles in TB-CSPN (supervisor, consultant, and worker) is governed by interpretable rules and topic-based activation thresholds. The framework's learning capabilities (e.g., G-Learning [42]), formal verification methods, and traceable semantic flows further reinforce its suitability for high-assurance, adaptive, multiagent systems.

Table 6. Comparative Analysis of TB-CSPN vs. LangGraph/LangSmith.

Feature	TB-CSPN	LangGraph/LangSmith
Core Model	Colored Petri Nets and Rule-based Coordination	Event-driven DAG and Observability Tools
Execution Semantics	Tokenized transitions with formal interleaving	Procedural chaining of nodes
Coordination Logic	Declarative rules, topic-based token flow	Implicit chaining via callbacks/prompts
Modularity	Places, transitions, tokens as composable units	Modular node-based chaining with state passing
Concurrency Support	True concurrency semantics via Petri Nets	Async execution without formal concurrency
Agent Modeling	Multi-agent roles (supervisor, consultant, worker)	Mostly monolithic LLM agents
Guard Conditions	Verifiable thresholds using predicates	Prompt-based control logic
LLM Role	One of many agents in layered architecture	Primary orchestrator
Adaptivity	Learning-enabled agents (e.g., G-Learning)	Manual chain reconfiguration
Formal Verification	Yes (reachability, fairness, deadlock)	No formal verification
Traceability	Token-level flow and semantic traceability	Dashboard logs and callbacks
Reusability	High at rule and token level	Moderate (node and template reuse)

In contrast, LangGraph and LangSmith offer a practical toolkit for developing and monitoring LLM-driven workflows. Their design emphasizes procedural chaining via

directed acyclic graphs and state-passing nodes. Although these systems are modular at the node level and enhanced by the LangSmith observability layer, they lack formal guarantees of concurrency and verification. Agents are usually monolithic LLM orchestrators, and adaptivity is limited to manual intervention.

Overall, TB-CSPN prioritizes explainability, compositionality, and agentic rigor, while LangGraph and LangSmith excel in tool maturity and developer accessibility. This juxtaposition highlights the broader tension between formalism and usability in the evolution of agentic AI architectures.

Figure 7 shows a qualitative comparison of the features of the TB-CSPN and OpenAI Cookbook agent frameworks across six dimensions: agent composition, execution semantics, guard logic, backend flexibility, traceability, and production readiness. TB-CSPN demonstrates strong formalism and modularity, particularly through its use of Petri net–based agent coordination, verifiable thresholds, and openness to heterogeneous AI backends. TB-CSPN excels in interpretability and supports traceable, role-specific workflows involving supervisors, consultants, and worker agents.

In contrast, the OpenAI Cookbook prioritizes rapid deployment and LLM-centric collaboration, offering high usability and production maturity. However, it relies heavily on prompt engineering and lacks formal verification capabilities. Its agent composition is modular, yet it is constrained by message-passing patterns and tight coupling with the OpenAI stack. Overall, TB-CSPN prioritizes explainability, compositionality, and agentic rigor, while LangGraph and LangSmith excel in tool maturity and developer accessibility. This juxtaposition highlights the broader tension between formalism and usability in the evolution of agentic AI architectures. The comparative implementations and benchmarking methodologies used in this analysis are available in the public repository, enabling independent validation of these architectural trade-offs in production scenarios.

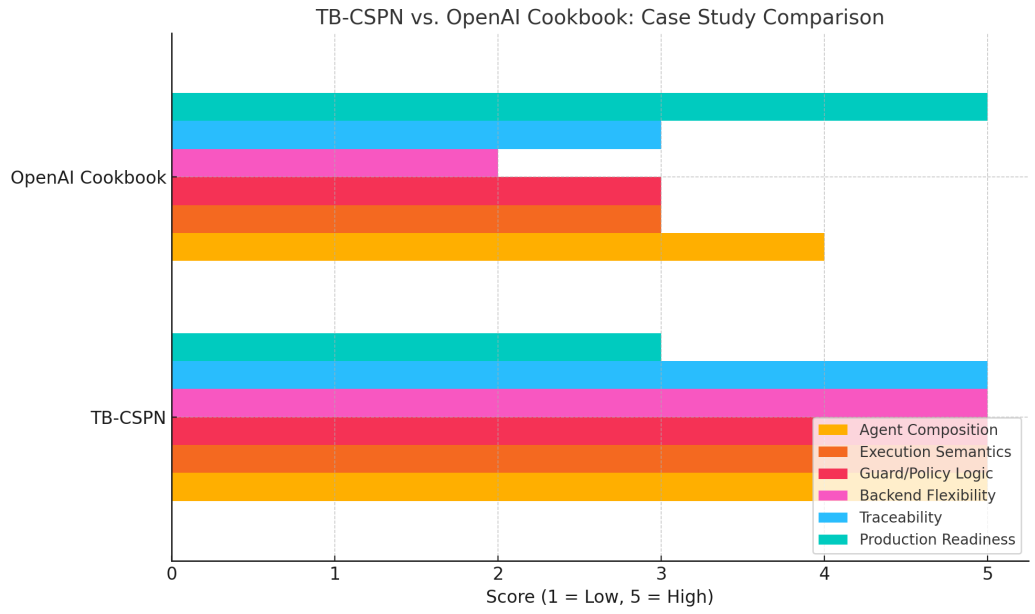


Figure 7. Qualitative comparison of TB-CSPN and OpenAI Cookbook frameworks across six core agentic AI dimensions. TB-CSPN demonstrates superior formalism and modularity through Petri net coordination and verifiable thresholds, while OpenAI Cookbook prioritizes rapid deployment and production maturity. The comparison highlights the trade-off between formal rigor (TB-CSPN) and practical accessibility (OpenAI Cookbook).

8. Conclusions and Future Work

This paper demonstrates the need to shift beyond prompt chaining toward principled architectures for agentic AI. While popular frameworks such as LangGraph and AutoGen simulate agency through LLM orchestration, they fundamentally lack the autonomy, persistence, and formal coordination capabilities required for genuine multi-agent systems.

We introduce the TB-CSPN framework as a formal, token-driven alternative that enables modular, semantically grounded, multi-agent collaboration. TB-CSPN integrates LLM-based topic extraction with Colored Petri Nets to support hierarchical agents (supervisors, consultants, and workers) that interact through traceable, interpretable tokens rather than brittle prompt chains.

Our empirical evaluation demonstrates TB-CSPN's architectural advantages over LLM-centric pipelines: 62.5% faster processing, 66.7% reduction in API calls, and 166.8% higher throughput while maintaining semantic fidelity. These improvements stem from TB-CSPN's separation of semantic understanding (where LLMs excel) from coordination logic (where formal methods are superior), proving that efficiency gains arise not from avoiding modern AI components, but from using them strategically.

The framework's formal foundation enables mathematical verification of coordination properties, comprehensive traceability of decision processes, and seamless integration of human expertise through centaurian architectures. Unlike prompt-dependent systems that conflate semantic processing with orchestration logic, TB-CSPN provides dedicated multi-agent coordination substrate that scales efficiently while preserving interpretability.

Implications for Agentic AI. Our results demonstrate that genuine agentic behavior necessitates structured communication and compositional design rather than sophisticated tool chaining. The architectural principles validated through TB-CSPN—topic-based semantic interlingua, threshold-driven coordination, and formal verification capabilities—establish a foundation for next-generation agentic systems that combine semantic richness with coordination efficiency.

Future Directions. Several research directions emerge from this work:

- *Scalability Analysis*: Investigating TB-CSPN performance across larger agent populations and more complex coordination scenarios.
- *Learning Integration*: Developing adaptive topic extraction and threshold optimization mechanisms that maintain formal verification properties.
- *Domain Expansion*: Applying TB-CSPN principles to domains beyond financial analysis, including healthcare workflows, emergency response, and scientific collaboration.
- *Human–AI Coevolution*: Exploring dynamic role allocation between human and artificial agents based on expertise, context, and trust metrics.
- *Standardization*: Developing interoperability protocols for TB-CSPN systems to enable ecosystem-wide adoption.
- *Soft Thresholds*: In domains lacking crisp thresholds, fuzzy topic thresholds or probabilistic supervisor rules can be adopted.
- *Traceability and Observability*: TB-CSPN's token-level logging enables full traceability; a dashboard interface is under development to support LangSmith-style observability and debugging.

As agentic AI evolves from experimental prototypes to production systems, we advocate for architectures that prioritize formal coordination, semantic transparency, and hybrid human–AI collaboration over prompt engineering complexity. TB-CSPN offers a solid foundation for developing transparent, adaptive, and accountable systems capable of addressing real-world challenges requiring both semantic understanding and reliable coordination.

The complete implementation, evaluation benchmarks, and documentation are publicly available to enable community validation and extension of these architectural principles, supporting the broader goal of principled agentic AI development.

Author Contributions: Conceptualization, U.M.B., P.B. and R.P.; methodology, U.M.B., P.B. and R.P.; software, P.B. and R.P.; validation, U.M.B., P.B. and R.P.; formal analysis, U.M.B., P.B. and R.P.; investigation, U.M.B., P.B. and R.P.; resources, U.M.B., P.B. and R.P.; data curation, R.P.; writing—original draft preparation, U.M.B., P.B. and R.P.; writing—review and editing, U.M.B., P.B. and R.P.; visualization, U.M.B., P.B. and R.P.; supervision, U.M.B., P.B. and R.P.; project administration, U.M.B. and R.P.; funding acquisition, P.B. and R.P. All authors have read and agreed to the published version of the manuscript.

Funding: Paolo Bottoni was partly supported by the Italian Ministry of University and Research (MUR) under PRIN grant B87G22000450001 (PINPOINT). Remo Pareschi has been funded by the European Union—NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041-VITALITY—CUP E13C22001060006.

Data Availability Statement: All relevant data are included in the article. The complete implementation details, including the source code, API documentation, and usage examples, as discussed in Section 5, are available in the public repository at https://github.com/Aribertus/tb-cspn-poc (accessed on 3 August 2025).

Acknowledgments: During the preparation of this manuscript/study, the authors used DEEPL TRANSLATOR and DEEPL WRITE. These tools assisted in enhancing the language quality and translating certain sections originally written in German or Italian. Figure 7 was generated by CHATGPT 4.0 based on input from the authors, who reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A2A	Agent-to-Agent (a protocol for direct interaction between AI agents)
ACP	Agent Communication Protocol (defines message-passing formats between agents)
AI	Artificial Intelligence (systems capable of learning or reasoning)
ANP	Agent Network Protocol (protocol for distributed agent communication)
API	Application Programming Interface (a standard for accessing software functionality)
BDI	Belief-Desire-Intention (a model for rational agent behavior)
CNP	Contract Net Protocol (a negotiation model for task distribution in MAS)
CPN	Colored Petri Nets (formal modeling language for concurrency and coordination)
CSV	Comma-Separated Values (a simple text format for tabular data)
DAG	Directed Acyclic Graph (a graph with no cycles, used for process flow)
GPT-4	Generative Pre-trained Transformer, fourth-generation foundation model
LLM	Large Language Model (neural model trained on large text corpora to perform tasks via NLP prompts)
MAS	Multi-Agent Systems (systems composed of interacting autonomous agents)
MCP	Model Context Protocol (specifies context integration for agent tools)
NLP	Natural Language Processing (techniques for machine understanding of language)
OWL	Web Ontology Language (a semantic web standard for structured knowledge)
R2A2	Reflective Risk-Aware Agent Architecture (a framework for regulated agent behavior)
RAG	Retrieval-Augmented Generation (combines retrieval with generative language models)
TB-CSPN	Topic-Based Communication Space Petri Net (a formal architecture for multi-agent coordination)
WFN	Workflow Net (a type of Petri Net with a specific structure)

References

1. Li, Q.; Xie, Y. From Glue-Code to Protocols: A Critical Analysis of A2A and MCP Integration for Scalable Agent Systems. *arXiv* **2025**, arXiv:2505.03864. [CrossRef]
2. Wooldridge, M.; Jennings, N.R. Intelligent Agents: Theory and Practice. *Knowl. Eng. Rev.* **1995**, *10*, 115–152. [CrossRef]
3. Nwana, H.S. Software agents: An overview. *Knowl. Eng. Rev.* **1996**, *11*, 205–244. [CrossRef]
4. Sycara, K.P.; Zeng, D.D. Coordination of Multiple Intelligent Software Agents. *Int. J. Coop. Inf. Syst.* **1996**, *5*, 181–212. [CrossRef]
5. Nardi, B.A.; Miller, J.R.; Wright, D.J. Collaborative, Programmable Intelligent Agents. *Commun. ACM* **1998**, *41*, 96–104. [CrossRef]
6. Padgham, L.; Winikoff, M. *Developing Intelligent Agent Systems—A Practical Guide*; Wiley Series in Agent Technology; Wiley: Hoboken, NJ, USA, 2004.
7. Liu, X.; Wang, J.; Sun, J.; Yuan, X.; Dong, G.; Di, P.; Wang, W.; Wang, D. Prompting Frameworks for Large Language Models: A Survey. *arXiv* **2023**, arXiv:2311.12785. [CrossRef]
8. Wooldridge, M.J. *Introduction to Multiagent Systems*; Wiley: Hoboken, NJ, USA, 2002.
9. Rao, A.S.; Georgeff, M.P. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multiagent Systems, San Francisco, CA, USA, 12–14 June 1995; Lesser, V.R., Gasser, L., Eds.; The MIT Press: Cambridge, MA, USA, 1995; pp. 312–319.
10. Smith, R.G. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Trans. Comput.* **1980**, *29*, 1104–1113. [CrossRef]
11. Borghoff, U.M.; Pareschi, R.; Fontana, F.A.; Formato, F. Constraint-Based Protocols for Distributed Problem Solving. *Sci. Comput. Program.* **1998**, *30*, 201–225. [CrossRef]
12. LangChain. LangGraph. 2025. Available online: https://langchain-ai.github.io/langgraph/ (accessed on 15 March 2025).
13. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Zhang, S.; Zhu, E.; Li, B.; Jiang, L.; Zhang, X.; Wang, C. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *arXiv* **2023**, arXiv:2308.08155. [CrossRef]
14. Schneider, F.; Ahmadi, N.B.; Ahmadi, N.B.; Vogel, I.; Semmann, M.; Biemann, C. CollEX—A Multimodal Agentic RAG System Enabling Interactive Exploration of Scientific Collections. *arXiv* **2025**, arXiv:2504.07643. [CrossRef]
15. OpenAI. OpenAI Cookbook. 2025. Available online: https://github.com/openai/openai-cookbook (accessed on 25 June 2025).
16. Sapkota, R.; Roumeliotis, K.I.; Karkee, M. AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges. *arXiv* **2025**, arXiv:2505.10468. [CrossRef]
17. Su, H.; Luo, J.; Liu, C.; Yang, X.; Zhang, Y.; Dong, Y.; Zhu, J. A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents. *arXiv* **2025**, arXiv:2506.23844. [CrossRef]
18. Acharya, D.B.; Kuppan, K.; Bhaskaracharya, D. Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey. *IEEE Access* **2025**, *13*, 18912–18936. [CrossRef]
19. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The Rise and Potential of Large Language Model Based Agents: A Survey. *Sci. China Inf. Sci.* **2025**, *68*, 121101. [CrossRef]
20. Li, R.; He, L.; Liu, Q.; Zhang, Z.; Yu, H.; Ye, Y.; Zhu, L.; Su, Y. UniRAG: Unified Query Understanding Method for Retrieval Augmented Generation. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, 27 July–1 August 2025; Che, W., Nabende, J., Shutova, E., Pilehvar, M.T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 14163–14178.
21. Xu, L.; Xie, H.; Qin, S.J.; Wang, F.L.; Tao, X. Exploring ChatGPT-Based Augmentation Strategies for Contrastive Aspect-Based Sentiment Analysis. *IEEE Intell. Syst.* **2025**, *40*, 69–76. [CrossRef]
22. Serral, E.; Smedt, J.D.; Vanthienen, J. Extending CPN Tools with Ontologies to Support the Management of Context-Adaptive Business Processes. In Proceedings of the Business Process Management Workshops–BPM 2014 International Workshops, Eindhoven, The Netherlands, 7–8 September 2014; Revised Papers; Fournier, F., Mendling, J., Eds.; Lecture Notes in Business Information Processing; Springer: Berlin/Heidelberg, Germany, 2014; Volume 202, pp. 198–209. [CrossRef]
23. Westergaard, M. CPN Tools 4: Multi-formalism and Extensibility. In Proceedings of the Application and Theory of Petri Nets and Concurrency–34th International Conference, PETRI NETS 2013, Milan, Italy, 24–28 June 2013; Proceedings; Colom, J.M., Desel, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7927, pp. 400–409. [CrossRef]
24. Schlosser, M.E. Embodied Cognition and Temporally Extended Agency. *Synthese* **2018**, *195*, 2089–2112. [CrossRef]
25. Schlosser, M. Agency. In *The Stanford Encyclopedia of Philosophy*, Winter 2019 ed.; Zalta, E.N., Ed.; Metaphysics Research Lab, Stanford University: Stanford, CA, USA, 2019. Available online: https://plato.stanford.edu/archives/win2019/entries/agency/ (accessed on 3 August 2025).
26. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Prentice Hall: Englewood Cliffs, NJ, USA, 2010.
27. Churchland, P.M. Eliminative Materialism and the Propositional Attitudes. *J. Philos.* **1981**, *78*, 67–90.
28. Dennett, D.C. True Believers: The Intentional Strategy and Why It Works. In *Scientific Explanation: Papers Based on Herbert Spencer Lectures*; University of Oxford: Oxford, UK, 1981; pp. 17–36.
29. Dennett, D.C. *The Intentional Stance*; MIT Press: Cambridge, MA, USA, 1987.

30. Quine, W.V.O. *Word and Object*; MIT Press: Cambridge, MA, USA, 1960.
31. Brooks, R.A. Intelligence without Representation. *Artif. Intell.* **1991**, *47*, 139–159. [CrossRef]
32. Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE J. Robot. Autom.* **1986**, *2*, 14–23. [CrossRef]
33. Borghoff, U.M.; Bottoni, P.; Pareschi, R. An Organizational Theory for Multi-Agent Interactions Integrating Human Agents, LLMs, and Specialized AI. *Discov. Comput.* **2025**, *28*, 138. [CrossRef]
34. Pareschi, R. Beyond Human and Machine: An Architecture and Methodology Guideline for Centaurian Design. *Sci* **2024**, *6*, 71. [CrossRef]
35. Saghafian, S.; Idan, L. Effective Generative AI: The Human-Algorithm Centaur. *arXiv* **2024**, arXiv:2406.10942. [CrossRef]
36. Jensen, K. *Coloured Petri Nets–Basic Concepts, Analysis Methods and Practical Use–Volume 3*; Monographs in Theoretical Computer Science. An EATCS Series; Springer: Berlin/Heidelberg, Germany, 1997. [CrossRef]
37. Van Der Aalst, W.M.P. The Application of Petri Nets to Workflow Management. *J. Circuits Syst. Comput.* **1998**, *08*, 21–66. [CrossRef]
38. Borghoff, U.M.; Bottoni, P.; Pareschi, R. Human-Artificial Interaction in the Age of Agentic AI: A System-Theoretical Approach. *Front. Hum. Dyn.* **2025**, *7*, 1579166. [CrossRef]
39. Jensen, K.; Kristensen, L. *Coloured Petri Nets–Modelling and Validation of Concurrent Systems*; Springer: Berlin/Heidelberg, Germany, 2009. [CrossRef]
40. Ratzer, A.V.; Wells, L.; Lassen, H.M.; Laursen, M.; Qvortrup, J.F.; Stissing, M.S.; Westergaard, M.; Christensen, S.; Jensen, K. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In *Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, 23–27 June 2003*; van der Aalst, W.M.P., Best, E., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2679, pp. 450–462. [CrossRef]
41. Pommereau, F. SNAKES: A Flexible High-Level Petri Nets Library (Tool Paper). In *Application and Theory of Petri Nets and Concurrency–36th International Conference, PETRI NETS 2015, Brussels, Belgium, 21–26 June 2015*; Devillers, R., Valmari, A., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9115, pp. 254–265. [CrossRef]
42. Fox, R.; Pakman, A.; Tishby, N. Taming the Noise in Reinforcement Learning via Soft Updates. In Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, New York, NY, USA, 25–29 June 2016; Ihler, A., Janzing, D., Eds.; AUAI Press: Arlington, VA, USA, 2016.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.